

The Statistics of Word Cooccurrences

Word Pairs and Collocations

Von der Philosophisch-Historischen Fakultät der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von
Stefan Evert
aus Ludwigsburg

Hauptberichter:	Prof. Dr. C. Rohrer
Mitberichter:	Apl. Prof. Dr. D. Kahnert
Mitberichter:	HD Dr. U. Heid

Tag der mündlichen Prüfung: 30. August 2004

Institut für maschinelle Sprachverarbeitung
Universität Stuttgart

2005

Heartfelt thanks

- ... to my supervisors Christian Rohrer and Ulrich Heid for giving me the opportunity to develop my own ideas and the time to write them down.
- ... to my supervisor Dietmar Kahnert for many of my favourite mathematics lectures, and for charting the linguistic unknown with me so willingly.
- ... to Brigitte Krenn and Ulrich Heid for introducing me to the world of collocations. A considerable part of the research presented here was inspired by or developed in joint projects with them.
- ... to Anke Lüdeling for being my linguistic conscience and a good friend at the same time; and for insisting that I must see the words behind the numbers.
- ... to Harald Baayen for introducing me to R and to word frequency distributions.
- ... to the R development team and the Perl community for first-class software that has become a cornerstone of my research.
- ... to everyone at the IMS for an environment that was fun and inspiring, and for great coffee breaks.
- ... to my parents for their support and patience on the many days when I was fighting writer's block. Without them, none of this would have been possible.
- ... and to Elke for making it all worth-while.

Contents

1	Introduction	15
1.1	About cooccurrences	15
1.1.1	Cooccurrences and collocations	15
1.1.2	Types of cooccurrences	18
1.1.3	Association measures	20
1.1.4	A first example	21
1.2	Applications of cooccurrence data	22
1.2.1	Applications of cooccurrences and collocations	22
1.2.2	Extracting collocations from text	25
1.3	Motivation and goals	28
1.3.1	The state of the art	28
1.3.2	Goals and Objectives	30
1.3.3	Limitations	31
2	Foundations	33
2.1	Corpus data	33
2.1.1	Frequency counts	33
2.1.2	Contingency tables and frequency signatures	35
2.1.3	Examples	37
2.1.4	Filtering cooccurrence data	40
2.2	A statistical model of cooccurrences	42
2.2.1	Cooccurrence data as a random sample	44
2.2.2	Independent Poisson sampling	47
2.2.3	The null hypothesis	49
2.2.4	Conditioning on fixed marginal frequencies	51
2.2.5	Measuring statistical association	54
2.3	Adequacy of the statistical models	57
2.3.1	Assumptions of the random sample model	57
2.3.2	Clustering and dispersion	60
2.3.3	Extraction noise	63
2.4	Positional cooccurrences	65
2.4.1	Segment-based cooccurrences	66
2.4.2	Distance-based cooccurrences	68
2.4.3	Examples	71
2.4.4	Discussion	71

3	Association Measures	75
3.1	An inventory of association measures	75
3.1.1	General remarks	75
3.1.2	Likelihood measures	77
3.1.3	Exact hypothesis tests	79
3.1.4	Asymptotic hypothesis tests	80
3.1.5	Point estimates of association strength	84
3.1.6	Conservative estimates of association strength	86
3.1.7	Measures from information theory	88
3.1.8	Heuristic, parametric and combined measures	89
3.2	Implementation	91
3.2.1	Know your numbers	91
3.2.2	The UCS toolkit	94
3.3	A geometric model of association measures	94
3.3.1	The coordinate space	94
3.3.2	Generalised association measures	96
3.3.3	Iso-surfaces and iso-lines	101
3.4	Comparing association measures	107
3.4.1	Goals and methods	107
3.4.2	The major groups	110
4	Quantisation Effects	119
4.1	Frequency distributions	119
4.1.1	A thought experiment	119
4.1.2	Introduction to lexical statistics	121
4.1.3	The conditional parameter distribution	123
4.2	The Zipf-Mandelbrot population model	124
4.2.1	Zipf's law	124
4.2.2	The Zipf-Mandelbrot model	125
4.2.3	The finite Zipf-Mandelbrot model	128
4.2.4	Evaluation of the models	129
4.3	Interpretation of the theoretical results	130
4.3.1	Sample-size independent results (ZM model)	130
4.3.2	Sample-size dependent results (fZM model)	132
4.3.3	Discussion	132
5	Evaluation	137
5.1	Evaluation of association measures	137
5.1.1	Evaluation methods and reference data	138
5.1.2	Precision and recall graphs	140
5.1.3	Fine-grained comparative evaluation	145
5.2	The significance of result differences	150
5.2.1	Evaluation as a random experiment	150
5.2.2	Confidence intervals and significance tests	153
5.2.3	Empirical validation	156
5.3	Evaluation based on random samples	159

6 Conclusion and Future Work	165
A Proofs and Mathematical Background	169
A.1 Proofs from Chapter 2	169
A.2 Proofs from Chapter 3	176
A.3 Proofs from Chapter 4	179
A.4 Some mathematical background	181
B UCS Software Documentation	185
B.1 UCS/Perl	185
B.1.1 General Documentation	186
B.1.2 UCS/Perl Programs	202
B.1.3 UCS/Perl Modules	220
B.2 UCS/R	268
Zusammenfassung	333
Summary	337

List of Tables

1.1	Highly associated verb + noun (direct object) pairs from the <i>British National Corpus</i> (BNC), ranked according to the log-likelihood measure.	22
2.1	List of special situations for the comparison of different coefficients of association strength. The symbol ϵ in Equations B and E indicates a first-order approximation for $\epsilon \rightarrow 0$.	57
2.2	Values of various coefficients of association strength for the special cases of independence (A), minimal association (B), total negative association (C), total positive association (D), nearly total association (E), and total determination (F and F')	58
2.3	Results of dispersion test for the AN-FR data set with $K = 200$ and $S = 8\,975$. The expected number of underdispersed types is rounded to the nearest integer. All observed results are significant at a level of $\alpha = .001$.	62
2.4	Results of dispersion test for the AN-FR data set with $K = 17\,950$ and $S = 100$. The expected number of underdispersed types is rounded to the nearest integer. All observed results are significant at a level of $\alpha = .001$.	62
2.5	Evaluation results for the extraction of German adjective-noun co-occurrences (from Evert and Kermes 2003).	64
4.1	Estimated shape parameter α , population size S , and goodness-of-fit statistic χ^2 for the ZM and fZM models applied to the AN-BNC and AN-HGC data sets.	130
5.1	Table of n -best precision values for various n -best lists and 5 different association measures on the PNV-FR-30 data set. The n -best lists marked ◀ are indicated by vertical lines in Figure 5.1.	141

List of Figures

2.1	Example of adjective-noun cooccurrences. The arrows indicate structural relations between a prenominal adjective and the noun that it modifies, corresponding to pair tokens in the formal model.	35
2.2	Contingency table of observed frequencies	36
2.3	Contingency table for the adjective-noun pair type (<i>black, box</i>) in the <i>British National Corpus</i>	37
2.4	Contingency table with row and column sums.	37
2.5	Contingency table for (<i>black, box</i>) with row and column sums.	38
2.6	Example for the extraction of PP-verb cooccurrences from a partial syntactic analysis produced by the YAC chunk parser.	41
2.7	Random variables representing the contingency table of a sample. . . .	46
2.8	The variability of the sample size: Histogram for the number of adjective-noun pair tokens extracted from subsets of the <i>Frankfurter Rundschau</i> corpus, containing 100,000 running words each. The solid curve shows the distribution expected by the independent Poisson sampling model.	48
2.9	Comparison of population probabilities with observed frequencies. . . .	49
2.10	Number of sentence repetitions in the <i>Frankfurter Rundschau</i> corpus, broken down by sentence length.	63
2.11	Contingency table for segment-based cooccurrences.	66
2.12	Contingency table for distance-based cooccurrences.	69
2.13	Alternative contingency table for distance-based cooccurrences.	70
2.14	Distribution of the lengths of articles and sentences in the <i>Frankfurter Rundschau</i> corpus.	72
3.1	Expected vs. observed frequencies.	76
3.2	Yates' continuity correction.	82
3.3	The three-dimensional parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set (stereograph for cross-eyed viewing).	95
3.4	The top row shows a rotated view of the parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set. The e -axis is nearly horizontal in this view, while the b -axis is oriented from background to foreground. The bottom right panel shows a projection of the point cloud into the (e, o) plane, and the bottom left panel shows the same data without jittering.	97

3.5	Parameter space with point cloud representing the PNV-SLICES-01 data set and iso-surfaces of the log-likelihood and Dice measures. The top row shows the iso-surface $\{g_{\log\text{-likelihood}} = 22.6\}$, corresponding to $p\nu = 10^{-6}$. The bottom row shows a 200-best iso-surface for Dice.	103
3.6	The top row shows a rotated view of the parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set and the iso-surface $\{g = 6\}$ of the Poisson measure (corresponding to coordinates with $p\nu = 10^{-6}$). The bottom row shows the orthogonal projection of both the point cloud and the iso-surface into the (e, o) plane. In the bottom left panel, the projection of the corresponding acceptance region $A_g(6)$ is shaded in the plane (corresponding to coordinates with $p\nu \geq 10^{-6}$).	105
3.7	Families of iso-lines representing the generalised association measures Poisson (left panel) and z-score (right panel). The threshold values for the iso-lines were chosen to correspond to specific p -values, including the common significance levels $p\nu = .01$ and $p\nu = .001$	106
3.8	Rotated view of 200-best iso-surfaces for the Dice (fine grid) and Poisson (coarse grid) measures, with the b -axis running from background to foreground.	107
3.9	Comparison of p -values for measures from the significance of association group, using Fisher as a reference point (labels on the axes refer to $-\log_{10} p\nu$).	111
3.10	The roots of overestimation: comparison of the Fisher and chi-squared p -values according to observed (left) and expected (right) frequency.	112
3.11	Comparison between likelihood measures ($-\log_{10} l\nu$, y -axis) and the corresponding exact hypothesis tests ($-\log_{10} p\nu$, x -axis).	113
3.12	Comparison of p -values between central and non-central variants of measures from the significance of association group.	114
3.13	Iso-surfaces of the log-likelihood measure g (fine grid) and its centralised version g^c (coarse grid) for the same threshold value (corresponding to $p\nu = 10^{-6}$).	115
3.14	Iso-lines for t-score, Poisson, the centralised version of log-likelihood and z-score with Yates' correction applied (all corresponding to $p\nu = 10^{-6}$).	115
3.15	Iso-lines for the MI measure as a point estimate (MLE) of $\log_{10} \mu$ and conservative estimates for different confidence levels α ($MI_{\text{conf},\alpha}$ measure).	116
4.1	Development of relative frequency spectrum and relative error of Herdan law (Heaps' law) with $\alpha = 0.87$ for the AN-HGC data set.	129
4.2	Expected frequency spectrum of ZM (left panel) and fZM (right panel) models compared to observed spectrum for the AN-HGC data set (logarithmic scale).	131

4.3	Comparison of the p -value computed by the Poisson association measure against the expected proportion of low-probability types in frequency classes $m = 1, 2, 3$ and 5 , for a population described by a ZM model with shape parameter α . The graphs in the bottom row cover a wider range of expected frequencies for $m = 1, 2$	134
4.4	Comparison of the p -value computed by the Poisson association measure against the expected proportion of low-probability types in frequency classes $m = 1, 2, 3$, and 5 . These graphs show the predictions of a fZM model estimated from the AN-HGC data set for three different sample sizes.	135
5.1	Graphs of n -best precision for five association measures evaluated on the PNV-FR-30 data set. The vertical lines mark n -best lists for $n = 800$ and $n = 2\,300$	142
5.2	Precision graphs for n -best lists with $n \leq 2\,300$ on the PNV-FR-30 data set.	143
5.3	Recall graphs for the PNV-FR-30 data set. The vertical lines mark n -best lists for $n = 800$ and $n = 2\,300$	144
5.4	Assessing the practical usefulness of association measures with precision-by-recall graphs (on the PNV-FR-30 data set). The diagonal lines indicate n -best lists for $n = 800$ and $n = 2\,300$	144
5.5	Comparison of the performance of association measures for figurative expressions (top panel) vs. support-verb-constructions (bottom panel). It is pure coincidence that the baseline precision is the same for both types of collocations.	146
5.6	Estimates of the local precision in different parts of the ranked candidate lists for the extraction of support-verb-constructions.	147
5.7	Comparison of precision of n -best lists ($n \leq 1\,450$) for pair types accepted by the kwic filter (Krenn 2000, 120) in the left column vs. the rejected pair types in the right column. The top row shows overall precision, the middle row precision for figurative expressions, and the bottom row precision for support-verb constructions.	148
5.8	Comparison of the precision of n -best lists ($n \leq 2\,000$) extracted with a combination of the kwic filter (Krenn 2000, 120) and a frequency threshold of $f \geq 10$ (left column) vs. a frequency threshold of $f \geq 30$ but no filter (right column). The top row shows overall precision, the middle row precision for figurative expressions, and the bottom row precision for support-verb constructions.	149
5.9	Illustration of evaluation experiment as the random selection of true and false positives from a hypothetical population.	152
5.10	Precision graphs for G^2 and X^2 with 95% confidence intervals.	154
5.11	Illustration of the significance of precision differences between two association measures (here, G^2 and X^2 are compared (left panel: overlapping acceptance regions A and B ; right panel: difference regions D_1 and D_2).	155
5.12	Significant differences between G^2 and X^2 at a confidence level of 95%.	156

5.13	Distribution of the observed precision P_A for γ -acceptance regions of the association measures G^2 (left panel) and t (right panel). The solid curves indicate the expected distribution according to Eq. (5.2).	157
5.14	Empirical confidence intervals for the n -best precision $p_{g,n}$ of the association measures G^2 (top right panel), X^2 (bottom right panel) and t (bottom left panel).	158
5.15	An illustration of the use of random samples for evaluation: precision graphs for the PNV-KRENN data set (left) and the corresponding estimates obtained from a 10% sample (right).	159
5.16	Sample estimates for the true precision with confidence intervals based on a 10% random sample. The dashed lines show the true precision computed from the full candidate set.	160
5.17	Chart of binomial confidence intervals for selected sample sizes.	161
5.18	Random sample evaluation of German adjective-noun combinations.	163

Chapter 1

Introduction

1.1 About cooccurrences

1.1.1 Cooccurrences and collocations

You shall know a word by the company it keeps! With this slogan, Firth (1957) drew attention to a fact that language scholars had intuitively known for a long time: In natural language, words are not combined randomly into phrases and sentences, constrained only by the rules of syntax. The particular ways in which they go together are a rich and important source of information both about language and about the world we live in. In the 1930s, J. R. Firth coined the term **collocations** for such characteristic, or “habitual” word combinations (as he called them). While Firth used to be lamentably vague about his precise understanding of this concept (cf. Lehr 1996, 21), the term itself and the general idea behind it – that collocations “correspond to some conventional way of saying things” (Manning and Schütze 1999, 151) – were eagerly taken up by researchers in various fields, leading to the serious terminological confusion that surrounds the concept of collocations today. As Choueka puts it: “even though any two lexicographers would agree that ‘once upon a time’, ‘hit the road’ and similar idioms are collocations, they would most certainly disagree on almost anything else” (Choueka 1988, 4). Feel free to replace “lexicographers” with any profession that is concerned with language data.¹

The diverse notions of collocations that have evolved over the past fifty years can generally be divided into two groups: a *distributional* and an *intensional* approach. The **distributional** approach is mainly due to Firth’s successors and disciples in the United Kingdom, most notably M. A. K. Halliday. Often referred to as the Neo-Firthian school, they lapsed on to the empirical side of Firth’s notion of collocations as recurrent word combinations in a particular text, gradually developing a formal and operational definition of this concept. The Neo-Firthians understood collocations as a directly observable quantity that serves a purely descriptive purpose. Some proponents of this point of view go so far as to rule out any automatic processing or linguistic interpretation of the source text. Lehr (1996) – who speaks of the deliberate renunciation of additional information, “bewußter Informationsverzicht” (Lehr 1996, 50f) – provides an example of the most extreme kind, allowing only fully

¹Hausmann (1989), a German lexicographer, might even disagree about the status of Choueka’s examples as collocations.

deterministic operations such as the identification of graphemic words (sequences of alphabetic characters delimited by whitespace or punctuation) in her unimplemented design of an extraction system. The distributional notion of collocations has also become one of the fundamentals of a recent corpus-oriented lexicographic tradition in the United Kingdom (see Sinclair 1991). Williams (2003) gives a concise and well-written overview of the Neo-Firthian concept of collocations. See Lehr (1996) for a more detailed account and Monaghan (1979) for an in-depth discussion.

Outside the Neo-Firthian tradition, the term *collocation* has been applied to a wide range of lexicalisation phenomena, giving rise to a variety of **intensional** definitions. Collocations are usually placed somewhere in the grey area between fixed idioms and free combinations, often in a phraseological framework (e.g. Burger *et al.* 1982). In a narrower sense, they are understood as semi-compositional word pairs, with one “free” element (the *base*) and the other element lexically determined (the *collocate*). Well-known examples of collocations in this sense are *a pride of lions*, *a school of fish*, *reckless abandon*, *heavy smoker*, as well as support verb constructions such as *give a speech* and *set an alarm*. While the free element retains its independent meaning in the combination, the collocate often contributes a meaning component that it cannot have on its own. This concept has come to play an important role in computational lexicography (Hausmann 1989; Grossmann and Tutin 2003) and can be formalised in terms of lexical functions (see Mel’čuk (2003) for a concise summary). Until recently, the mainstream of theoretical linguistics has shown little interest in collocations. Under the influence of Chomsky, the lexicon was reduced to a mere list of fully interchangeable words. When syntax was inadequate to account for the combinatorics of words, they were explained as selectional restrictions (or preferences) at a conceptual level. Bartsch (2004, 27–64) gives an excellent overview of the diverse theoretical approaches to the concept of collocations.

In the field of natural-language processing (NLP), the combinatorics of words have always played an important role, even in the very early days when researchers still referred to their work as “mechanized documentation” (Stevens *et al.* 1965). While the first publications spoke of *associations* between words (Giuliano 1965a), the term *collocations* was soon adopted (Berry-Rogghe 1973; Choueka *et al.* 1983). Since much of the research in NLP is driven by the requirements of applications, it is hardly surprising that the term is used in a much broader and more practical sense than in linguistics. Word combinations that are considered as collocations range from compound nouns (*black box*), over semantically opaque idiomatic expressions (*kick the bucket*), to fully compositional combinations that are only lexically restricted (*handsome man* vs. *beautiful woman*). This variability in definition is mirrored by a large number of alternative terms that are used almost interchangeably, such as *multi-word expressions* (MWE), *multi-word units* (MWU), *bigrams* and *idioms*.

Three characteristic properties emerge as a common theme in the linguistic treatment of collocations: semantic *non-compositionality*, syntactic *non-modifiability*, and the *non-substitutability* of components by semantically similar words (Manning and Schütze 1999, 184).² Collocation definitions in the field of natural-language processing are usually based on the same three criteria, which are used in various combinations and interpreted in a more or less strict sense. However, for most researchers, any definition according to linguistic criteria has to be complemented – and is some-

²See also Krenn (2000, 14–18) and Bartsch (2004, 58f).

times overridden – by the relevance of the respective word combination for an intended application. For instance, Choueka (1988) gives a relatively precise definition of a “collocational expression” as a “syntactic and semantic unit whose exact and unambiguous meaning or connotation cannot be derived directly from the meaning or connotation of its components”. However, realising the multitude of borderline cases that such definitions are bound to create, he proposes some guidelines for the distinctions between collocations and non-collocations. These guidelines can be boiled down to the central question: “Does it deserve a special entry in a dictionary or lexical database of the language?” – which is the intended application of Choueka’s work. A similar example is provided by Schone and Jurafsky (2001), who are interested in MWUs to be used as headwords in machine-readable dictionaries. Again, the practical relevance is an essential ingredient of their definition, which is otherwise based on the criteria listed above (citing Manning and Schütze 1999; Choueka 1988). Note that the distinction between collocations and non-collocations is ultimately based on the intuition of a lexicographer, for instance, in contrast to the formal and unambiguous definitions that linguistic research aims for.

In order to make a clear distinction between the two approaches to collocations, I refer to the distributional notion as **cooccurrences**, which encompasses both the observable (cooccurrence) frequency information and its interpretation as an indicator of statistical association.³ This description seems fully adequate for the Neo-Firthian understanding of a collocation as a recurrent word combination, cf. the definition “collocation is the occurrence of two or more words within a short space of each other in a text” (Sinclair 1991, 170). By contrast, I reserve the term *collocation* for an intensionally defined concept that does not depend on corpus frequency information. Not wanting to embrace any particular theory of collocations, I propose the following partial definition that encompasses both the criteria of Manning and Schütze (1999) and the criterion of application relevance:

A **collocation** is a word combination whose semantic and/or syntactic properties cannot be fully predicted from those of its components, and which therefore has to be listed in a lexicon.

I use *collocation* thus as a generic term whose specific meaning can be narrowed down according to the requirements of a particular research question or application. The precise interpretation of the definition depends on the properties that are considered (e.g. semantic compositionality vs. syntactic modifiability), on the processes involved in “predicting” the properties of the combination (e.g. composition of literal meanings vs. metaphoric interpretation), and on the form and intended usage of the lexicon (which may range from the word list of a syntactic parser to the human mental lexicon in psycholinguistic research). Bartsch (2004, 58f) makes a similar distinction between cooccurrences and collocations, although she gives a much narrower and more concrete definition for the latter.

My thesis is primarily concerned with cooccurrences and their statistical association. Since the generic definition of collocations above implies some degree of lexicalisation (because their unpredictable properties have to be learned and stored), they must be recurrent combinations (so that they can be learned, and in order to

³I will explain what it means for words to “cooccur” in more detail in Section 1.1.2.

warrant the effort of storing them). Therefore, measures of statistical association abstracted from the cooccurrence frequency data should provide evidence for collocational word combinations. Such collocation extraction tasks, which are addressed in Chapter 5, link my work to the research on collocations and their applications. The evaluation methods presented there apply equally well to any specialisation of the generic collocation definition: only the evaluation results will differ.

1.1.2 Types of cooccurrences

In this section, I give a more precise definition for the concept of word cooccurrences. As a first problem, it is not at all obvious how to define a “word”. A considerable amount of work – especially in English-speaking countries and by the proponents of so-called “knowledge-free” approaches – has been based on graphemic words that are delimited by whitespace and punctuation. However, variants such as *whitespace* vs. *white-space* vs. *white space*⁴ show the inconsistency of such an approach. Moreover, relevant lexical items may be substrings of graphemic words (e.g. in German compounds⁵) or comprise multiple graphemic words (as in the *white space* example). Cooccurrences can provide useful information at all these levels, and they will exhibit similar properties. Therefore, I use **word** as an entirely generic term which may refer to any kind of lexical item, depending on the underlying theory or intended application.

Similarly, the **cooccurrence** of words can be defined at different levels: as mere (graphemic) adjacency or proximity, as occurrences within the same linguistic unit (sentence, paragraph, article, etc.), or as a specific structural (usually syntactic) relationship between the words. Especially in the latter case, cooccurrences may be relations between more than two words (the expression *to keep a straight face* is clearly a systematic combination of verb, adjective, and noun). In my thesis, I will only consider cooccurrences of two words, though, which I refer to as **word pairs**.⁶ A motivation for this restriction can be found at the end of Section 1.3.2. There is a broad distinction between two **types of cooccurrences**, which I call *relational* and *positional* cooccurrences. This distinction is not merely conceptual: the different types of cooccurrences also require different counting methods and statistical models (cf. Chapter 2).

Positional cooccurrences represent the historically older approach, where words are said to cooccur when they appear within a certain distance from each other. This distance is typically measured by the number of intervening words (usually in the sense of graphemic words) and referred to as the **(collocational) span** (Sinclair 1991, 175). Alternatively, linguistically motivated windows (clauses, sentences, paragraphs, documents, etc.) may be used. The positional approach has been widely

⁴Evidence for all three spellings can easily be found with an Internet search engine, e.g. <http://www.google.com/>. A similar example is *fulltime* vs. *full-time* vs. *full time*, all of which are attested in the *British National Corpus* (Aston and Burnard 1998).

⁵In analogy to multi-word nouns such as *hard disk* in English, German compounds (e.g. *Festplatte*, the German translation of *hard disk*) can be interpreted as cooccurrences of (free) morphemes.

⁶Some authors use the term *bigram*, and more generally *n-gram* for the combination of *n* words. However, I try to avoid this term as it is often understood to imply adjacency (*n*-grams being *uninterrupted* sequences of *n* words).

adopted by the Neo-Firthian school (e.g. Lehr 1996) and by early work in computational linguistics (e.g. Stevens *et al.* 1965) before automatic syntactic analysis became feasible. An advantage of positional cooccurrences is that they are directly observable in corpus data. Especially when based on a graphemic definition of words, positional cooccurrence frequencies can be determined reliably with fully automatic means.⁷

Relational cooccurrences, on the other hand, are based on a *linguistic interpretation* of the observable corpus data. Each cooccurrence corresponds to an instance of a specific structural relation. Typical examples of such relations are graphemic adjacency (but *not* a collocational span or window), dependency relations and (underspecified) subtrees in a phrase-structure analysis. In the latter case, relations between words are usually mediated by larger syntactic units. For instance, consider a prenominal adjective *A* and the modified noun *N*: a direct syntactic relation holds between the AP of which *A* is the head and the NP headed by *N* (although the precise relation between the AP and the NP depends on the particular flavour of syntax being used). Syntactic relations that are often considered by work on English (and German) collocations include the following: (i) verb + noun (direct object), e.g. *commit suicide*; (ii) adjective + noun, e.g. *reckless abandon*; (iii) adverb + verb, e.g. *tacitly agree*; (iv) verb + predicative adjective, e.g. *keep sth handy*; (v) verb-particle constructions, e.g. *bring sth up*; and (vi) verb + prepositional phrase, e.g. *set in motion*. When cooccurrence data are intended for knowledge extraction, there is usually a strong emphasis on verb + noun relations.

The identification of relational cooccurrences in any substantial amount of text requires automatic linguistic pre-processing (typically including a partial or full syntactic analysis), which will invariably introduce errors into the results. Critics also point out that the relational approach precludes an unbiased analysis of the observable facts by imposing the preconceived notions of a particular linguistic theory on the data. Nonetheless, I believe that the following three arguments outweigh any disadvantages:

1. On a theoretical level, it is quite obvious that the results of a quantitative analysis will be much more clear-cut and meaningful when they are based on linguistic understanding rather than just mindless computation. As Greenbaum (1970, 13) puts it: “A more valuable, if more modest, contribution might be made to the study of collocations if a relatively homogeneous class of items were selected and an investigation undertaken of the collocation of each item in the class with other items that are related syntactically in a given way.”
2. On a practical level, positional cooccurrences represent a mixture of many different kinds of structural relations, at least as many as there are different reasons for words to cooccur within a given span or a sentence. It should be obvious that the various types of relations follow substantially different frequency distributions. Statistical methods that are based on simple frequency counts will produce much better results when they are applied to a single “homogeneous” frequency distribution rather than to such a mixture. Support for this claim comes from many studies (e.g. Daille 1994; Justeson and Katz 1995a; Breidt

⁷Lehr (1996) argues in some detail that this is a desirable property, although she refers to *collocations* rather than *cooccurrences*, of course.

1993; Smadja 1991, 1993; Lezius 1999), who obtain substantial improvements from part-of-speech tagging and the use of (simple) syntactic patterns.⁸

3. On a mathematical level, the statistical model for relational cooccurrences (Section 2.2) is simpler and more elegant than the models required for the analysis of positional cooccurrences (Section 2.4). It is the same random-sample model that is used in biometrics (for a famous example, see Good 1953) and lexical statistics (see Baayen 2001), so that important results from these fields can be applied to the analysis of cooccurrence data (especially in Section 2.3 and Chapter 4).

For these reasons, the present thesis concentrates on **relational cooccurrences**. Considering that tools for linguistic pre-processing and automatic syntactic analysis are widely available nowadays,⁹ I believe that applications in computational linguistics that involve the extraction of cooccurrence data should always be based on a relational model. Possible exceptions are purposely “knowledge-free” approaches (whose aim is to avoid pre-conceived linguistic notions) as well as some cases where the desired relation cannot be identified reliably with the current technology. Positional cooccurrences are briefly considered in Section 2.4, where I distinguish between two subtypes (*segment-based* vs. *distance-based* cooccurrences), describe appropriate counting methods, and introduce the corresponding statistical models. These are found to be similar to the model for relational cooccurrences (Section 2.2), so that most of the methods and results in the present thesis apply equally well to positional data.

1.1.3 Association measures

Raw cooccurrence data – in the form of frequency counts for word pairs – have two serious shortcomings. First, the plain frequencies are often not meaningful as a measure for the amount of “glue” between two words. Provided that both words are sufficiently frequent, their cooccurrences might be pure coincidence. Therefore, a **statistical interpretation** of the frequency data is necessary, which determines the degree of statistical association between the words. Second, the observed cooccurrences only provide information about the one particular corpus they were extracted from. It is usually desirable to make **generalisations** about the language as a whole (or, more realistically, about a well-defined sub-language). This is achieved by methods of statistical inference that interpret the source corpus – and hence the cooccurrence data – as a random sample from the language or sub-language of interest. A

⁸Cf. Goldman *et al.* (2001, 61): “There is no doubt that syntactic dependencies, such as the ones expressed by grammatical functions or by modification relations between two terms constitute a more appropriate criterion of relatedness than simple linear proximity, such as 2 or 3 words away.” The authors support this claim with some examples of verb-object combinations such as *éprouver difficultés* ‘to experience problems’, arguing that a verb and its object can be more than 30 words apart in French (Goldman *et al.* 2001, 62).

⁹Compare this with the situation in 1988, as perceived by (Choueka 1988, 3): “morphological, syntactical or semantical modules, . . . , even when available, certainly cannot be applied to large corpora in any reasonably efficient way”. Only three years later, however, Frank Smadja strikes a more optimistic note: “the advent of robust parsers such as **Cass** [Abney, 1990], **Fidditch** [Hindle, 1983] has made it possible to process large amounts of text with good performance” (Smadja 1991, 280).

statistical model can then be formulated that allows us to predict to what extent the observed cooccurrences may be merely due to chance (i.e. the particular choice of source corpus), or whether they provide sufficient evidence for a “true” association between the words (that holds for the entire language or sub-language).

The most widely used method for distinguishing between random cooccurrences and true statistical association is the application of so-called **association measures**. Such measures compute an **association score** for each word pair, which can then be used for ranking (putting pairs with high scores at the top) or selection (by setting a cutoff threshold). It should be obvious now that both tasks, statistical interpretation and generalisation, are closely related, being based on the same notions of coincidence and chance. It is an advantage of association measures that they can address both problems simultaneously, but it is also a great challenge that they *have* to address both problems: an association score is a single real number that serves as a compound measure both for the degree of association and for the amount of evidence supporting it. As we will see in Chapter 3, some measures focus on either of the two while others attempt to strike a good balance.

The earliest reports of the application of association measures to language data go back to Stevens *et al.* (1965). Even at that time, an enormous range of different measures was available, borrowed from mathematical statistics and related fields. During the past 40 years, various new association measures have been suggested, while others were forgotten and later re-discovered (sometimes tagged with a different name). However, only a few have achieved sustained popularity. Among the best-known measures are MI (Mutual Information, an information-theoretic notion suggested by Church and Hanks (1990)), the t-score measure t (Church *et al.* 1991), the log-likelihood ratio G^2 (Dunning 1993), and to a lesser extent also the chi-squared statistic X^2 , which is the standard method to distinguish between chance cooccurrence and true association in mathematical statistics (Agresti 1990, 47f).¹⁰ These measures, as well as many less well-known alternatives, are described in detail in Section 3.1.

1.1.4 A first example

As a first example, let us consider English verb + noun (direct object) cooccurrences extracted from the *British National Corpus* (BNC) with the help of simple part-of-speech patterns. This resulted in 5 365 different word pairs with at least 25 occurrences in the BNC (see the description of the VN-BNC data set in Section 2.1.3 for details). In order to identify strongly associated word pairs, the log-likelihood measure¹¹ was applied, and the word pairs were ranked according to the association scores. The forty strongest associations found in this way are listed in Table 1.1, together with their cooccurrence frequency and association score.

Among the list of word highly associated word pairs, many different linguistic phenomena can be found: fixed idiomatic expression (*take place* and *give rise (to)*), support verb constructions and other lexically determined combinations (*make sense*, *play (a) role*, *solve (a) problem*, and *shut (the) door*), stereotypes and formulaic ex-

¹⁰The names of association measures are printed in a sans-serif face in this thesis.

¹¹The log-likelihood measure is widely used for this purpose in the field of computational linguistics. From the discussion in Chapter 3, it emerges as a meaningful, sound and robust association measure.

word pair	freq.	association	word pair	freq.	association
<i>take place</i>	7606	41942.15	<i>meet needs</i>	520	4183.22
<i>play role</i>	1488	11710.46	<i>make mistake</i>	763	4114.63
<i>open door</i>	1438	11299.73	<i>make decision</i>	1172	3943.51
<i>see chapter</i>	1461	9795.36	<i>keep eye</i>	577	3671.20
<i>give rise</i>	1499	9521.99	<i>tell storey</i>	527	3616.61
<i>make sense</i>	1888	7996.27	<i>show sign</i>	533	3577.90
<i>take advantage</i>	1557	7529.19	<i>pay tribute</i>	336	3390.79
<i>see page</i>	1294	7374.60	<i>thank goodness</i>	224	3338.03
<i>play part</i>	1331	7359.75	<i>take action</i>	1023	3302.98
<i>draw attention</i>	836	6610.98	<i>shake hand</i>	342	3289.48
<i>answer question</i>	743	6558.03	<i>take step</i>	759	3271.63
<i>take part</i>	2358	6424.23	<i>get hold</i>	614	3265.61
<i>ask question</i>	898	6373.39	<i>form basis</i>	448	3191.22
<i>take care</i>	1295	6196.21	<i>ring bell</i>	235	3093.21
<i>ask secretary</i>	621	5932.39	<i>closed door</i>	346	3091.96
<i>solve problem</i>	645	5706.53	<i>shut door</i>	322	3039.70
<i>wait minute</i>	428	5422.05	<i>write letter</i>	445	3023.47
<i>make use</i>	1441	4954.34	<i>give impression</i>	638	2948.46
<i>take account</i>	1164	4626.66	<i>make contribution</i>	682	2890.16
<i>form part</i>	886	4335.17	<i>raise question</i>	555	2882.07

Table 1.1: Highly associated verb + noun (direct object) pairs from the *British National Corpus* (BNC), ranked according to the log-likelihood measure.

pressions (*see chapter . . .*, *wait (a) minute*), but also free and compositional combinations that reflect facts of life, typical behaviour, or just happen to be frequent in the corpus (*ask (the) Secretary (of State)* and *write (a) letter*). Some of the entries point to potential problems of the automatic processing and data extraction: *ask secretary* is a misleading reduction and normalisation of *ask the Secretary of State*. Similarly, *ring (a) bell* can have both a literal and a figurative meaning, both of which are likely to occur in the corpus. It is impossible to find out from the cooccurrence data alone which of the two meanings is more frequent and hence contributes more to the association of the word pair. What all forty entries have in common, though, is that they sound very familiar to anyone with a good command of English.

A quick look at the numbers shows that statistical association (at least according to log-likelihood) is closely linked to cooccurrence frequency. However, this cannot be the only determining factor: the relatively frequent word pair *make think* ($f = 512$) obtains a low association score and is ranked at the 1722nd position.

1.2 Applications of cooccurrence data

1.2.1 Applications of cooccurrences and collocations

For applications in the field of natural-language processing, both cooccurrences and collocations play an important role. **Cooccurrences** represent the observable evidence that can be distilled from a corpus by fully automatic means. After statistical

generalisation, this information can be used to predict which word combinations are most likely to appear in another corpus. In such a way, cooccurrence data have been applied to the following tasks:

- resolving ambiguities in PP-attachment (Hindle and Rooth 1993; Volk 2002), syntactic parse trees (Alshawi and Carter 1994) and the internal structure of compound nouns (Yoon *et al.* 2001);
- the identification of sentence boundaries (Kiss and Strunk 2002b,a) and constituent boundaries (Magerman and Marcus 1990);
- lexical choice in natural language generation and gap-filling tasks (Edmonds 1997; Terra and Clarke 2004);¹²
- the adaptation of *n*-gram language models, using known associations as *triggers* to adjust occurrence probabilities (Rosenfeld 1996; Beeferman *et al.* 1997), as well as improvements on language models based on probabilistic context-free grammars (Eisele 1999, 109-125);
- the prediction of human word association norms from psycholinguistic experiments (Rapp 2002);¹³
- contrastive cooccurrence data (obtained from different corpora or subcorpora) have been used for a variety of classification tasks, ranging from word sense disambiguation (Biber 1993; Justeson and Katz 1995b; Pedersen 2001; Resnik 1997; Rapp 2004) to the detection of topic shifts (Ferret 2002) and subjectivity (Wiebe *et al.* 2001).

In addition to these direct uses, cooccurrence data often serve as a basis for distributional methods, which compare the “cooccurrence profile” of a given word, a vector of association scores for its cooccurrences, with the profiles of other words. The distance between two such vectors (which can be defined in various ways) is interpreted as an indicator of their semantic similarity. Clustering and dimensionality reduction methods (such as factor analysis or singular-value decomposition) can then be used to identify classes of semantically related words. Some applications of such distributional techniques are:

- detecting semantic similarities between words (Landauer and Dumais 1997; Läuter and Quasthoff 1999; Heyer *et al.* 2001; Biemann *et al.* 2004), especially for the identification of synonyms (Turney 2001; Rapp 2002; Terra and Clarke 2003);¹⁴

¹²Gap-filling tasks are often used in language tests, where students are presented with a choice of four near-synonyms and have to select the word which fits most naturally into a given sentence. Terra and Clarke (2004) evaluated their methods on such a gap-filling exercise from the verbal section of the GRE test (see <http://www.gre.org/>). The problem of lexical choice in natural language generation involves a similar task, where an appropriate lexical item has to be chosen to express a given concept.

¹³Rapp (2002) compares his cooccurrence data with the responses of human subjects to stimulus words, collected in the Edinburgh Associative Thesaurus (Kiss *et al.* 1973).

¹⁴Such algorithms are often evaluated on the synonym task from the TOEFL language test, where students are presented with a choice of four or more words and have to select the one that is closest in meaning to a given keyword. Since the alternatives usually represent quite distinct semantic concepts

- the unsupervised induction of word senses, usually combined with disambiguation of the automatically identified senses (Pantel and Lin 2002; Rapp 2003; Tamir and Rapp 2003; Dorow and Widdows 2003);
- the identification of translation equivalents (which are semantically related, of course) from non-parallel corpora, i.e. unrelated texts in two or more languages (Rapp 1999);
- distinguishing between compositional and lexicalised compound nouns, based on the assumption that the former are more similar to their head noun (Zinsmeister and Heid 2004);
- the selection of informative clauses for the compilation of biographical summaries (Schiffman *et al.* 2001).

Several authors use association scores directly for such tasks, relying on the tendency of semantically related words to cooccur within sentences or within specific syntactic patterns:

- cooccurrences within sentence windows provide evidence for the identification of synonyms (Terra and Clarke 2003) as well as antonyms (Justeson and Katz 1991);
- translation equivalents can be obtained from the cooccurrences of words in aligned sentence pairs (Church and Gale 1991; Smadja *et al.* 1996);
- Hisamitsu and Niwa (2001) extract (fully interchangeable) term variants and expansions of acronyms from parenthetical expressions of the form *A (B)*;
- Baroni *et al.* (2002) use semantic similarities found in such a way as one criterion for the identification of morphologically related words.

In contrast to the distributional character of cooccurrences and statistical association, **collocations** represent intrinsic properties of word combinations. Depending on the specific collocation definition used, these properties can be relevant for various applications. The field of lexicography, which has always been a driving force behind theoretical and practical work on collocations, provides also their most immediate application. However, collocations will not only be found in traditional paper dictionaries, but also in “mechanized dictionaries” (Choueka 1988), machine-readable lexical resources that range from simple lists of collocations to databases containing rich amounts of information at various levels. In this way, collocational knowledge becomes an essential part of many language processing applications:

- Collocations are an essential part of the microstructure both of monolingual dictionaries (cf. Heid 2004) and bilingual dictionaries (see e.g. Heid *et al.* 2000; Smadja 1993, 171–174), where they are particularly important because of their

(but may be similar in form, so they are easily confused by language learners), this is a fairly easy task for a computer program that has access to a sufficient amount of corpus data. Unsurprisingly, the automatic methods are almost on par with human native speakers and perform better than most non-native speakers (e.g. Rapp 2002; Terra and Clarke 2003).

contrastive relevance (i.e. because their component words cannot be translated individually). Monolingual learner dictionaries need to provide a wide range of collocations (even those that are not semantically opaque) to aid non-native speakers in text production (Hausmann 1989, 2004).

- Similar to the learner dictionaries, collocational knowledge is essential for natural language generation in order to ensure that the generated text reads smoothly (e.g. Stone and Doran 1996).
- Information about the semantic and syntactic irregularities of word combinations is important for symbolic approaches to deep syntactic analysis, especially with lexicalised grammars such as HPSG and LFG (e.g. Erbach and Krenn 1993).
- Machine-readable dictionaries of collocations and their translation equivalents are indispensable for high-quality machine translation (Smadja *et al.* 1996, 5–6), especially when using symbolic methods with hand-crafted rules.
- Other applications of such dictionaries include machine-assisted translation, multilingual information retrieval and multilingual summarisation (Smadja *et al.* 1996, 30–31).

Applications of collocations in natural language generation, computational lexicography and information retrieval are also described by Manning and Schütze (1999, 152, 187–189).

One of the most important applications of cooccurrence data is the semi-automatic identification of collocations, which is described in more detail in the following section. Based on the intuition that statistical association should correlate with collocativity up to a certain degree,¹⁵ cooccurrences with high association scores are interpreted as **collocation candidates**. The correspondence is far from perfect, of course. For this reason, the candidates are usually validated by human annotators, who identify true collocations among them manually. Sometimes automatically extracted cooccurrence data are used directly as a “noisy” substitute for a list of manually validated collocations, but Lemnitzer (1998) argues for the necessity of a semi-automatic procedure.

1.2.2 Extracting collocations from text

The standard design of a collocation extraction tool has the form of an **extraction pipeline** as described by Evert and Kermes (2003). First, the corpus is pre-processed and often also syntactically annotated. Then cooccurrences are extracted and may be filtered to improve accuracy. Typical filters set a minimal threshold for the cooccurrence frequency, remove stopwords or discard certain patterns (cf. Section 2.1.4). An

¹⁵This assumption is reasonable at least for collocation definitions that are compatible with the generic definition given in Section 1.1.1. A word combination with unpredictable properties has to be stored as a unit in the lexicon (whether mental or computational), which should make it more easily accessible in language production, increasing its probability of occurrence. On the other hand, collocations must be sufficiently frequent in the language so that their idiosyncratic properties can be learned.

association measure is chosen and applied to the frequency data. Finally, the collocation candidates are either classified into accepted and rejected candidates or they are ranked according to the association scores. In the first case, the classification is sometimes based on a pre-defined threshold for the association scores, but it is more common to accept the n highest ranking candidates, which are also referred to as an **n -best list**. Only the accepted candidates are passed on to the human annotators, and they will often be sorted alphabetically or grouped by one of the component words. In the second case, the full ranked lists are given to the annotators, who work their way down from the top of the list until the true collocations become too few and far between. Most approaches assume that there is a **binary distinction** between collocational and non-collocational pairs. Therefore, the candidates accepted by the extraction pipeline are classified as **true positives** (if they are in fact collocations) or **false positives** (otherwise) by the human annotators. When there are more than two categories (for instance, Krenn (2000) distinguishes between figurative expressions, support-verb constructions and free combinations), they can often be seen as more fine-grained subdivisions of the sets of true and false positives (cf. Krenn *et al.* 2004). Graded judgements of the degree of collocativity may be more informative, but they require ratings by multiple human subjects obtained in a carefully designed psycholinguistic experiment (e.g. Lapata *et al.* 1999).

The literature abounds with descriptions of collocation extraction tools, most of which are intended for applications in computational lexicography, terminology or the compilation of machine-readable dictionaries:

- Prototypical examples of the semi-automatic extraction pipeline design are Lin (1998) for English, Nerima *et al.* (2003) for French, Lemnitzer (1998) for German, as well as Kaalep and Muischnek (2003) for Estonian multi-word verbs. The first two examples extract head-modifier pairs from deep syntactic analyses, i.e. relational cooccurrences *par excellence*.
- Kermes and Heid (2003) extract adjective-verb cooccurrences as base data for the identification of collocations, but do not apply association measures. The same holds for one of the earliest publications on collocation extraction tools (Choueka 1988), which mentions preliminary experiments with filtering and ranking by association scores.
- XTRACT (Smadja 1993), perhaps the most well-documented collocation extraction system so far, combines association scores with various heuristics, syntactic patterns, and other filters.¹⁶ In addition, the word pairs extracted in a first step are combined into longer sequences, which may include optional or unspecified elements.
- Systems for the extraction of compound terms and terminologically relevant collocations are described by Daille (1994, 1996) and Justeson and Katz (1995a). A recent more advanced system combines association measures with other extraction techniques using a range of voting schemes (Vivaldi and Rodríguez

¹⁶In particular, Smadja applies statistical filters that are based on the frequency distribution of the collocates of a given keyword as well as the distance between the cooccurring words. Such empirical methods can only be applied to high-frequency keywords. In this case, a threshold of $f > 100$ was used (Smadja 1993, 168).

2001). See Kageura and Umino (1996) for an overview of standard term extraction methods.

- Dias (2003) extracts relational cooccurrences with local syntactic patterns that are automatically learned from the source corpus.
- Bannard *et al.* (2003) identify phrasal verbs in English, i.e. non-compositional verb-particle pairs.
- The unsupervised learning of the subcategorisation frames of verbs can be interpreted as the identification of collocations between verbs and (surface hints for) argument structures (Brent 1993).

The collocation extraction task also has theoretical interest because it can help to throw light on the relation between cooccurrences and collocations. Extraction pipelines as described above are particularly suitable for the **empirical evaluation** and comparison of association measures, which are a pivotal element in the extraction process. Thus, an evaluation of the collocation candidates allows us to assess how well the scores assigned by an association measure correspond to the collocativity of the respective word pairs. The application background defines an evaluation goal and thus helps to interpret the results. The relevant evaluation criterion is the usefulness of each association measure for the extraction of collocations. A quantitative measure (the **precision**) is given by the proportion of true positives in an n -best list of pre-defined size (see Section 5). Of course, the evaluation results are also highly relevant for applications, where they help select the most appropriate association measure for a given task (mainly the evaluated collocation extraction task itself, but also for other applications that use automatically extracted collocation candidates as a knowledge source).

In this thesis, I adhere to the “symmetric” view of collocations as opaque units that are largely independent from their component words. The goal of this approach, which is prevalent in computational linguistics, is to obtain a high proportion of true positives in n -best lists selected from all candidate pairs. An alternative is the “directional” view, which starts from a given *keyword* (also called the *base*, usually a high-frequency noun or verb) and aims to identify its *collocates*. This approach is natural when collocations are formalised in terms of lexical functions (Mel’čuk 2003; Kahane and Polguère 2001), and it is widely used in British computational lexicography (Sinclair 1991). The goal is usually to identify those collocates which are the most characteristic for the keyword (the collocation definition has to be chosen accordingly, but is often left implicit). Since the search space is reduced to candidates that contain the keyword as one of their components, the extraction task is simplified considerably. On the other hand, the evaluation of “directional” methods is more complicated and not as clear-cut. So far, published experiments have been limited to impressionistic case studies for a small number of keywords (e.g. Church *et al.* 1991; Sinclair 1991; Stubbs 1995).

1.3 Motivation and goals

1.3.1 The state of the art

Section 1.2 has demonstrated the importance of cooccurrence data and statistical associations for various applications in natural-language processing, and for collocation extraction in particular. The cornerstone of all these applications is the statistical analysis with association measures, and the quality of the results depends crucially on the felicitous choice of a measure. As early as 1964, Vincent Giuliano reflected after the *Symposium on Statistical Association Methods For Mechanized Documentation*:

[First,] it soon becomes evident [to the reader] that at least a dozen somewhat different procedures and formulae for association are suggested [in the book]. One suspects that each has its own possible merits and disadvantages, but the line between the profound and the trivial often appears blurred. One thing which is badly needed is a better understanding of the boundary conditions under which the various techniques are applicable and the expected gains to be achieved through using one or the other of them. This advance would primarily be one in theory, not in abstract statistical theory but in a problem-oriented branch of statistical theory. (Giuliano 1965b, 259)

Giuliano also emphasises the need for empirical evaluation:

[Secondly,] it is clear that carefully controlled experiments to evaluate the efficacy and usefulness of the statistical association techniques have not yet been undertaken except in a few isolated instances. . . . Nonetheless, it is my feeling that the time is now ripe to conduct carefully controlled experiments of an evaluative nature, . . . (Giuliano 1965b, 259)

Let us have a look at the current state of the art, almost exactly forty years after the Symposium was held in Washington, DC in March 1964. There are five major strands of research that might lead to a better understanding of association measures and their usefulness for collocation extraction tasks:

1. In mathematical statistics, there is a large body of work on measuring association in 2×2 contingency tables (see Yates (1984) for an overview), as well as the underlying random sample model, which is perhaps the most fundamental and widely used statistical model (Agresti 1990; Lehmann 1991). Problems of the randomness assumption on which this model rests have been discussed in the fields of lexical statistics, corpus linguistics, and natural-language processing (Baayen 1996; Katz 1996; Church 2000; Kilgarriff 2001).
2. In computational linguistics, various association measures were suggested, usually for the task of extracting collocation candidates (e.g. Stevens *et al.* 1965; Church and Hanks 1990; Dunning 1993; Pedersen 1996). Some papers describe complete extraction systems that employ various kinds of filtering in addition to the statistical analysis (e.g. Choueka 1988; Smadja 1993; Daille 1996). There is no reasonably comprehensive listing of the large number of available association measures. Manning and Schütze (1999, Ch. 5) describe the three most

widely-used measures, although the mathematical presentation is a little vague. More explicit equations with example calculations are given by Pearce (2002) and Weeber *et al.* (2000), while Schone and Jurafsky (2001) present concise equations for nine different measures.

3. A few attempts were made to evaluate different association measures (or entire collocation extraction systems) and compare their performance in a specific task (e.g. Breidt 1993; Daille 1994; Lezius 1999; Evert *et al.* 2000; Krenn 2000; Evert and Krenn 2001; Pearce 2002; Schone and Jurafsky 2001). In particular:
 - Breidt (1993) evaluates a combination of the MI and t-score measures for the extraction of German noun-verb collocations. This preliminary study is based on small corpus and a list of 16 verbs that are typically found in support-verb constructions (Breidt states that they are also commonly found with other types of noun-verb collocations). Rather than comparing different association measures, she varies conditions such as the corpus size and the strategies used for the extraction of cooccurrences.
 - Daille (1994) compares a total of 18 statistical measures for the extraction of French multi-word compound nouns, terminology in the telecommunications domain (see also Daille 1996).
 - Krenn (2000) compares four association measures (Dice, MI, log-likelihood, and average-MI) as well as cooccurrence frequency (as a non-statistical “baseline”) for the extraction of German PP-verb collocations (she also considers two other approaches that are not based on association scores and the corresponding rankings, so it is difficult to compare them directly with the association measures). In later publications, the comparison is extended to t-score and eventually to a wide range of measures (Evert and Krenn 2001).
 - Schone and Jurafsky (2001) compare 9 measures (plus several strategies for filtering and enriching the extraction results), with the goal of extracting multi-word headwords for dictionaries (such as *compact disk*).
4. Some articles concentrate on a small case study rather than a full-scale evaluation, often trying to gain an intuitive understanding of the differences between association measures rather than evaluate their performance on large amounts of data. For instance, Lapata *et al.* (1999) correlate the association scores of different measures with native-speaker judgements of plausibility. Stubbs (1995) essentially performs a lexicographic analysis of cooccurrences involving the lemma *cause* (both as noun and verb), with a perfunctory look at six other “semantically related” lemmata. A complementary approach is interested in the mathematical characteristics of association measures. Stubbs (1995) compares and manipulates equations to get a feel for their mathematical properties, while Tan *et al.* (2002) study the behaviour of a large number of measures under various “extreme” conditions. Some authors focus on the properties of a single association measure, e.g. Smadja *et al.* (1996, 9–12) with an intuitive description of the Dice measure and Dunning (1998) with a more mathematical look at log-likelihood.

5. Finally, there is an enormous volume of literature on lexical statistics (for an overview, see Baayen 2001) and especially on Zipf's law (Zipf 1949).¹⁷ This research provides a different angle on the random sample model behind association measures, but the results have hardly ever been applied to cooccurrence data so far (except for Ha *et al.* 2002).

In addition, a large amount of research on the linguistic properties of collocations and on their formal definition has been carried out in various areas of linguistics, lexicography, etc. (cf. the references in Section 1.1.1). Since I am primarily concerned with cooccurrences and their statistical association, such linguistic issues are not directly relevant for my thesis. In Chapter 5, which explores the connection between cooccurrences and collocations, the evaluation *results* may depend on the precise definition of collocations and their properties, but the evaluation *methods* do not.

1.3.2 Goals and Objectives

It is amazing to see how little progress has been made in the understanding of word cooccurrences, association measures and their relation to collocations in the forty years that have passed since the *Washington Symposium*. The reference work that Giuliano felt was so urgently needed – a compendium that lists, explains and compares the multitude of available association measures – has never seen the light of day. The closest approximation, Chapter 5 of Manning and Schütze (1999), is routinely cited in this context nowadays. However, the authors only find room¹⁸ to discuss three widely-used association measures (plus one that has seldom been employed, Pearson's X^2), and the evaluation and comparison of these measures is restricted to lists of twenty-odd “interesting bigrams”, as it is so often the case. Tellingly, they cannot point to a more comprehensive listing and discussion of association measures recommended for further reading (Manning and Schütze 1999, 187–189). My thesis aims to fill this gap and provide a reference for future research on the statistics of word cooccurrences. This includes the following goals:

1. An explicit description of the statistical model underlying association measures and the appropriate counting methods, both for relational and positional cooccurrences. This task includes a discussion of the adequacy of the model assumptions and some remarks on the problem of extraction noise (Chapter 2).
2. A comprehensive inventory of association measures, collecting the wide variety of available suggestions into groups of measures that have a similar theoretical background. For each measure, an explicit and readable equation is given (expressed in terms of expected and observed frequencies), and its mathematical derivation is discussed with key references (Chapter 3).¹⁹

¹⁷See <http://linkage.rockefeller.edu/wli/zipf/> for a collection of references on Zipf's law.

¹⁸They even afford some of their scant space to a superficial treatment of frequency comparisons between different corpora, which has little to do with cooccurrences or collocations save for the fact that some statistical measures can be applied to both tasks.

¹⁹For instance, the equation commonly used for the t-score measure involves some approximations which neither Church and Hanks (1990) nor Manning and Schütze (1999) explain clearly.

3. Reference implementations of all association measures in this inventory, with attention to details, robustness under boundary conditions and numerical accuracy. With these implementations being available, it should no longer be necessary to resort to a piece of code “sent to me by a friend” or “grabbed off the net” (see the software documentation in Appendix B.1).
4. A geometric model of association measures, which provides a framework for a better intuitive understanding and comparison of their properties. As a result of this analysis (and the theoretical background of the measures), two major groups of association measures emerge. The measures in each group have similar properties and are based on the same reasoning, so most of them can be represented by one or two “group prototypes” (Sections 3.3 and 3.4).
5. Lowest-frequency word pairs have always been a challenge for the statistical analysis of cooccurrences. I apply the tools of lexical statistics and Zipf’s law in order to show that such problems are caused by a fundamental quantisation effect for the skewed distributions that are characteristic of lexical frequency data. Therefore, it is impossible *in principle* to compute meaningful association scores for the lowest-frequency data, providing theoretical support for the application of frequency cutoff thresholds (Chapter 4).
6. Finally, I aim to provide tools and methods for the empirical evaluation of association measures in collocation extraction tasks. After describing the general precision/recall-based evaluation methodology and the graphical presentation, I address the significance of result differences. This issue is surrounded by much confusion about the choice of an appropriate significance test. Furthermore, evaluation based on random samples opens up new possibilities to perform experiments under a wider range of conditions. Such experiments are necessary both in order to find suitable association measures for specific applications and in order to improve our understanding of their properties. Implementations of all evaluation methods described in the thesis are freely available (Chapter 5 and the software documentation in Appendix B.2).

Some preliminary results from the research presented here have previously been published in the following papers: the discussion of extraction noise in Section 2.3.3 (Evert and Kermes 2003), the Zipf-Mandelbrot population model in Section 4.2 (Evert 2004b), the evaluation procedure and graphical presentation in Section 5.1 (Evert *et al.* 2000; Evert and Krenn 2001; Krenn and Evert 2001), the significance of result differences in Section 5.2 (Evert 2004a), and the random sample evaluation method in Section 5.3 (Evert and Krenn 2005).

1.3.3 Limitations

There are several aspects of word cooccurrences and methods for their statistical analysis that I do not consider in the present thesis, namely: cooccurrences of more than two words (often referred to as *n*-grams, for $n > 2$), possibly also including categorial elements (such as function words); **variable-length sequences** (where the number of cooccurring elements is not fixed in advance); **distributional** methods

(which consider e.g. the frequency distribution over all cooccurents of a given word or phrase); and **higher-order statistics** (which compare and cluster similar frequency distributions).

My reasons for limiting my work in this way – apart from plain time and space complexity – are the following: (i) Association measures for word pairs are easy to compute and can be applied to large numbers of pairs without too much overhead, which can be relevant when they are used as one module in a complex collocation extraction system. (ii) Association scores often form the basis of further statistical analyses, especially higher-order statistics and clustering techniques on cooccurrence vectors, as well as methods that operate on cooccurrence graphs. It is therefore important to have meaningful association score with well-understood properties from which to proceed. (iii) Association measures can be applied to individual word pairs without knowledge of the full range of cooccurring pairs (called a data set in Chapter 2). They are thus applicable in situations where it is practically impossible to obtain accurate frequency data for all the cooccurrences of a given word, e.g. when cooccurrence frequencies are obtained from internet search engines (cf. Keller and Lapata 2003).

In addition to all the gaps and open questions that need to be filled in (as listed in Section 1.3.2), I see a thorough understanding of the properties of association measures for word pairs as a necessary prerequisite for an extension to more complex kinds of cooccurrences, for which the mathematical theory offers considerably less help and guidance. Or, as D. R. Cox put it: “Nevertheless points remain for discussion, in particular so as to understand what to do in more complicated cases for which the single 2×2 table is a prototype” (Yates 1984, 451).

Chapter 2

Foundations

2.1 Corpus data

2.1.1 Frequency counts

This section explains how **cooccurrence data** are obtained from a source corpus.¹ The following discussion assumes a relational model of cooccurrences, whose advantages have been explained in Section 1.1.2. Some remarks on positional cooccurrence data (which are either based on a segmentation of the corpus into non-overlapping regions or some measure of the distance between words) can be found in Section 2.4, showing how the counting methods have to be modified so that the interpretation in terms of a random sample model (as defined in Section 2.2) is still possible.

The term *word* is used both for certain syntactic units in a text (“running words”) and for lexical items as listed in a dictionary (“headwords”). For the purpose of obtaining frequency counts, it is essential to make a clear distinction between these two aspects: lexical items are called **types**, while their instances in a text are referred to as **tokens**. The same distinction between types and tokens has to be made when counting other entities such as cooccurrences or syntactic constructions.

The general formal model for frequency counts is based on a pre-determined set C of **types**, which is often defined in a very general manner (e.g. as the set $C = \Sigma^*$ of all strings over some alphabet Σ). Variables for types are written as lowercase letters $u, v \in C$. By some means, a set T of **tokens** is identified in the source corpus. In order to simplify notation, I assume that T is well-ordered, so that we can write $T = \{t_1, t_2, \dots, t_N\}$. The precise arrangement is not important, though, and need not correspond to a sequential ordering of the tokens in the source corpus. N is referred to as the **sample size** (for reasons that will become clear in Section 2.2). Each token t is labelled with a type $\phi(t) = u \in C$. The function $\phi : T \rightarrow C$ is called a **label mapping**. I use the notation $U_i := \phi(t_i)$ for the label of the i -th token, so that the data extracted from the corpus can be represented by the sequence (U_1, U_2, \dots, U_N) . Normally, ϕ is not surjective and only the **observed types** $\phi(T) \subseteq C$ are considered in the statistical analysis. The **corpus frequency** $f(u)$ of a type $u \in C$ is given by the

¹I use the term **corpus** in the sense it often has in natural-language processing, i.e. as any collection of machine-readable texts, not as a clean and representative sample from a well-defined frame of reference (cf. McEnery and Wilson 2001, 78f).

number of tokens labelled with u (called the **instances** of u). Formally,

$$f(u) := |\phi^{-1}(u)| = |\{i \mid U_i = u\}|. \quad (2.1)$$

When this general model is applied to word frequency counts, a token t usually corresponds to a contiguous sequence of characters, which may or may not contain blanks. In some cases, however, a token may also represent a non-contiguous sequence (e.g. a German particle verb with separated particle) or a linguistic interpretation of the text without reference to surface forms (e.g. a non-terminal node in a syntax tree that is not overtly realised). Of course, the way in which tokens are identified in the source corpus depends on the precise definition of what constitutes a *word* as a syntactic unit.

As an example, consider the approach to corpus frequency data described by McEnery and Wilson (2001, 82), which is widely used in the field of corpus linguistics. In this approach, tokens are contiguous, non-overlapping sequences of characters in a text corpus, and types are defined as equivalence classes of tokens. An equivalence class may collect all tokens that represent exactly the same sequence of characters (also called a *word form type*), or collect all word forms that belong to an inflectional paradigm (called a *lemma type*). My formalisation of the counting process is more general than this process, and compatible with it. A translation can be made in two ways: (i) Let C be the infinite set of all possible character sequences ($C = \Sigma^*$) and construct the type mapping ϕ according to the definition of equivalence; or (ii) identify C with the set of equivalence classes *a posteriori*. Then the type mapping is given by the membership relation between tokens and equivalence classes.

When applied to cooccurrence data, each token t represents a pair of cooccurring word tokens r and s , i.e. $t = (r, s)$. Hence, t is called a **pair token**, and $T_p = \{t_1, \dots, t_N\}$ is the set of all pair tokens in the source corpus. Formally, r and s may belong to different sets of word tokens, $r \in T_1$ and $s \in T_2$. In practice, r and s usually belong to the same set T of word tokens, but they will often be restricted to different subsets $T_1, T_2 \subseteq T$ (e.g. the adjectives T_1 and nouns T_2 in the corpus). T_p is a subset of all possible pairs of word tokens: $T_p \subseteq T_1 \times T_2$. In the relational model, each pair token t corresponds to an instance of a particular structural relation in the source corpus, represented by the two word tokens that are its arguments. Consider the example sentence in Figure 2.1,² showing cooccurrences between nouns and modifying adjectives. This sample contains two pair tokens, $T_p = \{(r_6, r_{10}), (r_9, r_{10})\}$. All word tokens are taken from the same set $T = \{r_1, \dots, r_{11}\}$, but the first components of the pairs are restricted to adjective tokens ($T_1 = \{r_6, r_9\}$) and the second components to noun tokens ($T_2 = \{r_2, r_{10}\}$).

Each pair token $t = (r, s)$ is labelled with the types of its two **components** r and s . Therefore, the set C_p of possible **pair types** is the Cartesian product of the two sets of word types, $C_p = C_1 \times C_2$, and each pair type $w = (u, v)$ consists of the components $u \in C_1$ and $v \in C_2$. The label mapping $\phi : T_p \rightarrow C_p$ is given by the Cartesian product of the corresponding word label mappings $\phi_1 : T_1 \rightarrow C_1$ and $\phi_2 : T_2 \rightarrow C_2$, so that $\phi(t) = (\phi_1(r), \phi_2(s)) \in C_p$. As above, I use the notation $W_i = (U_i, V_i) := \phi(t_i)$ for the labels of the i -th pair token and its components. In the example of Figure 2.1, we

²This example has been adapted from the novel *Dombey and Son* by Charles Dickens, Chapter 62. Parts of speech are indicated by tags from the Penn Treebank tagset. See <http://www.ims.uni-stuttgart.de/projekte/CQPDemos/cqpdemo.html> for more information.

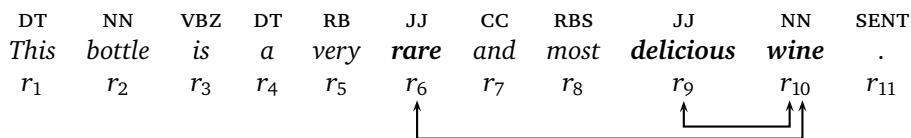


Figure 2.1: Example of adjective-noun cooccurrences. The arrows indicate structural relations between a prenominal adjective and the noun that it modifies, corresponding to pair tokens in the formal model.

have $N = 2$, $W_1 = (\textit{rare}, \textit{wine})$ and $W_2 = (\textit{delicious}, \textit{wine})$.³ Note how a word token (r_{10}) may belong to more than one pair token, or to none at all (r_2). In the following section we will see that, although there is only one instance of the type *wine* in this corpus, it has a frequency of 2 as a component of adjective-noun pairs (with instances t_1 and t_2).⁴ The sequence (W_1, \dots, W_N) of pair labels forms the **base data** (or base cooccurrence data) extracted from the source corpus. It provides the basis for the statistical model in Section 2.2, while the actual set T_p of pair tokens plays no role in the analysis. The **pair frequency** or **cooccurrence frequency** $f(w) = f(u, v)$ of a pair type $w = (u, v)$ is the number of tokens labelled w (i.e. the instances of w). Formally, we have

$$f(w) := |\phi^{-1}(w)| = |\{i \mid W_i = w\}| = |\{i \mid U_i = u \wedge V_i = v\}|, \quad (2.2)$$

with $\phi^{-1}(w) = \{t \in T_p \mid \phi(T_p) = w\}$. Recall that only the observed types $\phi(T_p) \subseteq C_p$ are usually considered, i.e. all pair types with zero frequency are discarded.

2.1.2 Contingency tables and frequency signatures

For each pair type $w = (u, v) \in C_p$, not only the cooccurrence frequency $f(w)$ is of interest, but also the cooccurrences of u and v with other words. This frequency information is usually collected in a **contingency table**, representing a four-way classification of the base data according to the components of the labels (i.e. whether $U_i = u$ or not, and whether $V_i = v$ or not). This classification yields the four cell counts

$$\begin{aligned} O_{11} &:= |\{i \mid U_i = u \wedge V_i = v\}| & O_{12} &:= |\{i \mid U_i = u \wedge V_i \neq v\}| \\ O_{21} &:= |\{i \mid U_i \neq u \wedge V_i = v\}| & O_{22} &:= |\{i \mid U_i \neq u \wedge V_i \neq v\}| \end{aligned} \quad (2.3)$$

³In this example, there is no difference between word form types and lemma types. Most applications will use lemma types (because the larger cooccurrence frequencies obtained by pooling morphological variants of the same lemma pair translate into more significant statistical results), provided that the necessary technology is available.

⁴It may seem counter-intuitive to assign a frequency count greater than one to a single word token, but keep in mind that the relational model of cooccurrences is based on instances of structural relations, not on instances of words. This strategy may lead to inflated frequency counts for examples such as *a beautiful, beautiful, beautiful speech* (found in the *British National Corpus*). For optimal results, the repetition of the adjective should be identified as a rhetoric device during cooccurrence extraction (or syntactic pre-processing) and be replaced by a single adjective-noun relation (perhaps with an annotation indicating the rhetoric effect).

	$V = v$	$V \neq v$
$U = u$	O_{11}	O_{12}
$U \neq u$	O_{21}	O_{22}

$$O_{11} + O_{12} + O_{21} + O_{22} = N$$

Figure 2.2: Contingency table of observed frequencies

which are usually presented in the form of a 2×2 table as shown in Figure 2.2. A more compact notation is the quadruple $(O_{11}, O_{12}, O_{21}, O_{22}) =: \vec{O}$. The cooccurrence frequency information can also be represented by the **pair frequency** $f(w) = f(u, v)$ and the **component frequencies** $f_1(u) := |\{i \mid U_i = u\}|$ and $f_2(v) := |\{i \mid V_i = v\}|$ (also called the **joint** and **marginal frequencies**). Note that the marginal frequencies are not based on the total number of instances of u or v in the corpus, but rather on the number of pair tokens with u as first label or v as second label, respectively. This method for obtaining contingency tables may be easier to implement in a computer program. I refer to the quadruple

$$(f(u, v), f_1(u), f_2(v), N) =: (f, f_1, f_2, N)_{(u, v)},$$

as the **frequency signature** of a pair type (u, v) . The subscript (u, v) is usually omitted, writing (f, f_1, f_2, N) unless there is a need to distinguish between the signatures of different pair types. The information contained in the contingency table \vec{O} is fully equivalent to that in the frequency signature. Conversion rules are given by (2.4).

$$\begin{aligned}
 f &= O_{11} & O_{11} &= f \\
 f_1 &= O_{11} + O_{12} & O_{12} &= f_1 - f \\
 f_2 &= O_{11} + O_{21} & O_{21} &= f_2 - f \\
 N &= \sum_{ij} O_{ij} & O_{22} &= N - f_1 - f_2 + f
 \end{aligned} \tag{2.4}$$

Here and in the following, \sum_{ij} is used as a shorthand notation for the summation $\sum_{i=1}^2 \sum_{j=1}^2$ over the rows and columns of a contingency table. As an example, consider adjacent English adjectives and nouns extracted from the *British National Corpus* (BNC) and labelled with lemma types (see Section 2.1.3 for a more detailed description of the extraction process). For the pair type $w = (\textit{black}, \textit{box})$, we obtain the contingency table shown in Figure 2.3. There are 123 instances of w (corresponding to the surface strings *black box* and *black boxes*), 13 168 cooccurrences of *black* with a different noun than *box*, and 1 810 cooccurrences of *box* with a different adjective than *black*. The corresponding frequency signature is $(f, f_1, f_2, N) = (123, 13\,291, 1\,933, 4\,966\,984)$. Note that the marginal frequency of *box*, $f_2(v) = 1\,933$, is much smaller than the total number of instances of the noun *box* in the BNC (which may be written as $f(v) = 7\,970$).

	$V = \textit{box}$	$V \neq \textit{box}$
$U = \textit{black}$	123	13 168
$U \neq \textit{black}$	1 810	4 951 883

Figure 2.3: Contingency table for the adjective-noun pair type (*black*, *box*) in the *British National Corpus*.

	$V = v$	$V \neq v$		
$U = u$	O_{11}	+	O_{12}	= R_1
	+		+	
$U \neq u$	O_{21}	+	O_{22}	= R_2
	= C_1		= C_2	

Figure 2.4: Contingency table with row and column sums.

When cooccurrence frequencies are given in the form of a contingency table, the row sums $R_1 = O_{11} + O_{12}$ and $R_2 = O_{21} + O_{22}$ as well as the column sums $C_1 = O_{11} + O_{21}$ and $C_2 = O_{12} + O_{22}$ are often included since they play an important role in the statistical analysis (cf. Figure 2.4).

From the transformation rules above it is obvious that $f_1 = R_1$ and $f_2 = C_1$. The row and column sums, and hence also the component frequencies f_1 and f_2 , are often referred to as **marginal frequencies**, being written in the margins of the table. A concrete example for the pair (*black*, *box*) is shown in Figure 2.5. I use the term **data set** for the set of pair types extracted from a source corpus together with their frequency signatures or contingency tables. A data set is the result of performing frequency counts on the base data.

2.1.3 Examples

As concrete examples, consider the following English and German data sets which are referred to in various places throughout the thesis. The data sets are based on three different source corpora:

1. For English, the *British National Corpus* (BNC) was used, a balanced sample of written and (transcribed) spoken English running up to a total of ca. 100 million words of text (Aston and Burnard 1998). The version of the corpus used here

	$V = v$		$V \neq v$	
$U = u$	123	+	13 168	= 13 291
	+		+	
$U \neq u$	1 810	+	4 951 883	= 4 953 693
	= 1 933		= 4 956 051	$N = 4 966 984$

Figure 2.5: Contingency table for (*black, box*) with row and column sums.

is annotated with part-of-speech tags and lemma types.

2. Most German examples are based on the *Frankfurter Rundschau* (FR) corpus, a newspaper corpus comprising ca. 40 million words of text from the years 1992 and 1993.⁵ The corpus was part-of-speech tagged with the TreeTagger (Schmid 1994) and annotated with lemma types as well as morpho-syntactic information from the IMSLex morphology (Lezius *et al.* 2000).
3. In order to study very large amounts of data, an extension of the FR corpus with material from various other newspapers (all from the 1990s) was used. With a total size of ca. 225 million words of text, this corpus is referred to as the *Huge German Corpus* (HGC).

AN-BNC: One of the simplest examples of relational cooccurrences are prenominal adjectives in English, seen as a cooccurrence of the adjective and the modified noun. It is fairly easy to identify these cooccurrences in a part-of-speech tagged corpus when the adjective and the noun are directly adjacent. The targeted structural relation can be defined as a combination of (syntactic) modification and (graphemic) adjacency, which does make sense e.g. when the cooccurrence data are used to extract multi-word compound nouns or dictionary headwords (such as the example *black box* from the previous section). When the relation of interest is adjective-noun modification (without the additional constraint), the extraction will miss a considerable number of cooccurrences, trading recall for a high degree of precision. Some inaccuracies in the base data (referred to as *noise*) always have to be expected when automatic methods are used for extraction (see Section 2.3.3 for a brief discussion).

In this way, $N = 4\,250\,139$ adjective-noun pair tokens were found in the *British National Corpus* as base data. The frequency analysis, based on lemma types, resulted in a data set of $V = 1\,205\,637$ pair types with cooccurrence frequencies ranging from $f = 1$ (for 813 498 types) to $f = 8\,847$ (for the pair *prime minister*).

⁵The FR corpus is part of the ECI Multilingual Corpus 1 distributed by ELSNET. ECI stands for European Corpus Initiative, and ELSNET for European Network in Language And Speech. See <http://www.elsnet.org/resources/ecicorpus.html> for details.

AN-FR: In a similar way, German adjective-noun pairs were extracted from the *Frankfurter Rundschau* corpus. Since most compound nouns are written as single graphemic words in German, the adjacency requirement did not seem justified. Instead, simple part-of-speech patterns were applied that allow a number of intervening words between the adjective and the noun, excluding certain parts of speech (see Evert and Kermes 2003). In the English example of Figure 2.1, a pattern that excludes nouns and verbs between the cooccurring adjective (JJ) and noun (NN), but allows conjunctions (CC), adverbs (RB) and other adjectives, would correctly identify both pair tokens.

The AN-FR base data consist of $N = 1\,618\,799$ pair tokens, resulting in a data set of $V = 605\,030$ pair types. Here, frequencies range from $f = 1$ (for 427 946 pair types) to $f = 7\,430$ (for the pair *vergangenes Jahr*, ‘last year’).

AN-HGC: This data set uses the same method to extract adjective-noun cooccurrences from the full HGC corpus. For technical reasons, exactly 12 million pair tokens were used, resulting in a data set of $V = 3\,621\,708$ pair types.

VN-BNC: For this data set, verb-noun pairs (where the noun is the direct object of the verb) were extracted from the *British National Corpus* (BNC). In contrast to the adjective-noun data, simple adjacency would identify only a limited subset of the cooccurrences (and would not even find well-known idioms such as *kick the bucket*). Therefore, a more complex part-of-speech pattern was used for the extraction, which can be described informally as

[*verb particle?*] *det?* *adjective** [*noun*],

i.e. a verb, optionally followed by a particle, then followed by a simple noun phrase that may contain an optional determiner and an arbitrary number of adjectives in addition to the head noun. The square brackets indicate which parts of the pattern were extracted as the components of the pair tokens (namely, *verb(+particle)* as first component and *noun* as second component).

In this way, $N = 1\,345\,935$ pair tokens were extracted from the BNC, resulting in a data set of $V = 496\,249$ lemma types. Of these, 5 365 satisfied the frequency threshold condition $f \geq 25$ that was applied.

PNV-FR: A more complex example is the extraction of preposition-noun-verb (PNV) combinations from the *Frankfurter Rundschau* corpus, as used by Krenn (2000) for the identification of PP-verb collocations. In order to fit the PNV triples into the framework used here, they are interpreted as (PN,V) pairs, where a combination of preposition (functional head) and noun (lexical head) represents the PP (this combination is thus treated as a “complex word”). The structural relation between PP and verb can be defined in terms of a phrase structure analysis, where the PP must be attached to some projection of the verb. The relation may be refined to allow only PPs that function as P-object rather than adjunct, when such a distinction is made in the theory.

Ideally, a full syntactic analysis of the source corpus would allow us to extract the cooccurrences directly from parse trees. Since a parser with the required

coverage was not available, a partial syntactic analysis was performed with the YAC chunk parser (Kermes 2003). In addition to noun phrases (NP) and prepositional phrases (PP), YAC identifies verbal complexes (VC) and subordinate clauses in the text. All chunks are annotated with the corresponding head lemma. PPs are annotated both with the preposition and the nominal head. The head lemma annotations of VCs are particularly useful because they recombine separated particle verbs. Based on these annotations, all possible combinations of a VC and a PP (labelled with their respective head lemma annotations) within the same main or subordinate clause were extracted as cooccurrences.

Figure 2.6 shows the partial syntactic analysis of the sentence *Ein mit Kaffee beladenes Schiff sticht bei gutem Wetter in See*. ‘A ship loaded with coffee beans puts to sea in fine weather.’ From this tree structure, the pair tokens (*bei Wetter, stechen*) and (*in See, stechen*) are extracted (because the corresponding PP nodes are attached to the same S node as the VC). The embedded PP *mit Kaffee* is ignored because it is not directly attached to the S node.

This extraction strategy resulted in $N = 5\,082\,148$ pair tokens and a data set of $V = 3\,346\,843$ pair types. Because the structural relation – and especially the extraction technique – is much less constrained than in the adjective-noun examples, the proportion of types with $f = 1$ is particularly high (2 711 356 types).

PNV-SLICES: For an empirical validation experiment in Section 5.2.3, the *Frankfurter Rundschau* corpus was divided into 80 contiguous, non-overlapping parts (called “slices”), each one containing approx. 500 000 running words. PP-verb cooccurrences were extracted from each slice as described above for the PNV-FR data set, with a frequency threshold of $f \geq 3$. This procedure resulted in 80 data sets containing between 536 and 867 pair types (with an average of 658).

PNV-HGC: An extension of PNV-FR to the HGC corpus yielded more than 32 million pair tokens. For technical reasons, exactly 32 million tokens were used, resulting in a data set of 18 529 301 pair types.

2.1.4 Filtering cooccurrence data

Cooccurrence data are often **filtered**, removing certain “undesirable” pair tokens or types. Some filters are outlined shortly for the examples in Section 2.1.3, and presented in more detailed at the end of the section. For instance, adjective-noun pairs may be suppressed if the adjective is deverbal and subcategorises a PP (which is then interpreted as a noun-verb relation “in disguise”, possibly also including the PP as an argument). Note how the application of such a filter requires additional information to be annotated with the base data (regarding the presence of a PP). Filtering is particularly common in NLP applications (tools for collocation extraction, cf. Section 1.2.2).

There are two different kinds of filtering: **token filtering**, where pair tokens are removed *before* obtaining frequency counts; and **type filtering**, where pair types are removed *after* obtaining the frequency counts.

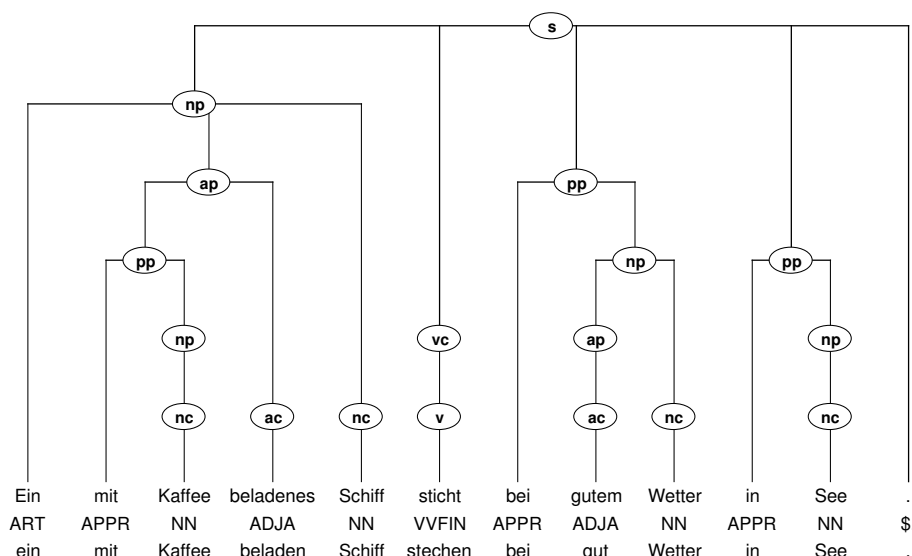


Figure 2.6: Example for the extraction of PP-verb cooccurrences from a partial syntactic analysis produced by the YAC chunk parser.

Token filtering affects the sample size and (more importantly) the frequency signatures of pair types. Token filtering can be understood as a set of additional rules for the identification of pair tokens and has no further implications for the model. However, there should be some (theoretical) justification of why the deleted pair tokens are not considered instances of the targeted structural relation, which may involve narrowing down that relation. It is *not* sufficient to note that e.g. certain general adjectives in adjective-noun pairs “usually produce uninteresting results”: they are still instances of adjectival modification of nouns and have to be counted as such. (It is perfectly valid, though, to remove such “uninteresting” pairs by means of type filtering.)

Type filtering deletes certain pair types from a data set without affecting the frequency signatures of the remaining pair types, or divides a data set into two or more subsets, which are then processed separately. Type filtering is often done in an attempt to improve the statistical analysis of cooccurrences by teasing apart different frequency distributions that are overlaid in the original data set. Sichel (1975, 547) uses the same argument for word frequency distributions. It can also be understood as a pragmatic means of improving the performance of a collocation extraction tool (an example are the above-mentioned general adjectives in adjective-noun pairs, which are seldom of interest to lexicographers).

Example 1: In the case of prenominal adjectives, it might make sense to ignore noun phrases whose head is a proper noun, the underlying structural relation being defined as “adjectival modification of *common* nouns”. Technically, filters of this type are usually implicit in the syntactic analysis and the identification of cooccurrences rather than being explicitly applied to a the base data. The part-of-speech patterns used for the construction of the AN-FR data set include such a constraint.

Example 2: As mentioned above, for German prenominal adjectives it can be useful

to eliminate deverbal adjectives (i.e. present or past participles used as adjectives, which often take NP or PP complements), in order to avoid the problematic distinction between lexicalised adjectives and verb participles. It is not immediately clear whether such a filter should be applied to pair tokens or to types. Possible arguments are: (i) In favour of the token filter, that deverbal adjectives often express verb-subject (present participles) and verb-object (past participles) relations rather than adjectival modification. It may be even more appropriate to filter out only those pair tokens where the adjective is in fact accompanied by a PP or NP. (ii) In favour of the type filter, that the syntactic construction is identical for both types of adjectives, but that deverbal ones combine with a different set of nouns (those which can be subjects or objects of the corresponding verbs), so that we have two different, overlaid frequency distributions. Alternatively, deverbal adjectives may simply be seen as useless or problematic in a lexicographic application, and hence be deleted.

Example 3: An excellent example of type filtering is provided by PP-verb pairs. In her German data, Krenn (2000) distinguishes between figurative expressions and support-verb constructions (SVC, as Bußmann (1990)'s translation of the German term *Funktionsverbgefüge*). Breidt (1993) gives a list of 16 verbs that are often used as support verbs, namely: *bleiben, bringen, erfahren, finden, geben, gehen, gelangen, geraten, halten, kommen, nehmen, setzen, stehen, stellen, treten, ziehen* (Krenn 2000, 120). One may well expect these SVC verbs to have special distributional properties in PP+verb pairs, so it makes sense to split the data set into two parts that are analysed separately.

2.2 A statistical model of cooccurrences

Raw cooccurrence data (i.e. the observed frequency signatures or contingency tables) provide some evidence for recurrent word combinations (mainly in the form of cooccurrence frequencies). However, the plain numbers are difficult to interpret,⁶ and any conclusions drawn from them are only valid for the one particular corpus from which the data were extracted. When extraction involves automatic linguistic pre-processing or analysis, the observed frequencies will also be affected by the errors that these programs make (typical error rates range from some 2% for a well-trained part-of-speech tagger to almost 50% for broad-coverage syntactic analysis).

Statistical analysis is applied in order to overcome these problems. This analysis has three main goals (or *tasks*): (i) to interpret the observed frequency data as an indicator of statistical association between words and quantify the strength of this association; (ii) to generalise results beyond the particular source corpus from which the cooccurrence data were obtained; and (iii) to filter out noise introduced by automatic pre-processing and extraction. All three tasks are related in their underlying logic, which assumes that the object of interest – statistical association between the components of a pair type – is a hidden quantity that is reflected in the contingency

⁶Should one just look at the cooccurrence frequency f of a pair type? Or is the ratio between joint frequency and marginal frequency more meaningful? If so, which of the two ratios, f/f_1 or f/f_2 ? Or should they be combined in some way, be it arithmetic mean, geometric mean, harmonic mean, minimum, or maximum?

table of observed frequencies. However, those frequencies are also subject to other uncontrollable influences. Task (i) above consists in the identification and precise definition of the hidden quantity, while tasks (ii) and (iii) address the relation between this quantity and the observed data.

In the terminology of a **statistical model**, the hidden quantity is a **parameter** of the model (for a given pair type) and the contingency table extracted from the source corpus for this pair type is an **observation**; the link between parameters and observations is provided by the **sampling distribution**, which specifies the probability of a particular observation or group of observations (an **outcome** of the sample) given some hypothesis about the parameter values. The goal of the statistical analysis is to make inferences about the model parameters from the observed data, based on the assumed sampling distribution.

The core of any statistical model lies in the definition (or rather postulation) of a sampling distribution. This choice determines which external influences are taken into account by the model (cf. tasks (ii) and (iii) above) and how accurately they are represented. By the nature of statistical reasoning, the sampling distribution must always involve some element of **randomness**. For the purpose of task (ii), the sampling distribution should predict to what extent the contingency table of a given pair type varies from one source corpus to another. Randomness here lies in the arbitrary choice of a particular source corpus from a **set of alternatives** (often hypothetical ones). Task (iii) can also be accommodated in this framework, when the set of alternatives includes versions of the same corpus with different pre-processing and extraction errors (among them, presumably, a “perfect” error-free version). As Sinclair (1965) puts it: “Any stretch of language has meaning only as a sample of an enormously large body of text; it represents the results of a complicated selection process, and each selection has meaning by virtue of all the other selections which might have been made, but have been rejected” (cited from Stubbs (1995)).

Obviously, the shape and variability of the sampling distribution depends on the range of modalities, text types, genres, subject matters, etc. represented in the set of alternatives from which the source corpus has been selected, as well as the type and amount of noise that is taken into account. Just as obviously, the influence of such linguistic factors cannot be predicted by purely statistical means.⁷ Instead, it would be necessary to formulate an explicit model of linguistic variation. Therefore, we need to choose a more regular (imaginary) set of alternatives, for which the sampling distribution can be predicted. This approach will almost always provide a lower boundary on the true sampling variation (because the “regular” alternatives are less diverse than the “real” set, except when this real set is very restricted). Consequently, it represents the minimum amount of uncertainty that will be present in any inferences about the parameters of the “real” model. The regular set of alternatives is constructed in such a way that we can interpret (the pair tokens extracted from) a corpus as a **random sample** of (the pair tokens extracted from) a large hypothetical body of language data (the **population**). The model parameters describe properties of the full body of language data, and the random sample model allows us to make inferences about these properties from the observed data.

The random sample model can now be seen in two ways. (i) As a baseline for

⁷The same is true for *systematic* pre-processing and extraction errors, e.g. the proper name *New York* may be consistently identified as an adjective-noun sequence by a stochastic part-of-speech tagger.

the sampling variation that has to be expected and that needs to be corrected for. Any results that can be explained by this sampling variation alone may just be flukes. (ii) As a generalisation from the observed data to the properties of a (hypothetical) sublanguage. This sublanguage must be defined in such a way that the source corpus that was actually used can realistically be seen as a random sample from it. Taking the example of a newspaper corpus such as the *Frankfurter Rundschau* (containing one volume of a single newspaper), the hypothetical collection would contain more articles written by the *same* journalists during the *same* time on the *same* subjects. It might also be understood to comprise multiple volumes from the same newspaper, although the assumption of a random sample already becomes questionable in this case (because entirely different topics may be covered in different volumes – just think of the differences between articles written before and after September 2001).

2.2.1 Cooccurrence data as a random sample

The base data extracted from the corpus are interpreted as a random sample from an infinite population (so that sampling with replacement can be assumed). This **population** is characterised by a set of pair types w with **cooccurrence probabilities** π_w .⁸ The set of population pair types is usually equated with C_p . An “impossible” pair type w (i.e. a pair type that can never appear in the base data because it is ruled out by syntactic or semantic constraints, or because the morphology component used for lemmatisation rejects any word that is not listed in its lexicon) will have $\pi_w = 0$. It is important to make a clear distinction between such impossible pairs ($\pi_w = 0$) and unseen pairs that are not found in the sample ($f_w = 0$). Consequently, the population cannot be restricted to the set of observed types $\phi(T)$.

The random selection of a pair type from the population, according to the cooccurrence probabilities, is described by random variables U and V , which stand for the components of the selected type. For any pair type $w = (u, v)$ we have

$$\Pr(U = u \wedge V = v) = \pi_w .$$

The probabilities $\Pr(U = u) =: \pi_{1,u}$ and $\Pr(V = v) =: \pi_{2,v}$ are called the **marginal probabilities** of the component types u and v , and can be obtained by summation over pair types with the same first or second component:

$$\pi_{1,u} = \sum_{v' \in C_2} \pi_{(u,v')} \quad \text{and} \quad \pi_{2,v} = \sum_{u' \in C_1} \pi_{(u',v)} .$$

The population is fully determined by the **probability parameters** π_w . However, since the marginal probabilities are also important for the statistical analysis, the

⁸This random sample model can also be interpreted as a model of text production, where for each instance of a particular relation that is generated, the heads of its arguments are chosen randomly from the population (represented by the labels assigned to the corresponding pair token). In this view, pair tokens are randomly (and independently) generated: whenever a speaker produces an instance of the targeted relation, he or she randomly selects a pair type from the population. The chance of selecting type w is given by its population probability π_w . This model leads to the same distributions etc. as random sampling. It fits quite well into the framework of generative syntax and probabilistic context-free grammars (where π_w can be interpreted as a measure of selectional preference). It can also be understood as a (very simple) model of a speaker in psycholinguistic studies.

population probabilities (or **population parameters**) of a given pair $w = (u, v) \in C_p$ are usually taken to be the triple

$$(\pi_w, \pi_{1,u}, \pi_{2,v}) =: (\pi, \pi_1, \pi_2)_w$$

and the subscript is omitted unless there is danger of confusion. The population probabilities can also be expressed in the form of a contingency table:

$$\begin{aligned} \tau_{11} &:= \Pr(U = u \wedge V = v) & \tau_{12} &:= \Pr(U = u \wedge V \neq v) \\ &= \pi_{(u,v)} & &= \sum_{v' \neq v} \pi_{(u,v')} \\ \tau_{21} &:= \Pr(U \neq u \wedge V = v) & \tau_{22} &:= \Pr(U \neq u \wedge V \neq v) \\ &= \sum_{u' \neq u} \pi_{(u',v)} & &= \sum_{u' \neq u} \sum_{v' \neq v} \pi_{(u',v')} \end{aligned}$$

with $\tau_{11} + \tau_{12} + \tau_{21} + \tau_{22} = 1$. The transformation rules are similar to those for frequency signatures:

$$\begin{aligned} \pi &= \tau_{11} & \tau_{11} &= \pi \\ \pi_1 &= \tau_{11} + \tau_{12} & \tau_{12} &= \pi_1 - \pi \\ \pi_2 &= \tau_{11} + \tau_{21} & \tau_{21} &= \pi_2 - \pi \\ & & \tau_{22} &= 1 - \pi_1 - \pi_2 + \pi \end{aligned}$$

A random sample of size N is described by independent pairs of random variables U_i and V_i ($i = 1, \dots, N$),⁹ where the distribution of U_i is identical to that of U and the distribution of V_i is identical to that of V . U and V can be seen as *prototypes* for the sample variables U_i and V_i . When we interpret the base data as a random sample from the population, the pair (U_i, V_i) of random variables describes the label of the i -th token in the sample (i.e. we assume that each pair of labels is chosen randomly from the population). I use the notation $W_i = (U_i, V_i)$ for the pair describing the i -th token, and $W = (U, V)$ for the pair of prototypes.

The sample frequencies of a pair type w can be computed from the random variables W_i , based on a classification of the pair tokens into four bins. As functions of random variables, they are themselves random variables X_{ij} , corresponding to the cells O_{ij} of the observed contingency table. For notational convenience, I define additional random variables for the row and column sums. Figure 2.7 shows the full contingency table of random variables.

Since the pair tokens extracted from the source corpus are assumed to be one particular random sample from the population, the observed contingency table $\vec{O} = (O_{11}, O_{12}, O_{21}, O_{22})$ is interpreted as a particular realisation of the random variables $\vec{X} = (X_{11}, X_{12}, X_{21}, X_{22})$. Statistical analyses of the observed data are based on the probability of this realisation and similar ones (i.e. the **sampling distribution** of the random variables $(X_{11}, X_{12}, X_{21}, X_{22})$) under certain assumptions about the population parameters. Formally, the random variables X_{ij} can be defined with the help of **indicator variables** for a given pair type $w = (u, v) \in C_p$:

$$Y_k := I_{[U_k=u]} = \begin{cases} 1 & U_k = u \\ 0 & U_k \neq u \end{cases} \quad \text{and} \quad Z_k := I_{[V_k=v]} = \begin{cases} 1 & V_k = v \\ 0 & V_k \neq v \end{cases}$$

⁹i.e. U_i is independent from any U_j or V_j with $j \neq i$, but U_i and V_i are (usually) not independent

	$V = v$	$V \neq v$	
$U = u$	X_{11}	X_{12}	$= X_{R1}$
$U \neq u$	X_{21}	X_{22}	$= X_{R2}$
	$= X_{C1}$	$= X_{C2}$	

Figure 2.7: Random variables representing the contingency table of a sample.

as well as

$$\begin{aligned}
 I_{11}^{(k)} &= \begin{cases} 1 & \text{if } U_k = u \text{ and } V_k = v \\ 0 & \text{otherwise} \end{cases} = Y_k \cdot Z_k \\
 I_{12}^{(k)} &= \begin{cases} 1 & \text{if } U_k = u \text{ and } V_k \neq v \\ 0 & \text{otherwise} \end{cases} = Y_k \cdot (1 - Z_k) \\
 I_{21}^{(k)} &= \begin{cases} 1 & \text{if } U_k \neq u \text{ and } V_k = v \\ 0 & \text{otherwise} \end{cases} = (1 - Y_k) \cdot Z_k \\
 I_{22}^{(k)} &= \begin{cases} 1 & \text{if } U_k \neq u \text{ and } V_k \neq v \\ 0 & \text{otherwise} \end{cases} = (1 - Y_k) \cdot (1 - Z_k)
 \end{aligned}$$

for $k \in \{1, \dots, N\}$. With these indicator variables,

$$X_{ij} = \sum_{k=1}^N I_{ij}^{(k)}$$

for $i, j \in \{1, 2\}$. The random variables X_{ij} are not independent because they must sum to the sample size: $\sum_{ij} X_{ij} = N$. Their joint distribution is a **multinomial distribution** with parameters $(\tau_{11}, \tau_{12}, \tau_{21}, \tau_{22})$. For any numbers $k_{11}, k_{12}, k_{21}, k_{22} \in \mathbb{N}_0$ with $\sum_{ij} k_{ij} = N$, we have

$$\Pr(\vec{X} = \vec{k} \mid N) = \frac{N!}{k_{11}! k_{12}! k_{21}! k_{22}!} \cdot (\tau_{11})^{k_{11}} \cdot (\tau_{12})^{k_{12}} \cdot (\tau_{21})^{k_{21}} \cdot (\tau_{22})^{k_{22}}. \quad (2.5)$$

Here and in the following, I use the shorthand notation $\vec{X} = (X_{11}, X_{12}, X_{21}, X_{22})$, and similarly for \vec{k}, \vec{O} , etc. In particular, the vector equality

$$\vec{X} = \vec{k} : \iff (X_{11}, X_{12}, X_{21}, X_{22}) = (k_{11}, k_{12}, k_{21}, k_{22})$$

stands for the condition

$$X_{11} = k_{11}, X_{12} = k_{12}, X_{21} = k_{21}, X_{22} = k_{22}.$$

Furthermore, I use the notation $\vec{k}|N$ for a set of values k_{ij} that are compatible with the condition $\sum_{ij} X_{ij} = N$ when inserted into the equality $\vec{X} = \vec{k}$ (i.e., they must satisfy $\sum_{ij} k_{ij} = N$). A similar notation will later be used for other conditioning equations. $\Pr(\vec{X} = \vec{k} | N)$ is written as a conditional probability in order to indicate that we are considering a sample of fixed size N (Section 2.2.2 motivates this notation). Each random variable X_{ij} has a **binomial distribution** by itself, i.e.

$$\Pr(X_{ij} = k | N) = \binom{N}{k} (\tau_{ij})^k (1 - \tau_{ij})^{N-k} \quad (2.6)$$

with $E[X_{ij}] = N\tau_{ij}$ (but remember that these distributions are *not* independent). For the row and column sums $X_{R1} = X_{11} + X_{12}$, $X_{R2} = X_{21} + X_{22}$, $X_{C1} = X_{11} + X_{21}$, and $X_{C2} = X_{12} + X_{22}$, we obtain similar binomial distributions with

$$\begin{aligned} E[X_{R1}] &= N(\tau_{11} + \tau_{12}) = N\pi_1, & E[X_{C1}] &= N(\tau_{11} + \tau_{21}) = N\pi_2, \\ E[X_{R2}] &= N(\tau_{21} + \tau_{22}) = N(1 - \pi_1), & E[X_{C2}] &= N(\tau_{12} + \tau_{22}) = N(1 - \pi_2). \end{aligned}$$

2.2.2 Independent Poisson sampling

In the previous section, we have considered random samples of a fixed size N . Speaking in the terms of the introductory explanation at the beginning of Section 2.2, the set of alternatives is restricted to corpora containing exactly N pair tokens. This is quite unrealistic: it would be unusual to sample a pre-determined number of pair tokens. If anything is fixed in advance, a source corpus of pre-determined size (measured by the number of running words) might be used. However, two different corpora of the same size will usually contain a different number of pair tokens. In a more realistic model, the sample size itself becomes a random variable N^* . Figure 2.8 illustrates the distribution of N^* for 100 000-word subsets of the *Frankfurter Rundschau* corpus. On average, about 4 300 pair tokens were extracted from each slice, but the individual sample sizes N^* range from 3 741 to 4 770 tokens. Since the unconstrained sampling distribution now depends on the unknown distribution of N^* , Eq. (2.5) is conditioned on the observed sample size N (i.e. on the constraint $N^* = N$). The resulting conditional probabilities depend only on the population parameters, regardless of the distribution of N^* .

The unconstrained model is only manageable when we assume a specific distribution for the sample size N^* . In the field of biometrics, where types often correspond to different animal species and tokens to specimens caught in a trap during a fixed amount of time, it is reasonable to assume a Poisson distribution for N^* , which stipulates that specimens are caught on average at a constant rate. The mean $E[N^*] = \nu$, which is also the single parameter that determines the shape of a Poisson distribution, is given by this rate multiplied by the time the trap is open. When translated to cooccurrence data, ν is the average “sampling rate” at which cooccurrences are encountered in text multiplied by the (pre-determined) size of the source corpus. In this model, which I call **independent Poisson sampling**,¹⁰ the random variables X_{ij}

¹⁰Agresti (1990, 37) also makes a distinction between multinomial and Poisson sampling.

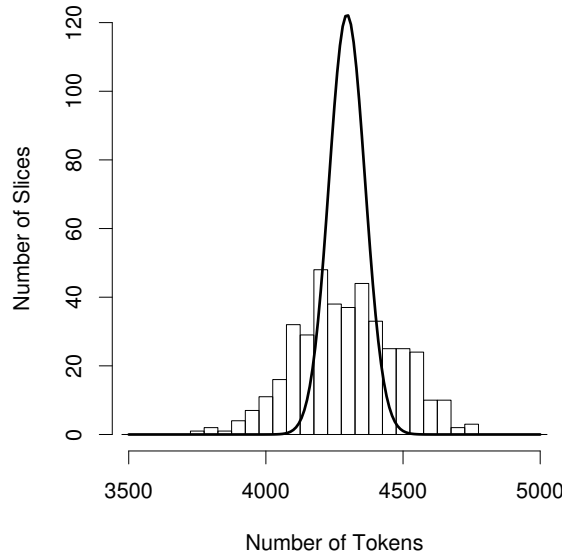


Figure 2.8: The variability of the sample size: Histogram for the number of adjective-noun pair tokens extracted from subsets of the *Frankfurter Rundschau* corpus, containing 100,000 running words each. The solid curve shows the distribution expected by the independent Poisson sampling model.

have independent Poisson distributions with mean $E[X_{ij}] = \nu\tau_{ij}$ and

$$\Pr(X_{ij} = k_{ij}) = e^{-\nu\tau_{ij}} \cdot \frac{(\nu\tau_{ij})^{k_{ij}}}{k_{ij}!}.$$

Their joint distribution is given by

$$\Pr(\vec{X} = \vec{k}) = e^{-\nu} \cdot \nu^n \cdot \frac{(\tau_{11})^{k_{11}}}{k_{11}!} \cdot \frac{(\tau_{12})^{k_{12}}}{k_{12}!} \cdot \frac{(\tau_{21})^{k_{21}}}{k_{21}!} \cdot \frac{(\tau_{22})^{k_{22}}}{k_{22}!} \quad (2.7)$$

with $n = \sum_{ij} k_{ij}$ unconstrained. The importance of this model lies in the fact that both the cell counts X_{ij} for a given pair type w and the cooccurrence frequencies f_{w_1}, f_{w_2} of different pair types (here interpreted as random variables) are independent. This simplifies complex statistical analyses, especially for modelling the distribution of word frequencies in Chapter 4. Fig. 2.8 shows that the Poisson distribution (indicated by the solid curve) underestimates the true variation of N^* : its sample standard deviation (≈ 185) is almost three times the standard deviation predicted by the Poisson model (65.5). Especially when N is very large, though, the relative variation of N^* (both the true variation and that predicted under the Poisson model) is comparatively small, and (2.7) can also be understood as a mathematically convenient approximation of the multinomial probabilities (2.5).

Even when we do not want to make any assumptions about the distribution of N^* , the independent Poisson model can be a useful analytical device, which was discovered by R.A. Fisher (Fisher 1922, 89). Note that (2.7) includes the additional unknown parameter ν , which is referred to as a **nuisance parameter** because it does not provide any information about the associations that are of interest to us. We can

Population	$V = \nu$	$V \neq \nu$
$U = u$	τ_{11}	τ_{12}
$U \neq u$	τ_{21}	τ_{22}

Sample	$V = \nu$	$V \neq \nu$
$U = u$	O_{11}	O_{12}
$U \neq u$	O_{21}	O_{22}

$$\tau_{11} + \tau_{12} + \tau_{21} + \tau_{22} = 1$$

$$O_{11} + O_{12} + O_{21} + O_{22} = N$$

Figure 2.9: Comparison of population probabilities with observed frequencies.

get rid of this nuisance parameter by conditioning on the observed sample size N , i.e. on the condition $\{N^* = N\}$, and obtain the multinomial probabilities (2.5):

$$\Pr(\vec{X} = \vec{k} | N) = \Pr(\vec{X} = \vec{k} | N^* = N) = \begin{cases} \frac{\Pr(\vec{X} = \vec{k})}{\Pr(N^* = N)} & \text{if } \vec{k}|N, \text{ i.e. } \sum_{ij} k_{ij} = N \\ 0 & \text{otherwise} \end{cases}$$

For any $\vec{k}|N$, this fraction evaluates to

$$\Pr(\vec{X} = \vec{k} | N) = \frac{e^{-\nu} \cdot \nu^N \cdot \frac{(\tau_{11})^{k_{11}}}{k_{11}!} \cdot \frac{(\tau_{12})^{k_{12}}}{k_{12}!} \cdot \frac{(\tau_{21})^{k_{21}}}{k_{21}!} \cdot \frac{(\tau_{22})^{k_{22}}}{k_{22}!}}{e^{-\nu} \cdot \frac{\nu^N}{N!}} = \frac{N!}{k_{11}! k_{12}! k_{21}! k_{22}!} \cdot (\tau_{11})^{k_{11}} \cdot (\tau_{12})^{k_{12}} \cdot (\tau_{21})^{k_{21}} \cdot (\tau_{22})^{k_{22}}.$$

It is sometimes possible to obtain results for the independent Poisson model more easily than in the multinomial model, and then translate them to the conditional probabilities. For instance, the maximum-likelihood estimates in the following section are derived in this way.

2.2.3 The null hypothesis

In the remainder of this chapter and in Chapter 3, we will use the observed frequencies in the corpus sample to make inferences about the unknown population parameters. The comparison of probability parameters with sample frequencies is schematised in Figure 2.9.

The **maximum-likelihood estimates (MLEs)** for the population parameters are those values which maximise the probability of the observed contingency table \vec{O} . Lemma A.1 derives the following MLE equations for the multinomial sampling distri-

bution (2.5):

$$\begin{aligned} \tau_{ij} &\approx \frac{O_{ij}}{N} \quad (\text{for } i, j \in \{1, 2\}) & \pi_1 &\approx \frac{R_1}{N} = \frac{f_1}{N} =: p_1 \\ \pi &\approx \frac{O_{11}}{N} = \frac{f}{N} =: p & \pi_2 &\approx \frac{C_1}{N} = \frac{f_2}{N} =: p_2 \end{aligned} \quad (2.8)$$

We are particularly interested in the **statistical association** between the components of a pair type. It is clear that this association is a property of the population, and our goal is to make inferences about it using information from the observed sample. However, it is not at all obvious how to measure the *strength* of the association within a pair type. I will first turn to an easier question: When is there no association at all between the components of a pair type? The answer lies in the concept of statistical **independence**. The components of a pair are completely unassociated when their occurrences (as the labels of pair tokens) have no influence on each other, i.e. when the indicator variables Y_k and Z_k (marking these occurrences) are statistically independent. Since Y_k and Z_k are binary variables, this reduces to the condition that

$$\Pr(I_{11}^{(k)} = 1) = \Pr(Y_k = 1, Z_k = 1) = \Pr(Y_k = 1) \cdot \Pr(Z_k = 1) \quad (2.9)$$

or, equivalently,

$$H_0 : \quad \pi = \pi_1 \cdot \pi_2 \quad (2.10)$$

for a given pair type $w \in C_p$.¹¹ In the terminology of statistical hypothesis tests, H_0 is the **null hypothesis of independence**, and we are interested in pair types where the sample provides clear evidence *against* H_0 . Under H_0 , the probability parameters $\vec{\tau}$ have a simple “regular” form

$$\begin{aligned} \tau_{11} &= \pi_1 \pi_2 & \tau_{12} &= \pi_1 (1 - \pi_2) \\ \tau_{21} &= (1 - \pi_1) \pi_2 & \tau_{22} &= (1 - \pi_1) (1 - \pi_2) \end{aligned} \quad (2.11)$$

and the multinomial sampling distribution (2.5) becomes

$$\Pr(\vec{X} = \vec{k} \mid N, H_0) = \frac{N!}{k_{11}! k_{12}! k_{21}! k_{22}!} \cdot (\pi_1)^{k_{11}+k_{12}} \cdot (1 - \pi_1)^{k_{21}+k_{22}} \cdot (\pi_2)^{k_{11}+k_{21}} \cdot (1 - \pi_2)^{k_{12}+k_{22}}$$

where the exponents are the row and column sums of the contingency table \vec{k} . In particular, for the probability of the observed table ($\vec{k} = \vec{O}$) we have

$$\Pr(\vec{X} = \vec{O} \mid N, H_0) = \frac{N!}{O_{11}! O_{12}! O_{21}! O_{22}!} \cdot [(\pi_1)^{R_1} \cdot (1 - \pi_1)^{N-R_1}] \cdot [(\pi_2)^{C_1} \cdot (1 - \pi_2)^{N-C_1}] \quad (2.12)$$

This version of the null hypothesis is somewhat inconvenient because the values of π_1 and π_2 (on which the sampling distribution under H_0 depends) are not determined. H_0 can be reduced to a **point hypothesis** H'_0 by adding maximum-likelihood estimates for π_1 and π_2 :

$$H'_0 : \quad \pi = \pi_1 \cdot \pi_2 \wedge \pi_1 = p_1 \wedge \pi_2 = p_2 \quad (2.13)$$

¹¹Note that (2.9) automatically holds for *all* $k \in \{1, \dots, N\}$ as soon as it holds for *some* k , because all the pairs (Y_k, Z_k) have identical distributions.

The parameters of the multinomial sampling distribution are now fully determined by H'_0 : $\tau_{11} = p_1 p_2$, $\tau_{12} = p_1(1 - p_2)$, $\tau_{21} = (1 - p_1)p_2$, and $\tau_{22} = (1 - p_1)(1 - p_2)$. Inserting these values into (2.6), with $p_1 = R_1/N$ and $p_2 = C_1/N$, we obtain the expected values of the variables X_{ij} under the null hypothesis H'_0 :

$$\begin{aligned} E_0[X_{11}] &= \frac{R_1 C_1}{N} =: E_{11} & E_0[X_{12}] &= \frac{R_1 C_2}{N} =: E_{12} \\ E_0[X_{21}] &= \frac{R_2 C_1}{N} =: E_{21} & E_0[X_{22}] &= \frac{R_2 C_2}{N} =: E_{22} \end{aligned} \quad (2.14)$$

as well as

$$\begin{aligned} E_0[X_{R1}] &= R_1 & E_0[X_{C1}] &= C_1 \\ E_0[X_{R2}] &= R_2 & E_0[X_{C2}] &= C_2 \end{aligned}$$

for the marginals. I will refer to the expectations E_{ij} under the null hypothesis H'_0 simply as **expected frequencies**, but it is important not to confuse them with the expected values $E[X_{ij}] = N\tau_{ij}$ of the general sampling distribution (without H'_0). Under H'_0 , each X_{ij} has a binomial distribution with success probability $\tau_{ij} = E_{ij}/N$,

$$\Pr(X_{ij} = k \mid N, H'_0) = \binom{N}{k} \cdot \left(\frac{E_{ij}}{N}\right)^k \cdot \left(1 - \frac{E_{ij}}{N}\right)^{N-k},$$

and the same holds for the row and column sums:

$$\begin{aligned} \Pr(X_{Ri} = k \mid N, H'_0) &= \binom{N}{k} \cdot \left(\frac{R_i}{N}\right)^k \cdot \left(1 - \frac{R_i}{N}\right)^{N-k} \\ \Pr(X_{Ci} = k \mid N, H'_0) &= \binom{N}{k} \cdot \left(\frac{C_i}{N}\right)^k \cdot \left(1 - \frac{C_i}{N}\right)^{N-k} \end{aligned}$$

The multinomial sampling distribution under H'_0 can be written as

$$\begin{aligned} \Pr(\vec{X} = \vec{k} \mid N, H'_0) &= \\ &= \frac{N!}{k_{11}! k_{12}! k_{21}! k_{22}!} \cdot \left(\frac{E_{11}}{N}\right)^{k_{11}} \cdot \left(\frac{E_{12}}{N}\right)^{k_{12}} \cdot \left(\frac{E_{21}}{N}\right)^{k_{21}} \cdot \left(\frac{E_{22}}{N}\right)^{k_{22}}. \end{aligned} \quad (2.15)$$

2.2.4 Conditioning on fixed marginal frequencies

The use of maximum-likelihood estimates in the point hypothesis H'_0 is somewhat problematic. Especially for small marginal frequencies R_1 and C_1 , they will introduce a considerable amount of uncontrolled error into the null distribution, and are likely to distort the results of hypothesis tests. (For instance, think of a situation where H_0 holds, but $p_1 \cdot p_2 \ll \pi_1 \cdot \pi_2$, so that H'_0 is rejected by a statistical hypothesis test.)

A different approach to resolving the uncertainty of the sampling distribution under H_0 is to condition on the observed row and/or column sums. In other words, we consider only those random samples where the marginal frequencies for a given pair type are identical to the ones observed in the corpus data, rather than all possible samples of size N . The advantage of such conditional distributions is that some of

the population parameters become irrelevant and hence do not have to be estimated. Conditioning the sampling distribution on the observed column sums leads to

$$\Pr(\vec{X} = \vec{k} \mid C_1, C_2) = \binom{C_1}{k_{11}} (\rho_1)^{k_{11}} (1 - \rho_1)^{C_1 - k_{11}} \cdot \binom{C_2}{k_{12}} (\rho_2)^{k_{12}} (1 - \rho_2)^{C_2 - k_{12}}, \quad (2.16)$$

for any $\vec{k} \mid C_1, C_2$, where

$$\rho_1 = \frac{\tau_{11}}{\tau_{11} + \tau_{21}} \quad \text{and} \quad \rho_2 = \frac{\tau_{12}}{\tau_{12} + \tau_{22}}$$

are the column ratios of the population parameters. Note that $\Pr(\vec{X} = \vec{k} \mid C_1, C_2) = 0$ when \vec{k} does not satisfy the conditioning equations $k_{11} + k_{21} = C_1$ and $k_{12} + k_{22} = C_2$. Also note that the conditional sampling distribution does no longer depend on the individual values of the parameters τ_{ij} , but only on the column ratios ρ_1 and ρ_2 . I have again used the shorthand notation

$$\Pr(\vec{X} = \vec{k} \mid C_1, C_2) := \Pr(\vec{X} = \vec{k} \mid X_{C_1} = C_1, X_{C_2} = C_2).$$

The conditional probability (2.16) is the product of two independent binomial distributions for the columns of a contingency table, with k_{1i} successful trials out of C_i and success probability ρ_i . The mathematical derivation is based on independent Poisson sampling, making use of the equivalence

$$\Pr(\vec{X} = \vec{k} \mid N, C_1, C_2) = \Pr(\vec{X} = \vec{k} \mid C_1, C_2),$$

which follows from the fact that $X_{C_1} = C_1$ and $X_{C_2} = C_2$ implies $\sum_{ij} X_{ij} = X_{C_1} + X_{C_2} = C_1 + C_2 = N$. Note that (2.7) implies that X_{C_1} and X_{C_2} are independent Poisson-distributed random variables with $E[X_{C_1}] = N(\tau_{11} + \tau_{21})$ and $E[X_{C_2}] = N(\tau_{12} + \tau_{22})$. The null hypothesis $H_0: \pi = \pi_1 \cdot \pi_2$ implies

$$H_{0, \text{hom}} : \quad \rho_1 = \rho_2 \quad (2.17)$$

for the conditional distribution (2.16). $H_{0, \text{hom}}$ is also called the **null hypothesis of homogeneity** (because it stipulates that the columns of the contingency table are homogeneous). Here, only a single parameter, the common column ratio $\rho_1 = \rho_2$, has to be estimated from the observed data to obtain a point null hypothesis:

$$H'_{0, \text{hom}} : \quad \rho_1 = \rho_2 = \frac{R_1}{N} =: r. \quad (2.18)$$

The sampling distribution under $H'_{0, \text{hom}}$ is

$$\begin{aligned} \Pr(\vec{X} = \vec{k} \mid C_1, C_2, H'_{0, \text{hom}}) &= \binom{C_1}{k_{11}} \cdot \binom{C_2}{k_{12}} \cdot r^{k_{11} + k_{12}} (1 - r)^{k_{21} + k_{22}} \\ &= \binom{C_1}{k_{11}} \cdot \binom{C_2}{k_{12}} \cdot \left(\frac{R_1}{N}\right)^{k_{11} + k_{12}} \left(\frac{R_2}{N}\right)^{k_{21} + k_{22}}, \end{aligned} \quad (2.19)$$

and each X_{ij} has a binomial distribution with $E_0[X_{ij}] = E_{ij}$ (the same expected frequencies as for the multinomial sampling distribution under H'_0). Conditioning on the row sums instead of the column sums leads to similar equations.

Going one step further, we will now condition on all observed marginal frequencies. Note that conditioning on $R_1, R_2, C_1,$ and C_2 is redundant because $R_1 + R_2 = N = C_1 + C_2$ (and likewise $X_{R1} + X_{R2} = \sum_{ij} X_{ij} = X_{C1} + X_{C2}$). I have therefore chosen to condition on $N, R_1,$ and C_1 :

$$\Pr(\vec{X} = \vec{k} \mid \sum_{ij} X_{ij} = N, X_{C1} = C_1, X_{R1} = R_1) = \Pr(\vec{X} = \vec{k} \mid X_{C1} = C_1, X_{C2} = C_2, X_{R1} = R_1, X_{R2} = R_2).$$

This leads to an unwieldy expression for the conditional probability, the non-central hypergeometric distribution (Agresti 1992, 134):

$$\Pr(\vec{X} = \vec{k} \mid N, C_1, R_1) = \frac{\binom{C_1}{k_{11}} \cdot \binom{C_2}{R_1 - k_{11}} \theta^{k_{11}}}{\sum_{l=\max\{0, R_1 + C_1 - N\}}^{\min\{R_1, C_1\}} \binom{C_1}{l} \cdot \binom{C_2}{R_1 - l} \theta^l} \quad (2.20)$$

for any $\vec{k} \mid N, R_1, C_1$, with the parameter

$$\theta = \frac{\tau_{11}\tau_{22}}{\tau_{12}\tau_{21}}$$

(θ is the odds ratio defined in Section 2.2.5). Note that $\Pr(\vec{X} = \vec{k} \mid N, C_1, R_1) = 0$ if \vec{k} does not satisfy the constraints on the marginal frequencies. In particular,

$$\max\{0, R_1 + C_1 - N\} \leq k_{11} \leq \min\{R_1, C_1\} \quad (2.21)$$

(most contingency tables will have $R_1 + C_1 < N$ so that the lower bound is 0). Equation (2.20) can be simplified to the **hypergeometric distribution** underlying Fisher's exact test when we also condition on the null hypothesis H_0 , which is equivalent to $\theta = 1$ (Agresti 1992, 134):

$$\Pr(\vec{X} = \vec{k} \mid N, C_1, R_1, H_0) = \frac{\binom{C_1}{k_{11}} \cdot \binom{C_2}{R_1 - k_{11}}}{\binom{N}{R_1}}. \quad (2.22)$$

As the unknown population parameters no longer appear in (2.22) above, it is not necessary to reduce H_0 to a point hypothesis by adding maximum-likelihood estimates. Note that with all marginal frequencies fixed, the sampling distribution only depends on the single value k_{11} , which is again constrained to the range (2.21).

This last distribution can also be derived directly from the column-conditioned null distribution (2.16), which assumes the form shown in the middle line of (2.19) under $H_{0, hom}$ (not $H'_{0, hom}$). The derivation conditions the probability on R_1, C_1, C_2 instead of N, R_1, C_1 and makes use of the fact that

$$\Pr(\vec{X} = \vec{k} \mid R_1, C_1, C_2, H_0) = \frac{\Pr(\vec{X} = \vec{k} \mid C_1, C_2, H_0)}{\Pr(X_{R1} = R_1 \mid C_1, C_2, H_0)}$$

for any table $\vec{k}|R_1, C_1, C_2$ that satisfies all conditioning equations. Moreover, X_{R_1} follows a binomial distribution with success probability $r = \rho_1 = \rho_2$ under H_0 . Starting from a row-conditioned sampling distribution, we obtain

$$\Pr(\vec{X} = \vec{k} \mid N, C_1, R_1, H_0) = \frac{\binom{R_1}{k_{11}} \cdot \binom{R_2}{C_1 - k_{11}}}{\binom{N}{C_1}},$$

which yields the same probabilities as the first form of the hypergeometric distribution (as can be shown by direct computation). For the derivation of the general hypergeometric sampling distribution (2.20), see (Lehmann 1991, 151–162).

2.2.5 Measuring statistical association

In Section 2.2.3 I have argued that complete absence of association is adequately described by the concept of statistical independence. However, when there is some association, we still have to find a way to quantify the size of the effect, which I call the **association strength**. In the following, I present several alternative formulae that compute a **coefficient of association strength** from the probability parameters of a pair type. (Recall that association strength is a property of pairs in the population.)

The null hypothesis $H_0: \pi = \pi_1 \cdot \pi_2$ suggests the ratio

$$\mu := \frac{\pi}{\pi_1 \cdot \pi_2} \quad (2.23)$$

as a measure for the association strength of a pair type $w \in C_p$, which I call the **μ -value**.¹² A value of $\mu = 1$ corresponds to statistical independence. For $\mu > 1$ we speak of **positive association** (where the components are *more* likely to occur together than if they were independent), and for $\mu < 1$ we speak of **negative association** (where the components are *less* likely to occur together than if they were independent).

In mathematical statistics, 2×2 contingency tables are most commonly interpreted as the result of two independent binomial samples with success probabilities ρ_1 and ρ_2 , which is sometimes referred to as a 2×2 *comparative trial* (Upton 1982, 87). This model is equivalent to the multinomial sampling distribution with one fixed margin (2.16) and the null hypothesis of equal success probabilities $H_{0,hom} : \rho_1 = \rho_2$. For this reason, coefficients of association strength are often based on a comparison of ρ_1 with ρ_2 . Such coefficients are not necessarily meaningful in the multinomial sampling model of Section 2.2.1 with its parameters π , π_1 , and π_2 . Here, the μ -value will often be found to be a more intuitive choice.

The simplest coefficient of association strength for the 2×2 comparative trial is the **difference of proportions** κ_u , i.e. the difference between the two success probabilities:

$$\kappa_u := \rho_1 - \rho_2 = \frac{\pi - \pi_1\pi_2}{\pi_2(1 - \pi_2)} = \frac{\tau_{11}\tau_{22} - \tau_{12}\tau_{21}}{\pi_2(1 - \pi_2)}.$$

¹²The letter μ is intended to be reminiscent of *mutual information*, since the quantity $\log \mu$ can be interpreted as point-wise mutual information. I have avoided using this term for μ , though, so as not to confuse information theory with population parameters.

This coefficient was used by Liddell (1976), for instance. A more useful coefficient is the ratio q of the success probabilities:

$$q := \frac{\rho_1}{\rho_2} = \frac{\pi - \pi\pi_2}{\pi_1\pi_2 - \pi\pi_2},$$

which is known as **relative risk**. Although $\mu = q = 1$ in the case of independence, the two measures judge strength of association differently. In particular,

$$q = \frac{1 - \pi_2}{\frac{1}{\mu} - \pi_2} = \frac{\mu - \mu\pi_2}{1 - \mu\pi_2},$$

so the relation between q and μ depends on the value of π_2 .

The most widely used coefficient of association strength is the **odds ratio** θ . The *odds* associated with a success probability ρ_i is the expected ratio of successes to failures $\rho_i / (1 - \rho_i) = \tau_{1i} / \tau_{2i}$, and θ is the quotient of these odds:

$$\theta := \frac{\rho_1}{1 - \rho_1} : \frac{\rho_2}{1 - \rho_2} = \frac{\tau_{11}\tau_{22}}{\tau_{12}\tau_{21}}.$$

The odds ratio has a meaningful interpretation in the 2×2 comparative trial (if one is willing to accept the odds as a measure of success probability), but it is difficult to express (and interpret) in terms of the probability parameters π , π_1 , and π_2 :

$$\theta = \frac{\pi - \pi(\pi_1 + \pi_2 - \pi)}{\pi_1\pi_2 - \pi(\pi_1 + \pi_2 - \pi)} = \frac{\tau_{22}}{\frac{1}{\mu} - (1 - \tau_{22})}.$$

The relation between μ and θ depends on $\tau_{22} = 1 - \pi_1 - \pi_2 + \pi$ and is thus influenced by all three parameters. The popularity of the odds ratio even for multinomial sampling (especially in the context of log-linear models) is due to its convenient formal properties rather than to its intuitive appeal. In particular, θ is the only parameter of the sampling distribution (2.20), conditioned on both the row and column sums but not on the null hypothesis H_0 . The coefficients κ_u , q and θ are also described by Agresti (1990, Ch. 2).

The following coefficients are based on the conditional probabilities

$$\Pr(V = v \mid U = u) = \frac{\pi}{\pi_1} \quad \text{and} \quad \Pr(U = u \mid V = v) = \frac{\pi}{\pi_2}.$$

In the literature, they are usually formulated for observed proportions rather than conditional probabilities. However, such equations involving the observed frequencies (as well as their row and column sums) can be translated to population probabilities and interpreted as coefficients of association strength. The original equations then become maximum-likelihood estimates for these coefficients (cf. Section 3.1.5). With an emphasis on cases of strong association rather than independence, the coefficients below are popular in application settings, especially in information retrieval and related fields. None of them is commonly used in mathematical statistics.

The **Dice coefficient** κ_{Dice} is the *harmonic mean* (Weisstein 1999, *s.v. Harmonic Mean*) of the two probabilities:

$$\begin{aligned} \kappa_{\text{Dice}} &:= 2 \cdot \left(\frac{1}{\Pr(V = v \mid U = u)} + \frac{1}{\Pr(U = u \mid V = v)} \right)^{-1} \\ &= 2 \cdot \left(\frac{\pi_1}{\pi} + \frac{\pi_2}{\pi} \right)^{-1} = \frac{2\pi}{\pi_1 + \pi_2}. \end{aligned}$$

The *average* κ_{mean} (or *arithmetic mean*) of the two probabilities is a possible alternative mentioned by Daille (1994, 137). Their *geometric mean* (Weisstein 1999, s.v. *Geometric Mean*) is

$$\begin{aligned}\kappa_{\text{gmean}} &:= \sqrt{\Pr(V = v \mid U = u) \cdot \Pr(U = u \mid V = v)} \\ &= \sqrt{\frac{\pi^2}{\pi_1 \pi_2}} = \frac{\pi}{\sqrt{\pi_1 \pi_2}},\end{aligned}$$

which I call the **gmean coefficient** κ_{gmean} . It is also possible to take the minimum or maximum of the two probabilities (cf. the MS measure in Section 3.1.5), yielding the **min** and **max coefficients** κ_{min} and κ_{max} .¹³ To my knowledge, κ_{max} has never been used, while κ_{min} was suggested by Pedersen and Bruce (1996) but has not been taken up by other researchers. Finally, the **Jaccard coefficient** κ_{Jaccard} is a similar conditional probability with a particularly intuitive interpretation:

$$\begin{aligned}\kappa_{\text{Jaccard}} &:= \Pr(U = u \wedge V = v \mid U = u \vee V = v) \\ &= \frac{\pi}{\pi_1 + \pi_2 - \pi} = \frac{\tau_{11}}{\tau_{11} + \tau_{12} + \tau_{21}} = \frac{\tau_{11}}{1 - \tau_{22}},\end{aligned}$$

i.e. the probability of a cooccurrence given that either u or v occurs in a pair.

I will now attempt to compare the coefficients of association strength, i.e. describe the circumstances under which they disagree about how far a given pair type deviates from independence. Note that two coefficients are **equivalent** iff there exists a monotonic transformation between their values. For instance, the Jaccard coefficient is fully equivalent to the Dice coefficient, as the following calculation shows:

$$\begin{aligned}\frac{\kappa_{\text{Dice}}}{\kappa_{\text{Jaccard}}} &= \frac{2\pi}{\pi_1 + \pi_2} \cdot \frac{\pi_1 + \pi_2 - \pi}{\pi} \\ &= 2 \cdot \left(1 - \frac{\pi}{\pi_1 + \pi_2} \right) = 2 - \kappa_{\text{Dice}}\end{aligned}$$

implies

$$\kappa_{\text{Jaccard}} = \frac{\kappa_{\text{Dice}}}{2 - \kappa_{\text{Dice}}}. \quad (2.24)$$

Since $f(x) = x/(2 - x)$ is a strictly increasing function of x on the interval $[0, 1]$, there is a monotonic bijective transformation between the values of κ_{Jaccard} and κ_{Dice} . There are no other equivalences between the coefficients, and their interrelations are usually complex and depend on various parameters.

Table 2.2 shows a comparison of the coefficient values for the special cases listed in Table 2.1. The values shown in columns B and E are first-order approximations for $\epsilon \rightarrow 0$. They converge to the values in columns A and D, respectively. Note that κ_{Jaccard} has been omitted because of its equivalence with κ_{Dice} , and the “unused” coefficients κ_{mean} and κ_{max} are not shown either. Proofs for these results can be found in Appendix A.1, Lemma A.2.

There are two major groups of coefficients: (i) μ , κ_u , q , and θ are equal to 1 (or 0 for κ_u) in the case of independence (A), and they lead to the same distinction

¹³Interestingly, κ_{max} is obtained by scaling μ to the range $[0, 1]$ for given values of π_1 and π_2 ($\pi \leq \min\{\pi_1, \pi_2\}$ implies $\mu \leq \min\{\pi_1^{-1}, \pi_2^{-1}\}$).

A:	$\pi = \pi_1\pi_2$	(independence)
B:	$\pi = (1 + \epsilon)\pi_1\pi_2$	(minimal association)
C:	$\pi = 0$	(total negative association)
D:	$\pi = \pi_1 = \pi_2$	(total positive association)
E:	$\pi_1 = \pi_2 = (1 + \epsilon)\pi =: \delta\pi$	(nearly total association)
F:	$\pi = \pi_1 \ll \pi_2$	(total determination)
F':	$\pi = \pi_2 \ll \pi_1$	

Table 2.1: List of special situations for the comparison of different coefficients of association strength. The symbol ϵ in Equations B and E indicates a first-order approximation for $\epsilon \rightarrow 0$.

between positive and negative association. However, they vary greatly in the case of strong positive association (D,E), and their values are difficult to interpret then. The most consistent results are obtained by the odds ratio θ , although it does not allow for a distinction between total association (D) and total determination (F). The μ -value is well-suited for measuring small degrees of association (B) and is often used for this purpose in mathematical statistics. Relative risk ρ and the difference of proportions κ_u in particular are not symmetric between rows and columns and seem less useful in this context. (ii) κ_{Dice} , κ_{gmean} and κ_{min} are good indicators of total (positive or negative) association (C,D), where they are all equal to 1 and 0, respectively. Their values are also easy to interpret in the case of strong positive association (E), but are less clear for total determination (F,F'). Differences between the coefficients are most conspicuous in this case, where κ_{Dice} and especially κ_{gmean} assign higher values to totally determined pairs with a strong imbalance between the marginal probabilities. The major disadvantage of the coefficients in this group is that none of them assumes a specific value in the case of independence (A), so that they cannot be used to measure small degrees of association or to distinguish between positive and negative association.

An ideal coefficient of association strength would combine the well-defined behaviour of μ for a small degree of association (A,B) with the equally well-defined behaviour of κ_{Dice} , κ_{gmean} and κ_{min} for nearly total association (D,E). Unfortunately, it is not clear at the moment how a statistically sound and mathematically convenient coefficient with these properties could be derived. Another point of uncertainty is the desirable behaviour for the edge case of total determination (F,F').

2.3 Adequacy of the statistical models

2.3.1 Assumptions of the random sample model

In mathematical terms, the random sample model of Section 2.2 makes two assumptions about the data: (i) the pairs of random variables (U_k, V_k) are statistically independent (**independence**) and (ii) their distributions are identical to the prototype (U, V) (**homogeneity**).

These assumptions can be violated by real-world data in various ways and for various reasons. Among the major causes of non-randomness in cooccurrence data

coefficient	A	B	C	D	E	F	F'
μ	1	$1 + \epsilon$	0	$\frac{1}{\pi}$	$\frac{1}{\pi} - \frac{2}{\pi}\epsilon$	$\frac{1}{\pi_2}$	$\frac{1}{\pi_1}$
ϱ	1	$1 + \frac{\epsilon}{1 - \pi_2}$	0	$+\infty$	$\frac{1}{\epsilon} \left(\frac{\pi}{1 - \pi} \right)^{-1}$	$+\infty$	$\frac{1 - \pi_2}{\pi_1 - \pi_2}$
θ	1	$1 + \frac{\epsilon}{(1 - \pi_1)(1 - \pi_2)}$	0	$+\infty$	$\frac{1}{\epsilon^2} \left(\frac{\pi}{1 - \pi} \right)^{-1}$	$+\infty$	$+\infty$
κ_u	0	$\frac{\epsilon \pi_1}{1 - \pi_2}$	$-\frac{\pi_1}{1 - \pi_2}$	1	$1 - \frac{\epsilon}{1 - \pi}$	$\frac{\pi_1}{\pi_2}$	$\frac{1 - \pi_1}{1 - \pi_2}$
κ_{Dice}	$\frac{2\pi_1\pi_2}{\pi_1 + \pi_2}$	$(1 + \epsilon) \frac{2\pi_1\pi_2}{\pi_1 + \pi_2}$	0	1	$1 - \epsilon$	$\frac{2}{1 + \pi_2/\pi_1}$	$\frac{2}{1 + \pi_1/\pi_2}$
κ_{gmean}	$\sqrt{\pi_1\pi_2}$	$(1 + \epsilon) \sqrt{\pi_1\pi_2}$	0	1	$1 - \epsilon$	$\sqrt{\frac{\pi_1}{\pi_2}}$	$\sqrt{\frac{\pi_2}{\pi_1}}$
κ_{min}	$\min \{\pi_1, \pi_2\}$	$(1 + \epsilon) \min \{\pi_1, \pi_2\}$	0	1	$1 - \epsilon$	$\frac{\pi_1}{\pi_2}$	$\frac{\pi_2}{\pi_1}$

Table 2.2: Values of various coefficients of association strength for the special cases of independence (A), minimal association (B), total negative association (C), total positive association (D), nearly total association (E), and total determination (F and F')

are the following:

Ordering dependencies impose constraints on the order in which tokens appear in the sample. One reason for such dependencies is the syntactic structure of sentences. To give a famous example, the token sequence *the the* is impossible in English, but a word-level random sample model assigns a non-zero probability of $p \approx 0.0036$ to it (Baayen 2001, 163), indicating an occurrence about every 300 words.¹⁴

Inhomogeneity of the sample causes the population parameters to change between different parts of the sample. The source corpus is made up from documents with different properties (e.g. different text types, different authors, the sections of a newspaper, a collection of novels or technical documents), so that the population parameters may be different for each of the documents. Baayen (2001) refers to this problem as *lexical specialisation*.

Clustering or **repetition** effects, where the probability of repeated occurrences is much higher than predicted by a random sample model. Repetitions typically occur within text-structural units such as newspaper articles or technical documents, and they are often linked to the topic of the respective unit (cf. Katz 1996; Church 2000).

Of course, the null hypothesis H_0 introduced in Section 2.2.3 is also highly unrealistic for natural language data. However, this does not affect the validity of the random sample model (but it will affect the interpretation of association scores in Chapter 3). For the same reason, only those violations of the random sample assumption which have a substantial influence on the multinomial sampling distribution (2.5) are problematic for our statistical model. In particular, ordering dependencies will usually not have a major effect on the joint and marginal frequencies. Inhomogeneity can be accounted for to a certain degree by interpreting the full sample of size N as a composite consisting of r smaller samples of sizes N_1, \dots, N_r (with $N_1 + \dots + N_r = N$), taken from different populations. As long as the number of samples r is relatively small, such a composite is highly similar to a sample of size N from a *mixture population* (this approximation can be motivated in terms of independent Poisson sampling). Therefore, the random sample assumptions are not violated, but strong associations in one of the component populations may be obscured in the mixture population.

The most serious problems for the statistical model of Section 2.2 are therefore created by the clustering of pair types (as well as their component types) within small text segments. Such clustering effects inflate both the joint and marginal frequencies. In particular, low-probability types are quite likely to appear twice or more rather than just once.¹⁵ In the following section, I present a method for estimating the extent to which clustering effects violate the randomness assumption for a given sample.

¹⁴As a matter of fact, the sequence *the the* was found twice in the final draft of this thesis, corresponding to a relative frequency of $p \approx 3.510^{-5}$. Chapter 4 explains why the relative frequency is not a valid estimate for the occurrence probability in this case.

¹⁵It is theoretically possible to explain clustering effects as a kind of inhomogeneity, where the corpus sample is a composite taken from a different population for each text segment. Pair types that are relevant to the topic of a segment and therefore likely to be repeated will have a highly increased cooccurrence probability in the corresponding population. However, this composite cannot be interpreted as a random sample from a mixture population because it consists of a large number of

2.3.2 Clustering and dispersion

It is usually not feasible to test the randomness assumption directly by comparing the *empirical* sampling distribution based on observed data from different source corpora to the *theoretical* sampling distribution predicted by the random sample model. Apart from the practical difficulty of finding and processing a sufficient number of comparable corpora (so that it is realistic to assume that they are random samples from the same population), we would need to know the exact population probabilities: even for high-frequency types we cannot simply use the maximum-likelihood estimates, which may have been affected by the consequences of non-randomness.

What we can do, though, is to check whether the observed instances of a given pair type are distributed evenly across the sample. The relevant kinds of non-randomness as discussed in the previous section, in particular clustering effects, will also cause an uneven distribution in large samples. These effects are most clearly visible – and have the most disastrous consequences – for the lowest-frequency pair types, and my evaluation will concentrate on those.

The standard randomness test used in statistics is the runs test, which is mainly intended for testing the non-independence of consecutive events (e.g. Siegel 1956). This test is not applicable here because cooccurrence probabilities are (almost always) very low so that runs of length greater than one are extremely rare. It is also highly sensitive to ordering dependencies that are not relevant for cooccurrence statistics. Baayen (2001) computes the **dispersion** of the instances of types across a corpus to test the randomness and independence assumptions of his statistical model for word frequency distributions (which is equivalent to independent Poisson sampling, cf. Section 2.2.2). This dispersion test can be applied to relational cooccurrence data in a straightforward way by splitting the base data (i.e. the sequence of N pair tokens) into K parts of equal size. If there is a natural segmentation of the source corpus, e.g. into newspaper articles or technical documents, the division could be based on this segmentation (and it should be, since we expect clustering effects within such text-structural units). However, due to the often wildly different sizes of these segments (see Figure 2.14 for the lengths of individual articles in the *Frankfurter Rundschau* corpus), the mathematical analysis becomes more involved (see e.g. Katz 1996) and may require computationally expensive methods or Monte Carlo sampling. Therefore, I use fixed-size parts, which have the further advantage that the dispersion test can be applied to the base data without additional information from the source corpus.

The dispersion test can detect both clustering and inhomogeneity effects, depending on the size $S = N/K$ of the individual parts. Baayen (2001, 165–167) divides his sample from *Alice in Wonderland* into $K = 40$ parts of equal size, with the express intention of measuring lexical specialisation. A division into smaller parts, on the other hand, allows the selective identification of clustering effects. The precise choice of S is a matter of experience. In general, it should be close to the average size of the smallest text segments within which clustering effects are expected.

The dispersion D of a given pair type w is the number of parts that contain at least one instance of w . The dispersion test is based on the comparison of the observed dis-

small samples (one for each text segment). Thus, clustering effects constitute a true violation of the randomness assumption.

person with the sampling distribution $\Pr(D = d)$ under the randomness assumption. Since we do not know the true probability parameter π and its maximum-likelihood estimate is unreliable for low frequency ranks, the standard procedure, once again, is to condition on the sufficient statistic f for π , resulting in conditional probabilities $\Pr(D = d \mid f = m)$ for a pair type with observed cooccurrence frequency m . A pair type with observed dispersion d is called **underdispersed** if the cumulative probability $p_{d,m} := \Pr(D \leq d \mid f = m)$ is sufficiently small. At first sight, one may be tempted to interpret all underdispersed types as evidence for clustering effects. However, due to the very large number of rare types that is characteristic for word frequency distributions (and those of word cooccurrences in particular, cf. Chapter 4), a substantial number of types may be underdispersed purely by chance, even if $p_{d,m}$ is small. Writing V_m for the number of types with $f = m$ in the sample, the expected number of types with dispersion $D \leq d$ in this frequency class is $V_m \cdot p_{d,m}$. There is evidence for clustering effects only when the observed number of such types is significantly larger (measured by a binomial test).

Baayen (2001) uses a Monte Carlo simulation to obtain approximate probabilities for the dispersion test. The exact values are derived in Lemma A.3:

$$\Pr(D = d \mid f = m) = \binom{N}{m}^{-1} \binom{K}{d} \sum_{j=1}^d (-1)^{d-j} \binom{d}{j} \binom{S \cdot j}{m} \quad (2.25)$$

for K parts of size S each, so that $N = K \cdot S$. These probabilities can easily be computed with the help of a recurrence formula. For given K , S and $N = K \cdot S$ we find

$$\Pr(D = d \mid f = m) = \binom{N}{m}^{-1} \binom{K}{d} A(d, m) \quad (2.26)$$

with

$$A(d, m) = \binom{S \cdot d}{m} - \sum_{j=1}^{d-1} \binom{d}{j} A(j, m) \quad (2.27)$$

(see Lemma A.4). Note that this formula still requires high-precision arithmetic to avoid catastrophic cancellation (the computed probabilities are only reliable for $m \leq 10$ otherwise). An accurate implementation of the dispersion test is provided within the UCS toolkit (cf. Sections 3.2.2 and B.1).¹⁶

As a case study, dispersion tests were performed for the AN-FR data set, with $K = 200$ ($S = 8\,975$, corresponding approximately to one day's worth of text) and $S = 100$ ($K = 17\,950$, so that each part covers two or three articles). The results are shown in Tables 2.3 and 2.4. In both cases, highly significant underdispersion was found for frequency ranks $m = 2, \dots, 5$ and all values $1 \leq d < m$. The difference $m - d$ can be interpreted as the amount by which the cooccurrence frequency of an underdispersed pair type is inflated. Totalling up the number of observed types with $d \leq m - 1$ in Table 2.3 and subtracting the corresponding expected numbers, we see that there are some 24 000 types whose cooccurrence frequency is inflated by non-randomness effects. Table 2.4 shows that at least 12 000 of those cases are almost certainly caused by clustering. Similar and even more drastic results were obtained for the same frequency ranks in the AN-HGC data set (see Evert 2004b).

¹⁶The `ucs-make-tables` command-line tool can be used to compute dispersion statistics, which are then evaluated and compared to the theoretical distribution with the `dispersion-test` script.

m	d	V_m	$\Pr(D \leq d \mid f = m)$	# of types with $D \leq d$	
				expected	observed
2	1	102 256	0.0049995	511	12 591
3	1	31 949	$2.499 \cdot 10^{-5}$	1	830
	2	31 949	0.0149484	478	5 726
4	1	17 538	$1.249 \cdot 10^{-7}$	0	178
	2	17 538	0.0001742	3	1 313
	3	17 538	0.0297225	521	4 479
5	1	9 956	$6.243 \cdot 10^{-10}$	0	49
	2	9 956	$1.866 \cdot 10^{-6}$	0	211
	3	9 956	0.0006173	6	887
	4	9 956	0.0491259	489	3 044

Table 2.3: Results of dispersion test for the AN-FR data set with $K = 200$ and $S = 8975$. The expected number of underdispersed types is rounded to the nearest integer. All observed results are significant at a level of $\alpha = .001$.

m	d	V_m	$\Pr(D \leq d \mid f = m)$	# of types with $D \leq d$	
				expected	observed
2	1	102 256	$5.515 \cdot 10^{-5}$	6	5 578
3	1	31 949	$3.011 \cdot 10^{-9}$	0	358
	2	31 949	0.0001655	5	2 863
4	1	17 538	$1.627 \cdot 10^{-13}$	0	56
	2	17 538	$2.117 \cdot 10^{-8}$	0	630
	3	17 538	0.0003309	6	2 281
5	1	9 956	$8.703 \cdot 10^{-18}$	0	15
	2	9 956	$2.474 \cdot 10^{-12}$	0	87
	3	9 956	$7.573 \cdot 10^{-8}$	0	420
	4	9 956	0.0005514	5	1 524

Table 2.4: Results of dispersion test for the AN-FR data set with $K = 17950$ and $S = 100$. The expected number of underdispersed types is rounded to the nearest integer. All observed results are significant at a level of $\alpha = .001$.

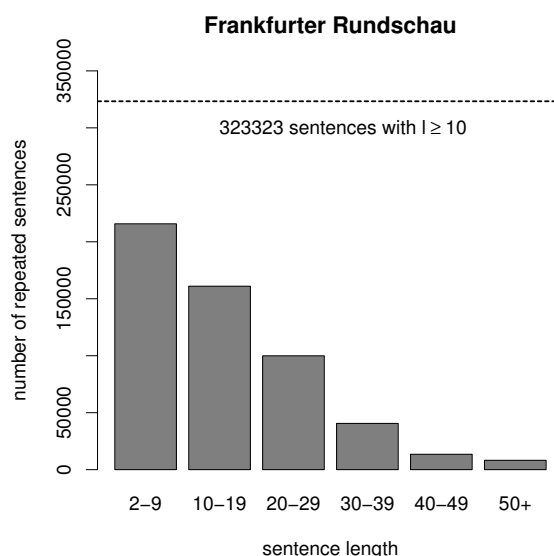


Figure 2.10: Number of sentence repetitions in the *Frankfurter Rundschau* corpus, broken down by sentence length.

The observed underdispersion for the larger segments shown in Table 2.3, may in part be due to an artefact of the corpus. The *Frankfurter Rundschau* contains many duplicates and, worse, near-duplicates of entire articles. The size of this problem can be estimated by counting the number of (exact) sentence duplicates. Of the 2 076 541 sentences in the corpus, 388 379 are *identical* repetitions of a previous sentence. Although many of those are one- or two-word “pseudo sentences” (such as a location or the name of a press agency), Figure 2.10 shows a substantial number of duplicates even for long sentences: 323 323 of them have length 10 or greater. In total, the sentence repetitions add up to more than 5 million running words, or 12.64% of the entire corpus. Article duplicates, most of which are published on different days, cannot account for the underdispersion shown in Table 2.4, though.

2.3.3 Extraction noise

Automatic pre-processing and extraction of cooccurrences invariably introduces **noise** into the base data because: (i) word tokens are not identified correctly; (ii) tokens are labelled with the wrong types; or (iii) there are errors in the detection of the targeted structural relation (i.e. wrong pair tokens are generated). All these errors produce **false negatives** (FN, missing pair tokens) and **false positives** (FP, spurious cooccurrences) in the base data.¹⁷ In the resulting data set, both the actual set of observed pair types and the frequency signatures are affected. The goals of this section are two-fold: (i) quantify the amount of noise in automatically extracted cooccurrence data (depending on the extraction methods used) and (ii) show how random extraction noise can be accounted for in the statistical model of Section 2.2.

¹⁷Note that in this view, a correctly identified pair token with wrong labels (e.g. because of a lemmatisation error) counts both as a false positive and as a false negative!

extraction method	perfect tagging		TreeTagger tagging	
	precision	recall	precision	recall
adjacent pairs	98.47%	90.58%	94.81%	84.85%
window-based	97.14%	96.74%	93.85%	90.44%
YAC chunks	98.16%	97.94%	95.51%	91.67%

Table 2.5: Evaluation results for the extraction of German adjective-noun cooccurrences (from Evert and Kermes 2003).

An **evaluation** of the extraction methods is carried out by counting false negatives and false positives in the base data (i.e. at the level of labelled pair tokens). The amount of noise is measured in terms of **precision** (proportion of true positives among all extracted pair tokens) and **recall** (proportion of the correct pair tokens in the source corpus that were found by the automatic extraction). An estimate for the precision value can be determined fairly easily by checking a sample of the base data manually. However, precise guidelines (on what counts as a true positive) have to be worked out for the manual annotation. In order to obtain an estimate for the recall value, all instances of the desired relation have to be identified in the source corpus. This labour-intensive task can be avoided – so that the evaluation becomes feasible – when a treebank corpus is available as a gold standard.

Evert and Kermes (2003) used the German Negra treebank (Skut *et al.* 1998) as a gold standard to evaluate the extraction of adjective-noun cooccurrences. The results of this evaluation are reproduced here in Table 2.5. Although adjective-noun cooccurrences are comparatively easy to extract, the excellent results achieved with simple part-of-speech patterns (referred to as “window-based” extraction in Table 2.5) are astonishing and motivated the use of the same patterns for the AN-FR and AN-HGC data sets.

In the following, I will show how **random noise** can be accounted for in our statistical model. **Systematic noise**, on the other hand, is always problematic and one just has to hope that there is only a limited amount of it. Evert and Kermes (2003) found only a small number of clearly systematic errors in their evaluation. Typical causes of such errors are: (i) extraction from a partial syntactic analysis, yielding systematic combinations of word tokens within certain segments or windows (leading to inflated marginal frequencies, cf. Section 2.4.1); and (ii) systematic errors in the part-of-speech tagging or morphological analysis (examples of such errors are given by Evert and Kermes (2003)).

Concerning random noise, **false negatives** simply reduce the sample size N . Although some information is lost, the results of the statistical analysis are not distorted (because the random deletion of pair tokens leads to an equally random sample of smaller size). Non-systematic **False positives** can themselves be interpreted as a random sample, but from a different population (which I call the **noise population**). Thus, the observed data is a composite of random samples from two populations. Following the argument in Section 2.3.1, we can interpret this composite as a random sample from a single **mixture population**, whose population probabilities are weighted mixtures of the true population parameters and the noise population parameters. Thus the random sample model is still valid, but the population proba-

bilities are distorted. Unfortunately, it is difficult to make any predictions about the parameters of the noise population. If we allow ourselves to assume that the false positives are entirely random combinations (i.e. all pair types in the noise population satisfy H_0), the noise population has essentially the effect of weakening the association strength of pair types in the true population. It will thus make it harder to detect highly associated pairs, but should not introduce spurious associations into the results. Unfortunately, available reference corpora are too small for a meaningful empirical study of the the shape of the noise population and the influence that false positives have on the statistical analysis.

2.4 Positional cooccurrences

Positional cooccurrences were commonly used in the early days of NLP before sophisticated linguistic pre-processing and syntactic analysis was possible (see Stevens *et al.* 1965; Choueka 1988; Breidt 1993). Nowadays, the main proponents of positional cooccurrences either advocate a radically different approach to syntax based on patterns and priming (cf. Barlow and Kemmer 2000), or they are interested in knowledge-free statistical processing (where “knowledge” refers mostly to *linguistic* knowledge and theories).¹⁸ A good example is the extreme standpoint of Lehr (1996) and others, rejecting any automatic processing that might introduce noise into the cooccurrence data. Her arguments are largely invalidated, though, by the evaluation of extraction methods in Section 2.3.3 and the unproblematic statistical interpretation of noise.

Positional notions of cooccurrences require different counting methods than relational cooccurrences in order to ensure that the data can be interpreted with random sample models. These methods are also based on a four-way classification, resulting in a contingency table for each pair type found in the corpus. It is not always possible to obtain marginal frequencies directly from the corpus, but the contingency tables can be translated into frequency signatures with the transformation rules from Section 2.1.2. Note, however, that neither the marginal frequencies nor the full contingency tables can be obtained by summation over the cooccurrence frequencies of different pair types (as was the case for relational cooccurrences).

Although the contingency tables of positional cooccurrences have a different interpretation from those of relational ones, most association measures can be applied to both types alike. In the following sections, I will formulate statistical models for the two subtypes of positional cooccurrences, namely **segment-based** and **distance-based** cooccurrences. I will then show that these models are (almost) equivalent to the model for relational data presented in Section 2.2.1.

Unlike relational cooccurrences, where labelled pair tokens can be extracted from the corpus directly, both kinds of positional cooccurrences require the explicit identification of **word tokens** as a first step. Except for the knowledge-free approaches that allow no *a priori* classification of words, cooccurrences will often be constructed from two different sets of word tokens, T_1 and T_2 , for the first and second compo-

¹⁸An exception are some applications that attempt to identify synonyms and antonyms from their cooccurrences in sentences (cf. Section 1.2.1). In this case, there may not be a direct syntactic relation between the cooccurring words, so that the relational model would fail.

	$v \in S$	$v \notin S$
$u \in S$	O_{11}	O_{12}
$u \notin S$	O_{21}	O_{22}

Figure 2.11: Contingency table for segment-based cooccurrences.

nents of the pairs. For instance, when looking for noun-verb combinations, T_1 would contain only nouns and T_2 would contain only verbs (so they are disjoint subsets of the set T of all word tokens). The token sets T_1 and T_2 , as well as the corresponding sets of types C_1 and C_2 and the type mappings $\phi_1 : T_1 \rightarrow C_1$ and $\phi_2 : T_2 \rightarrow C_2$, are assumed as prerequisites in the following sections, where the counting methods and statistical models are formulated. The set C_p of pair types is defined as the Cartesian product of the component types, $C_p = C_1 \times C_2$, and can later be restricted to the types that are actually observed in the corpus. In many cases, both the token sets and the type sets will be disjoint, i.e. $T_1 \cap T_2 = \emptyset$ and $C_1 \cap C_2 = \emptyset$. However, unless stated otherwise, the counting methods presented here can also be applied to non-disjoint sets of tokens and even to the case where they are identical ($T_1 = T_2$, $C_1 = C_2$ and $\phi_1 = \phi_2$) without modification.

2.4.1 Segment-based cooccurrences

Corpus data

For segment-based cooccurrence data, the source corpus is divided into a sequence of non-overlapping segments S_1, \dots, S_N , such that every word token (from T_1 and T_2) can be assigned to exactly one segment. Here, the sample size N corresponds to the number of segments rather than the number of extracted pair tokens. Segements are typically sentences (perhaps also smaller clauses), paragraphs, articles, or other text-structural units (see also the work on collections of technical documents such as Katz (1996) and Church (2000)).

For a given pair type $(u, v) \in C_p$, the N segments are classified into four bins according to whether they contain at least one instance of u and/or v . Thus, O_{11} is the number of segments containing instances of both u and v , O_{12} is the number of segments containing at least one instance of u , but no instances of v , O_{21} vice versa, and O_{22} is the number of segments containing neither instances of u nor of v . This four-way classification is schematised in the form of a contingency table in Figure 2.11. Since every segment is assigned to exactly one bin in the contingency table, we obviously have $O_{11} + O_{12} + O_{21} + O_{22} = N$.

It is important to remember that segments containing more than one instance of u or v are still counted just once. Hence, the marginal frequencies correspond to the number of segments containing an instance of u ($= R_1$) and the number of segments

containing an instance of v ($= C_1$), rather than the word frequencies f_u and f_v .¹⁹ (Although this situation is the same for relational cooccurrences, where component frequencies rather than word frequencies must be used, I emphasise the distinction here because I have the impression that researchers who use a segment-based model are prone to substitute the individual word frequencies f_u and f_v for the correct marginal frequencies R_1 and C_1). For the same reason, the marginal frequencies cannot be obtained by summation over different pair types.²⁰

The statistical model

With segment-based cooccurrences, we cannot simply extract all combinations of word tokens within segments and interpret them as a random sample of pair tokens in the sense of Section 2.2.1. This approach would inflate the component frequencies and violate the randomness assumption: for each instance of a pair type in the sample, both component types will also occur in many other pair tokens extracted from the same segment. Thus, pair tokens are not independent within each segment, and component frequencies are quantified in steps larger than 1.

For this reason we have to resort to another statistical model, which is based on a separate random sample for each pair type $w = (u, v)$.²¹ Each such sample corresponds to a different classification of the N segments into four bins, according to whether or not they contain at least one instance of u and/or v . It can be described by indicator variables

$$Y_k := \begin{cases} 1 & \text{if the } k\text{-th segment contains at least one instance of } u \\ 0 & \text{otherwise} \end{cases}$$

and

$$Z_k := \begin{cases} 1 & \text{if the } k\text{-th segment contains at least one instance of } v \\ 0 & \text{otherwise} \end{cases}$$

Note that due to the use of indicator variables we cannot distinguish between single and multiple occurrences of u and v within a segment. Y_k and Z_k correspond to the indicator variables defined in Section 2.2.1. $E[Y_k] = \Pr(Y_k = 1)$ is the probability that the k -th segment contains at least one instance of u , and $E[Z_k] = \Pr(Z_k = 1)$ is the probability that the k -th segment contains at least one instance of v . The second part of Section 2.2.1 (beginning with the introduction of Y_k and Z_k) and the following sections are thus equally valid for segment-based cooccurrences (note that all relevant probabilities can be defined in terms of the indicator variables $I_{ij}^{(k)}$, which are derived from Y_k and Z_k).

¹⁹The word frequencies can be computed from the individual type mappings, though: $f_u = |\{t \in T_1 \mid \phi_1(t) = u\}| = |\phi_1^{-1}(u)|$ and $f_v = |\{t \in T_2 \mid \phi_2(t) = v\}| = |\phi_2^{-1}(v)|$.

²⁰In particular, in all but the most trivial cases we find that $\sum_{u \in C_1} \sum_{v \in C_2} f_{(u,v)} \neq N$.

²¹Of course, the samples for different pair types are not independent because they are derived from the same distribution of word tokens across the segments. In the model of Section 2.2.1, this dependence is embodied by the multinomial distribution of the random variables W_i . However, the dependencies are not taken into account by the statistical analysis and the association measures of Chapter 3.1, which are applied to each pair type on its own. For the same reason, the statistical model for segment-based cooccurrences does not have to consider the statistical dependencies between the samples for different pair types.

The population parameters $\pi_1 = E[Y_k]$ and $\pi_2 = E[Z_k]$ are the probabilities that any given segment contains at least one instance of u and v , respectively. Likewise, $\pi = E[I_{11}^{(k)}]$ is the probability that a segment contains instances of both u and v .²² The alternative set of parameters $\tau_{ij} = E[I_{ij}^{(k)}]$ has a similar interpretation (e.g. τ_{12} is the probability that a segment contains an instance of u but not of v).

A distinct advantage of segment-based cooccurrences is that they are not sensitive to the repetition of words and pairs within segments. Katz (1996) and Church (2000) demonstrate that the likelihood of multiple occurrences of a *topical* word within a document, given that there is at least one occurrence, is much higher than the total likelihood of one or more occurrences (which is incompatible with a simple random-sample model of language). Katz formulates an empirical model for the probabilities of multiple occurrences, but we do not need such a model: π , π_1 , and π_2 measure the total probability of one or more (co-)occurrences in a segment.

2.4.2 Distance-based cooccurrences

Corpus data

Distance-based cooccurrences are *directional*: one has to choose one word type as a base and then determine its cooccurrents (cf. the discussion at the end of Section 1.2.2). In the following presentation of the counting methods, I assume the base to be the second component v of each pair (without loss of generality, of course). In order to simplify notation, I also assume that T_1 and T_2 are disjoint sets.²³ For each token $t \in T_2$, a **local window** $W(t) \subseteq T_1$ is determined which contains all possible cooccurrents of t , i.e. all tokens $s \in T_1$ within a maximal allowed distance. When the local windows are obtained from a distance measure d , we can use the following definition:

$$W(t) := \{s \in T_1 \mid d(s, t) \leq L\}$$

(where $d(s, t)$ is the distance between two tokens in the corpus). Note that additional constraints may be involved in the definition of $W(t)$, e.g. that base and cooccurrent must belong to the same sentence (which is reasonable for most applications) or that s must precede t (resulting in one-sided windows). For a given word type $v \in C_2$, the **window** $W(v)$ of v is the union of the local windows around its instances:

$$W(v) = \bigcup_{t \in \phi_2^{-1}(v)} W(t)$$

The window size is given by the number of tokens in $W(v)$, i.e. $|W(v)|$. When some of the local windows overlap, the total window size is not equal to the sum of the local window sizes:

$$|W(v)| \neq \sum_{t \in \phi_2^{-1}(v)} |W(t)|$$

²²Recall that the “marginal” parameters π_1 and π_2 cannot be computed by summation over the cooccurrence probabilities π in this case.

²³If this is not the case, the instances of a given base $v \in C_2$ have to be removed both from the window $W(v)$ and its complement $CW(v)$. When $T_1 = T_2 = T$, this can be achieved by setting $T_1 := T \setminus \phi_2^{-1}(v)$ for each choice of v .

	$W = W(v)$	$W = CW(v)$	
$u \in W$	O_{11}	O_{12}	$R_1 = f(u)$
$u \notin W$	O_{21}	O_{22}	$R_2 = N - f(u)$

$$C_1 = |W(v)| \quad C_2 = |CW(v)| \quad N = |T_1|$$

Figure 2.12: Contingency table for distance-based cooccurrences.

In order to compute the contingency table for a pair type $(u, v) \in C_p$, the word tokens in the set T_1 are cross-classified in two steps. First, T_1 is divided into the window $W(v)$ of v and the remaining set $CW(v) = T_1 \setminus W(v)$ (the window's complement). Then, each of these sets is split into instances of u and the remaining tokens. This two-by-two layout of bins is schematised in the form of a contingency table in Figure 2.12, with the columns corresponding to the top-level bins $W(v)$ and $CW(v)$, the first row corresponding to instances of u , and the second row corresponding to the remaining tokens. The sample size is defined as $N = |T_1|$, so that $O_{11} + O_{12} + O_{21} + O_{22} = N$ holds. Note the row and column sums shown in the margins of the contingency table, where $f(u) = |\phi_1^{-1}(u)|$ stands for the word frequency of u . The frequency signature of (u, v) , which is obtained from the contingency table, can also be computed directly: f is the number of instances of u within the window $W(v)$, f_1 is the total number of instances of u ($f_1 = f(u)$), f_2 is the total size of the window $W(v)$ ($f_2 = |W(v)|$), and N is the number of tokens in T_1 ($N = |T_1|$).

Both relational and segment-based positional cooccurrences are symmetric in the sense that exchanging T_1 and T_2 simply transposes the contingency table (which will usually not affect the statistical analysis). For distance-based cooccurrences, on the other hand, swapping the roles of u and v , so that the instances of v are cross-classified against the window $W(u)$, produces an entirely different result (see Figure 2.13). In the general case ($T_1 \neq T_2$), the sample size N will be different. Even for $T_1 = T_2 = T$ (where N does not change), the second contingency table is the transpose of the first *only* for $f(u) = f(v)$ and $|W(u)| = |W(v)|$ (which would be pure coincidence).

It is not easy to implement frequency counts for distance-based cooccurrences in an efficient manner. Terra and Clarke (2004) describe a fast algorithm based on a word index of the corpus, which they apply to a huge corpus of 53 billion running words harvested from the Internet.

The statistical model

The statistical model required for the interpretation of distance-based cooccurrences is quite different from the previous models. The random sample interpretation of Section 2.2.1 is ruled out for the same reasons as in the case of segment-based cooccurrences. On the other hand, there is no segmentation of the source corpus so the model of Section 2.4.1 is not applicable either.

	$v \in W$	$v \notin W$	
$W = W(u)$	O_{11}	O_{12}	$R_1 = W(u) $
$W = CW(u)$	O_{21}	O_{22}	$R_2 = CW(u) $

$$C_1 = f(v) \quad C_2 = N - f(v) \quad N = |T|$$

Figure 2.13: Alternative contingency table for distance-based cooccurrences.

The model suggested here divides the token set T_1 into the window $W(v)$ and its complement $CW(v)$ for a given pair type $w = (u, v)$. The number $f(v)$ of instances of v is interpreted as a pre-determined parameter (similar to the sample size) that is irrelevant for inferences in the model (formally, the sampling distribution is conditioned on the word frequency $f(v)$ and the corresponding window size $|W(v)|$). The model assumes that the tokens of $W(v)$ and $CW(v)$ are randomly generated, but possibly with different probability parameters: ρ_1 is the probability that a token in $W(v)$ is assigned the label u (so that it becomes an instance of u), and ρ_2 is the probability that a token in $CW(v)$ is assigned the label u . Again, we have a separate random sample for each pair type.

Under these assumption, the sampling probability of a given contingency table $\vec{k} | C_1, C_2$ (i.e. with the pre-determined column sums C_1 and C_2) is the product of two independent binomial distributions (the first with $|W(v)|$ trials and success probability ρ_1 , the second with $|CW(v)|$ trials and success probability ρ_2):

$$\Pr(\vec{X} = \vec{k}) = \binom{C_1}{k_{11}} (\rho_1)^{k_{11}} (1 - \rho_1)^{C_1 - k_{11}} \cdot \binom{C_2}{k_{12}} (\rho_2)^{k_{12}} (1 - \rho_2)^{C_2 - k_{12}}. \quad (2.28)$$

This probability is identical to the conditional sampling distribution (2.16) for fixed column sums that was described in Section 2.2.4. Therefore, any results and association measures that assume fixed column sums (such as the log-likelihood measure, see Section 3) are applicable to contingency tables for distance-based cooccurrences. The hypergeometric sampling distribution (where all marginals are fixed) can be derived from the column-conditioned distribution, hence it is also valid for distance-based cooccurrences, and so is the Fisher measure (which is based on this distribution). Strictly speaking, association measures that are directly based on the random sample model of Section 2.2.1 are not valid for distance-based cooccurrences. However, differences between the unconstrained and the column-conditioned distribution will often be minor, so that most association measures are applicable *in practice*. Note that exchanging the roles of u and v in the model leads to the conditional sampling distribution for fixed row sums.

2.4.3 Examples

Many examples both of segment-based and distance-based cooccurrences are presented in the reports of early work in the field of natural language processing (e.g. Stevens *et al.* 1965; Choueka 1988) as well as in more recent research in the Neo-Firthian tradition (e.g. Lehr 1996; Sinclair 1991). Most of these examples consider cooccurrences of arbitrary graphemic words (perhaps using stop word lists to exclude some closed-class items) within segments (typically sentences) or within a collocational span (Sinclair 1991, 115f). Formally, we have $T_1 = T_2 = T$, corresponding to all graphemic tokens in the corpus, and for segment-based cooccurrences the data set will contain “reflexive” pair types (u, u) with $O_{12} = O_{21} = 0$. Of course, these “trivial” cooccurrences should be excluded from the statistical analysis.

Examples for the general case (where T_1 and T_2 are different token sets) are (i) cooccurrences of a noun and a verb in a sentence and (ii) a PP and a full verb in an automatically identified clause (inviting a re-interpretation of the PNV-FR data set as segment-based cooccurrences). A particularly interesting example of segment-based cooccurrences is provided by attempts to identify translation equivalents from their cooccurrences in aligned sentence pairs (Church and Gale 1991; Smadja *et al.* 1996). In this case, T_1 corresponds to the word tokens of the source language text, and T_2 to those of the target language text. The segments are given by the alignment pairs, and two tokens $s \in T_1, t \in T_2$ cooccur iff they appear in the same alignment pair.

Research in the Neo-Firthian tradition often concentrates on a certain number of tokens to the left and right of a given keyword (the *base* of the cooccurrences), which is referred to as a *collocational span* (Sinclair 1991, 175). In this situation, we have $T_1 = T_2 = T$ and the distance $d(s, t)$ of two tokens $s, t \in T$ can be defined as the number of intervening tokens plus one. Given a collocational span L , the local window $W(t)$ around a token t contains L tokens to either side of t , but not t itself. Thus, $|W(t)| = 2L$. The window for a type v is the union of its local windows. If these do not overlap, i.e. all instances of v are at least $2L$ tokens apart, the window size can directly be computed: $|W(v)| = 2L \cdot f(v)$, and the size of the complement is $|CW(v)| = N - (2L + 1)f(v)$ (recall that the instances of v have to be excluded from the complement as well). However, when this cannot be guaranteed, there is no choice but to determine the window $W(v)$ explicitly and count the number of tokens, excluding all instances of v from the count. Berry-Rogghe (1973) effectively uses such a model in her definition of a z-score like measure.

Some researchers add a lower distance threshold, thus hoping to find semantic relations rather than syntactic or lexically determined ones (e.g. Baroni *et al.* 2002). Terra and Clarke (2003) state that the optimal span size for the identification of synonyms is a distance between 10 and 30 words.

2.4.4 Discussion

Testing the model assumptions

This section gives a brief summary of the assumptions behind the statistical models for positional cooccurrence data in contrast to the model of Section 2.2.

An important advantage of the segment-based model is its insensitivity to clustering effects within the segments, which may correspond to sentences, paragraphs,

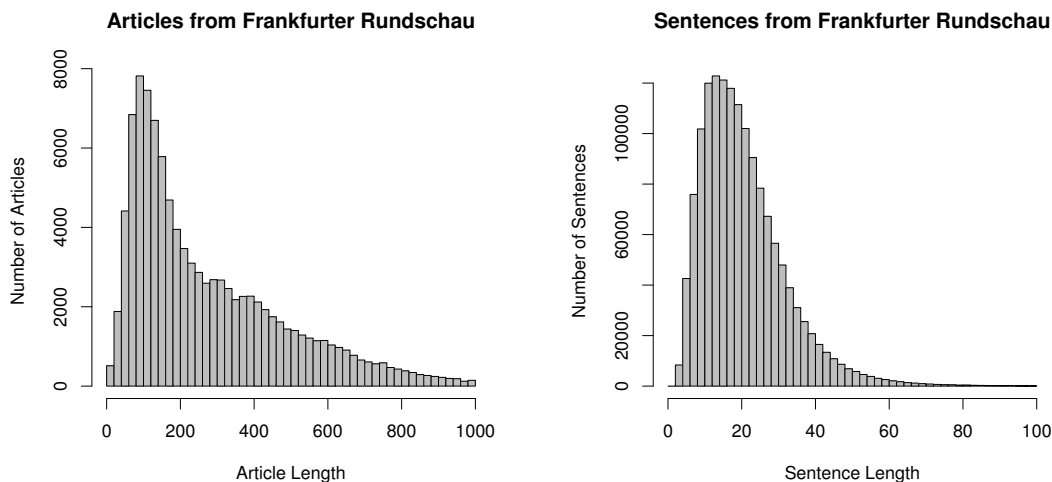


Figure 2.14: Distribution of the lengths of articles and sentences in the *Frankfurter Rundschau* corpus.

or entire documents. However, its homogeneity assumption (that the occurrence probabilities are the same for all segments) can be problematic when there is great variation in the size of individual segments. Figure 2.14 shows such variation for the lengths of articles and sentences in the *Frankfurter Rundschau* corpus. The dispersion test described in Section 2.3.2 can also be applied to segment-based cooccurrences (with each part containing exactly S segments), but it cannot detect inhomogeneity that is due to segment size (because segments of different sizes will usually be distributed randomly across the corpus).

The statistical model used for distance-based cooccurrences calls for more elaborate tests of randomness. For a given pair type (u, v) , the instances of u within the window $W(v)$ and those in the complement $CW(v)$ have to be tested separately, since the statistical model does not assume that the two distributions are identical. $W(v)$ is typically much smaller than its complement and the local windows may not be spread homogeneously across the source corpus. This suggests the following tests for non-randomness: (i) test the dispersion of the instances of u across the local windows that make up $W(v)$, using a dispersion test; (ii) test independence of the instances within each local window, e.g. with a runs test (although this kind of non-randomness usually has no influence on the contingency tables); (iii) test the distribution of instances of u across the complement of $W(v)$ with a standard dispersion test (using K equally sized chunks and ignoring the “holes” left by local windows). It is possible to pool the dispersion data for all pair types (u', v) that are based on the same window $W(v)$.

Extraction noise for segment-based cooccurrences can be discussed in terms of precision and recall for each pair type (u, v) . An additional source of errors here is the detection of segment boundaries, especially when segments are linguistically motivated units (such as sentences or clauses). It is more difficult to evaluate extraction quality for distance-based cooccurrences. However, most proponents of a distance-based approach use fully deterministic operational definitions of word tokens and windows in order to exclude noise altogether (again, Lehr (1996) provides the most striking example). Consequently, there is little point in an empirical study

of extraction errors.

Relational vs. positional cooccurrences

In general, the differences between relational and positional cooccurrences are so profound that the choice is largely determined by theoretical (linguistic) considerations. In some cases, however, the data extracted from a corpus can be interpreted either as relational or as positional cooccurrences. Consider the PNV-FR data set of PP-verb combinations, which were defined as relational cooccurrences. However, the extraction method described in Section 2.1.3 produces all possible combinations of PPs and full verbs that occur within the same segment (corresponding to a main or subordinate clause), which fits in better with the segment-based model. The resulting data set can thus be interpreted as relational cooccurrence data with a high proportion of noise (mostly false positives), taking the extraction method that was used as a fairly unsophisticated tool for identifying PP-verb relations. It can also be interpreted as segment-based data, extracting cooccurrences of PP-chunk tokens and verb tokens within clause segments. In the latter case, the marginal frequencies, which were computed from the cooccurrence frequencies by the procedure described in Section 2.1.2, have to be adjusted according to Section 2.4.1.

Chapter 3

Association Measures

3.1 An inventory of association measures

3.1.1 General remarks

An **association measure** is a formula that computes an **association score** from the frequency information in a pair type's contingency table. This score is intended as an indicator of how strong the association between the pair's components is, correcting for random effects (as predicted by the statistical model of Section 2.2). I use the **convention** that high association scores indicate strong association. Some of the published (and implemented) measures may be different, but their scores can easily be transformed to my convention (for instance, when the p -value computed by a statistical hypothesis test is used as an association score, its negative logarithm conforms with the convention, cf. Section 3.1.3).

The scores computed by an association measure can be interpreted in different ways: (i) They can be used directly to estimate the magnitude of the association between the components of a pair type. (ii) They can be used to obtain a ranking of the pair types in a data set. In this case, the absolute magnitude of the scores is irrelevant. (iii) They can also be used to rank pair types with a particular first or second component. Here, a comparison is made between contingency tables with fixed row (or column) sums only, and the relative scores of entirely different frequency signatures are irrelevant. Sections 1.2.1 and 1.2.2 give examples for all three applications. The interpretation of association scores has some influence on whether the logic behind a particular association measure seems appropriate, and on the relevant criteria for a comparison as in Section 3.4. In this chapter, I often make the tacit assumption that association scores are used for ranking a data set (ii). For some measures, the absolute magnitude of the scores can be given a meaningful interpretation (especially those listed in Sections 3.1.3 and 3.1.4). I do not go further into (iii), which is closely tied to a “directional” view of cooccurrences and casts an entirely different light on the properties of association measures.

There is a general division into **one-sided** and **two-sided** measures, depending on whether they distinguish between positive and negative association (one-sided measures) or not (two-sided measures).¹ Recall that positive association indicates that

¹The terms *one-sided* and *two-sided* are taken from the theory of statistical hypothesis tests. In Sections 3.1.3 and 3.1.4, one-sided hypothesis tests result in one-sided association measures, and vice

the components of a pair type cooccur *more* often than if they were independent, and negative association that they cooccur *less* often. Also recall that there is no “standard” way of measuring association strength, and Section 2.2.5 lists several possibilities that give different results. All the statistically reasonable ones among them (i.e. those that assume a well-defined value in the case of independence) should lead to the same distinction between positive and negative association, though.

For one-sided association measures, high scores indicate strong *positive* association. Low scores (including negative scores) indicate that there is no evidence for a positive association (which could mean either that the components are independent or that there is negative association). For two-sided association measures, on the other hand, high scores indicate any kind of strong association (*positive or negative*), whereas low scores indicate near-independence, regardless of their sign. A two-sided measure whose scores are always non-negative can easily be converted into a one-sided measure: for any pair type with negative association (as indicated e.g. by the maximum-likelihood estimate for the μ -value), multiply the association score by -1 . Thus, positive scores indicate positive association and negative scores indicate negative association. The absolute value of the score depends on the association strength, with values close to 0 indicating near-independence.²

The following sections present a wide range of association measures that have been suggested and used by various researchers. Wherever possible, a measure’s theoretical background and the derivation of its equation are explained, and key references are given. Most association measures compare the observed frequencies O_{ij} against the frequencies E_{ij} expected under the null hypothesis H'_0 (cf. Section 2.2.3) in some way. I formulate the equations of all measures in terms of O_{ij} and E_{ij} . The complete frequency information needed for their implementation is thus summarised in the two tables shown in Figure 3.1.

	$V = v$	$V \neq v$
$U = u$	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$
$U \neq u$	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$

	$V = v$	$V \neq v$	
$U = u$	O_{11}	O_{12}	$= R_1$
$U \neq u$	O_{21}	O_{22}	$= R_2$
	$= C_1$	$= C_2$	$= N$

Figure 3.1: Expected vs. observed frequencies.

There are four major approaches to measuring association:

1. The first approach aims to quantify the amount of evidence that the observed sample provides against the non-association of a given pair type (i.e. against

versa.

²For small absolute values, the distinction between positive and negative association is unreliable because of random effects. Such pair types should be interpreted as “roughly independent”, with no clear evidence for either positive or negative association.

either one of the null hypotheses introduced in Section 2.2.3 or the homogeneity variants from Section 2.2.4). Since most of these association measures are derived from statistical hypothesis tests, I refer to them as the **significance of association** group. Measures of the significance of association can be further subdivided into **likelihood measures** (which compute the *probability* of the observed contingency table, Section 3.1.2), **exact statistical hypothesis tests** (which compute the *significance* or *p-value* of the observed data, Section 3.1.3), and **asymptotic statistical hypothesis tests** (which compute a *test statistic* that can be translated into an approximate *p-value*, Section 3.1.4).

2. The second approach estimates one of the coefficients of association strength introduced in Section 2.2.5 from the observed data. I refer to such measures as the **degree of association** group. Note that the computed association score is an estimate for the *effect size*, while the significance of association group is more concerned with the amount of evidence provided by the sample. Measures of association strength are divided into **point estimates** (usually maximum-likelihood estimates, Section 3.1.5) and **conservative estimates** (based on confidence intervals obtained from a hypothesis test, Section 3.1.6).
3. The third approach is based on the information-theoretic concepts of *entropy*, *cross-entropy*, and *mutual information*. It is therefore referred to as the **information theory** group (Section 3.1.7). Intuitively, association measures from this group quantify the non-homogeneity of the observed contingency table, compared to the contingency table of expected frequencies. Alternatively, mutual information can be understood as a coefficient of association strength (which is 0 iff a pair's components are independent), and the corresponding association measures are point estimates of this coefficient.
4. The final approach encompasses a considerable number of **heuristic** formulae (Section 3.1.8). Such association measures combine sample values that are considered to be good indicators of (positive) association in various ways. They can also be modified versions of measures from other groups or combinations of such measures. It is sometimes not entirely clear whether a particular association measure should be classified as a heuristic or belongs to one of the other three groups. The most prominent example is t-score (Section 3.1.4): although derived from an asymptotic hypothesis test (Student's *t* test), its applicability to cooccurrence frequencies is highly questionable.

A comprehensive and regularly updated list of association measures is available online at the URL

<http://www.collocations.de/AM/>

3.1.2 Likelihood measures

Likelihood measures compute the probability of the observed contingency table (or part of it) under a null hypothesis of non-association (usually the point independence hypothesis $H'_0: \pi = p_1 \cdot p_2$, where p_i is a maximum-likelihood estimate for the unknown parameter π_i). The equations below compute a probability $lv \in (0, 1]$ (lv

stands for *likelihood value*). A small value of lv means that the observed data are unlikely given the null hypothesis, indicating strong evidence for association of the pair type in question. It is usually more convenient to report the negative decadic logarithm $-\log_{10} lv \in (0, \infty)$, which adheres to the convention that high scores should indicate strong association. All likelihood measures are two-sided. Their scores can be multiplied with -1 for negatively associated pairs to obtain a one-sided measure.

The most obvious likelihood measure is **multinomial likelihood**, which computes the probability of the observed contingency table under H'_0 (using the sampling distribution under H'_0 from Section 2.2.3).

$$\text{multinomial-likelihood} = \frac{N!}{N^N} \cdot \frac{(E_{11})^{O_{11}} \cdot (E_{12})^{O_{12}} \cdot (E_{21})^{O_{21}} \cdot (E_{22})^{O_{22}}}{O_{11}! \cdot O_{12}! \cdot O_{21}! \cdot O_{22}!}$$

It is also possible to use the general null hypothesis H_0 and avoid maximum-likelihood estimates for π_1 and π_2 by conditioning on the observed marginal frequencies (cf. Section 2.2.4), which leads to the **hypergeometric likelihood** measure.

$$\text{hypergeometric-likelihood} = \frac{\binom{C_1}{O_{11}} \cdot \binom{C_2}{R_1 - O_{11}}}{\binom{N}{R_1}}$$

These two association measures compute the likelihood of obtaining exactly the observed contingency table, provided that H'_0 (or H_0 is true). However, the top left cell (O_{11}) provides the most immediate evidence for an association between a pair's components. Therefore, it makes sense to compute the total probability of all contingency tables with $X_{11} = O_{11}$ under H'_0 , regardless of their row and column sums (which only plays a role in estimating E_{11}). This reasoning leads to the **binomial likelihood** measure, corresponding to the sampling distribution of X_{11} .

$$\text{binomial-likelihood} = \binom{N}{O_{11}} \left(\frac{E_{11}}{N}\right)^{O_{11}} \left(1 - \frac{E_{11}}{N}\right)^{N-O_{11}}$$

For computational efficiency, it is advantageous to replace the binomial probabilities with a Poisson distribution. The approximation of the **Poisson likelihood** to binomial likelihood is excellent for small values of E_{11}/N and O_{11} , which is usually the case with cooccurrence data. Note that Poisson-likelihood is the exact distribution of X_{11} when independent Poisson sampling is assumed (cf. Section 2.2.2).

$$\text{Poisson-likelihood} = e^{-E_{11}} \frac{(E_{11})^{O_{11}}}{O_{11}!}$$

Extending the approach of (Quasthoff 1998, 9), Quasthoff and Wolff (2002) take the negative logarithm of Poisson-likelihood (corresponding to the $-\log_{10} lv$ convention) and approximate the factorial with Stirling's formula (Weisstein 1999, *s.v. Stirling's Approximation*) to obtain the **Poisson-Stirling** measure.

$$\text{Poisson-Stirling}_{\log} = O_{11} \cdot (\log O_{11} - \log E_{11} - 1)$$

3.1.3 Exact hypothesis tests

A problem of the likelihood approach is that the computed probabilities may become quite small under certain circumstances, even when the null hypothesis H_0 is satisfied. Taking Poisson-likelihood as an example, the likelihood of $O_{11} = 1$ for an expected cooccurrence frequency of $E_{11} = 1$ is 0.3678794. However, for $O_{11} = E_{11} = 1\,000$, the likelihood is only 0.01261461 (which is similar to the likelihood of $O_{11} = 4$ for $E_{11} = 1$). Thus, a contingency table with $O_{11} = E_{11} = 1\,000$ seems to provide *more* evidence against H_0 than one with $O_{11} = 4$ and $E_{11} = 1$, even though its observed and expected frequency are equal.

(Exact) **statistical hypothesis tests** solve this problem by controlling the probability of a so-called **type I error**, which is an unjustified rejection of the null hypothesis H_0 . They do so by summing over all contingency tables that provide as much evidence against H_0 as the observed table, or even more. The resulting ***p*-value** (*pv* for short) can be interpreted as the amount of evidence provided by the observed data *against* the null hypothesis: the smaller it is, the less likely it is that a given pair type satisfying H_0 would lead to a similar or more “extreme” contingency table purely by chance. The *p*-value is a probability in the range $pv \in (0, 1]$, with smaller values indicating more evidence for a (usually positive) association. It is often convenient to use the negative decadic logarithm $-\log_{10} pv \in (0, \infty)$ instead, which adheres to the convention that high scores should indicate strong association.

A crucial problem in the design of exact hypothesis tests is the question how to compare different contingency tables and identify the ones that are more “extreme” than the observed table (and whose probabilities should be added up to obtain the *p*-value). A general solution exists only in simple cases with a single free parameter (which includes all likelihood measures described in Section 3.1.2 except for multinomial-likelihood). For the same reason, exact hypothesis tests are usually one-sided, summing over contingency tables that provide more evidence for *positive* association.

The only free parameter of the binomial and Poisson likelihood functions is the upper left corner of the contingency table, i.e. X_{11} in the sampling distribution. The greater its value, the more evidence there is for a positive association (because the observed cooccurrence frequency is higher than expected). Summation over the likelihood values for all possible values of $X_{11} \geq O_{11}$ leads to the **binomial** and **Poisson** tests and the corresponding association measures below.

$$\text{binomial} = \sum_{k=O_{11}}^N \binom{N}{k} \left(\frac{E_{11}}{N} \right)^k \left(1 - \frac{E_{11}}{N} \right)^{N-k}$$

$$\text{Poisson} = \sum_{k=O_{11}}^{\infty} e^{-E_{11}} \frac{(E_{11})^k}{k!}$$

The Poisson measure was suggested in 1970 by Robert Daley (published in Sinclair *et al.* 2004, 39). Neither the Poisson nor the binomial test are completely exact tests, because they depend on H'_0 and hence on the sample estimate for E_{11} .

A truly exact test can be obtained by the same procedure from the hypergeometric likelihood function, which depends on H_0 only. It is known as **Fisher’s exact test** (see

Agresti 1990, 60–66),³ and is even more computationally expensive than the other two tests.

$$\text{Fisher} = \sum_{k=O_{11}}^{\min\{R_1, C_1\}} \frac{\binom{C_1}{k} \cdot \binom{C_2}{R_1 - k}}{\binom{N}{R_1}}$$

The first application of the Fisher measure to cooccurrence data was reported by Justeson and Katz (1991), but without reference to Fisher’s test. It was later popularised by Pedersen (1996) as an alternative to the log-likelihood measure (cf. Section 3.1.4) that does not have to rely on approximations.

3.1.4 Asymptotic hypothesis tests

Asymptotic hypothesis tests are usually based on normal distributions and avoid the numerical difficulties of exact tests.⁴ An elementary example is the **z-score**, which simplifies the computation of (the *p*-value of) the binomial measure by approximating the discrete binomial distribution with a continuous normal distribution. When H'_0 holds and E_{11} is sufficiently large, the binomial distribution of X_{11} is approximately normal with mean E_{11} and standard deviation close to $\sqrt{E_{11}}$. Hence, the value $(X_{11} - E_{11}) / \sqrt{E_{11}}$ follows a standard normal distribution (Weisstein 1999, *s.v. Normal Distribution*). Setting $X_{11} = O_{11}$, we obtain

$$\text{z-score} = \frac{O_{11} - E_{11}}{\sqrt{E_{11}}}$$

The higher the z-score value, the more evidence there is for positive association. Using the theoretical distribution function of the normal distribution, it can be converted into a *p*-value (as an approximation of the *p*-value computed by the binomial measure). The z-score measure was used by Dennis (1965, 69) to identify “significant word-pair combinations” and later by Berry-Rogghe (1973, 104).

More generally, **asymptotic hypothesis tests** compute a **test statistic**, which indicates how far the observed contingency table deviates from what would be expected under the null hypothesis. The definition of the test statistic plays a crucial role in the design of such a test because it determines an ordering of all possible contingency tables, according to how much evidence they provide against H_0 . A *p*-value is then obtained by summation over all contingency tables that are more “extreme” than the observed one in this ordering. The expensive computation of this exact *p*-value is greatly simplified when the **limiting distribution** of the test statistic under H_0 is known for large samples. Its distribution function can then be used to transform the test statistic into the corresponding *p*-value. Since this transformation is monotonic, the test statistic itself can also be used as an association measure, and this is done in most cases.⁵

³This test was first described by Fisher (1934). In a later publication, a derivation of the test procedure is given for a concrete numerical example (Fisher 1935, 48–50).

⁴Historically, the asymptotic tests pre-date exact tests, having been developed at a time when the normal distribution was at the heart of all branches of statistics.

⁵It is still interesting to compute the *p*-values, though, in order to allow a comparison with the association scores computed by exact hypothesis tests (see Section 3.4).

The standard test for independence of the rows and columns in a contingency table, at least in the field of mathematical statistics, is **Pearson's chi-squared test**, based on H'_0 (DeGroot and Schervish 2002, 552). Its test statistic is often denoted by the symbol X^2 (cf. Pedersen 1996), and is a two-sided association measure. The limiting distribution of X^2 is a χ^2 -distribution with one degree of freedom ($df = 1$), which can be used to translate association scores into p -values. A one-sided measure can be obtained by the general method described in Section 3.1.1. However, the p -values according to the χ^2 distribution then have to be divided by 2 in order to allow direct comparison with one-sided tests.⁶

$$\text{chi-squared}_i = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Another version of the test is based on the sampling distribution for fixed column sums and the corresponding null hypothesis $H'_{0, \text{hom}}$ (cf. Section 2.2.4), known as Pearson's chi-squared test of homogeneity (DeGroot and Schervish 2002, 557f). For a 2×2 table, this version of the test is often written in the form

$$\text{chi-squared}_h = \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{R_1R_2C_1C_2}$$

For the comparison with other association measures in Section 3.4, the following "normal" form is particularly useful:

$$\text{chi-squared} = \frac{N(O_{11} - E_{11})^2}{E_{11}E_{22}}$$

Although this is not at all obvious, all three formulae are equivalent (Lemma A.5).

Surprisingly, chi-squared has not very often been used for cooccurrence analysis so far, although it is mentioned by Manning and Schütze (1999). Edmundson (1965) suggested a "correlation coefficient for events" $R(A, B)$ that is identical to chi-squared when these events are interpreted as occurrences of the pair type components u and v , i.e. $A = \{U = u\}$ and $B = \{V = v\}$ (Edmundson 1965, 44). Dennis (1965, 69) also considered the use of chi-squared, but finally chose z-score. Many years later, Church and Gale (1991) applied chi-squared to the extraction of translation equivalents from parallel text.

It is well known that many asymptotic hypothesis tests give a poor approximation of their limiting distribution when one or more of the entries in the contingency table are small numbers. The main reason for this effect lies in the approximation of the discrete binomial distribution by a continuous normal distribution, as exemplified in Figure 3.2. The graphs show a binomial distribution X with parameters $N = 15$ and $p = 1/3$ and its approximation by a normal distribution Y with the same expectation and variance (corresponding to parameters $\mu = 5$ and $\sigma^2 = 10/3$). The area of the coloured bars corresponds to the exact binomial probability $\Pr(X \geq 7)$ of 7 or more successes out of $N = 15$ trials, and the shaded area under the normal curve to its normal approximation. The left panel shows the normal probability for $\Pr(X \geq 7)$.

⁶The same strategy is used in mathematical statistics to perform one-sided tests based on Pearson's X^2 statistic (or similar two-sided test statistics).

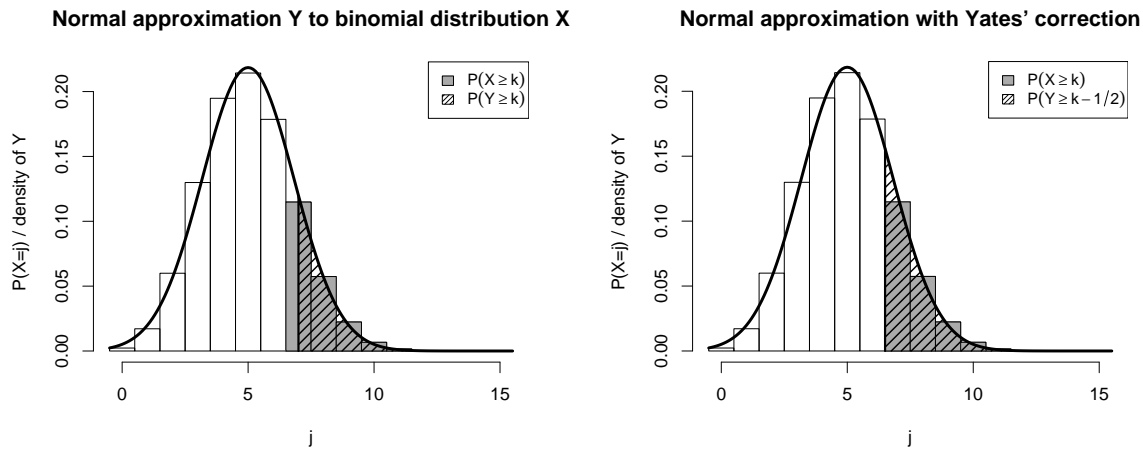


Figure 3.2: Yates' continuity correction.

It is obvious that the probability for $\Pr(X \geq 6.5)$ in the right panel is a much better approximation of the correct binomial probability (see Cox (1970) for an analytical derivation of this property and the offset of $1/2$).

Thus, **Yates' continuity correction** (Yates 1934) adjusts observed frequencies by $1/2$ towards the expected values. It is applied to asymptotic hypothesis tests that involve a normal approximation. When used for association measures, the following correction rules are applied to the observed frequencies O_{ij} in the contingency table after the expected frequencies E_{ij} have been determined.

$$\begin{aligned} O'_{ij} &:= O_{ij} - 1/2 & \text{if } O_{ij} > E_{ij} \\ O'_{ij} &:= O_{ij} + 1/2 & \text{if } O_{ij} < E_{ij} \end{aligned} \quad (3.1)$$

The application of Yates' continuity correction to the chi-squared test is not universally accepted. Statisticians differ as to when it should be applied and whether it is valid at all (e.g. Motulsky 1995, Ch. 37). The reason for this dispute seems to be that the chi-squared test with Yates' correction often gives a good approximation to the p -values computed by Fisher's test (Yates 1934), which some researchers consider to be too conservative.

The homogeneity version chi-squared_{*h*} has a special form that incorporates Yates' correction so that the observed frequencies do not have to be modified (Lemma A.6). This form is often used in applications.

$$\text{chi-squared}_{h,\text{corr}} = \frac{N(|O_{11}O_{22} - O_{12}O_{21}| - N/2)^2}{R_1 R_2 C_1 C_2}$$

Another asymptotic test is **Student's *t*-test**, whose test statistic has become known as the *t*-score measure (Church *et al.* 1991, Sec. 2.2). It is a one-sided test and has Student's *t* distribution with $df \approx \infty$ as its limiting distribution.

$$\text{t-score} = \frac{O_{11} - E_{11}}{\sqrt{O_{11}}}$$

From a theoretical perspective, Student's test is not applicable to cooccurrence frequency data. It is designed for a sample of n independent and identically distributed

normal variates. The null hypothesis is a condition on the mean of the normal distribution, while its variance is estimated from the sample. Under a null hypothesis that stipulates a specific value for the mean of the distribution, the test statistic has a t -distribution with $n - 1$ degrees of freedom. There are two ways in which such a test might be applied to the comparison of O_{11} and E_{11} : (a) the “sample” consists of a single item X_{11} (i.e. $n = 1$), which has an approximately normal distribution; or (b) the “sample” consists of N indicator variables, one for each pair token (i.e. $n = N$). In case (a), it is impossible to estimate the variance from a sample of size one. In case (b), which Manning and Schütze (1999, 164) refer to as the “standard way” of extending the t -test for use with frequency data, the sample variance can be estimated and corresponds to the value used by (Church *et al.* 1991) (O_{11} in the denominator of the t -score equation is a good approximation of the correct sample variance). However, the individual random variates of this “sample” are indicator variables with only two possible values 0 and 1 (and they are usually highly skewed towards 0). The normal approximation required by the t -test is highly questionable for such binary variables. In particular, the test assumes that the mean and variance of the distribution are independent, which is not the case for indicator variables. It may thus be more appropriate to interpret t -score as a heuristic variant of z -score that avoids the characteristic overestimation bias of the latter.

An entirely different class of test statistics are **likelihood ratio tests**, which are based on the ratio between the maximum likelihood of the observed data under H_0 and its unconstrained maximum likelihood (without making any assumptions about the population parameters).⁷ When this method is applied to the multinomial distribution (2.5) of contingency tables, we obtain the general form of the log-likelihood measure (see also Agresti 1990, 48).

$$\text{log-likelihood}_{\text{ratio}} = -2 \log \frac{\max \Pr(\vec{X} = \vec{O} \mid N \wedge \pi = \pi_1 \cdot \pi_2)}{\max \Pr(\vec{X} = \vec{O} \mid N)}$$

Use of the natural logarithm and the factor -2 ensure that the limiting distribution of the likelihood ratio statistic, which is often denoted with the symbol G^2 , is a χ^2 -distribution with one degree of freedom (Wilks 1935). A closed expression for this ratio can be derived from the multinomial sampling distribution (Lemma A.7).

$$\text{log-likelihood} = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

The use of log-likelihood as an association measure was originally suggested by Dunning (1993). He derived it from the sampling distribution for fixed column sums and the corresponding null hypothesis $H_{0, \text{hom}}$, resulting in the following rather unwieldy, but fully equivalent formula (Dunning 1993, 67):

$$\begin{aligned} \text{log-likelihood}_{\text{Dunning}} &= -2 \log \frac{L(O_{11}, C_1, r) \cdot L(O_{12}, C_2, r)}{L(O_{11}, C_1, r_1) \cdot L(O_{12}, C_2, r_2)} \\ L(k, n, r) &= r^k (1 - r)^{n-k} \\ r &= \frac{R_1}{N}, \quad r_1 = \frac{O_{11}}{C_1}, \quad r_2 = \frac{O_{12}}{C_2} \end{aligned}$$

⁷Note the use of H_0 rather than H_0' . Likelihood ratio tests do not depend on a point null hypothesis because they compute the maximal likelihood value consistent with H_0 .

Like chi-squared, the log-likelihood measure is two-sided. Dunning argues at length that G^2 approximates the limiting χ^2 distribution much better than X^2 for the highly skewed contingency tables (with N large and O_{11} small) that are typical of cooccurrence data (Dunning 1993, 1998).

3.1.5 Point estimates of association strength

Although measures of the significance of association have been widely and successfully applied, they have one important drawback: a high association score, corresponding to a large amount of evidence against independence, can result either from a high degree of association between the components of a pair type or from a large amount of evidence being available (i.e. a high cooccurrence frequency O_{11}). The association measures presented in preceding sections cannot distinguish between these two effects, and are thus often biased in favour of high-frequency pairs.⁸ The measures of association strength introduced in this section provide a different approach that focuses on the degree of association. They are point estimates (maximum-likelihood estimates) of the coefficients of association strength described in Section 2.2.5 (see Table 2.2 for an overview). In Section 3.1.6 they will be refined in order to take the amount of evidence supporting the estimated association strength into account and avoid overestimation for low-frequency pairs.

A problem with the maximum-likelihood estimates for coefficients of association strength is that the value of any such coefficient does not fully determine the sampling distribution. Therefore, any hypothesised value is consistent with an entire likelihood range for the observed data. The most sensible approach is to use the highest likelihood value within the range (which usually has a lower bound close to zero) as a criterion. Thus, the estimate for a coefficient of association strength is computed from the direct estimates of the population parameters τ_{ij} , for which the total likelihood of the observed contingency table assumes its global maximum.⁹ The maximum-likelihood estimates below simply replace the population parameters with the corresponding sample estimates (see Eq. (2.8)):

$$\begin{aligned}\pi &\approx p = \frac{O_{11}}{N} \\ \pi_1 &\approx p_1 = \frac{R_1}{N} = \frac{O_{11} + O_{12}}{N} \\ \pi_2 &\approx p_2 = \frac{C_1}{N} = \frac{O_{11} + O_{21}}{N}\end{aligned}$$

and likewise

$$\tau_{ij} \approx \frac{O_{ij}}{N}.$$

Note that this approach yields *unconditional* maximum-likelihood estimates. The arguments that are often presented in favour of Fisher's exact test (e.g. Yates 1984)

⁸This is not the case for the chi-squared measure. However, Dunning (1993) found the reason to be a poor approximation of the limiting distribution, which causes chi-squared to overestimate the significance of low-frequency pairs.

⁹Note that a Bayesian approach, assuming a prior distribution for the probability parameters, would arrive at different estimates.

suggest that a *conditional* estimate – where the marginal frequencies are fixed to the observed values – may be more useful. However, it is very difficult and expensive to compute such conditional estimates, which are in fact different from the unconditional maximum-likelihood estimates (Agresti 1992, 135).

The **MI** measure is an estimate for the logarithm of the μ -value, which can be interpreted as (a maximum-likelihood estimate for) point-wise **mutual information** (Church and Hanks 1990):

$$\text{MI} = \log \frac{O_{11}}{E_{11}}$$

The **relative risk** measure is an estimate for the logarithm of the ρ coefficient (but to my knowledge has never been used as an association measure):

$$\text{relative-risk} = \log \frac{O_{11}C_2}{O_{12}C_1}$$

The maximum-likelihood estimate for the **difference of proportions** κ_{μ} was used as a test statistic by Liddell (1976), but has not been applied to language data so far:

$$\text{Liddell} = \frac{N(O_{11} - E_{11})}{C_1C_2} = \frac{O_{11}O_{22} - O_{12}O_{21}}{C_1C_2}$$

The **Dice coefficient**, a point estimate of κ_{Dice} , is interesting because, as Smadja *et al.* (1996) point out, it identifies pairs with a particularly high degree of lexical cohesion (i.e. those with nearly total association). The same holds for the equivalent κ_{Jaccard} measure as well as the estimates of κ_{gmean} and κ_{min} , and κ_{Jaccard} below (cf. Table 2.2). Dias *et al.* (1999) introduced an n -gram generalisation of the Dice coefficient under the name **mutual expectation**.

$$\text{Dice} = \frac{2O_{11}}{R_1 + C_1}$$

The **Jaccard coefficient** is mentioned by Dunning (1998, 53), but is merely a monotonic transformation of Dice (see Section 2.2.5).

$$\text{Jaccard} = \frac{O_{11}}{O_{11} + O_{12} + O_{21}}$$

The **geometric mean** measure is a point estimate of the κ_{gmean} coefficient. Interestingly, gmean is the square root of the heuristic MI^2 measure from Section 3.1.8 (scaled with a constant factor).

$$\text{gmean} = \frac{O_{11}}{\sqrt{R_1C_1}} = \frac{O_{11}}{\sqrt{NE_{11}}}$$

Another association measure in this group that has not found widespread use is **minimum sensitivity (MS)**, a point estimate for the κ_{min} coefficient (Pedersen and Bruce 1996). In a recent experiment, however, it has unexpectedly performed better than all the established measures in a collocation extraction task (cf. Section 5.3).

$$\text{MS} = \min \left\{ \frac{O_{11}}{R_1}, \frac{O_{11}}{C_1} \right\}$$

The (logarithmic) **odds ratio**, an estimate of $\log \theta$, is particularly interesting: Blaheta and Johnson (2001) suggest the use of log-linear models (see Agresti 1990) as a generalisation of traditional association measures to n -grams. Their general measure of association is based on the n -way interaction term λ and its asymptotic standard error σ (Blaheta and Johnson 2001, 56). The authors note that in the case of bigrams (i.e. $n = 2$), λ is the logarithmic odds ratio and σ its asymptotic standard error (Hollander and Wolfe 1999).

$$\text{odds-ratio} = \log \frac{O_{11}O_{22}}{O_{12}O_{21}}$$

A problem of the odds-ratio measure is that it assumes an infinite value whenever any of the observed frequencies is zero ($-\infty$ for $O_{11} = 0$ or $O_{22} = 0$, $+\infty$ for $O_{12} = 0$ or $O_{21} = 0$). Many applications use a “discounted” version of the log odds ratio, where $\frac{1}{2}$ is added to each O_{ij} in order to avoid such infinite values. This adjusted estimator was shown to be “well-behaved” in various studies (see Agresti 1990, 54).

$$\text{odds-ratio}_{\text{disc}} = \log \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})}$$

In addition to its computational advantages, $\text{odds-ratio}_{\text{disc}}$ allows a distinction between total determination (where either $O_{12} = 0$ or $O_{21} = 0$) and total association (where $O_{12} = O_{21} = 0$), assuming a larger value in the latter case. However, the interpretation of the adjusted values is not as clear as in the case of the unmodified odds ratio (cf. Table 2.2).

Most of the coefficients listed in this group are already mentioned by Kuhns (1965). However, his equations replace O_{11} by the difference $O_{11} - E_{11}$, arguing that “the excess of x over its independence value is what will interest us” (Kuhns 1965, 34). In most cases, this heuristic modification is problematic for the interpretation of the coefficients as maximum-likelihood estimates for theoretical probabilities and coefficients of association strength (Section 2.2.5). In particular, Kuhns presents a variant of the MS measure under the name “rectangular distance” (Kuhns 1965, 35).

3.1.6 Conservative estimates of association strength

A serious problem of point estimates is that they are subject to the full random variation of the sample frequencies, and are therefore unreliable for low frequency data. In particular, O_{11} / N will often overestimate π drastically, which is responsible for the poor evaluation results of MI in Evert and Krenn (2001) and similar studies.¹⁰ The exact hypothesis tests in Section 3.1.3 (and some well-behaved asymptotic tests such as log-likelihood) are much less prone to overestimating low-frequency data because they explicitly take the random variation of the observed frequencies into account.

In an unpublished report, Johnson (2001) suggested the use of interval estimates from exact hypothesis tests to avoid inflated values for the coefficients of association strength. The underlying idea is that such conservative measures should correct for random variation and avoid overestimation in the same way as the exact hypothesis

¹⁰In fact, the highest MI scores are always assigned to pair types with $O_{11} = R_1 = C_1 = 1$, similar to what Dunning (1993) found for the chi-squared measure.

tests do this for H_0 . A simplified version of Johnson's method, using the logarithmic odds ratio and its asymptotic standard error (in a log-linear model), was presented by Blaheta and Johnson (2001).

Generally speaking, interval estimates replace the single point estimate for a population characteristic with the set of all possible values of the characteristic that are consistent with the observed data (according to an appropriate statistical hypothesis test). In many cases, this set of values will take the form of a connected interval, the **confidence interval** for the population characteristic. Any values outside this interval can be rejected with the confidence level chosen for the hypothesis test, so that the true value of the population characteristic should be somewhere within the confidence interval. In Johnson's application to association measures, the estimated population characteristic is some coefficient of association strength κ , and the lower bound of the confidence interval is used as a conservative estimate for κ .¹¹

In principle, any one of the hypothesis tests introduced in Sections 3.1.3 and 3.1.4 can be used to obtain confidence intervals and conservative estimates for a coefficient of association strength. One-sided and two-sided tests are equally applicable, and corrections may only become necessary when we want to compare conservative estimates from different tests (for the same coefficient). When the hypothesis test is applied, the null hypothesis of independence (or homogeneity) is replaced by a null hypothesis $H_{\kappa=x}$ that stipulates a particular value x for the coefficient κ . Depending on the test used, it may be necessary to reduce $H_{\kappa=x}$ to a point null hypothesis $H'_{\kappa=x}$. The confidence interval $I_{\kappa,\alpha}$ is the set of all values x for which the hypothesis test does *not* reject $H_{\kappa=x}$ at the chosen confidence level α .

$$I_{\kappa,\alpha} := \{x \mid H_{\kappa=x} \text{ is not rejected at confidence level } \alpha\}$$

The conservative estimate is then given by $\kappa_- := \min I_{\kappa,\alpha}$.¹²

The choice of hypothesis test depends largely on whether the sampling distribution under $H_{\kappa=x}$ can be easily obtained, and whether $I_{\kappa,\alpha}$ can be computed efficiently. As long as $I_{\kappa,\alpha}$ is guaranteed to be an uninterrupted interval, κ_- can be determined fairly quickly by a binary search algorithm.¹³ Another important choice concerns the confidence level α . In general statistics, commonly used levels of confidence are 95%, 99% and sometimes 99.9%, corresponding to significance levels $\alpha = .05$, $.01$ and $.001$, respectively. However, empirical results show that the significance of association computed by the measures from Sections 3.1.3 and 3.1.4 is typically much lower for most of the pair types in a data set (with a median $\leq 10^{-6}$, cf. Chapter 3.4). This suggests that a much higher confidence level, i.e. $\alpha \ll .05$, might be called for.

In his unpublished report, Johnson targeted the **μ -value**, referring to his new measure as a conservative version of MI. Applying a binomial test (corresponding to the binomial measure) with the point null hypothesis

$$H'_{\mu=x} := \mu = x \wedge \pi_1 = p_1 \wedge \pi_2 = p_2,$$

¹¹I assume here that high coefficient values indicate strong positive association, and that low (or negative) values indicate either non-association or negative association. Since we are only interested in positive association here, the lowest value consistent with the observed data is always the most conservative choice.

¹²In order to be mathematically precise, the infimum $\inf I_{\kappa,\alpha}$ should be used instead of the minimum $\min I_{\kappa,\alpha}$, since the set $I_{\kappa,\alpha}$ may not include its lower bound.

¹³The time complexity of the binary search should not exceed 100 times the complexity of a single application of the hypothesis test.

he obtained the conservative estimate μ_- . A numerically and analytically more tractable version of this conservative estimate substitutes a Poisson approximation for the binomial test. This leads to the following definition of the $\mathbf{MI}_{\text{conf}, \alpha}$ measure (with a free parameter α that has to be chosen manually):

$$\mathbf{MI}_{\text{conf}, \alpha} = \log \min \left\{ \mu > 0 \mid e^{-\mu E_{11}} \sum_{k=O_{11}}^{\infty} \frac{(\mu E_{11})^k}{k!} \geq \alpha \right\}$$

This definition has two drawbacks that may cause inaccuracies: (i) only information from a single cell in the contingency table is used by the hypothesis test; (ii) the maximum-likelihood estimates for π_1 and π_2 needed to compute the point null hypothesis $H'_{\mu=x}$. A variant of Fisher's exact test would provide an elegant solution for both problems. Unfortunately, as has been pointed out in Section 2.2.4, the underlying non-central hypergeometric distribution (2.20) for fixed row and column sums can be simplified to a manageable form *only* under the null hypothesis $H_0 : \mu = 1$. Johnson (2001) mentions in passing that he has computed exact confidence intervals for the logarithmic odds-ratio ($\log \theta$), but does not explain his implementation.

A two-step procedure can be used to compute approximate confidence intervals for any one of the coefficients described in Section 2.2.5. The first step applies the binomial or Poisson test (as above) to determine exact confidence intervals for the probability parameters π (from O_{11}), π_1 (from R_1) and π_2 (from C_1), as well as the ratios π/π_1 (from O_{11} and R_1) and π/π_2 (from O_{11} and C_1). In the second step, a likely range of values for a coefficient of association strength κ is computed by inserting the interval estimates into the equation of κ and taking the most "extreme" results.¹⁴ The resulting approximate confidence interval is usually larger than an exact interval would be (because it is determined from a "worst-case scenario"), leading to highly conservative estimates and association measures.

3.1.7 Measures from information theory

For an introduction to the key concepts of information theory, see e.g. Fano (1961). Generally speaking, the concept of **mutual information** expresses the "overlap" between two events or distributions.

Pointwise MI is used to compare two events A and B , and is simply the (logarithmic) ratio of their actual joint probability to the "expected" joint probability if A and B were independent: $\Pr(A \cap B) / \Pr(A) \Pr(B)$. When applied to the occurrences of types u and v , i.e. to the events $\{U = u\}$ and $\{V = v\}$, this definition leads to the μ -value $\mu = \pi / \pi_1 \pi_2$. Consequently, the maximum-likelihood estimate for pointwise MI is the MI measure from Section 3.1.5:

$$\mathbf{MI} = \log \frac{O_{11}}{E_{11}}$$

The overlap between two (binary) random variables is measured by **average MI**. Applied to the indicator variables $I_{[U=u]}$ and $I_{[V=v]}$, it results in the average-MI mea-

¹⁴Here, either the lower or the upper bound of the confidence intervals for π , π_1 and π_2 (or π/π_1 and π/π_2) may be used, depending on where in the equation they appear (e.g., in the numerator or denominator of a fraction). Note that confidence levels should be adjusted when more than one of the estimates is used (e.g., to $\sqrt[3]{.99}$ for 99% confidence when three parameters are needed to compute κ).

sure, which is again a maximum-likelihood estimate for the true mutual information between the indicator variables.¹⁵

$$\text{average-MI} = \sum_{ij} O_{ij} \cdot \log \frac{O_{ij}}{E_{ij}}$$

Fascinatingly, this equation is essentially identical to the log-likelihood measure, save for a factor of 2 (see also Dunning 1998, 75f). The average-MI value can also be interpreted as the cross-entropy between the observed and expected frequency tables, i.e. how accurately the expected frequencies predict the sample data. Dunning (1998) discusses MDL (minimum description length) approaches in this context.

Finally, the concept of mutual information can also be applied to the random variables U and V , in which case its value indicates how much information the components of word pairs provide about each other *in general*, i.e. averaged over all pair types in the population. The contribution of a given pair type $w = (u, v)$ to this “grand total” MI corresponds to the **local-MI** measure below.

$$\text{local-MI} = O_{11} \cdot \log \frac{O_{11}}{E_{11}}$$

Note that local-MI is nearly identical to the Poisson-Stirling measure (Section 3.1.2).

3.1.8 Heuristic, parametric and combined measures

The simplest possible association measure is the plain **cooccurrence frequency** of a pair type. Its use is motivated by the assumption that associated word pairs will in general occur more frequently than arbitrary combinations, i.e. as an operationalisation of Firth’s *recurrence* criterion (cf. Lehr 1996).

$$\text{frequency} = O_{11}$$

MI^2 is a heuristic variant of the MI measure that aims to increase the influence of the cooccurrence frequency in the numerator and avoid the characteristic overestimation effect for low-frequency pairs. This measure has some theoretical support because it is the square of the gmean measure.¹⁶

$$\text{MI}^2 = \log \frac{(O_{11})^2}{E_{11}}$$

Another variant, MI^3 , which uses a higher exponent in the numerator to boost the association scores of high-frequency pairs even further, represents a purely heuristic approach. Daille (1994) tested versions MI^k for $k = 2, \dots, 10$ and found $k = 3$ to

¹⁵Note how drastically the mutual information of the indicator variables $I_{[U=u]}$ and $I_{[V=v]}$ differs from that of the corresponding events $\{U = u\}$ and $\{V = v\}$. Likewise, the MI and average-MI measures have fundamentally different properties.

¹⁶More precisely, let g be the gmean association score for a given pair type (u, v) . Then the score h of the MI^2 measure is given by $h = \log(g^2) + \log N$, which is a monotonic transformation (for a fixed sample of size N).

give the best results in her application, noting that it is “un bon compromis entre ne retenir que les événements rares et trop les négliger” (Daille 1994, 139).

$$MI^3 = \log \frac{(O_{11})^3}{E_{11}}$$

Daille’s MI^k is a simple example of a **parametric association measure**. The value of the parameter k can be chosen freely (in principle, any $k > 0$ is possible) in order to modify the properties of the measure. In this way, it may be possible to “tune” such parametric measures to the needs of specific applications. It should be obvious now that $MI_{\text{conf}, \alpha}$ and other conservative estimates are also parametric measures, because a significance level α for the confidence interval has to be chosen more or less arbitrarily. Larger values of α will lead to a more conservative measure, especially for low-frequency data.

One motivation for using estimates of association strength is that the null hypothesis of independence is linguistically implausible: hardly any pair of words will cooccur in a completely random fashion. This means that hypothesis tests will always reject H_0 when enough data are available, i.e. for the higher-frequency pair types. It also leads to extremely small p -values far below customary significance levels. Another way around these problems is to use a more realistic null hypothesis, e.g. $H_0 : \mu \leq 10$ rather than $H_0 : \mu = 1$. This **modified null hypothesis** specifies an upper bound on the amount of “glue” between word types u and v that can be attributed to the general patterns of language (e.g., semantic compatibility of u and v), allowing cooccurrences to be up to ten times more frequent than expected by chance. The number 10 is entirely arbitrary here and may be replaced by any other value x , depending on the intuitions of the researcher.

For a one-sided hypothesis test, the modified hypothesis $H_0 : \mu \leq x$ is equivalent to $H_{\mu=x} : \mu = x$, because the highest p -value is obtained for $\mu = x$. Inserting this null hypothesis and the corresponding expected cooccurrence frequency $E[X_{11}] = xE_{11}$ into the Poisson equation yields the **modified Poisson** measure with one free parameter x :

$$\text{Poisson}_{\mu=x} = e^{-xE_{11}} \sum_{k=O_{11}}^{\infty} \frac{(xE_{11})^k}{k!}$$

Sometimes, it is desirable to combine the different properties of two or more association measures. Church *et al.* (1991) apply such a strategy for the purpose of collocation identification. They rank collocation candidates according to their association strength, measured by MI , but retain them only when there is also significant evidence for the association according to a t test (or, equivalently, the t -score measure). This procedure can be emulated by a **combined association measure** constructed from the MI and t -score formulae after suitable scaling. The general form of this measure is

$$MI/t\text{-score} = \min \left\{ h_1 \left(\log \frac{O_{11}}{E_{11}} \right), h_2 \left(\frac{O_{11} - E_{11}}{\sqrt{O_{11}}} \right) \right\}$$

where $h_1, h_2 : \mathbb{R} \rightarrow \mathbb{R}$ are monotonic scaling functions. When h_1 and h_2 are identity functions, the MI/t -score value is just the minimum of the two scores. This ensures that high-ranking pair types show strong association according to *both* measures. Linear functions can be used to transform the measures to a common scale when

necessary. In order to implement a significance filter in the sense of Church *et al.* (1991), h_1 is set to the identity $h_1(x) := x$ and h_2 to a threshold function:

$$h_2(x) := \begin{cases} 0 & x \geq \gamma \\ -\infty & x < \gamma \end{cases}$$

where γ is the t-score value that corresponds to the desired significance level α . With this definition, any pair types that do not pass the significance threshold are assigned the score $-\infty$ (in practice, $-\infty$ will be replaced by a large negative value). For all other pair types, the MI/t-score value is identical to the MI score. The same method can be used to “integrate” a frequency threshold into an association measure, replacing t-score by O_{11} in the equations above.

Many other combinations of association measures are possible, e.g. taking the maximum of the two scores or adding them up. Section 3.3 describes a general framework for the formal specification of parametric and combined measures.

3.2 Implementation

3.2.1 Know your numbers

With the enormous size of the source corpora that are available nowadays, numerical accuracy can become a serious problem for the implementation of association measures. A naive implementation will be based on standard (IEEE) double-precision **floating-point arithmetic**, which has an accuracy of about $\epsilon \approx 2 \cdot 10^{-16}$ and cannot represent numbers whose magnitude is less than $5 \cdot 10^{-324}$ (this is referred to as **underflow**). See Goldberg (1991) for an introduction to IEEE floating-point arithmetic.

For the measures listed in Sections 3.1.2 and 3.1.3, p -values easily become so small that they can only be represented in logarithmic form (e.g. following the $-\log_{10} p$ convention). In order to avoid underflow problems, however, it is necessary to carry out all computations with logarithmic values (otherwise, the p -value might underflow to 0, causing transformation into $-\log_{10} p$ to fail). A similar situation arises for asymptotic hypothesis tests (Section 3.1.4). Here, the test statistic (i.e. the association score) can safely be computed with standard arithmetic, but translation into a p -value requires an implementation of the theoretical distribution function that can directly return *logarithmic* values.

The most serious problem, however, is **catastrophic cancellation** (see Goldberg 1991) for measures based on exact hypothesis tests (including the conservative estimates from Section 3.1.6). As an example, consider the Poisson measure, whose definition involves an infinite sum for the probability $\Pr(X_{11} \geq O_{11})$. It is tempting to compute the complementary probability $\Pr(X_{11} < O_{11})$ instead and thus reduce the summation to a finite number of terms (especially since O_{11} will be quite small for most pair types):

$$\Pr(X_{11} \geq O_{11}) = 1 - \Pr(X_{11} < O_{11}) = 1 - \sum_{k=0}^{O_{11}-1} e^{-E_{11}} \frac{(E_{11})^k}{k!}. \quad (3.2)$$

For instance, Heyer *et al.* (2001) suggest an equation that is equivalent to (3.2). However, for highly associated pair types (especially high-frequency ones) the p -values become so small that this trick will lead to catastrophic cancellation. The summation yields the value $1 - pv$, close to 1. When it is subtracted from 1, many significant digits are lost. In particular, no p -values below the machine accuracy ϵ can be computed (and a naive implementation may even report the mathematical joke of a negative probability).¹⁷

Thus, the infinite summation is unavoidable (until the partial sum converges). Fortunately, the series converges geometrically once $k \geq E_{11}$. This can be seen by writing it in the form

$$\text{Poisson} = e^{-E_{11}} \sum_{k=O_{11}}^{\infty} t_k$$

with $t_k := (E_{11})^k / k!$. Because of

$$\frac{t_{k+1}}{t_k} = \frac{E_{11}}{k+1}, \quad (3.3)$$

the summation is then dominated by a geometric series $\sum_{j=0}^{\infty} q^j$ with $q < 1$. More precisely, we have

$$t_k \leq t_L \cdot q^{k-L}$$

with $q := E_{11} / (L + 1) < 1$ for any $L > E_{11}$. Since the expected cooccurrence frequency E_{11} is typically a comparatively small value, convergence is reached quickly.¹⁸ Eq. (3.3) can also be used to compute the terms t_k efficiently once $t_{O_{11}}$ has been determined. When direct computation of $-\log_{10} pv$ is necessary to avoid underflow, the recurrence relation becomes

$$\log t_{k+1} = \log t_k + \log E_{11} - \log(k+1).$$

Another approach is to express the probability $\Pr(X_{11} \geq O_{11})$ as an incomplete Gamma function (see Section A.4). Eq. (A.33) leads to this “closed form” of the Poisson measure:

$$\text{Poisson} = \frac{\gamma(O_{11}, E_{11})}{\Gamma(O_{11})}. \quad (3.4)$$

The incomplete Gamma function is provided by many numerical software libraries, and can be used for the implementation of the Poisson measure. (Internally, a power series similar to the infinite sum is computed until convergence.) In a similar way, the binomial measure can be computed accurately even for very small p -values as

$$\text{binomial} = \sum_{k=O_{11}}^N t_k$$

with

$$t_k := \binom{N}{k} \left(\frac{E_{11}}{N} \right)^k \left(1 - \frac{E_{11}}{N} \right)^{N-k}.$$

¹⁷Note that catastrophic cancellation already takes place at a magnitude of approx. 10^{-15} , long before underflow problems appear (approx. 10^{-320}).

¹⁸For instance, expected cooccurrence frequencies for high-frequency ($f \geq 30$) pair types in the PNV-FR data set range from $6 \cdot 10^{-4}$ to 483.5. More than half of the pair types have $E_{11} < 14$.

The ratio of consecutive terms

$$\frac{t_{k+1}}{t_k} = \frac{N - k}{k + 1} \cdot \frac{E_{11}}{N - E_{11}} \quad (3.5)$$

is also dominated by a geometric series for $k \geq E_{11}$. A “closed form” of the binomial measure can be derived from the incomplete Beta function, using Eq. (A.38):

$$\text{binomial} = \frac{B\left(\frac{E_{11}}{N}; O_{11}, N - O_{11} + 1\right)}{B(O_{11}, N - O_{11} + 1)} = I\left(\frac{E_{11}}{N}; O_{11}, N - O_{11} + 1\right). \quad (3.6)$$

Implementations of the incomplete Beta function in standard software libraries are not always reliable and the explicit summation above should be used to ensure accurate results.

The numerically most demanding association measure is Fisher. Although the summation is finite in this case, some implementations use complementary probabilities to speed up the calculation when O_{11} is comparatively small. One example is the otherwise excellent statistical environment R (R Development Core Team 2003). The Fisher probability for a contingency table with $O_{11} = 100$, $R_1 = C_1 = 1\,000$ and $N = 1\,000\,000$ (which is quite typical for cooccurrence data) can be computed with the R command

```
phyper(99, 1000, 999000, 1000, lower=FALSE).
```

At least for versions up to R-1.9.0 running under the Linux operating system, this will yield a negative p -value (the algorithm has been rewritten for R-2.0). Lemnitzer (1998, 87) is discouraged from using the Fisher measure by such difficulties, although he expects it to yield optimal results on theoretical grounds. The UCS toolkit (cf. Section 3.2.2), which relies heavily on R, adds a custom implementation of the direct summation to ensure that correct p -values are computed in such cases. As in the case of the Poisson measure, the series converges geometrically when $k > E_{11}$:

$$\text{Fisher} = \binom{N}{R_1}^{-1} \cdot \sum_{k=O_{11}}^{\min\{R_1, C_1\}} t_k$$

with

$$t_k := \binom{C_1}{k} \cdot \binom{C_2}{R_1 - k},$$

yielding the ratio

$$\frac{t_{k+1}}{t_k} = \frac{(C_1 - k)(R_1 - k)}{(k + 1)(C_2 - R_1 + k + 1)}. \quad (3.7)$$

Apart from the fundamental problems posed by the limited precision of floating-point arithmetic, it is important to be aware of corner cases that may lead to division by zero or other invalid operations. A particular example is the log-likelihood measure. When any one of the observed frequencies O_{ij} is zero, the corresponding term $\log \frac{O_{11}}{E_{11}}$ becomes undefined. However, this term can safely be dropped from the summation because $0 \cdot \log 0 = 0$ by continuous extension.

3.2.2 The UCS toolkit

This book is accompanied by a software, called the **UCS toolkit**,¹⁹ which provides implementations of all the association measures from Section 3.1, as well as most of the other mathematical procedures (including the dispersion test of Section 2.3.2, the frequency distribution models introduced in Chapter 4, and the evaluation methods described in Chapter 5). The toolkit is implemented in the Perl (Wall *et al.* 1996) and R (R Development Core Team 2003) languages, using the latter for most of the statistical functionality and for the graphical representation of data. Care was taken to ensure high accuracy of calculations, both for association measures (as detailed in Section 3.2.1) and for the implementation of the dispersion test (which requires high-precision integer arithmetic).

The UCS toolkit includes all the libraries, scripts and data sets that were used for the experiments and graphs in this book (with a small number of exceptions). The scripts are arranged by section and documented, allowing readers to replicate the main results of the thesis and encouraging them to continue the research with their own data. UCS is free software and can be downloaded from:

<http://www.collocations.de/phd.html>

Appendix B contains the complete software documentation of the UCS toolkit.

3.3 A geometric model of association measures

3.3.1 The coordinate space

For the statistical analysis and computation of association scores, each pair type w in a data set can be represented by its frequency signature (f, f_1, f_2, N) . Since N has the same value for all pair types in a data set, only the joint and marginal frequencies are relevant. I refer to the triple (f, f_1, f_2) as the **coordinates** of w . These coordinates describe a point x in the three-dimensional **coordinate space** $\mathcal{D} = (0, \infty)^3 = \{(x_1, x_2, x_3) \mid 0 < x_1, x_2, x_3 < \infty\}$ (more precisely, $x \in \mathbb{N}^3 \subseteq \mathcal{D}$). In this geometric view, a data set corresponds to a **point cloud** C in \mathcal{D} . There will usually be several pair types with identical frequency signatures, which are mapped to the same point in \mathcal{D} (for instance, there are 971 pair types with $f = f_1 = f_2 = 1$ in the AN-FR data set). For visualisation and similar purposes, such duplicates can be avoided by adding a small amount of random **jitter** to the coordinates (which are thus no longer integer values). Provided that the jitter is small enough, this will not affect the association scores in any substantial way. This device also allows a more elegant mathematical treatment, as we can now interpret the point cloud C as a subset of the coordinate space, i.e. $C \subseteq \mathcal{D}$ (otherwise, C would be a multi-set). In the following, I will always make the assumption that each pair type w has a unique frequency signature in the data set.

The coordinate space \mathcal{D} can be visualised as a three-dimensional cube (whose full size is determined by the sample size N). Since cooccurrence data often cover a wide range of frequencies, logarithmic coordinates are more appropriate for visualisation

¹⁹According to the `ucsintro` manpage, UCS stands for *Utilities for Cooccurrence Statistics*.

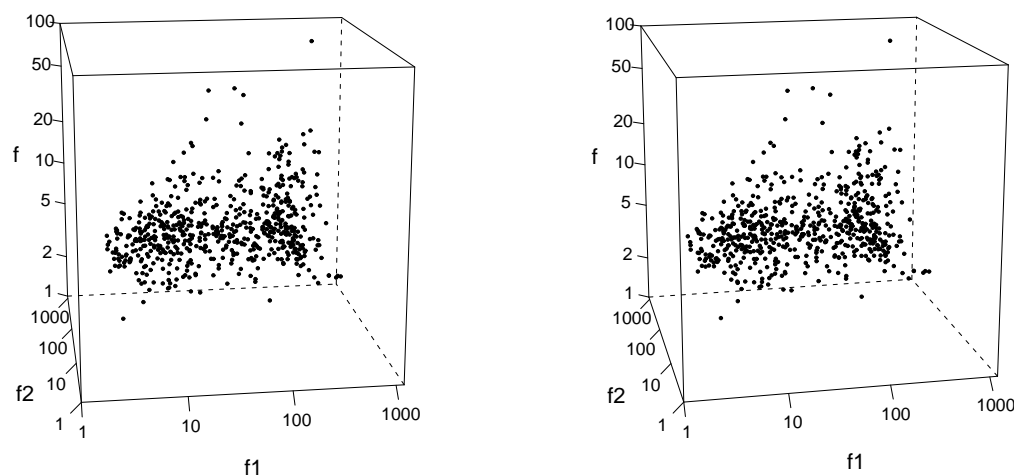


Figure 3.3: The three-dimensional parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set (stereograph for cross-eyed viewing).

(i.e. $\log_{10} f$, $\log_{10} f_1$ and $\log_{10} f_2$). Figure 3.3 shows a **stereographic image** of the subset of \mathcal{D} corresponding to $f_1, f_2 \leq 1000$ and $f \leq 100$ on a logarithmic scale. The point cloud represents the 617 pair types from the first data set (PNV-SLICES-01) in the PNV-SLICES collection, with jittered coordinates (Figure 3.4 shows a comparison of the point cloud with and without jittering). The stereograph has been designed for *cross-eyed viewing*. Look at the image from a normal reading distance, making sure that the paper is flat and evenly lit. Then cross your eyes slowly until you see the *left* image with your *right* eye, and the *right* image with your *left* eye. It often helps to tilt your head slightly in order to bring the two cubes into perfect alignment. A single point visible in the top right corner of the image represents the PP-verb combination *um ... Uhr beginnen* ‘start at ... o’clock’ with the coordinates $f = 76$, $f_1 = 458$ and $f_2 = 450$. The sample size of the PNV-SLICES-01 data set is $N = 61\,617$.

Various **coordinate transformations** can be applied to make the visualisation and the mathematical discussion more intuitive. The most useful of these transformations is the **ebo-system** defined as:

$$e := \frac{f_1 f_2}{N} = E_{11} \quad \text{“expectation”} \quad (3.8a)$$

$$b := \frac{f_1}{f_2} = \frac{R_1}{C_1} \quad \text{“balance”} \quad (3.8b)$$

$$o := f = O_{11} \quad \text{“observed”} \quad (3.8c)$$

The ebo-system is based on the observed cooccurrence frequency $o = O_{11}$ and the expected frequency $e = E_{11}$, which play a key role for most association measures. In addition to these two values, the balance b between the marginal frequencies is needed to determine the coordinates of a pair type uniquely. The ebo-coordinates also range across the space \mathcal{D} (the exact limits depending on the sample size N), but they are not constrained to integer values. On a logarithmic scale, the ebo-transformation rotates the coordinate system by 45 degrees around the $f = o$ axis.

The e -axis corresponds to the main diagonal in the (f_1, f_2) plane, and the b -axis to the main counterdiagonal. The top row of Figure 3.4 shows a rotated view of the parameter space \mathcal{D} where the e -axis is nearly horizontal and the b -axis runs from background to foreground.

Since the o and e coordinates provide the most relevant information, the visualisation and analysis of data sets (and later association measures) can be greatly simplified by ignoring the balance b . When logarithmic coordinates are used, this corresponds to an orthogonal projection onto the **two-dimensional** (e, o) plane, as shown in the bottom right panel of Figure 3.4 (note that the perspective of the three-dimensional view in the top row is nearly the same). As we will see in the following sections, many association measures do not depend on b at all or only to a very small degree, providing support for the use of (e, o) graphs.

In the bottom row of Figure 3.4, the data points corresponding to the pair types *auf (dem) Programm stehen* ‘be on the programme’ and *mit (dem) Bus fahren* ‘take a bus’ are marked with circles. The right panel uses jittered coordinates like the three-dimensional views, while the left panel uses the original integer values of f , f_1 and f_2 without adding jitter. The quantisation of the observed frequency $o = f$ becomes clearly visible as a band structure in the plot. The lowest band consists of all pair types with $f = 3$ (including *mit (dem) Bus fahren*), the next band of the pair types with $f = 4$, and so forth. Note that the distances between the bands decrease because of the logarithmic scale.

Formally, the ebo-system is not a fixed coordinate transformation because it depends on the sample size: the e -coordinate is scaled according to the value of N (which is not explicitly represented in the original coordinate space). This causes a “shift” along the e -axis in logarithmic coordinates. Since this shift is counter-balanced by a similar dependency of many association measures on sample size, these measures can be expressed by a fixed equation in terms of e , b and o , while their equations in the standard coordinate system also depend on N . Coordinates in the ebo-system are not size-invariant: when the complete frequency signature of a pair type w is multiplied with a constant factor k (corresponding to a larger corpus with the same relative joint and marginal frequencies for w), both the e and o coordinates are shifted by an amount of $\log_{10} k$ on a logarithmic scale (but b is not affected). This shift, which usually corresponds to higher association scores for the pair types, represents the greater amount of evidence provided by the larger sample.

3.3.2 Generalised association measures

An association measure assigns an association score to every possible frequency signature (f, f_1, f_2, N) , i.e. it assigns a score to every point in the coordinate space \mathcal{D} that corresponds to integer frequencies (these points form the **integer lattice** in \mathcal{D}). The precise values of the scores will usually depend on the sample size N . Every association measure has a continuous extension to the full coordinate space. For most measures, this extension is simply given by inserting non-integer frequencies into the equations. An exception are measures based on exact likelihood or exact hypothesis tests, which involve explicit summation or binomial coefficients.²⁰ However, since the

²⁰Note that the Poisson and binomial measures can easily be extended using Eq. (3.4) and (3.6), which accept non-integer frequency values. Binomial coefficients in other formulae (such as binomial-

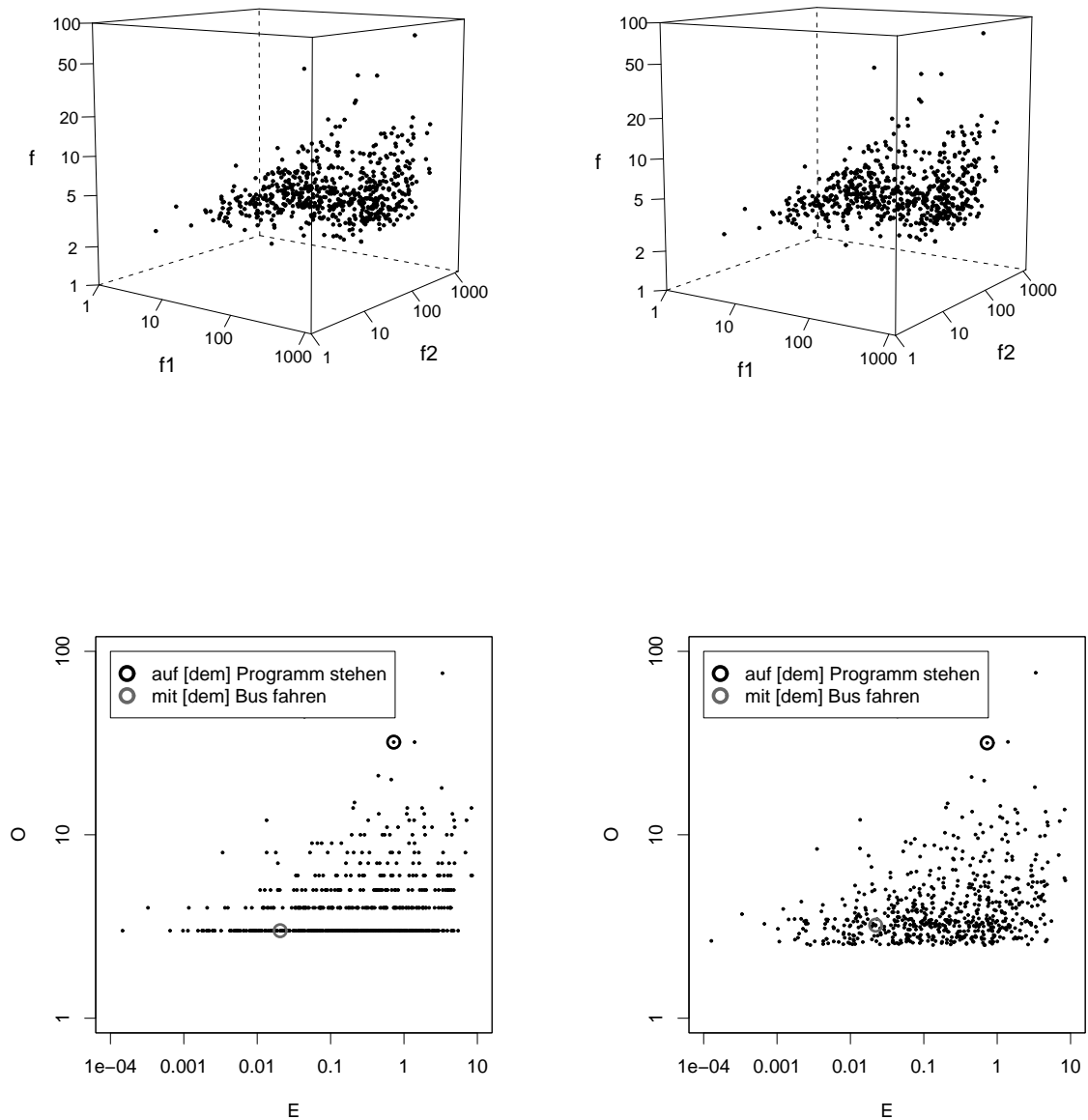


Figure 3.4: The top row shows a rotated view of the parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set. The e -axis is nearly horizontal in this view, while the b -axis is oriented from background to foreground. The bottom right panel shows a projection of the point cloud into the (e, o) plane, and the bottom left panel shows the same data without jittering.

integer lattice in \mathcal{D} is discrete, a continuous extension to the full space \mathcal{D} is always possible. In the following I will assume that the continuous extension of an association measure is *smooth* (i.e. all required derivatives exist and are continuous). It is always possible to find a smooth function that computes the correct scores on the integer lattice by a suitable interpolation procedure.

These considerations motivate the definition of a **generalised association measure (GAM)** as an arbitrary smooth function $g : \mathcal{D} \rightarrow \mathbb{R}$. A generalised association measure g is **sound** when it satisfies the following conditions:

$$\frac{\partial g}{\partial f} > 0, \quad \frac{\partial g}{\partial f_1} \leq 0, \quad \frac{\partial g}{\partial f_2} \leq 0. \quad (3.9)$$

These conditions capture reasonable intuitive assumptions about association scores: increasing the cooccurrence frequency while keeping the two marginal frequencies fixed should lead to a higher score; increasing one of the marginal frequencies with the other two frequencies fixed should lead to a lower score. A GAM g is called **semi-sound** when only the first condition holds. Most measures will not be **size-invariant**, i.e. their scores depend on the sample size N . Intuitively, a larger sample provides more evidence for positive or negative association, and this is reflected by the association scores. Formally, we should therefore write g_N instead of g , yielding a different score function for each sample size N (i.e. a family of functions). Sometimes the dependence on sample size can be “factored out”, i.e.

$$g_N(f, f_1, f_2) = c(N) \cdot g(f, f_1, f_2).$$

In these cases, the size factor $c(N)$ is only relevant when association scores computed on samples of different sizes have to be compared. Generalised association measures can also be defined over transformed coordinate systems, especially the ebo-coordinates. I do not make a formal distinction between the transformed measure and the original version, but simply write the transformed score function as $g(e, b, o)$ instead of $g(f, f_1, f_2)$. Many GAMs have a size-invariant equation in the (e, b, o) coordinate system: The shift of the data points along the logarithmic e and o axes compensates for changes in the score function g due to the larger sample size.

A generalised association measure g is **symmetric** iff $g(f, f_1, f_2) = g(f, f_2, f_1)$ (or, equivalently, $g(e, b, o) = g(e, b^{-1}, o)$). Such symmetric measures, which include virtually all the measures described in Section 3.1, make the implicit assumption that cooccurrences are “symmetric” rather than “directional”, i.e. exchanging the component types should yield the same association score for (v, u) as for (u, v) (cf. the remarks at the end of Section 1.2.2). A more important concept is centrality: a generalised association measure g is called **central** iff its scores do not depend on the balance b in ebo-coordinates, i.e. $g(e, b, o) = g(e, 1, o)$ for all $b > 0$. The score function of a central measure is completely determined by its values in the two-dimensional (e, o) plane.²¹ Section 3.3.3 explains how this property can be exploited for visualisation purposes. The **centralised** version g^c of a non-central measure g is obtained by setting $b = 1$, i.e. $g^c(e, b, o) := g(e, 1, o)$. A central measure is characterised by $g^c = g$ according to this definition.

likelihood and Fisher) can always be generalised with the help of the Beta function, cf. Eq. (A.36).

²¹In the standard coordinate system, the condition for centrality is much less intuitive, stating that $g(f, f_1, f_2) = g(f, \gamma f_1, \gamma^{-1} f_2)$ must hold for all $\gamma > 0$.

Two generalised association measures g_1 and g_2 are called **equivalent** iff there exists a strictly monotonic function h that transforms the scores assigned by g_1 into those of g_2 , i.e. $g_2(f, f_1, f_2) = h(g_1(f, f_1, f_2))$. Since both g_1 and g_2 are smooth, the link function h must necessarily be smooth as well. Equivalent GAMs lead to identical rankings of data sets. Examples of equivalence are: (i) the Dice and Jaccard measures, whose equivalence is established in the same way as that of the corresponding κ_{Dice} and κ_{Jaccard} coefficients (cf. Section 2.2.5); (ii) the test statistic of an asymptotic hypothesis test such as chi-squared and the corresponding p -value (in this case, h is the distribution function of the limiting distribution of the test statistic).

For a given data set $C \subseteq \mathcal{D}$ and generalised association measure g , the **n -best threshold** $\gamma_g(n)$ is defined as the largest value for which C contains at least n pair types with scores $g(x) \geq \gamma_g(n)$. When random jitter has been added to the coordinates of pair types and $\nabla g \neq 0$ throughout the coordinate space, it is almost impossible that two pair types are assigned exactly the same association score (i.e. the probability of such an event is zero).²² Therefore, it is almost certain that there are exactly n candidates with $g(x) \geq \gamma_g(n)$ and hence that $\gamma_g(n)$ is the association score of the n -th highest-ranking pair type.²³ Any GAM g can be transformed into an equivalent measure g' such that $\gamma_{g'}(n) = -n$ with respect to a given data set C (g' is called a **rank-transformed** measure). The negative sign is necessary to satisfy the soundness condition (3.9), i.e. the convention that higher association scores correspond to stronger association.

The following equations describe generalised versions of some important association measures presented in Section 3.1, formulated in ebo-coordinates. All these measures are *size-invariant* in the ebo-system (with the exception of g_{mean}) and *central* (because b does not appear in the equations).

$$g_{\text{frequency}}(e, b, o) = o \quad (3.10a)$$

$$g_{\text{MI}}(e, b, o) = \log \frac{o}{e} \quad (3.10b)$$

$$g_{\text{MI}^k}(e, b, o) = \log \frac{o^k}{e} \quad (3.10c)$$

$$g_{\text{gmean}}(e, b, o) = \frac{1}{\sqrt{N}} \cdot \frac{o}{\sqrt{e}} \quad (3.10d)$$

$$g_{\text{z-score}}(e, b, o) = \frac{o - e}{\sqrt{e}} \quad (3.10e)$$

$$g_{\text{t-score}}(e, b, o) = \frac{o - e}{\sqrt{o}} \quad (3.10f)$$

$$g_{\text{Poisson}}(e, b, o) = -\log_{10} \frac{\gamma(o, e)}{\Gamma(o)} \quad (3.10g)$$

²²This is usually not the case without random jitter, even if there are no data points with identical coordinates. For instance, the frequency measure assigns the same score to all candidates with the same cooccurrence frequency f .

²³Note that $\gamma_g(n)$ is not uniquely defined by the condition that there are exactly n pair types with $g(x) \geq \gamma_g(n)$. Since the set C is discrete, any value $\gamma_g(n)$ between the n -th and the $(n+1)$ -th highest score will satisfy the condition. The additional requirement that $\gamma_g(n)$ must be the largest such value ensures uniqueness and implies that $\gamma_g(n)$ equals the score of the n -th highest-ranking pair type.

$$g_{\text{MI}_{\text{conf}, \alpha}}(e, b, o) = \log \min \left\{ \mu > 0 \mid \frac{\gamma(o, e\mu)}{\Gamma(o)} \geq \alpha \right\} \quad (3.10h)$$

Poisson-Stirling_{log} cannot be extended to a sound GAM because the function $o \mapsto o \cdot (\log o - \log e - 1)$ has a minimum for $o = e$ and does not satisfy the monotonicity condition $\partial g / \partial o > 0$. The same problem is found for the local-MI measure. Eq. (3.10g) computes $-\log_{10} p\nu$ (according to the convention suggested in Section 3.1.3) instead of raw p -values, for which we would have $\partial g / \partial o < 0$.

The equations of non-central measures are typically much less elegant, especially when they are expressed in ebo-coordinates. It is convenient to use the following abbreviations for frequently needed quantities:

$$b^* := \frac{b+1}{\sqrt{b}} = \sqrt{b} + \frac{1}{\sqrt{b}} \quad (3.11a)$$

$$\bar{e} := N - \sqrt{Ne} \cdot b^* + e \quad (3.11b)$$

$$\|b\| := \begin{cases} \sqrt{b} & b \geq 1 \\ \sqrt{b^{-1}} & b < 1 \end{cases} \quad (3.11c)$$

The symbol b^* represents a *balance factor*, which does not distinguish between $f_1 > f_2$ and $f_1 < f_2$ and appears in many symmetric measures because of the identity $f_1 + f_2 = \sqrt{Ne} \cdot b^*$. The *conjugate expectation* \bar{e} corresponds to the term E_{22} in the contingency table of expected frequencies, while the *absolute balance* $\|b\|$ is only used for the MS measure. Using these abbreviations, the following non-central measures have relatively elegant equations:

$$g_{\text{Dice}}(e, b, o) = \frac{o}{\sqrt{e}} \cdot \frac{1}{\sqrt{N}} \cdot \frac{2}{b^*} \quad (3.12a)$$

$$g_{\text{Jaccard}}(e, b, o) = \frac{g_{\text{Dice}}(e, b, o)}{2 - g_{\text{Dice}}(e, b, o)} \quad (3.12b)$$

$$g_{\text{MS}}(e, b, o) = \frac{o}{\sqrt{e}} \cdot \frac{1}{\sqrt{N}} \cdot \frac{1}{\|b\|} \quad (3.12c)$$

$$g_{\text{chi-squared}}(e, b, o) = \pm \frac{N \cdot (o - e)^2}{e \cdot \bar{e}} \quad (3.12d)$$

The \pm in Eq. (3.12d) indicates that the GAM has to be converted into a one-sided measures in order to be sound. This can be achieved by writing the numerator as $(o - e) \cdot |o - e|$ instead of $(o - e)^2$. Looking at Eq. (3.12a), it is obvious how g_{Dice} differs from the central measure o / \sqrt{e} by a size factor and a balance factor. We can also see that association scores are reduced for $b \neq 1$ (because $b^* > 1$ in this case). Interestingly, the centralised version of Dice, which is obtained by setting $b^* = 1$, is identical to g_{mean} , i.e. $g_{\text{Dice}}^c = g_{\text{mean}}$. When applying Yates' continuity correction to the z-score and chi-squared measures, care has to be taken because the standard procedure (3.1) leads to an unsound GAM.²⁴ Therefore, an interpolation function

²⁴With Yates' correction applied, the score for $o = e + \frac{1}{4}$ is lower than the score for $o = e - \frac{1}{4}$, since $o' = o - \frac{1}{2}$ in the first case and $o' = o + \frac{1}{2}$ in the second.

such as

$$d_Y(x) := \begin{cases} x - 1/2 & x \geq 1 \\ x/2 & -1 < x < 1 \\ x + 1/2 & x \leq -1 \end{cases} \quad (3.13)$$

has to be used.²⁵ The continuity-corrected measures are then defined by:

$$g_{z\text{-score}_{\text{corr}}}(e, b, o) = \frac{d_Y(o - e)}{\sqrt{e}} \quad (3.14a)$$

$$g_{\text{chi-squared}_{\text{corr}}}(e, b, o) = \pm \frac{N(d_Y(o - e))^2}{e \cdot \bar{e}} \quad (3.14b)$$

Some other measures, such as odds-ratio, log-likelihood and Fisher, still lead to unwieldy equations. However, even without an explicit equation in ebo-coordinates, the score function $g(e, b, o)$ of such a measure can be evaluated numerically by transforming the ebo-coordinates (e, b, o) back into the corresponding frequency signature or contingency table, and then applying the standard equation of the measure as listed in Section 3.1.

In the general formal model presented here, **combined measures** as described at the end of Section 3.1.8 correspond to a combination of GAM functions g_1 and g_2 by some link operator. Well-known examples are the minimum $\min\{g_1, g_2\}$ or a linear combination $\alpha g_1 + \beta g_2$. A cutoff filter can be added to a generalised association measure with the help of a cutoff function

$$h_\gamma(x) := \begin{cases} 1 & x \geq \gamma + \epsilon \\ 0 & x \leq \gamma - \epsilon \end{cases}$$

for small $\epsilon > 0$, with a smooth extension to the interval $[\gamma - \epsilon, \gamma + \epsilon]$. The combined measure $h_\gamma(g_1) \cdot g_2$ equals 0 for $g_1 \leq \gamma - \epsilon$ and g_2 for $g_1 \geq \gamma + \epsilon$.²⁶ **Parametric measures** correspond to families $\{g_\alpha\}$ of sound GAMs, where α stands for the free parameter of the measure (cf. Eq. 3.10h).

3.3.3 Iso-surfaces and iso-lines

For any number $\gamma \in \mathbb{R}$, the threshold condition $g(x) \geq \gamma$ defines a subset

$$A_g(\gamma) := \{g \geq \gamma\} = \{(f, f_1, f_2) \mid g(f, f_1, f_2) \geq \gamma\} \quad (3.15)$$

of the coordinate space, which is called the γ -**acceptance region** of g . All pair types whose coordinates x fall into $A_g(\gamma)$ have an association score $g(x) \geq \gamma$. Hence, the intersection $A_g(\gamma) \cap C$ is the set of pair types in C that are “accepted” (e.g. as

²⁵Note that d_Y is continuous, but not differentiable at the contact points $x = -1$ and $x = 1$. Although generalised association measures are formally required to be smooth functions, the two kinks introduced by d_Y usually cause no problems in practice.

²⁶Note that this implementation of a cutoff filter as a combined association measure is slightly different from the one presented in Section 3.1.8. The formulation given here makes it easier for the new measure to satisfy the general conditions on GAMs.

collocation candidates) at a threshold of γ . By setting $\gamma = \gamma_g(n)$ with respect to the data set C , we obtain the n -**acceptance region**

$$A_{g,n} := A_g(\gamma_g(n)) = \{g \geq \gamma_g(n)\}, \quad (3.16)$$

and

$$C_{g,n} := A_{g,n} \cap C = \{x \in C \mid g(x) \geq \gamma_g(n)\} \quad (3.17)$$

is the n -**best list** for the measure g , i.e. the set of n highest-ranked pair types. Such n -best lists play a key role for collocation extraction tasks (Section 1.2.2) and for the empirical evaluation of association measures in Chapter 5. Under the assumptions made in the previous section, every n -best list contains exactly n pair types, i.e. $|C_{g,n}| = n$. For a rank-transformed measure g' , the n -acceptance region is simply given by $A_{g',n} = \{g' \geq -n\}$.

For any semi-sound measure, the γ -acceptance region $A_g(\gamma)$ is a connected three-dimensional region whose “lower” boundary is given by the **iso-surface** $\{g = \gamma\}$. This iso-surface can be parametrised over f_1 and f_2 , i.e.

$$\{g = \gamma\} = \{(h(f_1, f_2), f_1, f_2) \mid f_1, f_2 \in (0, \infty)\} \quad (3.18)$$

with a smooth function $h : (0, \infty)^2 \rightarrow (0, \infty)$. The acceptance region $\{g \geq \gamma\}$ contains all points $x \in \mathcal{D}$ for which $f \geq h(f_1, f_2)$, since the condition $\partial g / \partial f > 0$ implies $g(f, f_1, f_2) \geq g(h(f_1, f_2), f_1, f_2) = \gamma$ whenever $f \geq h(f_1, f_2)$. If g is sound, the “height” function h must be monotonic in f_1 and f_2 , i.e. $\partial h / \partial f_1 \geq 0$ and $\partial h / \partial f_2 \geq 0$.

The top row of Figure 3.5 shows the iso-surface $\{g_{\log\text{-likelihood}} = 22.6\}$, which is the (one-sided) log-likelihood threshold corresponding to a p -value of 10^{-6} , together with the PNV-SLICES-01 data set. The region of \mathcal{D} above the surface is the acceptance region $A_g(22.6)$. All points in this region represent pair types that show significant evidence for a positive association, at a confidence level of $\alpha = 10^{-6}$. In ebo-coordinates, the height function can be parametrised over e and b and we have

$$\{g \geq \gamma\} = \{(e, b, o) \mid o \geq h(e, b)\}. \quad (3.19)$$

For a sound measure, h satisfies the condition $\partial h / \partial e \geq 0$, but there is no corresponding constraint along the b -coordinate. This fact is illustrated by the U-shaped form of the Dice iso-surface along the b -axis (i.e. the counter-diagonal in the (f_1, f_2) plane), as shown in the bottom row of Figure 3.5.

The properties of an association measure are fully determined by the corresponding acceptance regions and the iso-surfaces that form their boundaries. Formally, a generalised association measure g is equivalent to a **monotonic family of iso-surfaces**, $\gamma \mapsto \{g = \gamma\}$. With respect to a particular data set C , only the surfaces $\{g = \gamma_g(n)\}$ are relevant, though. For most generalised association measures, explicit equations for the height functions of iso-surfaces $\{g = \gamma\}$ can be derived by “solving” Eq. (3.10), (3.12) and (3.14) for the variable o :²⁷

$$g_{\text{frequency}} : \quad o = \gamma \quad (3.20a)$$

$$g_{\text{MI}} : \quad o = 10^\gamma \cdot e \quad (3.20b)$$

²⁷Note the use of base-10 logarithms for MI and similar measures, corresponding to the convention followed by the UCS toolkit. This leads to a factor of 10^γ in the equations for iso-surfaces.

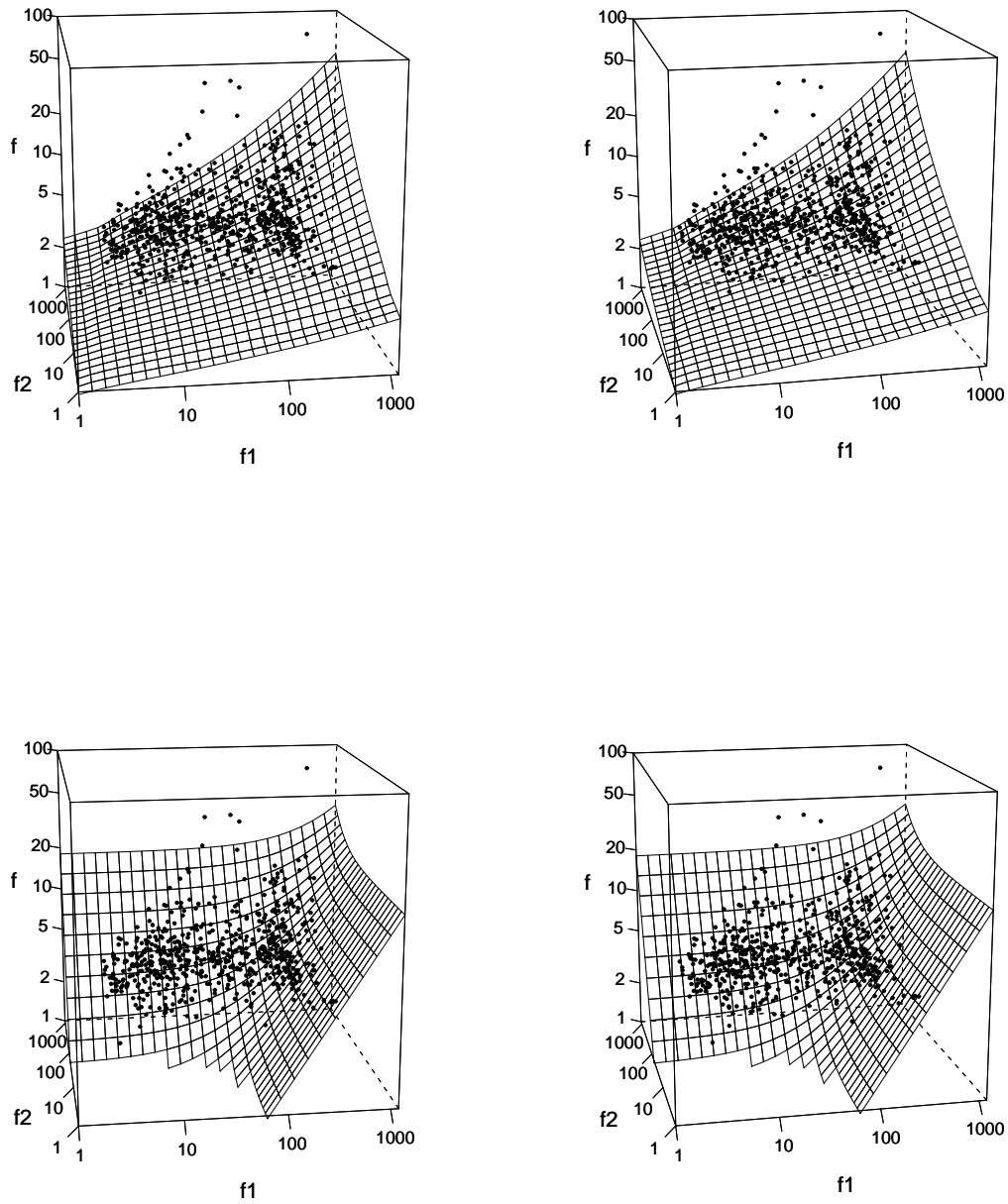


Figure 3.5: Parameter space with point cloud representing the PNV-SLICES-01 data set and iso-surfaces of the log-likelihood and Dice measures. The top row shows the iso-surface $\{g_{\log\text{-likelihood}} = 22.6\}$, corresponding to $p_v = 10^{-6}$. The bottom row shows a 200-best iso-surface for Dice.

$$g_{\text{gmean}} : o = \sqrt{N} \cdot \gamma \cdot \sqrt{e} \quad (3.20c)$$

$$g_{\text{z-score}} : o = \gamma \cdot \sqrt{e} + e \quad (3.20d)$$

$$g_{\text{t-score}} : o = \frac{\gamma^2}{2} + \gamma \cdot \sqrt{e + \frac{\gamma^2}{4}} + e \quad (3.20e)$$

$$g_{\text{MI}^k} : o = 10^{\gamma/k} \cdot \sqrt[k]{e} \quad (3.20f)$$

$$g_{\text{Dice}} : o = \sqrt{N} \cdot \gamma \cdot \frac{b^*}{2} \cdot \sqrt{e} \quad (3.20g)$$

$$g_{\text{Jaccard}} : o = \sqrt{N} \cdot \frac{\gamma}{1 + \gamma} \cdot b^* \cdot \sqrt{e} \quad (3.20h)$$

$$g_{\text{MS}} : o = \sqrt{N} \cdot \|b\| \cdot \gamma \cdot \sqrt{e} \quad (3.20i)$$

$$g_{\text{Liddell}} : o = \sqrt{N} \cdot \frac{\gamma}{\sqrt{b}} \cdot \sqrt{e} + \left(1 - \frac{\gamma}{b}\right) \cdot e \quad (3.20j)$$

$$g_{\text{chi-squared}} : o = \pm \sqrt{\gamma} \cdot \sqrt{\frac{\bar{e}}{N}} \cdot \sqrt{e} + e \quad (3.20k)$$

$$g_{\text{z-score}_{\text{corr}}} : o = d_Y^{-1}(\gamma \sqrt{e}) + e \quad (3.20l)$$

$$g_{\text{chi-squared}_{\text{corr}}} : o = d_Y^{-1} \left(\pm \sqrt{\gamma} \cdot \sqrt{\frac{\bar{e}}{N}} \cdot \sqrt{e} \right) + e \quad (3.20m)$$

In Eq. (3.20k) and (3.20m), $\pm\sqrt{\gamma}$ denotes the signed square root function, with $\pm\sqrt{\gamma} = -\sqrt{|\gamma|}$ for $\gamma < 0$. Note that the factor $\sqrt{\bar{e}/N}$ is usually close to 1. In Eq. (3.20l) and (3.20m), d_Y^{-1} is the inverse of the generalised Yates' correction. Eq. (3.20j) can be written more concisely in the form $o = \frac{\gamma}{N} f_2(N - f_2) + e$. For the other generalised association measures, there is no (obvious) closed-form solution to the iso-surface equation. However, the soundness condition $\partial g / \partial o > 0$ ensures that iso-surfaces for these measures can be computed efficiently with a binary search algorithm.

The association scores of a central measure g do not depend on the “balance” coordinate b in the ebo-system. Therefore, it is sufficient to know the values of g in the (e, o) plane in order to compute the score of any point $x = (e, b, o)$, by orthogonal projection $P_b x = (e, o)$ into the (e, o) -plane: $g(x) = g(e, o)$. A data set C can thus be replaced by its projection $P_b C$, and g is reduced to a function $g : (0, +\infty)^2 \rightarrow \mathbb{R}$ in the two-dimensional (e, o) -plane. This transformation simplifies the mathematical discussion, visualisation and empirical study of generalised association measures and data sets considerably. Assuming that random jitter has been added to C , the projections $P_b x$ of points $x \in C$ will almost certainly have unique coordinates in the (e, o) plane as well. Figure 3.6 illustrates this situation, showing the PNV-SLICES-01 data set together with the iso-surface $\{g_{\text{Poisson}} = 6\}$ of the central Poisson measure in the top row (this corresponds to a p -value of 10^{-6} , as in Figure 3.5). The view has been rotated in order to align the b -axis with the position of the observer. It is now clearly visible that the height of the iso-surface is constant along this axis, i.e. the score of the Poisson measure does not change as long as e and o are held constant. If the view were rotated a little further until the b -axis is exactly perpendicular to

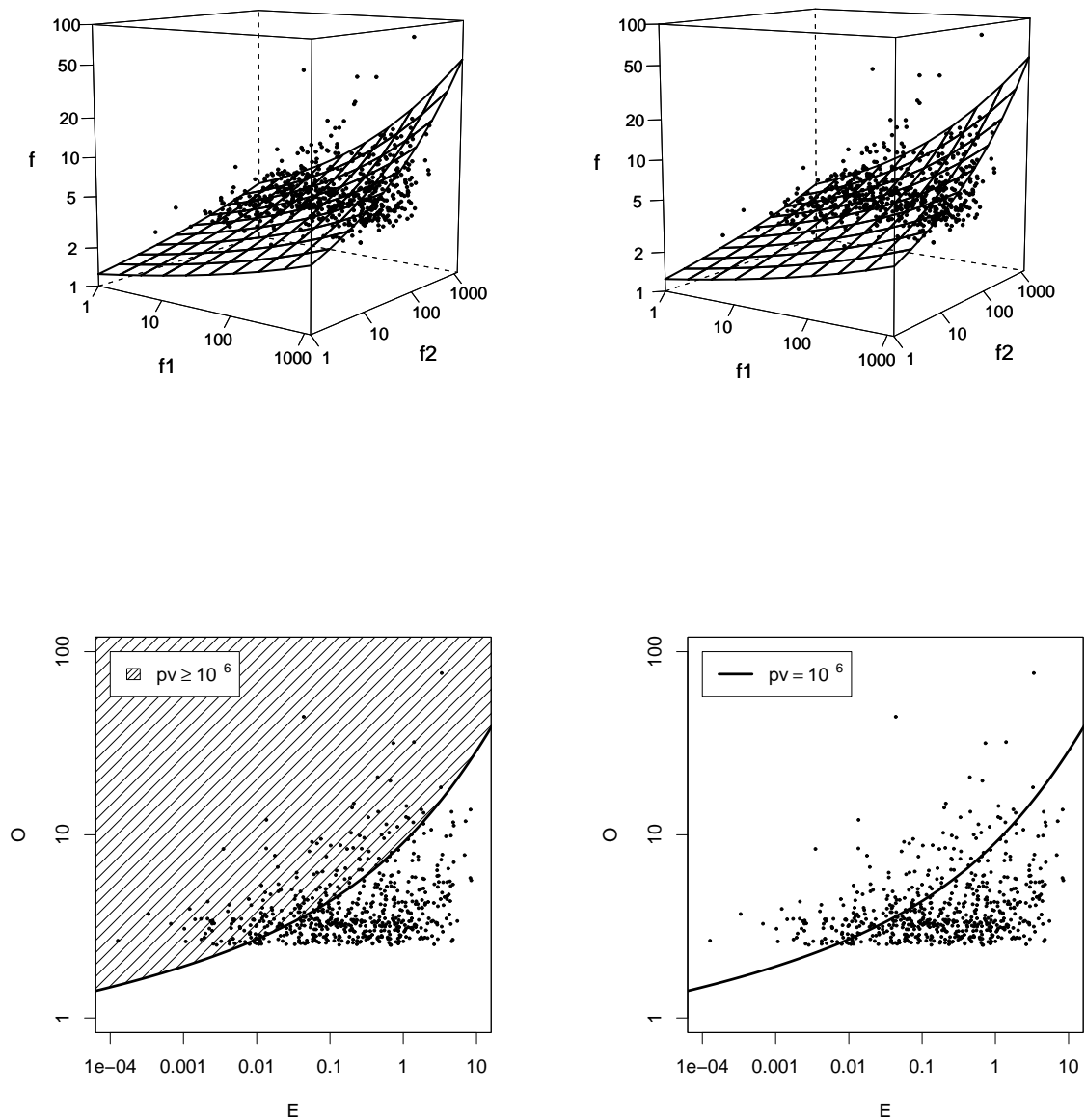


Figure 3.6: The top row shows a rotated view of the parameter space \mathcal{D} with a point cloud representing the PNV-SLICES-01 data set and the iso-surface $\{g = 6\}$ of the Poisson measure (corresponding to coordinates with $pv = 10^{-6}$). The bottom row shows the orthogonal projection of both the point cloud and the iso-surface into the (e, o) plane. In the bottom left panel, the projection of the corresponding acceptance region $A_g(6)$ is shaded in the plane (corresponding to coordinates with $pv \geq 10^{-6}$).

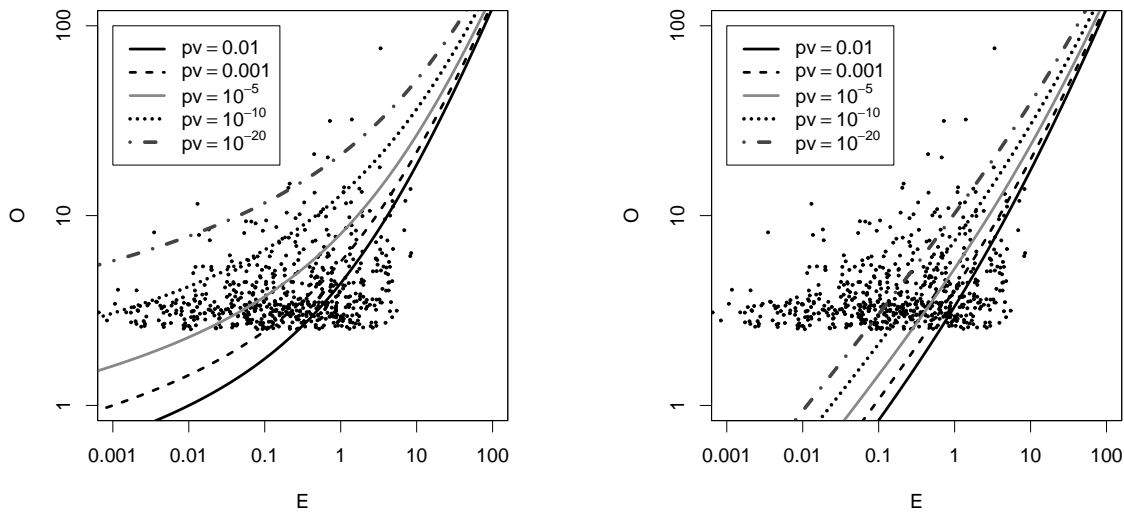


Figure 3.7: Families of iso-lines representing the generalised association measures Poisson (left panel) and z-score (right panel). The threshold values for the iso-lines were chosen to correspond to specific p -values, including the common significance levels $p\upsilon = .01$ and $p\upsilon = .001$.

the paper plane, the iso-surface would appear as simple curve, corresponding to its orthogonal projection into the (e, o) plane. This projection $P_b \{g = \gamma\}$ is referred to as an **iso-line** of the GAM g .

The bottom right panel of Figure 3.6 shows the iso-line for $\{g_{\text{Poisson}} = 6\}$. Since the height function $h(e, b)$ of the iso-surface of a central measure does not depend on b , the corresponding iso-line is defined by the equation $o = h(e) := h(e, 1)$. A point $x \in C$ belongs to the acceptance region $A_g(\gamma)$ iff its projection $P_b x$ satisfies $o \geq h(e)$. In other words, the iso-line $o = h(e)$ is the lower boundary of the projection $P_b A_g(\gamma)$ of the acceptance region into the (e, o) -plane, as illustrated by the bottom left panel of Figure 3.6. Thus, any central GAM g is equivalent to a **monotonic family of iso-lines** in the (e, o) plane, and the properties of g are determined by the shapes of these lines. Figure 3.7 uses this technique to visualise the Poisson (left panel) and z-score (right panel), drawing iso-lines corresponding to selected p -values (note that the visible range on the e -axis has been shifted compared to previous plots).

It has to be kept in mind that two-dimensional visualisation techniques are usually not suitable for measures that are not central *per se* (an exception being those with only a weak dependency on b). This includes, in particular, many of the measures that estimate coefficients of association strength (see Section 3.1.5). As an example, Figure 3.8 compares the 200-best iso-surface of the Poisson measure (coarse grid) with the 200-best iso-surface of Dice (fine grid), showing the strong dependency of the latter on the b coordinate. It is obvious from this graph that Dice cannot simply be reduced to a two-dimensional function.

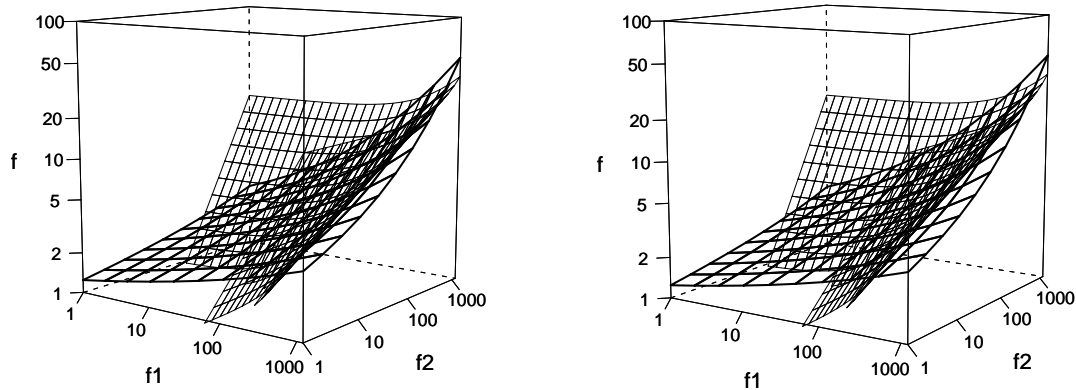


Figure 3.8: Rotated view of 200-best iso-surfaces for the Dice (fine grid) and Poisson (coarse grid) measures, with the b -axis running from background to foreground.

3.4 Comparing association measures

3.4.1 Goals and methods

As has already been pointed out at the beginning of this chapter, association measures are expected to perform two tasks: (i) estimate the “true” association strength of pair types in the population from the observed frequencies; (ii) correct this estimate for sampling variation in the observed data. Optimally, a comparison of association measures and a mathematical discussion of their properties should take both aspects into account. Given a (hypothetical) pair type with parameters (π, π_1, π_2) , such a discussion would proceed in two steps:

1. Identify the “ideal” association score which the measure would assign if there were no sampling variation. This score can be computed from the population parameters and is usually related to one of the coefficients of association strength presented in Section 2.2.5. The value of the ideal score should be meaningful and interpretable. Whether this is indeed the case depends on the intended application and the intuitions of the researcher, so it cannot be established on purely mathematical grounds. A possible approach is to list and compare the ideal scores for special (boundary) values of the population parameters as has been done in Table 2.2. It has to be kept in mind, though, that there is also no general agreement on a “best” measure of association strength in the field of mathematical statistics.
2. Study the sampling error of the real association scores computed by a measure, i.e. how close they are to the ideal value and how great their variation is. In principle, the exact distribution of the association scores can be determined from the multinomial sampling distribution for the given probability parameters (π, π_1, π_2) (though in practice this involves unwieldy mathematical expres-

sions or time-intensive numerical computations). Such an experiment has been performed by Dunning (1998, 73f).

The procedure outlined above faces several practical problems: (i) For many association measures, especially those connected to statistical hypothesis tests (Sections 3.1.2, 3.1.3, and 3.1.4), it is entirely unclear how the “ideal” association score should be defined (since Table 2.2 does not apply in this case). (ii) It is very difficult to compute the exact distribution of association scores. Analytical results have only been obtained for certain special cases (e.g. Good *et al.* 1970), and a numerical approach presents problems of accuracy and performance, especially for large sample size N ; even the less satisfactory Monte Carlo sampling is computationally expensive. (iii) The results obtained in the second step are valid for one particular set of parameters (π, π_1, π_2) only. The analysis or simulation would have to be repeated for many different parameter values throughout the population parameter space, and then the systematic effects of changes in the joint and marginal probabilities would have to be studied. Therefore, I advocate a much simpler and quite intuitive empirical approach: namely, to study generalised association measures as arbitrary real-valued functions, without reference to “ideal” scores or to the population parameters.

One possibility is a **direct comparison** of the scores computed by two or more association measures for a either real or an invented data set. The use of a real data set highlights practically relevant differences between measures. On the other hand, invented data sets (called **dummies**), where the joint and marginal frequencies vary in a systematic way across a wide range of values, can throw light on the behaviour of the measures under boundary conditions. The results of such a comparison can be visualised in the form of a scatter plots, using the scores assigned by one measure as x coordinates and those assigned by another one as y coordinates. This approach is especially useful for association measures that are intended to compute the same quantity, or whose scores can at least be interpreted in the same way. The best example are the p -values computed by likelihood measures and exact hypothesis tests, which measure the amount of evidence against the null hypothesis of independence. The scores of asymptotic tests are also comparable when they are translated into the corresponding p -values according to the theoretical limiting distribution of the test statistic. Examples of direct comparison plots (used for this purpose) can be found in Section 3.4.2 and in Dunning (1998, 74f). Plots of completely unrelated measures (say, log-likelihood and Dice) at best have artistic value.

Another possibility is an **intellectual comparison** of the equations that define the association measures. For instance, Stubbs (1995) performs an intuitive analysis of MI and t-score, where he manipulates and approximates the formulae in order to understand their behaviour under different conditions and to identify the “main factors” of each measure. One of his conclusions is that t-score is closely linked to the observed cooccurrence frequency: $t \approx \sqrt{O_{11}}$. However, it is sensitive to an increase in the expected frequency E_{11} , which he interprets as a bias against combinations of high-frequency words. In a similar way, Smadja *et al.* (1996, 9–12) embark on a lengthy intuitive discussion of the properties of Dice and its supposed advantages for the identification of translation equivalents. Generalised equations in the ebo-system, as given in Eq. (3.10), (3.12) and (3.14) in Section 3.3.2, are an excellent starting point for such analyses because they already make the influence of expected frequency e , observed frequency o and balance b explicit. While this approach can

be very successful for simple measures (those considered by Stubbs are arguably among the most easily interpretable ones), many other measures resist such direct interpretation.

The coordinate space and generalised association measures introduced in Section 3.3 provide a natural framework for the **geometric interpretation** and comparison of association measures, viewing them as families of iso-surfaces in the three-dimensional space \mathcal{P} , or families of iso-lines in the (e, o) plane for central measures (cf. Section 3.3.3). A visual analysis of the iso-surfaces or iso-lines can help us to reach an intuitive understanding of the properties of individual association measures and the differences between them. For instance, from Figure 3.7 we get a good idea what level of significance the Poisson and z-score measures assign to different combinations of expected and observed frequency. In this case, iso-lines for the same p -values are directly comparable between the two measures. We can thus see that Poisson and z-score agree about the level of significance for higher-frequency data ($e \geq 10$), but Poisson has much less faith in small amounts of evidence (for $e \ll 10$).²⁸ In particular, z-score considers any pair type with $e < .01$ highly significant, regardless of its observed frequencies. For Poisson, on the other hand, a pair type that occurs just once in a sample of this size could never achieve a significance of $p_v = 10^{-5}$ (indicated by the grey iso-line).

In contrast to the direct comparison of association scores described above, the geometric approach also allows comparisons between unrelated measures. It is often possible to describe the properties of iso-lines by looking at the formal mathematical properties of their height functions $o = h(e)$ (as listed in Section 3.3.3), paying special attention to the behaviour of h for high ($e \gg 1$) and low ($e \rightarrow 0$) frequencies. From Eq. (3.20), we see that the height functions of all association measures listed there are linear combinations of the identity function e and the square root function \sqrt{e} , with an additional balance factor for non-central measures. Only t-score adds a constant term, which is otherwise a unique characteristic of the frequency measure.

The practical relevance of the properties of association measures and the differences between them is highlighted when the iso-surfaces or iso-lines are compared to the point cloud C representing a real-life data set, or to its projection into the (e, o) plane. Ideally, such studies should be combined with an **empirical evaluation** of the measures as described in Chapter 5, as well as a linguistic appraisal of the word pairs hidden behind all the points in the graphs.

In order to come to terms with the multitude of known association measures (or at least the ones presented in Section 3.1), a good strategy is to divide them into groups of measures that (purport to) measure similar quantities. From each such group, a measure with particularly desirable or typical properties should be chosen as a **prototype**. Whenever possible, it is advantageous to choose a central measure as a prototype so that its analysis and visualisation are reduced to a two-dimensional problem. The measures within a group can then be described by comparison with the prototype, e.g. with respect to their behaviour for high- and low-frequency data (e), for different sample sizes (N), and for unbalanced data ($b \neq 1$). In order to understand the differences between groups, it is only necessary to compare the group prototypes, either by geometric interpretation or through an empirical evaluation of

²⁸For $e < 10$, the iso-lines of Poisson are higher than those of z-score because the observed frequency o must be larger to provide the same amount of evidence against H_0 , i.e. the same significance.

real-life data sets.

3.4.2 The major groups

There are two large and important groups of association measures. The first group collects measures based on statistical hypothesis tests (from Sections 3.1.3 and 3.1.4) or sample probabilities (the likelihood measures from Section 3.1.2). Together, these association measures form the **significance of association** group.²⁹ They estimate the amount of evidence provided by the observed data against the null hypothesis of independence. This estimate can be expressed in the form of a p -value, so that a direct comparison of the association scores is possible.

The second group of measures is concerned more with the **degree of association** (quantified by any one of the coefficients of association strength introduced in Section 2.2.5) than with the amount of evidence supporting it. Most association measures in this group are maximum-likelihood estimates of the respective coefficients (Section 3.1.5). In addition, conservative estimates (confidence intervals) for some coefficients of association strength were obtained in Section 3.1.6.

There are several association measures that do not fit in either group (at least according to their theoretical derivation, or lack thereof). Surprisingly, some of these measures are found to be equivalent (or nearly equivalent) to other measures that belong to one of the major groups. The remaining “outliers” have to be studied individually, provided that they show promise of any practical relevance.

Significance of association

This group includes all exact and asymptotic hypothesis tests from Sections 3.1.3 and 3.1.4. In view of its questionable theoretical foundation, the t -score measure may not fit into the group. However, it is used as a hypothesis test by many authors (e.g. Church *et al.* 1991) and has therefore been included.

In principle, all these tests should compute (more or less) the same p -values, although there are certain differences between asymptotic tests and exact tests (especially Fisher’s test, which is based on the conditional distribution for fixed row and column sums). These differences have been discussed at length in mathematical statistics. After decades of controversy, most experts seem to agree now that Fisher’s test produces the most meaningful p -values (cf. Yates 1984). We can thus take the Fisher association measure as a reference point for the significance of association group.

From the mathematical literature, we can predict how well the various asymptotic tests approximate the exact value of Fisher’s test, and which circumstances (such as sample size or the application of Yates’ continuity correction) have an influence on the quality of the approximations (e.g. Yates 1934; Barnard 1947; Yates 1984; Haberman 1988; Agresti 1990; Lehmann 1991). These predictions are not always borne out for word cooccurrences, though, as was shown by Dunning (1993, 1998). The reason is that mathematical discussions often assume roughly uniform distributions and are more concerned with small samples, while we have to deal with large sample sizes

²⁹The terms *significance of association* and *degree of association* are also used by Hollander and Wolfe (1999, 467).

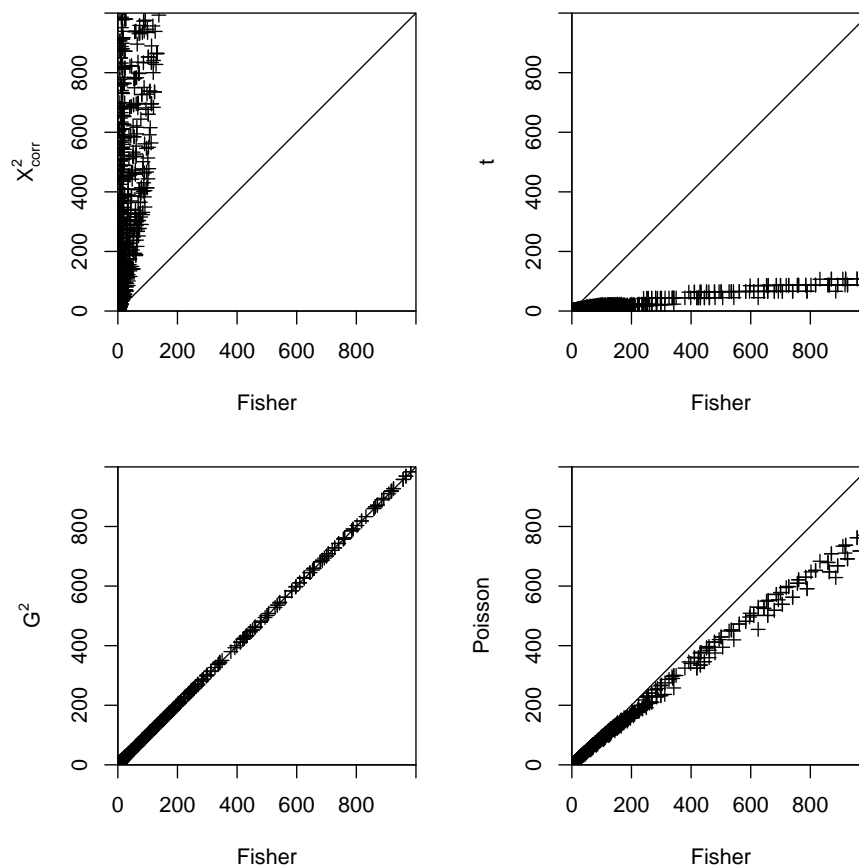


Figure 3.9: Comparison of p -values for measures from the significance of association group, using Fisher as a reference point (labels on the axes refer to $-\log_{10} p\nu$).

but highly skewed contingency tables (where O_{11} is very small and O_{22} is extremely large, cf. the examples in Section 2.1).

In order to find out how good the approximations of different measures to the Fisher p -values really are, we can directly compare their scores as described in Section 3.4.1. Here, a dummy data set was used with $N = 100\,000$ and f, f_1, f_2 ranging systematically between 1 and 1 000.³⁰ When looking at such invented data sets, it is important to keep in mind that they serve to explore the full range of possible situations rather than to have a realistic distribution. Therefore, the distribution of scores (which tend to cluster in a particular region of the graph, with only few outliers) must not be taken too seriously and may be entirely different for real-life data sets (which contain a higher proportion low-frequency pair types, for instance).

Figure 3.9 shows a comparison between Fisher and the measures chi-squared (X^2), t -score (t), log-likelihood (G^2) and Poisson. The association scores computed by the asymptotic tests have been converted to $-\log_{10} p\nu$ for this purpose, according to the respective limiting distributions. The thin diagonal line indicates the desired result of equal p -values. From the panels in the top row, we see that chi-squared overestimates significance dramatically, even when Yates' continuity correction is applied. This is

³⁰Results for larger sample sizes are qualitatively similar, although they show greater variation and differences between the measures become more pronounced.

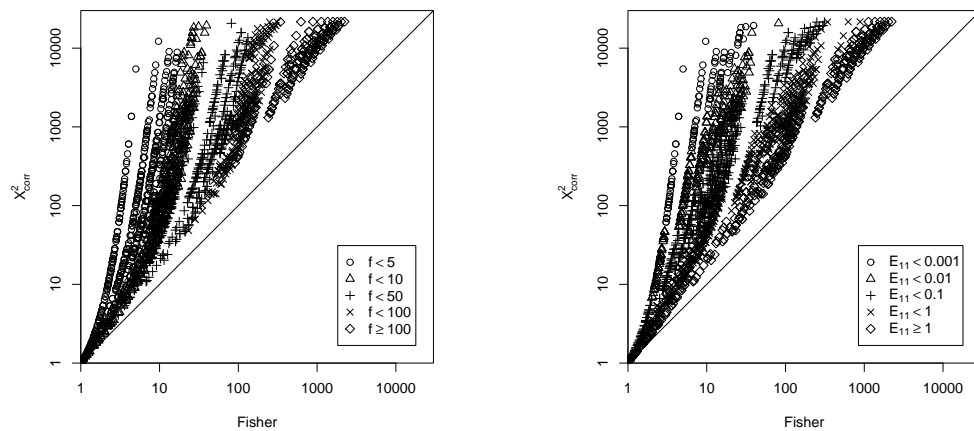


Figure 3.10: The roots of overestimation: comparison of the Fisher and chi-squared p -values according to observed (left) and expected (right) frequency.

not just a matter of scaling: a closer look reveals that even pair types with the same small Fisher scores (down to $-\log_{10} p\nu \approx 10$) may be assigned widely different scores by chi-squared (up to a significance of $-\log_{10} p\nu \geq 1\,000$). The t-score measure, on the other hand, turns out to be a highly conservative measure, underestimating significance substantially. The results of Dunning (1993, 1998) are corroborated by the bottom left panel, where log-likelihood gives an excellent approximation to the Fisher p -values across the entire range of frequency signatures. The best approximation by a central measure is given by Poisson in the bottom right panel, which underestimates significance only by a moderate amount (the binomial measure gives almost identical results).

Figure 3.10 explores the causes of the chi-squared overestimation, using a logarithmic scale to make the graphs more readable.³¹ The dummy data set was divided into frequency bands according to observed frequency ($o = f$, left panel) and expected frequency ($e = E_{11}$, right panel). The right panel shows clearly that the expected cooccurrence frequency is at the heart of the problem: the smaller E_{11} , the more inflated the chi-squared values are. The observed cooccurrence frequency is closely linked to the magnitude of the association scores (with respect to either measure) and is responsible for the band-like structure of the scatterplot (the leftmost “band” of points corresponds to $f = 1$, the next one to $f = 2$, etc.). A similar plot with a subdivision according to b shows that the balance between f_1 and f_2 does not contribute to the overestimation in any substantial way.

The field of mathematical statistics provides convincing arguments against likelihood measures (see the example at the beginning of Section 3.1.3), which are prone to overestimating the significance of high-frequency data in large samples. However, a direct comparison of the scores computed by Poisson-likelihood, binomial-likelihood and hypergeometric-likelihood with the corresponding exact tests (Poisson, binomial and Fisher) reveals that for the very high association scores which are mainly of in-

³¹Note that the values on both axes are already logarithms of p -values ($-\log_{10} p\nu$), but an additional logarithmic scale is needed to cover the enormous range of significance values produced by the measures. Also note that the scores of both measures would lead to floating-point underflow if they were computed as raw p -values (cf. Section 3.2.1).

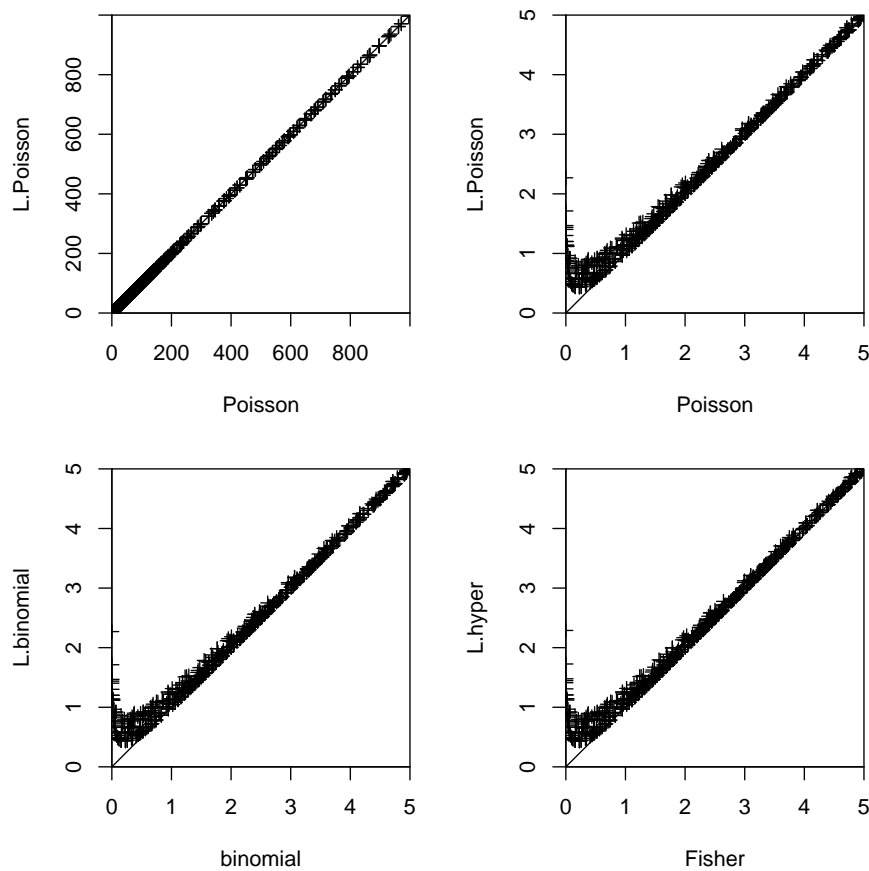


Figure 3.11: Comparison between likelihood measures ($-\log_{10} l_V$, y -axis) and the corresponding exact hypothesis tests ($-\log_{10} p_V$, x -axis).

terest in the analysis of cooccurrence data, they give a very good approximation to the exact tests (shown for Poisson in the left top panel of Figure 3.11). Substantial differences are only found for pair types that do not show significant evidence against H_0 even at the traditional significance level of $\alpha = .01$ (Figure 3.11).

A last question is motivated by the observation that log-likelihood scores are virtually identical to the reference values given by Fisher, while the best central measure (Poisson) deviates considerably even though as an exact test it is mathematically much more similar to Fisher's test than the asymptotic likelihood ratio test. This suggests that differences between these measures may be due to non-centrality, i.e. the influence of the balance b . In order to test this hypothesis, we have to compare a non-central measure with a similar central one, e.g. chi-squared and z-score. From Eq. (3.12d), we see that

$$g_{\text{chi-squared}}(e, b, o) = \pm \left(g_{\text{z-score}}(e, o) \right)^2 \cdot \frac{N}{\bar{e}}.$$

Therefore, chi-squared differs from a central measure only by a factor of $\bar{e}/N = (1 - f_1/N)(1 - f_2/N) \approx 1$ (because the marginal frequencies are usually small compared to the sample size). This conclusion is supported by the left panel of Figure 3.12, which shows a direct comparison of these measures on the dummy data set. The log-likelihood measure (abbreviated here as g) does not have a central equivalent, but

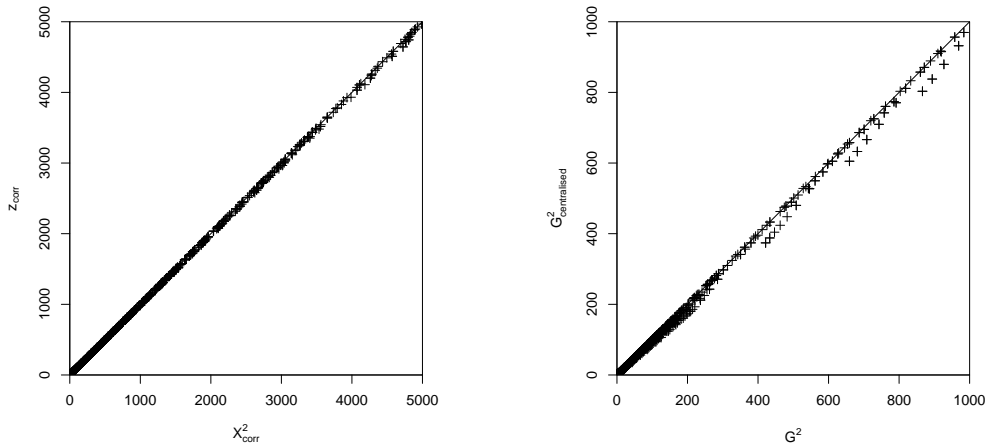


Figure 3.12: Comparison of p -values between central and non-central variants of measures from the significance of association group.

it can be compared to its centralised version $g^c(e, b, o) = g(e, 1, o)$. Since neither g nor g^c has a simple generalised equation in ebo -coordinates, we have to rely on the empirical comparison in the right panel of Figure 3.12. It is obvious that the deviation of g^c from log-likelihood is much smaller than that of the Poisson measure. Further evidence is provided by Figure 3.13, which compares iso-surfaces of g (fine grid) and g^c (coarse grid) for the same p -value 10^{-6} . Thus, balance can finally be ruled out as a major factor for measures of the significance of association group.

These findings also allow us to visualise the association measures in the form of iso-lines in the (e, o) plane, replacing each non-central measure with a central approximation. Figure 3.14 shows iso-lines of t -score (t), Poisson, the centralised version of log-likelihood ($(G^2)^c$) and z -score with Yates' correction applied (z_{corr}) for $p\nu = 10^{-6}$. For high expected frequencies ($e \geq 10$), all measures agree (t reaches good agreement only for $e \geq 100$). The overestimation of z_{corr} and the underestimation of t are clearly visible, while the Poisson iso-curve is very close to that of $(G^2)^c$. It is astonishing that the small gap between these two curves accounts for the considerable differences seen in the bottom right panel of Figure 3.9. The most interesting aspect of the graph is certainly the t -score measure, whose curve flattens out to a horizontal line for $e \rightarrow 0$. Unlike all other measures in this group, t -score sets an implicit frequency threshold: no pair type with $o \leq 22$ can achieve a significance of $p\nu = 10^{-6}$, regardless of its expected frequency. Even for the customary significance level of $p\nu = .01$, there is an implied frequency cutoff at $o = 5$. This unique property of t -score might explain its success for filtering out unwanted candidates in collocation extraction tasks (Church *et al.* 1991), where it has possibly worked more as a frequency filter than as a test of significance.

To sum up, the measures in the significance of association group are represented by a theoretically motivated **prototype**, namely the Fisher measure. For practical applications, log-likelihood is a convenient and numerically unproblematic alternative that gives very good approximations to the exact p -values. Its centralised version can be used for visualisation in the (e, o) plane and for empirical studies, with only minor score differences for unbalanced data points. The Poisson measure achieves the best approximation among the inherently central measures and its elegant form is useful

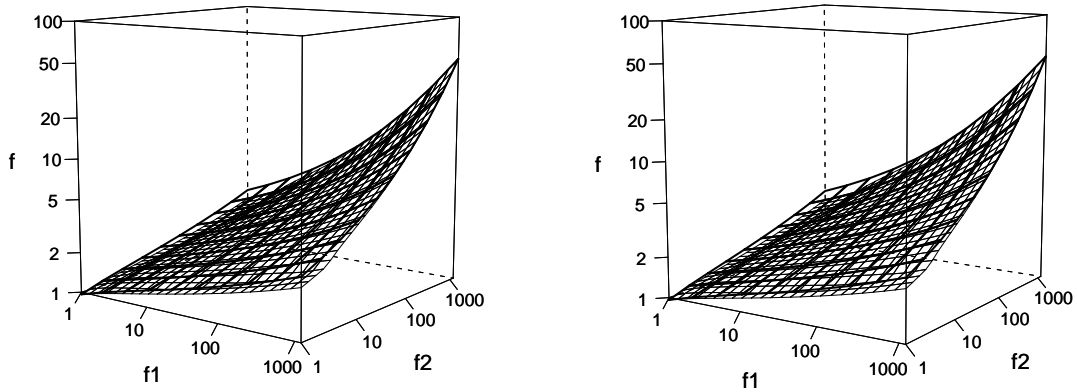


Figure 3.13: Iso-surfaces of the log-likelihood measure g (fine grid) and its centralised version g^c (coarse grid) for the same threshold value (corresponding to $p\nu = 10^{-6}$).

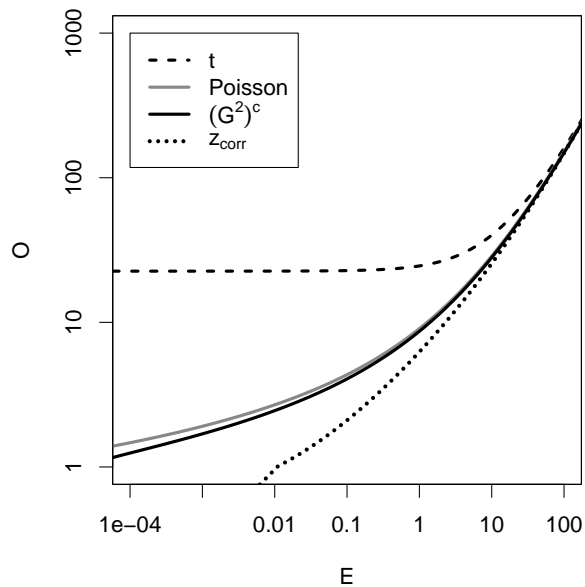


Figure 3.14: Iso-lines for t-score, Poisson, the centralised version of log-likelihood and z-score with Yates' correction applied (all corresponding to $p\nu = 10^{-6}$).

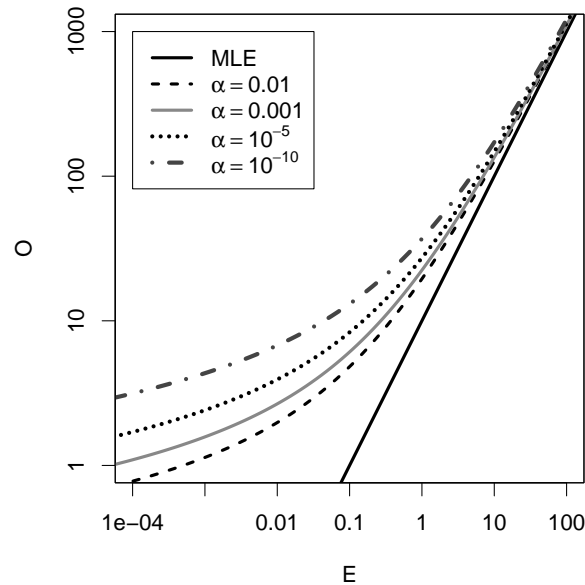


Figure 3.15: Iso-lines for the MI measure as a point estimate (MLE) of $\log_{10} \mu$ and conservative estimates for different confidence levels α ($MI_{\text{conf},\alpha}$ measure).

for mathematical discussions.

Degree of association

This group, which includes all the maximum-likelihood estimates for coefficients of association strength from Section 3.1.5, is much more diverse than the significance of association group. Since the various coefficients have quite different properties, there is no obvious group prototype.

Some measures or coefficients form subgroups that can be represented by a single prototype, or that are even fully equivalent. One example are the measures from the MI family, including the heuristic MI^k variants as well as $g\text{mean}$. Another example are Dice (with the equivalent Jaccard measure) and MS, which differ only in how the balance b affects the scores as can be seen from Eq. (3.12a) and (3.12c). In either case, the scores decrease for $b \neq 1$. Interestingly, the centralised versions of all three measures are equivalent to $g\text{mean}$, but the balance-dependency is so strong that the latter cannot be used as a prototype.

Conservative estimates for some coefficients of association strength were presented in Section 3.1.6. These association measures are parametric because they depend on the chosen confidence level (cf. Section 3.1.8), so they cannot be represented by a single prototype. However, each parametric family of measures can be compared to, and grouped with, the corresponding maximum-likelihood estimate. Figure 3.15 compares the maximum-likelihood estimate for $\log_{10} \mu$ (given by the MI measure) with conservative estimates (given by $MI_{\text{conf},\alpha}$) at different confidence levels α . The iso-lines in this graph represent an estimated value of $\log_{10} \mu = 1$. The differences between the MLE and the conservative estimates quickly become huge for $e < 1$, while they are practically indistinguishable for $e \geq 10$. This suggests that conservative estimates may indeed be able to overcome the overestimation bias of MI

for low-frequency data.

It is more difficult to visualise the properties of other measures in this group, such as Dice, MS and odds-ratio, because of their non-centrality. In these cases, three-dimensional graphs of iso-surfaces as in Figure 3.8 would be required.

Other measures

Of the remaining association measures, all three information-theoretic measures are fully or nearly equivalent to one from either of the two main groups: MI can be interpreted as a maximum-likelihood estimate of $\log_{10} \mu$ (and has been introduced as such in Section 3.1.5), average-MI is fully equivalent to log-likelihood, and local-MI can be seen as an approximation of the Poisson-Stirling measure.³² Therefore, these measures need not be treated separately, despite their different theoretical background.

The t-score measure, on the other hand, has unique properties that suggest that it may not belong into the significance of association group (although it agrees with the other measures for high expected frequencies).

The frequency measure ranks a data set by cooccurrence frequencies and is the intuitive non-mathematical choice for collocation extraction, based on the intuition that collocations are recurrent combinations. It is thus used as a non-statistical baseline for the evaluation experiments in Chapter 5. Again, the measure is most clearly characterised by its iso-lines in the (e, o) plane, which are parallel to the e -axis (compare this to the iso-lines of t-score for $e \rightarrow 0$).

There is an almost infinite range of possibilities for defining combined and parametric measures (cf. Section 3.1.8, and these will naturally be difficult to classify. The t-score measure provides an interesting example with its implied frequency threshold: it can be seen as a combination of the frequency measure (for $e < 1$) with a conservative significance of association measure (for $e \geq 10$).

³²Poisson-Stirling and local-MI are excluded from further discussion on the grounds that they cannot be extended to sound generalised association measures.

Chapter 4

Quantisation Effects

4.1 Frequency distributions

4.1.1 A thought experiment

Imagine a population consisting of 500 high-frequency pair types, each one of which occurs once every two thousand tokens ($\pi = 5 \cdot 10^{-4}$), and 750 000 low-frequency pair types, each one of which occurs once in a million pair tokens ($\pi = 10^{-6}$). Note that this is indeed a valid probability distribution because $500 \cdot 5 \cdot 10^{-4} + 750\,000 \cdot 10^{-6} = 0.25 + 0.75 = 1$. Assume further that all component types occur once in a thousand tokens ($\pi_1 = \pi_2 = 10^{-3}$), so that the null probability of *any* pair type under the independence hypothesis H_0 is $\pi_1 \cdot \pi_2 = 10^{-6}$. Thus, the low-frequency types are random combinations of their components (since they satisfy the null hypothesis $\pi = \pi_1\pi_2$), while the high-frequency types show strong positive association ($\pi \gg \pi_1\pi_2$).¹

If we take a sample of size $N = 2\,000$ from this population, most of the high-frequency pair types will occur exactly once ($O_{11} = f = 1$, which is the expected value given the true cooccurrence probability $\pi = 5 \cdot 10^{-4}$) or not at all in the sample. The expected marginal frequencies are $f_1 = f_2 = 2$, both for high-frequency and for low-frequency pair types. For the sake of the argument, I will ignore sampling variation of the marginal frequencies in the following discussion, so that we have $f_1 = f_2 = 2$ and the expected cooccurrence frequency (under the point null hypothesis of independence H'_0) is $E_{11} = f_1f_2/N = 0.002$ for *all* pair types in the sample.² Note

¹It is in fact possible to construct such a population. Each component set C_1 and C_2 consists of 1 000 types with equal marginal probabilities $\pi_1 = 10^{-3}$ and $\pi_2 = 10^{-3}$. Both sets are divided into two subsets of equal size: $C_1 = \{u_1, \dots, u_{500}, u'_1, \dots, u'_{500}\}$ and $C_2 = \{v_1, \dots, v_{500}, v'_1, \dots, v'_{500}\}$. Each type u'_i combines randomly with all $v_j \in C_2$, each v'_i combines randomly with all $u_j \in C_1$, and all $u'_i \in C_1$ and $v'_j \in C_2$ also combine randomly, yielding 750 000 pair types with $\pi = 10^{-6}$. In addition, each u_i combines with v_i but no other of the v_j ($j \neq i$), yielding another 500 pair types whose probability parameters must be equal to $\pi = 5 \cdot 10^{-4}$ in order to satisfy the summation conditions for marginal probabilities.

²This simplification allows us to ignore the differences between the general null hypothesis H_0 (where the expected number of random cooccurrences, given the true marginal probabilities π_1 and π_2 , equals $N\pi_1\pi_2$) and the point null hypothesis H'_0 (where the marginal probabilities are estimated from the sample frequencies, $\pi_1 \approx p_1$ and $\pi_2 \approx p_2$). Even when sampling variation is taken into account, the value $E_{11} = Np_1p_2 = f_1f_2/N$ estimated from the observed frequencies is unlikely to be much larger than $N\pi_1\pi_2$, so that the conclusions of the thought experiment remain fully valid. In particular, the computed association scores will not be much lower than the idealised scores used here.

that E_{11} is identical to the expected number of random cooccurrences given the true marginal probabilities: $E_{11} = N\pi_1\pi_2$.

Under these circumstances, even a single instance of a pair type in the sample is considered significant evidence for a positive association. With $O_{11} = 1$ and $E_{11} = 0.002$, the Poisson measure (as a representative of the significance of association group, cf. Section 3.4.2) computes a p -value of $p\nu \approx 0.002$ for these single occurrences, which are called *hapax legomena* (see Baayen 2001, 8). The degree of association is estimated by $MI = \log 500$, corresponding to a μ -value of $\mu = 500$ (cf. Section 2.2.5), and the other measures from this group compute similarly high scores. Even the 95% confidence interval for the μ -value (corresponding to the MI_{conf} measure introduced in Section 3.1.6) gives evidence for some degree of association ($\mu > 12.6$). For the strongly associated high-frequency pair types, this is the desired behaviour.

However, with only 500 high-frequency pairs in the population the remaining approx. 1 500 hapax legomena in the sample must belong to the low-frequency class.³ Although these cooccurrences are indeed pure coincidence (because the low-frequency pair types are random combinations), they obtain the same association scores as the high-frequency types. Thus, the degree of association is greatly overestimated compared to its true value of $\mu = 1$. What is even more disturbing is the apparent failure of statistical hypothesis tests to correct for the effects of chance. The p -value of $p\nu \approx 0.002$ computed by the significance-of-association measures indicates a risk of one in 500 for a non-associated pair type (satisfying H_0) to appear once or more in the sample. How can it be, then, that more than three quarters of all hapax legomena are such chance cooccurrences?

The answer to this question lies in a combination of three effects: (i) the very large number of low-frequency pair types in the population, (ii) the different statistical properties of single events vs. classes of events, and (iii) the quantisation of frequency counts. A statistical hypothesis test as used above predicts how likely it is for one particular low-frequency pair type, chosen *a priori*, to occur in the sample, namely $\Pr(X_{11} \geq 1) \approx 0.002$. (This is the p -value computed by Poisson and similar measures, and it is also the basis for conservative estimates such as MI_{conf} .) Although the occurrence probability is fairly small for each individual type, the large number of low-frequency types causes some of them to “leak through” into the sample. (The decisive factor is the total probability mass of such pair types, which is 0.75 in our thought experiment. Consequently, about three quarters of all pair tokens in the sample will be random cooccurrences.) Therefore, when we look at a *class* of types chosen *a posteriori*, namely the class of hapax legomena in the sample, the proportion of low-frequency pair types is determined as much by the shape of the population as by the individual occurrence probabilities. Quantisation effects allow the influence of the shape of the population to become dominant for lowest-frequency

Monte Carlo simulation shows that E_{11} tends to be larger for the high-frequency pair types than for the low-frequency ones, so that the random combinations will on average obtain even higher association scores than the associated pairs!

³On average, each of the 500 high-frequency pair types occurs once in a sample of size $N = 2000$. Therefore, the high-frequency types account for a total of approximately 500 tokens in the sample. The remaining $\approx 1\,500$ tokens must belong to the low-frequency class. Since low-frequency pair types are highly unlikely to occur more than once (in a sample of this size), almost all of these tokens will be hapax legomena.

data, especially the hapax legomena. In the thought experiment, this influence all but vanishes for higher frequency ranks. The expected number of high-frequency pair types among the *dis legomena* (double occurrences, i.e. $O_{11} = 2$) is 184, with hardly any low-frequency types present (less than five such types with 98% certainty). Here, the statistical tests have successfully filtered out random cooccurrences.

4.1.2 Introduction to lexical statistics

The thought experiment in Section 4.1.1 has demonstrated the important role that the distribution of probability parameters in the population plays for word cooccurrences. This is the domain of lexical statistics and word frequency distributions – see Baayen (2001, Ch. 2) for an introduction, notation, and detailed proofs. Here, I will just review the key concepts and some fundamental results.

The theory of lexical statistics provides a different perspective on the random sample model introduced in Section 2.2. While the statistical methods considered in previous chapters (on which most association measures are based) are applied to individual pair types, we will now study the behaviour of classes of types (without respect to the particular types making up the class) and the distribution of probability parameters in the population. For the purposes of lexical statistics, we only consider the pair types w and their cooccurrence probabilities π , ignoring the components and marginal probabilities. The population types are enumerated w_1, \dots, w_S such as to arrange their probability parameters in descending order: $\pi_1 \geq \pi_2 \geq \dots \geq \pi_S$ (in this chapter, π_1 refers to the cooccurrence probability of the pair type w_1 rather than a marginal probability). Likewise, the random variable f_i represents the cooccurrence frequency of the i -th pair type w_i in a sample. The **population size** S is the number of different types in the population. S may be finite ($S \in \mathbb{N}$) or infinite ($S = \infty$), with $\{1, \dots, S\}$ standing for the full set \mathbb{N} in the latter case. Work in lexical statistics usually assumes independent Poisson sampling (and so does related work, e.g. Good (1953)), so that the f_i are *independent* Poisson-distributed random variables:

$$\Pr(f_i = k) = e^{-N\pi_i} \frac{(N\pi_i)^k}{k!}. \quad (4.1)$$

Since we are not interested in individual type frequencies, but rather in their distribution across the entire population, all types w_i with the same frequency $f_i = m$ are collected into the **frequency class** m . The class size V_m , i.e. the number of different types in frequency class m , can easily be determined from the observed sample. In the statistical model, it can be defined as a sum over indicator variables:

$$V_m := \sum I_{[f_i=m]}. \quad (4.2)$$

The sequence of all class sizes (V_1, V_2, \dots) is called the **frequency spectrum**. Note that all but finitely many of the V_m equal zero (in particular, the largest non-empty frequency class is $V_{f_1^*}$). Using the same definition, V_0 is the number of unseen types and cannot be determined from the sample. The vocabulary size V is the total number of types observed in the sample:

$$V := \sum_{i=1}^S I_{[f_i>0]}.$$

The frequency spectrum is related to V and N through the identities $V = \sum_{m=1}^{\infty} V_m$ and $N = \sum_{m=1}^{\infty} mV_m$. The expectations of V and V_m can easily be computed from (4.1) and (4.2):

$$E[V_m] = \sum_{i=1}^S e^{-N\pi_i} \frac{(N\pi_i)^m}{m!} \quad \text{and} \quad E[V] = \sum_{i=1}^S (1 - e^{-N\pi_i}), \quad (4.3)$$

but it is more difficult to obtain variances and the full distributions. The variances are related to the expected values for a sample of twice the size:

$$\text{Var}[V_m(N)] = E[V_m(N)] - \binom{2m}{m} 2^{-2m} E[V_{2m}(2N)] \quad (4.4a)$$

$$\text{Var}[V(N)] = E[V(2N)] - E[V(N)] \quad (4.4b)$$

(Baayen 2001, 120–121).⁴ A **population model** describes the distribution of type probabilities in the population, based on a small set of parameters (usually two or three).⁵ While it is in principle possible to formulate a population model directly for the type probability parameters (e.g. Holgate 1969), it is usually more convenient to refer to the **structural type distribution**, which is a step function given by

$$G(\rho) := |\{i \in \{1, \dots, S\} \mid \pi_i \geq \rho\}|. \quad (4.5)$$

$G(\rho)$ specifies the number of types whose occurrence probability is $\geq \rho$. $E[V_m]$ and $E[V]$ can then be expressed in terms of Stieltjes integrals

$$E[V_m] = \int_0^{\infty} \frac{(N\pi)^m}{m!} e^{-N\pi} dG(\pi), \quad E[V] = \int_0^{\infty} (1 - e^{-N\pi}) dG(\pi) \quad (4.6)$$

(Baayen 2001, 47f). Most population models approximate $G(\rho)$ by a continuous function with the **type density** $g(\pi)$, i.e.

$$G(\rho) = \int_{\rho}^{\infty} g(\pi) d\pi. \quad (4.7)$$

Note the use of $+\infty$ as an upper integration limit although all type probabilities must fall into the range $0 \leq \pi \leq 1$. This device allows for more elegant mathematical formulations, but care has to be taken that $G(1) \ll 1$ (otherwise the model would predict the existence of types with $\pi > 1$). For a population model based on a type density function $g(\pi)$, the expectations of V_m and V become

$$E[V_m] = \int_0^{\infty} \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi, \quad E[V] = \int_0^{\infty} (1 - e^{-N\pi}) g(\pi) d\pi, \quad (4.8)$$

and the variances can again be computed from (4.4). The normalisation condition for type density functions is

$$\int_0^{\infty} \pi \cdot g(\pi) d\pi = 1, \quad (4.9)$$

and the population size is given by $S = \int_0^{\infty} g(\pi) d\pi$.

⁴These equations, which Baayen describes as approximations, are exact when independent Poisson sampling is assumed.

⁵Baayen (2001) uses the term LNRE model for such a population model, where LNRE stands for Large Number of Rare Events, a term introduced by Khmaladze (1987). It refers to the very large number of types with low occurrence probabilities that are characteristic of word frequency distributions and the associated population models.

4.1.3 The conditional parameter distribution

As we have seen from the thought experiment in Section 4.1.1, the critical problem of low-frequency data is that the observed frequency O_{11} may be much higher than the expected value $E[X_{11}]$ (given the true cooccurrence probabilities), leading to inflated estimates for coefficients of association strength such as the MI measure. This effect is much greater than predicted by the sampling distribution, and thus statistical hypothesis tests cannot correct for it. As a consequence, both the significance-of-association measures and conservative estimates for coefficients of association strength are subject to the same overestimation. In Section 4.1.1, the problem was brought down to a comparison of (a) the probability $\Pr(X \geq m)$ that a type w is observed at least m times in the sample, under the hypothesis that its probability parameter satisfies $\pi \leq \rho$ for some value ρ ; and (b) the proportion of types in frequency class m whose probability parameter does indeed satisfy $\pi \leq \rho$. The latter is strongly influenced by the population distribution. If it is much larger than the probability (a), statistical tests (and all inferences and association measures based on them) will fail to control the risk of type I errors properly. The extent of this failure is given by the ratio between (b) and (a).

Our goal in this section is to compute the proportion (b) – or rather its sampling distribution, since it is a random variable – from a population model. This will allow us to estimate the consequences of quantisation effects given assumptions about the population (in the form of the population model). Let $V_{m,\rho}$ stand for the number of types in frequency class m with probability parameter $\pi \leq \rho$, and $V_{m,>\rho}$ for the number of types with parameter $\pi > \rho$:

$$V_{m,\rho} := \sum_{\pi_i \leq \rho} I_{[f_i=m]} \quad \text{and} \quad V_{m,>\rho} := \sum_{\pi_i > \rho} I_{[f_i=m]}. \quad (4.10)$$

Since $V_{m,\rho}$ and $V_{m,>\rho}$ are obtained by summation over disjoint sets of types, they are independent (for the same value ρ). We can easily compute the expectation of $V_{m,\rho}$ and $V_{m,>\rho}$ from a population model in the form of a type density function, using Eq. (4.8):

$$E[V_{m,\rho}] = \int_0^\rho \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi \quad (4.11a)$$

and

$$E[V_{m,>\rho}] = \int_\rho^\infty \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi. \quad (4.11b)$$

The corresponding variances can then be obtained from Eq. (4.4). The proportion of low-probability types in frequency class m is given by the ratio $R_{m,\rho} := V_{m,\rho}/V_m$. Unfortunately, the computation of $E[V_{m,\rho}/V_m]$ leads to a mathematical problem that I have not solved (Good 1953, 242). However, given that the sample size N is large, the value of ρ is not too extreme and we are only interested in small m , the distributions of V_m , $V_{m,\rho}$ and $V_{m,>\rho}$ are approximately normal by the central limit theorem (since each is the sum of a large number of independent indicator variables, cf. (4.10)). Writing

$$R_{m,\rho} = \frac{V_{m,\rho}}{V_m} = \frac{V_{m,\rho}}{V_{m,\rho} + V_{m,>\rho}}, \quad (4.12)$$

we can express the proportion $R_{m,\rho}$ as a function of two independent, approximately normal random variables $V_{m,\rho}$ and $V_{m,>\rho}$. Lemma A.8 derives the distribution of $R_{m,\rho}$ and shows that – except for some extreme cases – it is approximately normal and the expectation is given by

$$E[R_{m,\rho}] \approx \frac{E[V_{m,\rho}]}{E[V_m]} . \quad (4.13)$$

In Section 4.3, we will use (4.13) as an estimate for the average value of $R_{m,\rho}$ to study the relation between the proportion of low-probability types and the p -values computed by statistical hypothesis tests. Some example calculations for the population models of Section 4.2 fitted to the data sets described in 4.2.4 have shown that the standard deviations of $V_{m,\rho}$ and $V_{m,>\rho}$ are much smaller than their expected values, so that the approximations of Lemma A.8 are indeed valid. Moreover, the relative standard error of $R_{m,\rho}$ is almost always below 1%, which implies that $E[R_{m,\rho}]$ is a good estimate for the proportion of low-probability types in any given sample.

4.2 The Zipf-Mandelbrot population model

4.2.1 Zipf’s law

Zipf’s law (Zipf 1949), which states that the frequency of the r -th most frequent type is proportional to $1/r$, was originally formulated for the Zipf ranking of observed frequencies ($f_r^* \approx Cr^{-1}$) and (more or less equivalently) for the observed frequency spectrum ($V_m \approx C/m(m+1)$). In its first form, Zipf’s law describes a fascinating property of the higher-frequency words in a language, for which explanations related to Zipf’s principle of least effort have been put forward (e.g. Mandelbrot 1962; Powers 1998). In its second form, it is a statement about the enormous abundance of lowest-frequency types, which has many consequences for the statistical analysis of word frequency data and for applications in natural-language processing.

It has long been known that the word frequency distributions obtained from random text are strikingly similar to Zipf’s law (Miller 1957; Li 1992). Formally, random text is understood as a character sequence generated by a Markov process, with word boundaries indicated by a special “space” character. Rouault (1978) shows that, under very general conditions, this segmented character sequence is equivalent to a random sample of words (with replacement, corresponding to the model introduced in Section 4.1.2) and that the population probabilities of low-frequency types asymptotically satisfy the Zipf-Mandelbrot law

$$\pi_i = \frac{C}{(i+b)^a} \quad (4.14)$$

with parameters $a > 1$ and $b > 0$ (Baayen 2001, 101ff). In Sections 4.2.2 and 4.2.3, I will formulate population models for random character sequences based on the Zipf-Mandelbrot law. Although Baayen remarks that “for Zipf’s harmonic spectrum law and related models, no complete expression for the structural type distribution is available” (Baayen 2001, 94), this need not discourage us: (4.14) refers to the population parameters rather than to the observed Zipf ranking. The Zipf-Mandelbrot

law for random text is a population model, while the original formulation of Zipf's law and its variants (Baayen 2001, 94f) have a purely descriptive nature.

These considerations open up an entirely new perspective on Zipf's law: If a population model based on (4.14) can be shown to agree with the observed data, we must conclude that – as far as statistical analysis is concerned – such language data are not substantially different from random text. As a consequence, the statistical analysis faces all the problems of making sense from random noise, and these problems can be predicted with the population models of Sections 4.2.2 and 4.2.3.

One of the characteristics of random text is an infinite population size, since there can be words of arbitrary length, leading to an extremely skewed population distribution. It has often been noted that this does not accord well with real-world data, especially when there are narrow restrictions and the data have been cleaned up manually. Examples are studies of (morphological) productivity (e.g. Baayen and Renouf 1996) or the word frequency distributions of small literary texts (see Baayen 2001). However, the situation is different when one considers “raw” data obtained from a large corpus of hundreds of millions of words, which is the input that statistical methods in natural-language processing typically have to deal with. The similarity to random text becomes even more striking for combinations of two or more words (cf. Baayen 2001, 221). Most techniques for the extraction of collocations from text corpora apply statistical independence tests to such base material (e.g. Evert and Krenn 2001), and are thus also affected by the consequences of the Zipf-Mandelbrot law. Ha *et al.* (2002) demonstrate this effect for Mandarin Chinese ideographs: while the number of different graphs is comparatively small and does not exhibit a highly skewed LNRE distribution, the situation changes when sequences of two or more such graphs are examined. The longer the sequences, the more closely their frequency distribution agrees with the Zipf-Mandelbrot law.

4.2.2 The Zipf-Mandelbrot model

In order to derive a useful population model from the Zipf-Mandelbrot law, it is necessary to reformulate (4.14) in terms of a type density function $g(\pi)$. The structural type distribution corresponding to the Zipf-Mandelbrot law is a step function with $G(\pi_i) = i$ (since there are exactly i types with $\pi \geq \pi_i$, namely w_1, \dots, w_i). Solving (4.14) for i , we obtain

$$G(\pi) = \frac{C^{1/a}}{\pi^{1/a}} - b \quad (4.15)$$

for $\pi = \pi_i$, and $G(\pi)$ is constant between these steps. Differentiation of (4.15) suggests a type density of the form

$$g(\pi) := \begin{cases} C \cdot \pi^{-\alpha-1} & 0 \leq \pi \leq B \\ 0 & \text{otherwise} \end{cases} \quad (4.16)$$

with two free parameters $0 < \alpha < 1$ and $B > 0$.⁶ The normalising constant C can be determined from (4.9):

$$1 = \int_0^B \pi g(\pi) d\pi = \int_0^B C\pi^{-\alpha} d\pi = C \cdot \left[\frac{\pi^{1-\alpha}}{1-\alpha} \right]_0^B = C \cdot \frac{B^{1-\alpha}}{1-\alpha},$$

which evaluates to

$$C = \frac{1-\alpha}{B^{1-\alpha}}.$$

The ZM model describes an infinite population, since $S = \int_0^B g(\pi) d\pi = \infty$, and its structural type distribution

$$\begin{aligned} G(\rho) &= \int_\rho^B g(\pi) d\pi = C \cdot \int_\rho^B \pi^{-\alpha-1} d\pi = C \cdot \left[\frac{\pi^{-\alpha}}{-\alpha} \right]_\rho^B \\ &= \frac{C \cdot \rho^{-\alpha}}{\alpha} - \frac{C \cdot B^{-\alpha}}{\alpha} = \frac{C/\alpha}{\rho^\alpha} - \frac{1-\alpha}{B \cdot \alpha} \end{aligned}$$

is identical to (4.15) with $a = \alpha^{-1}$ and $b = (1-\alpha)B^{-1}\alpha^{-1}$ for any values of ρ where $G(\rho) \in \mathbb{N}$. Thus, (4.16) can indeed be understood as a continuous extension of the Zipf-Mandelbrot law.

$$\begin{aligned} E[V_m] &= \int_0^\infty \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi = \frac{C}{m!} \int_0^B (N\pi)^m e^{-N\pi} \pi^{-\alpha-1} d\pi \\ &= \frac{C}{m!} \int_0^{NB} t^m e^{-t} \left(\frac{t}{N} \right)^{-\alpha-1} \frac{1}{N} dt = \frac{C}{m!} N^\alpha \int_0^{NB} t^{m-\alpha-1} e^{-t} dt \\ &\approx \frac{C}{m!} N^\alpha \int_0^\infty t^{m-\alpha-1} e^{-t} dt \end{aligned}$$

In the second line, the substitution $t := N\pi$ has been made. The approximation in the last line is justified for $NB \gg m$ (which should always be the case for the large samples that are of interest here) where the integral $\int_{NB}^\infty t^{m-\alpha-1} e^{-t} dt$ is vanishingly small. Thus, $E[V_m]$ is reduced to the Gamma integral (A.26) and we obtain the concise expression

$$E[V_m] = \frac{C}{m!} \cdot N^\alpha \cdot \Gamma(m-\alpha). \quad (4.17)$$

⁶The constraints on the parameter α follow from $0 < 1/a < 1$. C is a normalising constant and will be determined from (4.9). The upper cutoff point B is necessary since the model would predict types with probability $\pi > 1$ otherwise. B should roughly correspond to the probability π_1 of the most frequent type.

The computation of $E[V]$ involves an improper integral solved by partial integration:

$$\begin{aligned}
E[V] &= \int_0^\infty (1 - e^{-N\pi})g(\pi) d\pi \approx CN^\alpha \int_0^\infty (1 - e^{-t})t^{-\alpha-1} dt \\
&= CN^\alpha \cdot \lim_{A \downarrow 0} \left(\int_A^\infty t^{-\alpha-1} dt - \int_A^\infty e^{-t}t^{-\alpha-1} dt \right) \\
&= CN^\alpha \cdot \lim_{A \downarrow 0} \left(\left[\frac{t^{-\alpha}}{-\alpha} \right]_A^\infty - \left[e^{-t} \frac{t^{-\alpha}}{-\alpha} \right]_A^\infty - \int_A^\infty e^{-t} \frac{t^{-\alpha}}{-\alpha} dt \right) \\
&= CN^\alpha \cdot \lim_{A \downarrow 0} \left(\underbrace{(1 - e^{-A}) \cdot \frac{A^{-\alpha}}{\alpha}}_{= O(A^{1-\alpha}) \rightarrow 0} + \underbrace{\frac{\Gamma(1 - \alpha, A)}{\alpha}}_{\rightarrow \Gamma(1-\alpha)/\alpha} \right)
\end{aligned}$$

where $\Gamma(1 - \alpha, A)$ is the upper incomplete Gamma function (see A.4, Eq. (A.29)).

$$E[V] = C \cdot N^\alpha \cdot \frac{\Gamma(1 - \alpha)}{\alpha}. \quad (4.18)$$

Consequences of (4.17) and (4.18) are the recurrence relation

$$\frac{E[V_{m+1}]}{E[V_m]} = \frac{\Gamma(m + 1 - \alpha)}{(m + 1)!} \cdot \frac{m!}{\Gamma(m - \alpha)} = \frac{m - \alpha}{m + 1}, \quad (4.19)$$

a relative frequency spectrum

$$\frac{E[V_m]}{E[V]} = \frac{\alpha \cdot \Gamma(m - \alpha)}{\Gamma(m + 1) \cdot \Gamma(1 - \alpha)} \quad (4.20)$$

which is independent of the sample size N (cf. Baayen 2001, 118), and a power law

$$E[V(N)] = C' \cdot N^\alpha \quad \text{with} \quad 0 < \alpha < 1 \quad (4.21)$$

for the vocabulary growth curve. Equation (4.21) is known as Herdan's law (Herdan 1964) in quantitative linguistics and as Heaps' law (Heaps 1978) in information retrieval.

The appeal of the ZM model lies in its mathematical elegance and numerical efficiency. Computation of the expected frequency spectrum and similar statistics is fast and accurate, using the complete and incomplete Gamma function. Moreover, due to the simple form of $g(\pi)$, we obtain a closed-form expression for the expected number of low-probability types

$$\begin{aligned}
E[V_{m,\rho}] &= \int_0^\rho \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi \\
&= \frac{C}{m!} \cdot N^\alpha \cdot \gamma(m - \alpha, N\rho)
\end{aligned} \quad (4.22)$$

and the corresponding proportion

$$E[R_{m,\rho}] \approx \frac{E[V_{m,\rho}]}{E[V_m]} = \frac{\gamma(m - \alpha, N\rho)}{\Gamma(m - \alpha)} \quad (4.23)$$

for $0 < \rho < B$.

4.2.3 The finite Zipf-Mandelbrot model

Although the ZM model is theoretically well-founded as a model for random character sequences, its assumption of an infinite vocabulary is unrealistic for natural-language data. In order to achieve a better approximation of such frequency distributions, the finite ZM model introduces an additional lower cutoff point $A > 0$ for the type density:

$$g(\pi) := \begin{cases} C \cdot \pi^{-\alpha-1} & A \leq \pi \leq B \\ 0 & \text{otherwise} \end{cases}, \quad (4.24)$$

which implies that there are no types with probability $\pi < A$ in the population. The normalising constant C is determined from (4.9):

$$1 = \int_A^B \pi g(\pi) d\pi = \int_A^B C \pi^{-\alpha} d\pi = C \cdot \left[\frac{\pi^{1-\alpha}}{1-\alpha} \right]_A^B = C \cdot \frac{B^{1-\alpha} - A^{1-\alpha}}{1-\alpha},$$

which evaluates to

$$C = \frac{1-\alpha}{B^{1-\alpha} - A^{1-\alpha}}. \quad (4.25)$$

The population size is

$$S = C \cdot \int_A^B \pi^{-\alpha-1} d\pi = \frac{C}{\alpha} \cdot (A^{-\alpha} - B^{-\alpha}) = \frac{1-\alpha}{\alpha} \cdot \frac{A^{-\alpha} - B^{-\alpha}}{B^{1-\alpha} - A^{1-\alpha}}. \quad (4.26)$$

Again, the structural type density $G(\rho)$ is identical to (4.15), with $G(\rho) = S$ for $\rho \leq A$. The expectations of V_m and V are calculated in analogy to those for the ZM model:

$$\begin{aligned} E[V_m] &= \int_0^\infty \frac{(N\pi)^m}{m!} e^{-N\pi} g(\pi) d\pi = \frac{C}{m!} \int_A^B (N\pi)^m e^{-N\pi} \pi^{-\alpha-1} d\pi \\ &= \frac{C}{m!} N^\alpha \int_{NA}^{NB} t^{m-\alpha-1} e^{-t} dt \approx \frac{C}{m!} N^\alpha \int_{NA}^\infty t^{m-\alpha-1} e^{-t} dt \end{aligned}$$

reduces by (A.29) to

$$E[V_m] = \frac{C}{m!} \cdot N^\alpha \cdot \Gamma(m - \alpha, NA). \quad (4.27)$$

For the calculation of

$$E[V] = \int_0^\infty (1 - e^{-N\pi}) g(\pi) d\pi \approx CN^\alpha \int_{NA}^\infty (1 - e^{-t}) t^{-\alpha-1} dt,$$

we use partial integration

$$\begin{aligned} \int_{NA}^\infty (1 - e^{-t}) t^{-\alpha-1} dt &= \int_{NA}^\infty t^{-\alpha-1} dt - \int_{NA}^\infty e^{-t} t^{-\alpha-1} dt \\ &= \left[\frac{t^{-\alpha}}{-\alpha} \right]_{NA}^\infty - \left[e^{-t} \frac{t^{-\alpha}}{-\alpha} \right]_{NA}^\infty - \int_{NA}^\infty e^{-t} \frac{t^{-\alpha}}{-\alpha} dt \\ &= \frac{(NA)^{-\alpha}}{\alpha} - e^{-NA} \frac{(NA)^{-\alpha}}{\alpha} + \frac{1}{\alpha} \int_{NA}^\infty e^{-t} t^{-\alpha} dt \\ &= (1 - e^{-NA}) \frac{N^{-\alpha} A^{-\alpha}}{\alpha} + \frac{\Gamma(1 - \alpha, NA)}{\alpha} \end{aligned}$$

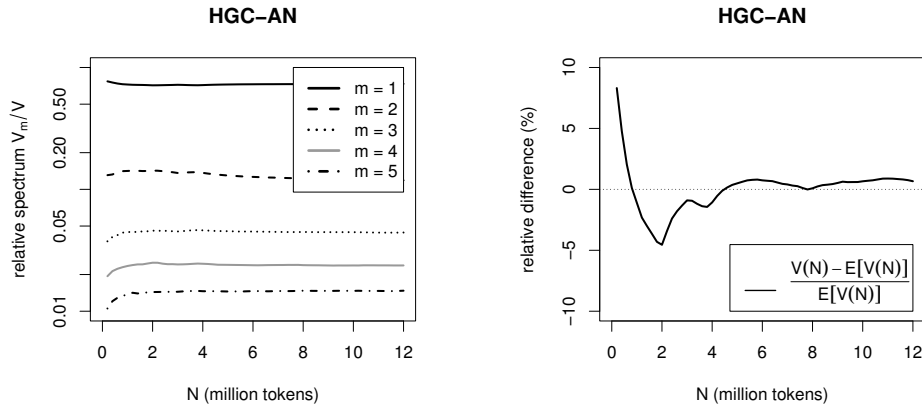


Figure 4.1: Development of relative frequency spectrum and relative error of Herdan law (Heaps' law) with $\alpha = 0.87$ for the AN-HGC data set.

to obtain

$$E[V] = C \cdot N^\alpha \cdot \frac{\Gamma(1 - \alpha, NA)}{\alpha} + \frac{C}{\alpha \cdot A^\alpha} (1 - e^{-NA}). \quad (4.28)$$

There are no simple expressions for the recurrence relation (4.19) and the relative frequency spectrum (4.20). Although much of the mathematical elegance of the ZM model has been lost, the fZM model is still numerically efficient and both $E[V_{m,\rho}]$ and $E[R_{m,\rho}]$ have closed-form solutions. For $A \leq \rho \leq B$, replacing the upper bound in the integral for $E[V_m]$ with ρ yields

$$E[V_{m,\rho}] = \frac{C}{m!} \cdot N^\alpha \cdot (\Gamma(m - \alpha, NA) - \Gamma(m - \alpha, N\rho)), \quad (4.29)$$

and in combination with (4.27)

$$E[R_{m,\rho}] \approx \frac{E[V_{m,\rho}]}{E[V_m]} = 1 - \frac{\Gamma(m - \alpha, N\rho)}{\Gamma(m - \alpha, NA)}. \quad (4.30)$$

4.2.4 Evaluation of the models

In order to see how well the ZM and fZM models describe real-world data, they have been applied to the AN-BNC and AN-HGC data sets. The Herdan law and the size-invariant relative frequency spectrum, which are characteristic properties of the ZM model, have repeatedly been criticised as unrealistic (e.g. Baayen 2001, 118). Figure 4.1 shows the development of the relative frequency spectrum up to $m = 5$ for the AN-HGC data set (left panel). After approximately 2 million tokens, the relative spectrum has converged and is nearly constant afterwards. Likewise, the relative error of the Herdan law $E[V(N)] = C \cdot N^\alpha$ with $\alpha = 0.87$ (determined by linear regression) remains below 1% after the first 4 million tokens (right panel). This is a strong indication that the ZM and fZM models may indeed be well suited for the type of frequency distribution represented by these data sets.

Using an implementation provided as part of the UCS toolkit (see Appendix B.2), the ZM and fZM model were fitted to the two data sets. For the infinite ZM model,

data set	ZM model		fZM model		
	α	χ_{14}^2	α	S	χ_{13}^2
AN-BNC	0.7145849	313472.66	0.9168508	9 048 002	9364.46
AN-HGC	0.7441247	441448.77	0.9134667	37 983 975	1855.59

Table 4.1: Estimated shape parameter α , population size S , and goodness-of-fit statistic χ^2 for the ZM and fZM models applied to the AN-BNC and AN-HGC data sets.

the parameter α can be estimated directly from (4.20) for $m = 1$:

$$\alpha = \frac{E[V_1]}{E[V]} \approx \frac{V_1}{V} \quad (4.31)$$

(see also Rouault 1978, 172). However, Equation (4.31) turned out to give unsatisfactory results, so the parameters for both models were estimated through non-linear minimisation of a multinomial goodness-of-fit chi-squared statistic for the first 15 spectrum elements, with the additional constraint $E[V] = V$. Goodness-of-fit was then evaluated with a multivariate chi-squared test, following Baayen (2001, Sec. 3.3). The results are shown in Table 4.1.⁷

The fZM model achieves a considerably better approximation to the observed frequency spectrum than the ZM model on all data sets. Evert (2004b) shows that the fZM model also compares favourably with several other population models described in (Baayen 2001). A graphic representation of the accordance between the expected and observed frequency spectrum for the AN-HGC data set is shown in Figure 4.2. Surprisingly, the estimated lower cutoff points ($A = 9.267 \times 10^{-9}$ for AN-BNC and $A = 1.576 \times 10^{-9}$ for AN-HGC) are already quite close to the observed relative frequency of the hapax legomena ($p = 1/N$). According to the predictions of the fZM model, increasing the sample 100-fold ($N \approx 10^9$) would already leave the LNRE zone, with all expected frequencies greater than 1 (cf. Baayen 2001, Sec. 2.4). A possible explanation for this counter-intuitive result lies in the term clustering effects discussed in Section 2.3.2.

4.3 Interpretation of the theoretical results

4.3.1 Sample-size independent results (ZM model)

Figure 4.3 compares the p -values computed by a Poisson test for the types in a given frequency class m with the expected proportion $E[R_{m,\rho}]$ of low-probability types in this frequency class. Given a value $0 < \rho < 1$, the x -axis shows the expected frequency $N\rho$ of a pair type with $\pi = \rho$. The solid vertical line indicates the maximum-likelihood estimate for the unknown probability parameter π (again scaled to the cor-

⁷The multivariate chi-squared test for the ZM and fZM models is also implemented in the UCS toolkit. Note that the χ^2 statistic for the ZM model has $df = 14$ because 2 parameters were estimated from the observed spectrum. Likewise, the statistic for the fZM model with 3 estimated parameters has $df = 13$.

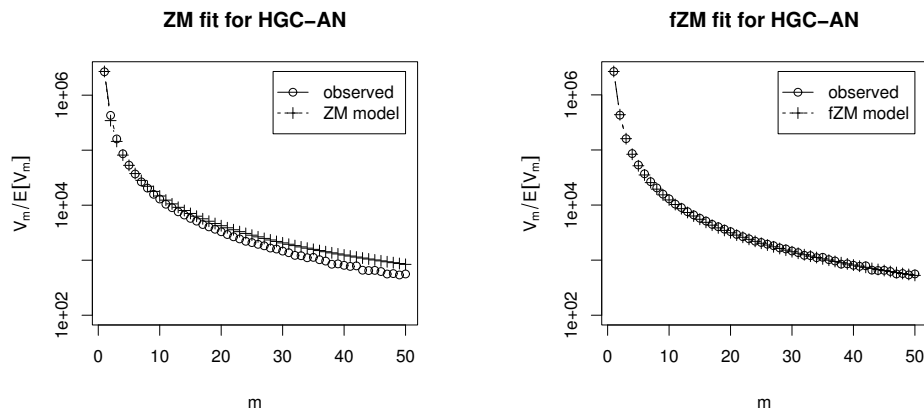


Figure 4.2: Expected frequency spectrum of ZM (left panel) and fZM (right panel) models compared to observed spectrum for the AN-HGC data set (logarithmic scale).

responding expected frequency $N\pi$), which is the same for all types in a frequency class. The solid curve gives the Poisson probability that a type with probability parameter $\pi = \rho$ will occur at least m times in the sample, $\Pr(O_{11} \geq m \mid \pi = \rho)$, which provides an estimate for the risk of an individual low-probability type with $\pi \leq \rho$ to appear in frequency class m purely by chance. The other curves show the expected proportion $E[R_{m,\rho}]$ of such low-probability types in frequency class m , for different values of the model parameter α . The model with $\alpha = 0.9$ represents a population with a particularly large number of low-probability types, corresponding to an exponent of $a \approx 1.11$ in the Zipf-Mandelbrot law. It is close to the shape parameter estimated for the fZM model on the AN-BNC and AN-HGC data sets. The population with $\alpha = 0.5$, on the other hand, is only moderately skewed, corresponding to a Zipf exponent of $a = 2$.

The top rows in Figure 4.3 show such graphs for frequency classes $m = 1, 2, 3$ and 5, while the bottom row gives a wider range of expected frequencies for $m = 1, 2$. Especially for the hapax and dis legomena, it is obvious that the proportion $R_{m,\rho}$ of low-probability types is considerably larger than the p -value computed by the Poisson test for individual types. This observation has two important consequences for the behaviour of association measures:

1. When ρ is interpreted as the cooccurrence probability under H'_0 , i.e. $\rho = p_1 p_2$, the solid curve indicates the p -value assigned by one of the significance-of-association measure to a pair type with $O_{11} = m$ and $E_{11} = N\rho$. For instance, a hapax legomenon ($O_{11} = 1$) with $E_{11} = 10^{-6}$ obtains a relatively high association score of $-\log_{10} p_v = -\log_{10} 10^{-6} = 6$. When sampling from a ZM population with shape parameter $\alpha = 0.9$, however, as many as 25% of all hapax legomena may be such low-probability types that satisfy H'_0 . In other words, the evidence for positive association attested by Poisson and similar measures is entirely spurious.
2. When coefficients of association strength are estimated from the observed data (either point estimates or conservative estimates), the maximum-likelihood estimate for the cooccurrence probability, $\pi \approx m/N$, usually plays a central role.

This estimate is indicated by a vertical line in Figure 4.3. The curves for $E[R_{m,p}]$ show the expected proportion of types in frequency class m for which this estimate is substantially too high. Again, for a ZM population with $\alpha = 0.9$, some 25% of the hapax legomena will have a true cooccurrence probability of $\pi \leq 10^{-6}/N$, and as many as 10% will even have $\pi \leq 10^{-10}/N$, so that the maximum-likelihood estimate is wrong by ten orders of magnitude. When conservative estimates are used, the Poisson test (or a similar statistical test) is meant to correct for this sampling error, but it will only reduce the estimate to $\pi \approx 10^{-2}/N$ (99% confidence) or $\pi \approx 10^{-3}/N$ (99.9% confidence), which is still off by several orders of magnitude for many of the pair types.

These problems are particularly serious because of the large number of lowest-frequency types that will be found in a sample from a Zipfian population. A proportion of 10% of the hapax legomena translates into a substantial number of pair types whose association is severely overestimated. Naturally, these effects are most pronounced for a highly skewed distribution ($\alpha = 0.9$ in the graphs) and for the lowest frequency classes $m = 1, 2$. While a moderately skewed distribution ($\alpha = 0.5$ in the graphs) still exhibits a considerable overestimation bias, the effect all but vanishes for $m = 5$ and higher frequency classes, irrespective of the shape parameter α .

4.3.2 Sample-size dependent results (fZM model)

Since the fZM model depends on two parameters (α and A) and its expected frequency spectrum and conditional parameter distribution are not size-independent, it is impossible to draw conclusions from it that are valid as generally as those in Section 4.3.1. However, we can plot the expected proportion of low-probability types for a specific population (estimated from a given data set) and for different sample sizes. Figure 4.4 shows the predictions of a fZM model fitted to the AN-HGC data set, for three different sample sizes. The four panels correspond to the top rows of Figure 4.3.

The overestimation exhibited by these graphs is much less severe than for the ZM model, except when the sample is very small (this effect is caused by the lower threshold A for population probabilities). Intriguingly, there is now a converse underestimation effect, due to the high value of A estimated from the AN-HGC data set. The conservative estimate for the cooccurrence probability of a hapax legomenon is $\pi \approx 10^{-2}/N$ (99% confidence). However, the dotted line in the top left panel of Figure 4.4 shows that there will be no pair types in the sample whose cooccurrence probability is this small (because $10^{-2}/N < A$). Again, the effect is reduced for $m = 3$ and all but vanishes for $m = 5$.

4.3.3 Discussion

The results presented in Sections 4.3.1 and 4.3.2 demonstrate that the distribution of type probability parameters among the hapax and dis legomena is entirely dominated by the shape of the population distribution. Depending on this distribution, statistical tests may drastically under- or over-correct for the effects of chance. Neither the ZM nor the fZM model is fully consistent with the observed data (which would be

surprising in view of the simplicity of these models), but both achieve a satisfactory goodness of fit that compares favourably with other widely-used population models (cf. Section 4.2.4). However, their predictions for quantisation effects in the lowest frequency classes (overestimation and underestimation of the true cooccurrence probabilities, respectively) are contradictory, and depend crucially on the sample size for the fZM model.

Since we must assume that neither of the models gives a fully accurate picture of the distribution of probability parameters in the population, it is impossible to correct for quantisation errors unless better population models become available. One point that is particularly disturbing is the high value of the cutoff threshold A estimated for the fZM model. This may well be the result of a distortion of the frequency spectrum by the clustering effects described in Section 2.3.2. If it is possible to correct for these effects, the threshold A may be lowered sufficiently to achieve better agreement between the predictions of the ZM and fZM models.

For the time being, however, we must assume that probability estimates and p -values for the lowest-frequency types are distorted in unpredictable ways. Fortunately, the influence of quantisation effects and the specific shape of the population is minimal for frequency classes $m \geq 5$, so that statistical inference is accurate. Taken together, these conclusions provide theoretical support for **frequency cutoff thresholds**. Data with cooccurrence frequency $f < 3$, i.e. the hapax and dis legomena, should *always* be excluded from the statistical analysis. On the other hand, the shape of the population has little effect for $f \geq 5$ and the data can safely be used.

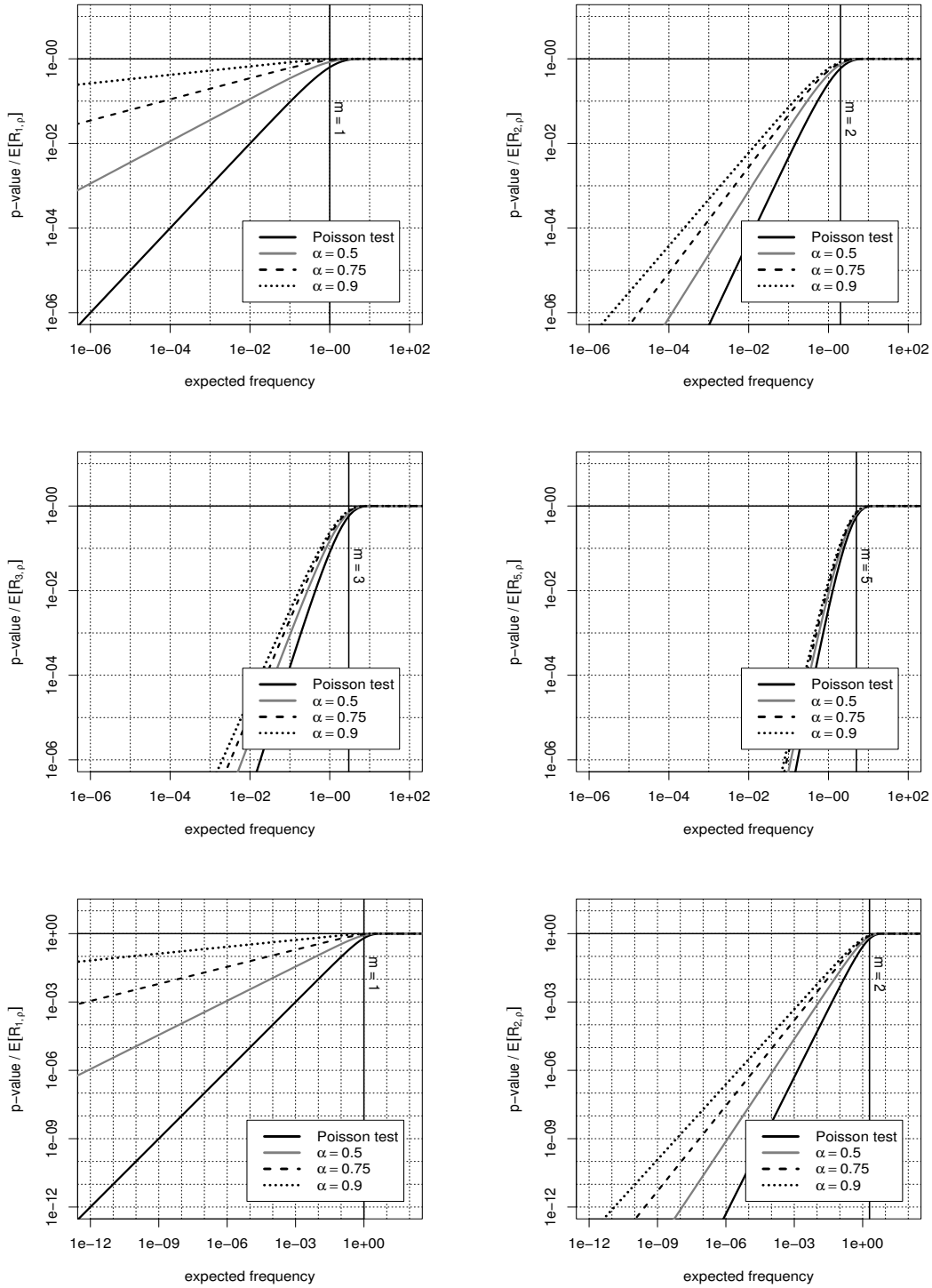


Figure 4.3: Comparison of the p -value computed by the Poisson association measure against the expected proportion of low-probability types in frequency classes $m = 1, 2, 3$ and 5 , for a population described by a ZM model with shape parameter α . The graphs in the bottom row cover a wider range of expected frequencies for $m = 1, 2$.

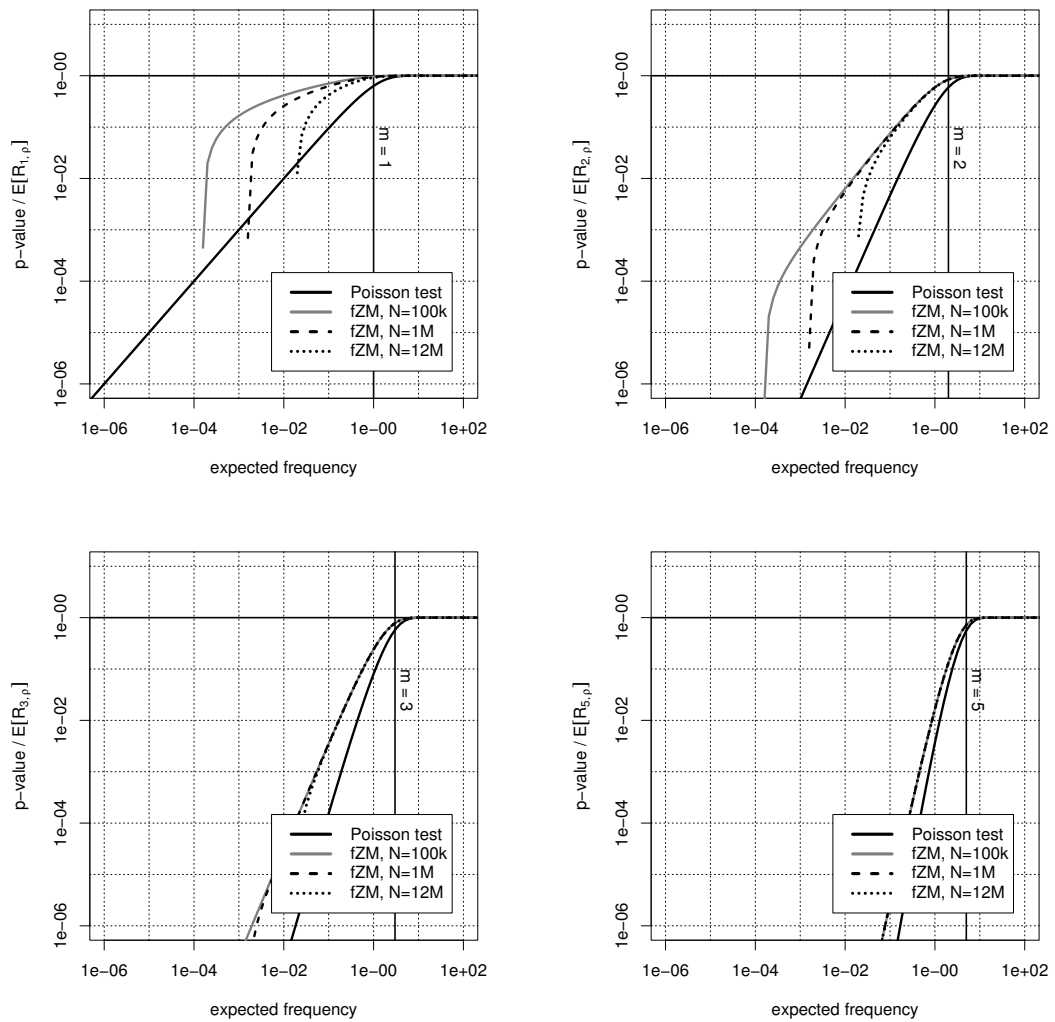


Figure 4.4: Comparison of the p -value computed by the Poisson association measure against the expected proportion of low-probability types in frequency classes $m = 1, 2, 3,$ and 5 . These graphs show the predictions of a fZM model estimated from the AN-HGC data set for three different sample sizes.

Chapter 5

Evaluation

5.1 Evaluation of association measures

With the wide range of association measures available, some guidance is needed for choosing an appropriate measure to be used in an application of cooccurrence data. While the theoretical discussion of Chapter 3 has helped to narrow down the number of options by grouping similar measures together, it cannot provide a definitive answer. The significance of association is a meaningful and well-defined concept, and Fisher's exact test is now widely accepted in mathematical statistics as the most appropriate quantitative measurement of this significance. The log-likelihood association measure gives an excellent approximation to the p -values of Fisher's test and has convenient mathematical and numerical properties. Consequently, it has recently become a *de facto* standard in the field of computational linguistics for the purpose of measuring the statistical association between words or similar entities.

However, there are many alternatives with entirely different characteristics, especially measures from the degree of association group as well as various heuristics and the new parametric measures. The statistical soundness of log-likelihood does not always translate into better performance. A conclusive answer can therefore only come from a comparative empirical evaluation of association measures, which plugs different measures into the intended application. In this way it is possible to determine what influence the choice of an association measure has on the performance and quality of the application, and to identify the measure that is best suited for the task. The general usefulness of cooccurrence data for the application can be assessed by comparison with random association scores as a baseline, and the frequency measure is sometimes used as a non-statistical baseline.

The range of possible settings for evaluation experiments is as broad as the range of applications for cooccurrence data (cf. Section 1.2.1). For instance, Dunning (1998) uses cooccurrence data scored with the log-likelihood measure in an information retrieval system and tests whether the performance of the system is improved. The clearest results can be expected from a collocation extraction task, however, especially when it is based on the standard pipeline design presented in Section 1.2.2. The computed association scores, which form the central component of the extraction pipeline, have an immediate influence on the quality of the results. Besides the practical importance of collocation extraction, such evaluation studies can also contribute to our understanding of the relation between the statistical association

of cooccurrences and any given notion of collocations. The stronger an association measure is correlated with collocativity, the better it should be suited for extracting the respective type of collocations from a text corpus.

In this chapter, I will only consider quantitative evaluation methods that should also be objective to the extent possible. All too often, especially when authors suggest a new association measure and want to substantiate its usefulness, the “evaluation” consists of looking at a small number of cooccurrences with high association scores and declaring them to be of good quality: “Table 1 shows some interesting Japanese collocations extracted using respectively mutual information and cost criteria. Table 2 shows some English ones” (Kita *et al.* 1994, 26). Case studies like the examples discussed by Church *et al.* (1991) or the much more detailed lexicographic analysis of Stubbs (1995) can make an important contribution to our understanding of the empirical properties of association measures and their relation to collocations, but impressionistic conclusions alone are not sufficient for an objective comparison of different measures.

5.1.1 Evaluation methods and reference data

An objective quantitative evaluation of association measures can be carried out in various ways, such as the following:

1. Determine the statistical correlation of association scores with a gradient notion of collocativity that is measured on an *interval scale* (e.g. plausibility ratings from psycholinguistic experiments), an *ordinal scale* (several levels of collocativity, e.g. the number of annotators accepting a candidate), or a *nominal scale* (a binary distinction between collocational and non-collocational pairs). A well-known experiment of this type was carried out by Lapata *et al.* (1999) with data on an interval scale. They correlated the association scores of different measures with native-speaker judgements of plausibility obtained by a magnitude estimation technique, evaluating a total of 90 adjective-noun combinations (plus 30 filler pairs) rated by 24 subjects. Drawbacks of such methods are the difficulty of obtaining gradient reference data and the limited relevance that even a significant correlation may have for practical applications.
2. Use a pre-determined threshold γ for association scores to extract collocation candidates from a text corpus, then determine the precision (and perhaps also recall) of the resulting γ -**acceptance set**. The threshold may be derived from a theoretical argument (e.g. $p < .001$ for measures from the significance of association group), selected by manual experimentation or determined automatically by the system. Evaluation in terms of precision and recall requires a binary distinction between collocations and non-collocations (possible sources of reference data are listed below). Smadja (1993, 166–170) is an excellent example of an evaluation experiment of this type, although he does not compare different association measures. The collocation candidates were manually evaluated by a professional lexicographer in this case.
3. Use association scores to rank the collocation candidates extracted from a text corpus. Precision and recall can then be computed for sets of n highest-ranking

candidates, called *n*-best lists. This procedure is analogous to the evaluation of γ -acceptance sets and uses the same types of reference data. The ranking-based evaluation has two important advantages: (i) it allows for a “fair” comparison of different measures because exactly the same number of candidates are evaluated from each ranking; (ii) when all possible values of *n* are considered, the method gives a much more complete picture of a measure’s performance than the fleeting glimpse provided by a single candidate set. In addition, it provides the most realistic evaluation framework for semi-automatic collocation extraction as described in Section 1.2.2: the human annotators will only have time to look at a limited number of candidates, so it is more likely that the number of candidates *n* will be pre-defined rather than the cutoff threshold γ . I will henceforth refer to this evaluation method as *n*-best precision.

In this chapter, I present methods for the third approach, i.e. evaluation in terms of *n*-best precision (and *n*-best recall, if possible). In order to compute precision and recall, the candidate set has to be compared with a **gold standard** that identifies candidates as **true positives** (TP, collocations) or **false positives** (FP, non-collocations). Possible sources of such reference data are:

Manual annotation: Ideally performed by two or more annotators who should be experts in a field relevant to the collocation definition and intended application (linguistics, terminology, lexicography, etc.). For such annotations to be meaningful, a precise definition of collocations is needed and should be accompanied by detailed guidelines.¹ Even then, annotations are not always reproducible and it is important to test the degree of intercoder agreement (e.g. Carletta 1996). See Krenn *et al.* (2004) for a study of intercoder agreement on an annotation database of German PP-verb combinations (Krenn 2000), which was used for most of the evaluation examples in this chapter. In other studies, manual evaluation was performed by the author herself (Breidt 1993), by a professional lexicographer (Smadja 1993), by domain experts for terminology extraction (Daille 1994) or by averaging over native speaker judgements (Blaheta and Johnson 2001).

Machine-readable dictionaries: Some authors use existing lexical resources in order to avoid the often unmanageable task of manual annotation. Of course one cannot expect that all true positives extracted from a corpus are covered by the database. A much more critical assumption, though, is that those TPs which are in the database form a random sample of the set of all true positives, so that the methods for random sample evaluation described in Section 5.3 can be applied (with the additional complication that the sampling rate is not known). Otherwise, the evaluation results may be completely distorted.

Considering the increasing number of corpus-based dictionaries – many of which are influenced by Church *et al.* (1991) and use the MI measure suggested there

¹Such a precise definition of true positives is not necessarily based on formal, testable criteria. Especially when the main goal of a study is relevance for a particular application, the intuitions of experts may play an essential role. For instance, the definition of true positives in a lexicographic setting might encompass “all candidates that provide useful information for the compilation of a large bilingual dictionary”.

to extract raw material for lexicographers – this assumption is becoming more and more doubtful. Nonetheless, Pearce (2002) uses 17 485 word pairs that were automatically extracted from a machine-readable version of the New Oxford Dictionary of English (Pearsall and Hanks 1998) as his gold standard. It is hardly surprising that he reports very low precision values (below 3%) and that the best results are obtained by MI combined with a frequency threshold.

Schone and Jurafsky (2001), whose goal is to extract MWU headwords for dictionaries, use multi-word units from the WordNet database (Miller 1990) for their evaluation. In order to achieve better coverage, they repeat the evaluation with various online resources (including <http://www.onelook.com/>, a website that “interfaces with over 600 electronic dictionaries” of untraceable provenance). The term “gold standard” seems almost cynical in this context.

Paper dictionaries: Breidt (1993) and Daille (1994) considered the use of a paper dictionary as a gold standard but found the overlap with the automatically extracted collocations too low to be useful (in the case of Daille, the collocation candidates were verified by domain experts).

Terminological resources: When the evaluation goal is the extraction of technical terminology, an existing terminological resource can be used as a gold standard. Daille (1994) used a telecommunications term bank provided by the European Commission in the form of a flat list of ca. 6000 multi-word terms. Similar to the problems with paper dictionaries she found the coverage of the term bank wanting and had to complement it with a manual evaluation by three domain experts that found as many as 900 true collocations among 1 900 putative false positives (Daille 1994, 143–145).

A small number of serious comparative evaluation experiments have been carried out so far (listed in Section 1.3.1). Most of them consider only a small number of popular association measures (and sometimes a few obscure ones, too). Exceptions are Daille (1994), who compares 18 different measures, and Evert and Krenn (2001), who make results for some additional measures available online. On the whole, these experiments have found that the log-likelihood measure achieves the highest n -best precision (Daille 1994; Lemnitzer 1998; Lezius 1999; Evert and Krenn 2001). Plain cooccurrence frequency also turned out to be a reliable indicator of collocativity. On the other hand, t -score and frequency seem to be better suited for the extraction of German PP-verb collocations than log-likelihood (Krenn 2000; Krenn and Evert 2001).

5.1.2 Precision and recall graphs

The formal definition of evaluation in terms of n -best precision and recall is based on the geometric interpretation of cooccurrence data and association measures introduced in Section 3.3. The point cloud $C \subseteq \mathcal{P}$ now represents a data set of **collocation candidates**. By manual evaluation (or comparison with some other gold standard), this data set is divided into disjoint sets of **true positives** C_+ and **false positives** C_- (i.e. $C = C_+ \cup C_-$ and $C_+ \cap C_- = \emptyset$). We know from Eq. (3.17) that the n -best list for

n	G^2	t	X^2	MI	f
100	42.00%	38.00%	24.00%	19.00%	27.00%
200	37.50%	35.00%	23.50%	16.50%	26.50%
500	30.40%	30.20%	24.60%	18.00%	23.00%
800	29.00%	30.38%	23.75%	19.50%	19.88% ◀
1500	25.33%	24.80%	25.00%	24.27%	18.00%
2000	23.35%	21.95%	23.35%	23.10%	16.30%
2300	21.61%	21.00%	21.61%	21.35%	15.30% ◀
3000	17.90%	17.90%	17.87%	17.83%	13.60%

Table 5.1: Table of n -best precision values for various n -best lists and 5 different association measures on the PNV-FR-30 data set. The n -best lists marked ◀ are indicated by vertical lines in Figure 5.1.

an association measure g is given by $C_{g,n} = A_{g,n} \cap C$ with $|C_{g,n}| = n$.² Likewise, the number of true positives in the n -best list is

$$T_{g,n} := A_{g,n} \cap C_+,$$

and the number of false positives is

$$F_{g,n} := A_{g,n} \cap C_- = n - T_{g,n}.$$

Then, n -best precision $P_{g,n}$ and recall $R_{g,n}$ are given by

$$P_{g,n} := \frac{|T_{g,n}|}{n} \quad \text{and} \quad R_{g,n} := \frac{|T_{g,n}|}{|C_+}|. \quad (5.1)$$

In this way, precision and recall can be computed for a variety of arbitrarily selected n -best lists, leading to large evaluation tables such as the one shown in Table 5.1. The results in this table were obtained for high-frequency candidates from the PNV-FR data set, which were manually annotated as TPs and FPs according to the criteria of Krenn (2000). With a frequency threshold of $f \geq 30$, the resulting data set contains 5 102 candidates and is referred to as PNV-FR-30 in the following. I will use this data set for all examples in the present section. For illustrative purposes, the evaluation is only carried out for five widely-used association measures, which are referred to by their customary symbols: log-likelihood (G^2), t-score (t), chi-squared_{corr} (X^2), MI (MI) and frequency (f).

Evaluation tables are often confusing and difficult to read, especially when a large number of association measures and n -best lists are considered. Interesting effects may be hidden beneath an endless procession of figures. **Evaluation graphs** as shown in Figure 5.1 present the same information in a more intuitive and readable way. In this plot, all n -best precision values of a given association measure g (corresponding to one of the columns in Table 5.1) are combined into a single graph. For

²Recall that the existence of “exact” n -best lists was enforced by adding random jitter to the coordinates of pair types. In a practical evaluation experiment it is usually more convenient to retain the original frequency signatures and break ties in the rankings in a random (but reproducible) fashion. The two approaches are (almost) equivalent.

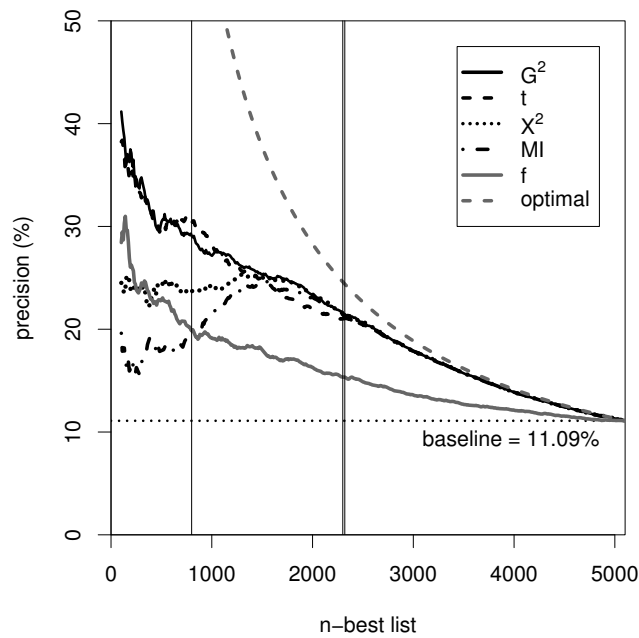


Figure 5.1: Graphs of n -best precision for five association measures evaluated on the PNV-FR-30 data set. The vertical lines mark n -best lists for $n = 800$ and $n = 2300$.

each n -best list indicated by the x -coordinate of the graph, the y -coordinate gives the corresponding n -best precision $P_{g,n}$. Precision graphs for up to five association measures can easily be combined into a single plot, which then provides a complete picture of differences between the measures at a single glance.

The vertical lines in Figure 5.1 indicate n -best lists for $n = 800$ and $n = 2300$, corresponding to the rows marked ◀ in Table 5.1. The corresponding n -best precision for the five evaluated measures can be determined from the intersection of each vertical line with the respective precision graphs, allowing the reader to reconstruct the detailed information provided in the evaluation table. The **baseline**, shown as a dotted horizontal line, corresponds to a random selection of n candidates from the data set. This provides a point of reference for the evaluation: the application of association measures to the data is useful only when they achieve an n -best precision that is substantially higher than the baseline. While there are considerable differences between the measures for small n , the graphs are almost identical for $n \geq 2300$ (except for the frequency measure) and slowly converge to the baseline precision. The reason is quite simple: once recall is close to 100%, even the best-performing measure cannot find any new TPs and keeps adding FPs to the n -best list when n is increased. The dashed grey line in Figure 5.1 represents the precision achieved by an “ideal” measure that ranks all TPs at the top of the list. This *optimal* measure provides an upper limit for the performance of association measures in the evaluation. In this case, we see that while there is considerable room for improvement in the range $n \leq 1500$, the association measures obtain nearly optimal results for $n \geq 2300$.

Figure 5.2 shows a “zoomed” version of the precision plot, where only the interesting range $n \leq 2300$ is displayed. The evaluation results largely agree with our expectations from previous studies. G^2 achieves the best performance and is on par

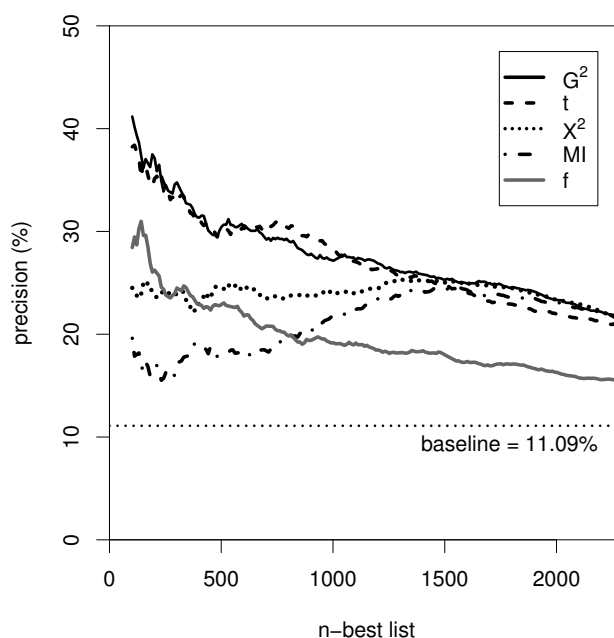


Figure 5.2: Precision graphs for n -best lists with $n \leq 2300$ on the PNV-FR-30 data set.

with t , which is known to be well suited for extracting German PP-verb collocations (cf. Evert and Krenn 2001). X^2 and MI give considerably worse results. The only surprise is the frequency measure f , whose precision remains well below the graphs of G^2 and t , contrary to most previous findings. This effect seems to be caused by the unusually high frequency threshold ($f \geq 30$) that was applied to the data set.

While n -best precision is of paramount importance for most applications, it is just one side of the coin. In real life, the goal of an extraction tool is to identify a substantial proportion of the collocations hidden in the data set. It is not enough to achieve excellent precision for the 100 highest-ranking candidates, as e.g. the mini-evaluation of Dunning (1993) would make us believe. **Recall graphs**, which simply substitute $R_{g,n}$ for $P_{g,n}$, give a different angle on the evaluation results. Figure 5.3 shows that the 2000 highest-ranking candidates according to the G^2 measure include more than 80% of the true positives in the data set. Such a high coverage is especially important when collocations are extracted from small, domain-specific corpora. On the other hand, a 100-best list will miss more than 90% of the true positives.

A third type of plot combines both aspects into a single **precision-by-recall graph** (Figure 5.4). The x -coordinate of such a graph represents the n -best recall $R_{g,n}$ and its y -coordinate represents the n -best precision $P_{g,n}$. Note that n -best lists correspond to diagonal lines in this view, since $P_{g,n} = T_{g,n}/n = R_{g,n} \cdot |C_+|/n$. This plot, which is in fact just a transformation of Figure 5.1, is the most intuitive form of presentation. Differences between the measures are amplified visually, and the question that is most relevant for applications can directly be answered: Which n -best list gives the best trade-off between precision and recall?

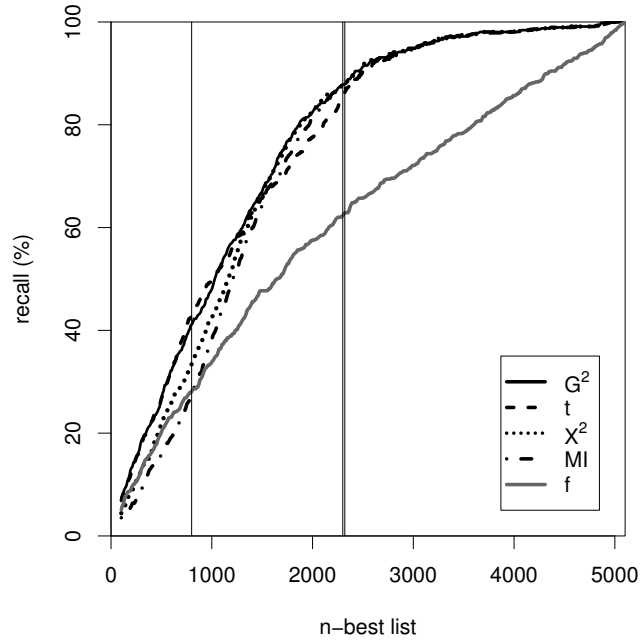


Figure 5.3: Recall graphs for the PNV-FR-30 data set. The vertical lines mark n -best lists for $n = 800$ and $n = 2300$.

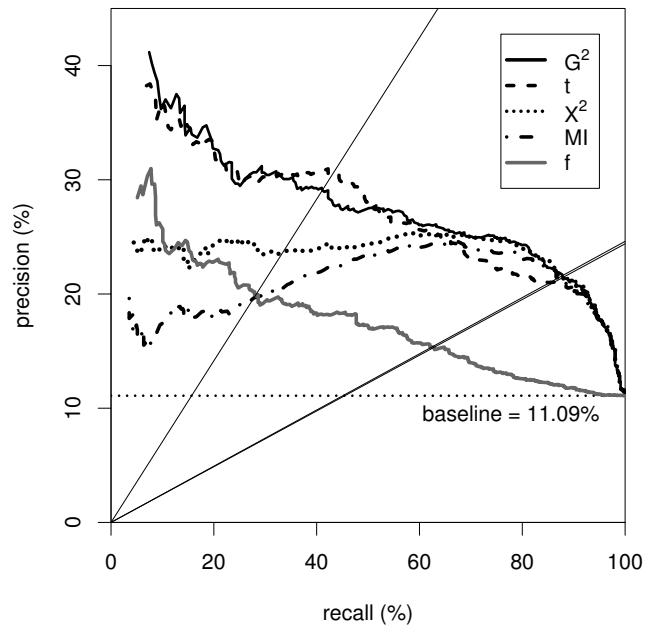


Figure 5.4: Assessing the practical usefulness of association measures with precision-by-recall graphs (on the PNV-FR-30 data set). The diagonal lines indicate n -best lists for $n = 800$ and $n = 2300$.

5.1.3 Fine-grained comparative evaluation

This section presents a small case study to demonstrate how the overall evaluation results (Section 5.1.2) can be refined to give a more detailed and accurate picture. Such a fine-grained evaluation is usually achieved by splitting the data set e.g. into different frequency layers (see Evert and Krenn (2001) for an example), which is formally equivalent to the application of a type filter (cf. Section 2.1.4). In this case study, I begin with a separate look at two types of PP-verb collocations that were both treated as true positives in Section 5.1.2.

In her annotations, Krenn (2000) divides collocations into *figurative expressions* (*figur*) and *support-verb constructions* (FVG, from German *Funktionsverbgefüge*). These two types of collocations have different syntactic and semantic properties, which are reflected in their cooccurrence frequency profiles. Figure 5.5 compares the performance of association measures for the extraction of *figur* and FVG, showing entirely different strengths and patterns. Both G^2 and X^2 are reasonably useful for extracting figurative expressions, with t somewhat below the two. This result agrees well with earlier studies on other types of data (e.g. Daille 1994; Evert *et al.* 2000).³ For support-verb constructions, on the other hand, t is clearly the best-performing measure. The evaluation results of Section 5.1.2 average over the two situations, hiding some of the characteristic strengths and weaknesses of the measures. For instance, a comparison of the two best-performing measures, whose overall results are nearly identical, reveals that G^2 achieves roughly the same precision in both tasks while t is much better suited for the extraction of support-verb constructions.

A puzzling observation is the shape of the FVG precision graph for MI , whose performance is even below the baseline for small n -best lists, but becomes much better when larger lists are considered. For $n \geq 1\,500$, it is on par with the best-performing measures. Krenn and Evert (2001) refer to this as the “mutual information mystery”. What makes this plot so difficult to interpret is the fact that precision graphs display the *cumulative* precision for n -best lists. This mode of presentation makes sense because the “concentration” of true positives is normally greatest at the top of the ranking and gradually decreases as one moves down the list. However, when an association measure achieves the highest concentration of TPs somewhere in the middle of its ranking, the cumulative precision will start off low and then increase.

For this reason, Daille (1994, 145ff) divides the ranked lists into non-overlapping segments of 50 candidates each and computes precision individually for each segment. While this mode of presentation shows clearly where the concentration of TPs is highest in the ranked list, the graphs are very jagged (since precision values computed from 50 candidates can vary only in increments of 2 percentage points) and difficult to compare between measures. It is therefore advisable to use a more sophisticated method that computes the precision on a moving window. Figure 5.6 uses kernel density estimates with a Gaussian kernel (Venables and Ripley 1999, 132–139) to estimate the **local precision** in different parts of the ranked candidate list, averaging over some 500 candidates at each point. While t shows the expected pattern, with the highest density of TPs at the top of the ranking, X^2 and MI reach

³The overestimation bias of X^2 , which often causes it to be distinctly inferior to G^2 , is reduced by the high frequency threshold of $f \geq 30$. Therefore, it is hardly surprising that X^2 reaches a similar performance.

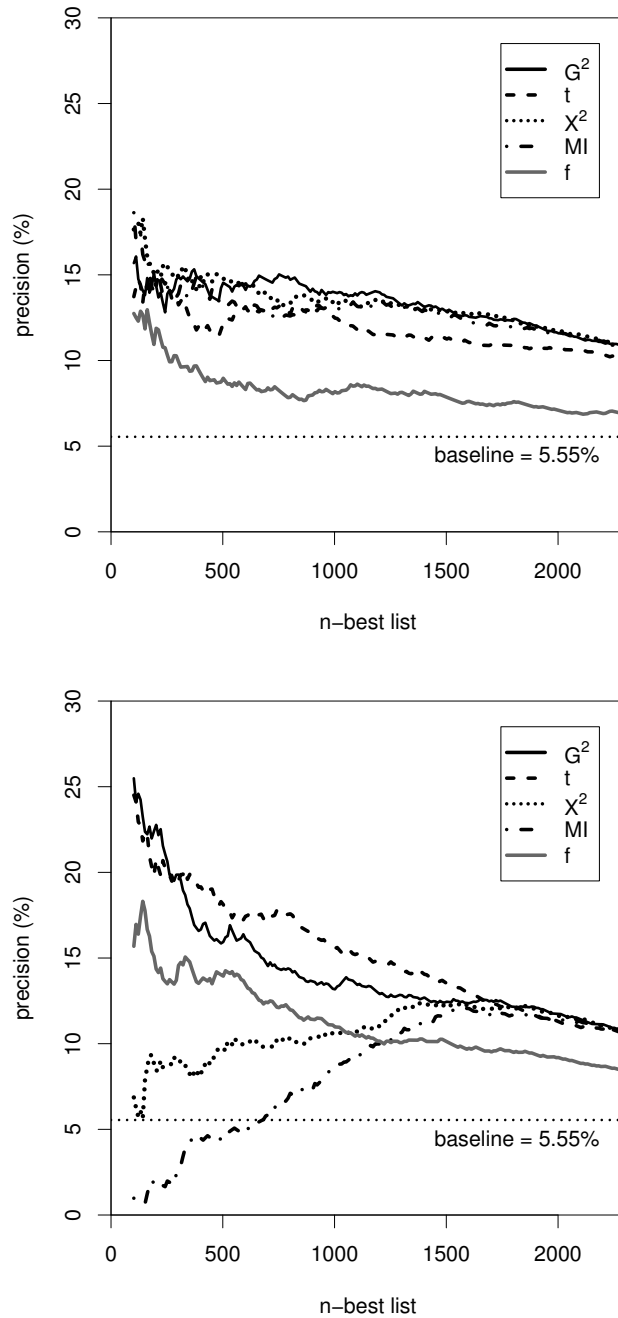


Figure 5.5: Comparison of the performance of association measures for figurative expressions (top panel) vs. support-verb-constructions (bottom panel). It is pure coincidence that the baseline precision is the same for both types of collocations.

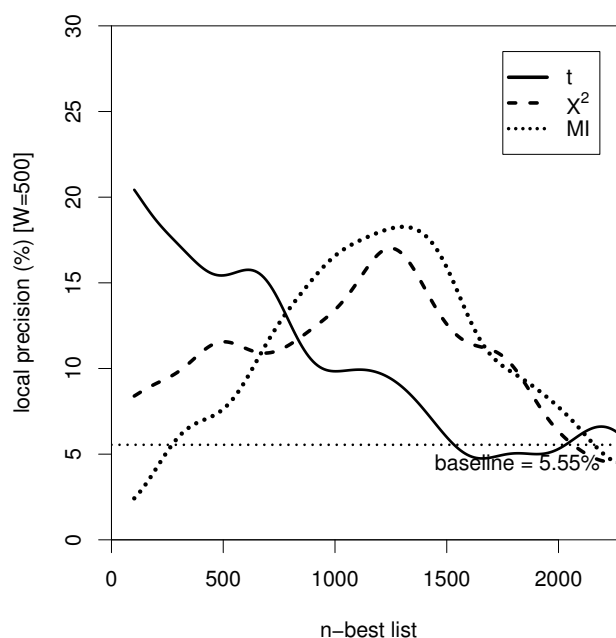


Figure 5.6: Estimates of the local precision in different parts of the ranked candidate lists for the extraction of support-verb-constructions.

optimal precision for ranks between $n = 1000$ and $n = 1500$, corresponding to average rather than particularly high association scores. In fact, the precision achieved by *MI* in this range is comparable to the highest precision of *t* (at the top of the ranking).

A second refinement of the evaluation reveals that the mutual information mystery is an artefact, introduced by the fact that most PP-verb collocations involve a small number of high-frequency verbs that Breidt (1993) identified as “typical” support-verbs.⁴ Based on this intuition, Krenn (2000) used what she called a kwic filter to improve extraction results. With this type filter, we can divide the PNV-FR-30 data set into two subsets, depending on whether the second component of a pair type belongs to the list of support-verbs or not. The first set contains 1450 pair types involving one of the support-verbs, while the second set contains the remaining 3652 pair types. The results of a separate evaluation of the two subsets are shown in Figure 5.7, which displays n -best precision up to $n = 1450$ for both sets. It is obvious that the kwic filter improves the precision for *both* types of collocations substantially. Moreover, the differences between the association measures (except for frequency) all but vanish after application of the filter. Thus, the poor performance of *MI* on the full data set was just due to its inability to single out typical support-verbs.

Fine-grained evaluation does not only have a high explanatory potential, which allowed it to solve the *MI* mystery, but it can also lead to improvements in extraction quality that are relevant for applications. The graphs in Figure 5.7 show that the kwic filter is highly discriminative between collocations and non-collocations. However, n -best precision decreases rapidly for $n \geq 800$, when most of the true positives in the

⁴The verbs are *bleiben*, *bringen*, *erfahren*, *finden*, *geben*, *gehen*, *gelangen*, *geraten*, *halten*, *kommen*, *nehmen*, *setzen*, *stehen*, *stellen*, *treten* and *ziehen*.

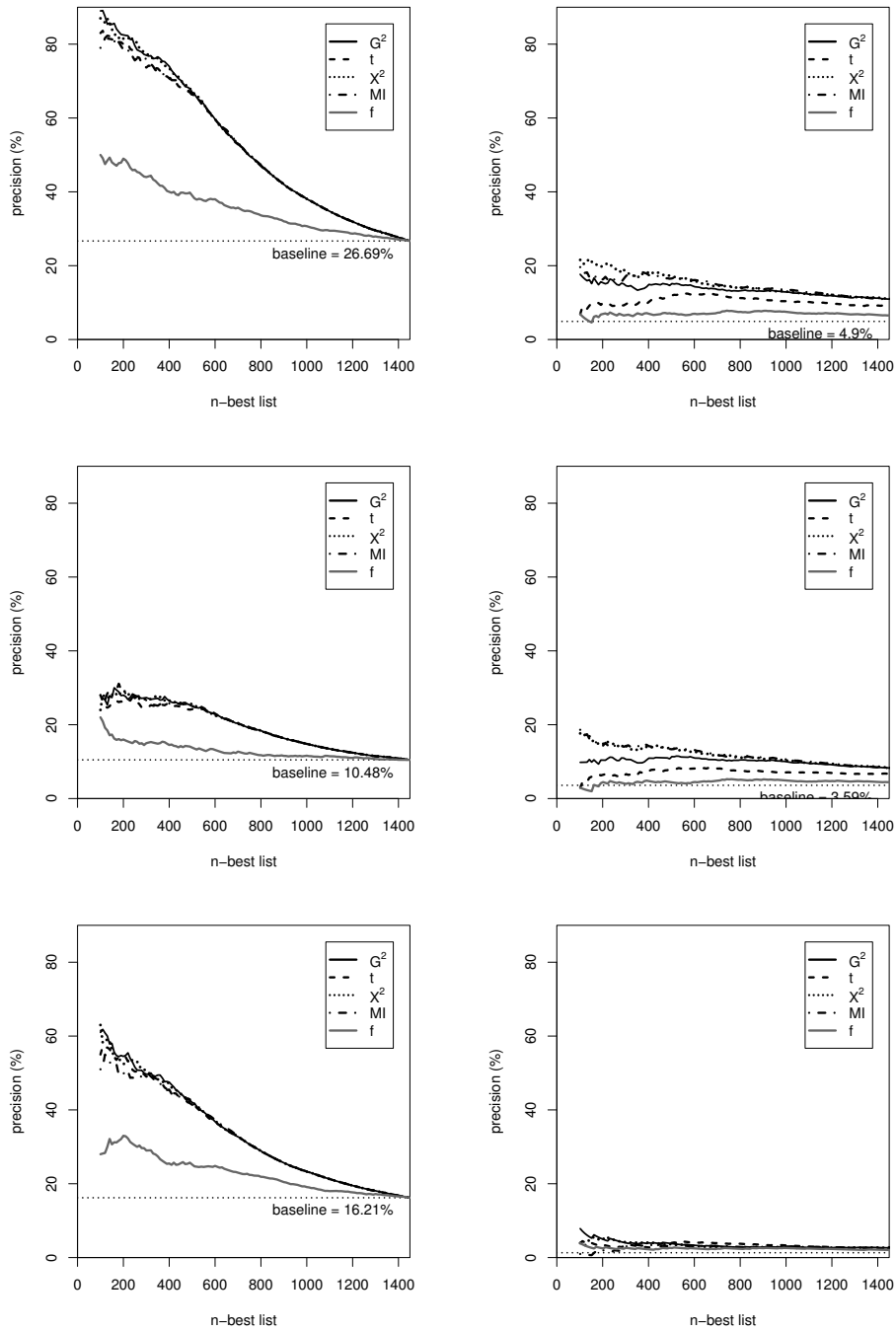


Figure 5.7: Comparison of precision of n -best lists ($n \leq 1450$) for pair types accepted by the kwic filter (Krenn 2000, 120) in the left column vs. the rejected pair types in the right column. The top row shows overall precision, the middle row precision for figurative expressions, and the bottom row precision for support-verb constructions.

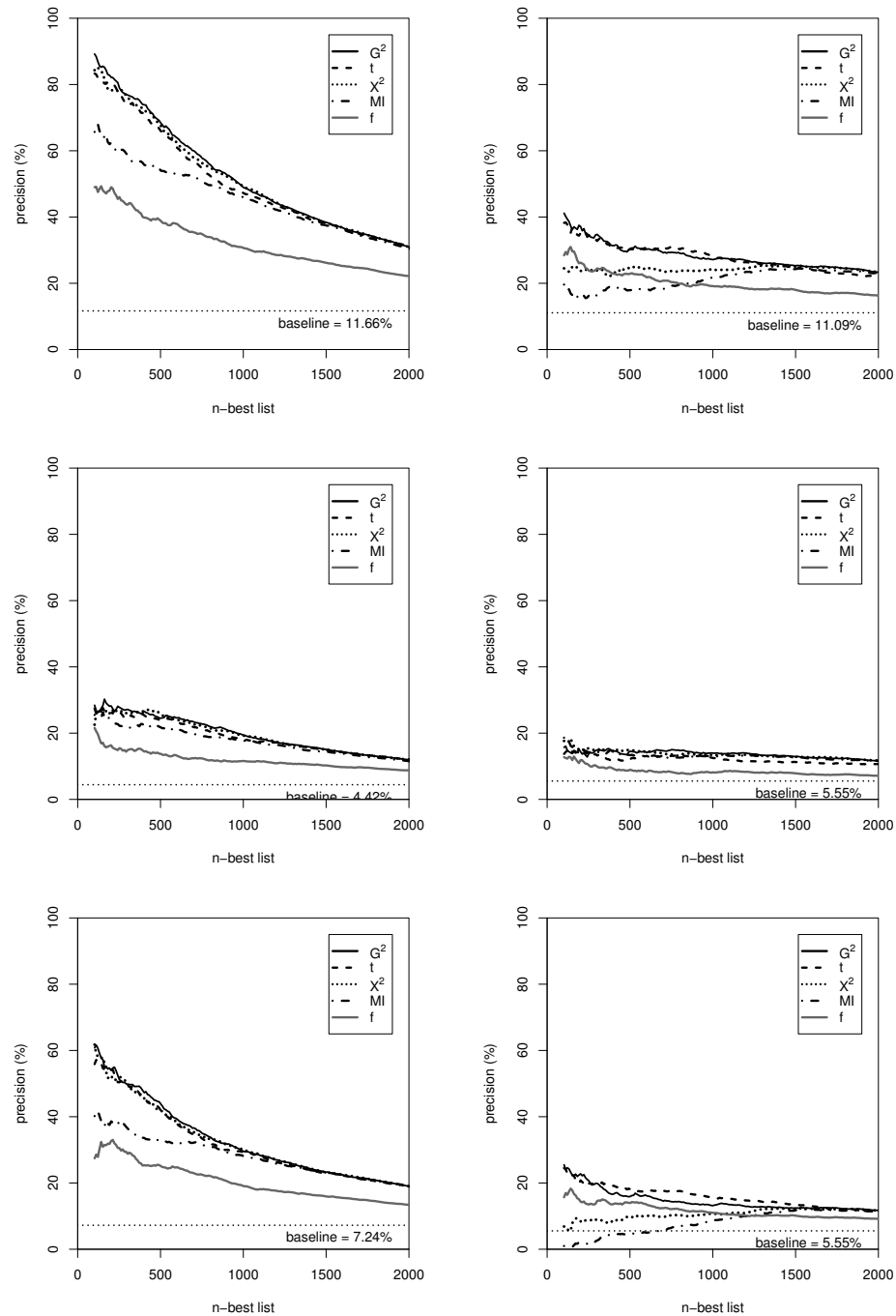


Figure 5.8: Comparison of the precision of n -best lists ($n \leq 2000$) extracted with a combination of the κ wic filter (Krenn 2000, 120) and a frequency threshold of $f \geq 10$ (left column) vs. a frequency threshold of $f \geq 30$ but no filter (right column). The top row shows overall precision, the middle row precision for figurative expressions, and the bottom row precision for support-verb constructions.

small filtered data set (of only 1 450 candidates) have already been identified. This suggests that further improvements may be possible by combining the successful kwic filter with a lower frequency threshold (here $f \geq 10$). For $n \leq 2000$, this combination (shown in the left column of Figure 5.8) achieves higher precision than the best results for the PNV-FR-30 data set (shown in the right column for comparison). At the same time, the influence of individual association measures on the extraction results is greatly diminished. For $n \leq 1000$, the improvement in performance is striking.

An implementation of the various types of evaluation graphs is available in the UCS toolkit as a UCS/R module (see Section B.2). All plots in this section were created with the UCS/R implementation.

5.2 The significance of result differences

In Section 5.1 we have seen that a fine-grained comparative evaluation of association measures can reveal a wealth of detail about the empirical properties of the measures and their respective differences. Sometimes, the observed effects are minuscule, though, and the question arises whether they reflect a true difference between the measures or whether they may simply be due to chance. A major source of such random variation is the choice of a particular source corpus for the evaluation experiment, but extraction noise and the uncertainty of human annotators will also play a role. The necessity for testing whether evaluation results are *statistically significant* is widely accepted, but there is much uncertainty about the appropriate choice of a significance test. For instance, Krenn (2000) applies Pearson’s chi-squared test, but she is aware that this test assumes independent samples and is hardly suitable for the comparison of different rankings of the same candidate set. Later, Krenn and Evert (2001) suggest several alternative tests for related samples. A wide range of exact and asymptotic tests as well as computationally expensive randomisation tests (Yeh 2000) are available and add to the confusion.

The following discussion concentrates on the uncertainty of precision values (and the significance of differences between them), which are of greater importance to most evaluation studies than recall values. Moreover, for n -best lists precision and recall are fully equivalent: $P_{g,n} = R_{g,n} \cdot |C_+| / n$, where $|C_+| / n$ is the “proportionality factor” between precision and recall.

5.2.1 Evaluation as a random experiment

The aim of this section is to formulate a statistical model that interprets the evaluation of ranking methods as a random experiment. This model defines the degree to which evaluation results are affected by random variation, allowing us to derive appropriate significance tests. Although evaluation is usually based on n -best lists, this model concentrates on the precision achieved by an arbitrary fixed acceptance region $A \subseteq \mathcal{D}$. The resulting estimates and significance tests can then be translated to n -best precision by setting $A = A_{g,n}$.

When an evaluation experiment is repeated, the results will not be exactly the same. There are many causes for such variation, including different source material used by the second experiment, changes in the pre-processing or extraction tools,

changes in the evaluation criteria, or the different intuitions of human annotators. Statistical significance tests are designed to account for a small fraction of this variation that is entirely due to random effects, assuming that all parameters that may have a systematic influence on the evaluation results are kept constant. Thus, they provide a lower limit for the variation that has to be expected in an actual repetition of the experiment. Only when results are significant can we expect them to be reproducible, but even then a second experiment may draw a different picture.

In particular, the influence of qualitatively different source material or different evaluation criteria can never be predicted by statistical means alone. Randomness is mainly introduced into the evaluation results by the selection of the source corpus, e.g. the choice of one particular newspaper volume rather than another. Disagreement between human annotators and uncertainty about the interpretation of annotation guidelines may also lead to an element of randomness in the evaluation. However, even significant results cannot be generalised to a different type of collocation (such as adjective-noun instead of PP-verb), different evaluation criteria, a different domain or text type, or even a source corpus of different size (cf. Evert and Krenn 2001; Krenn and Evert 2001).

A first step in the search for an appropriate significance test is to formulate a (plausible) model for random variation in the evaluation results. Because of the inherent randomness, every repetition of an evaluation experiment – even when it is performed under similar conditions – will lead to a different candidate set C , and to different sets of true positives C_+ and false positives C_- . Some elements will represent entirely new pair types, sometimes the same pair type will appear at a different point in the coordinate space, and sometimes a candidate that was annotated as a TP in one experiment may be annotated as a FP in the next. In order to encapsulate all three kinds of variation, let us assume that C_+ and C_- are randomly selected from a large hypothetical set of possible candidates. Every pair type (u, v) is represented by many different incarnations with different coordinates in this hypothetical set, some of which may be TPs and some FPs. Of course, only one incarnation of (u, v) may be selected for a given experiment, and it cannot belong both to C_+ and to C_- at the same time. Provided that the number of different pair types is sufficiently large, though, we can ignore the risk of such an event.

For any acceptance region $A \subseteq \mathcal{D}$, both the number of TPs in A , $T_A := |C_+ \cap A|$, and the number of FPs in A , $F_A := |C_- \cap A|$, are thus *random variables*. Figure 5.9 illustrates this model with four similar data sets, showing true positives as solid points and false positives as empty circles. The shaded acceptance region A belongs to the log-likelihood measure with a cutoff threshold of $\gamma = 32.5$. T_A is the number of solid points in the region A , and F_A is the number of empty circles in A . Obviously, both numbers vary from panel to panel. We do not know the precise distributions of these random variables, but it is reasonable to assume that (i) T_A and F_A are always independent and (ii) T_A and T_B (as well as F_A and F_B) are independent for any two disjoint regions $A \cap B = \emptyset$. Note that T_A and T_B cannot be independent for $A \cap B \neq \emptyset$ because they always include the same number of TPs from the region $A \cap B$. The total number of candidates in the region A is also a random variable $N_A := T_A + F_A$, and the same follows for the precision P_A of A , which is defined by $P_A := T_A / N_A$.⁵

⁵In the definition of the n -best precision $P_{g,n}$, i.e. for $A = A_{g,n}$, the number of candidates in A is constant: $N_A = n$, cf. (5.1). At first sight, this may seem to be inconsistent with the interpretation of

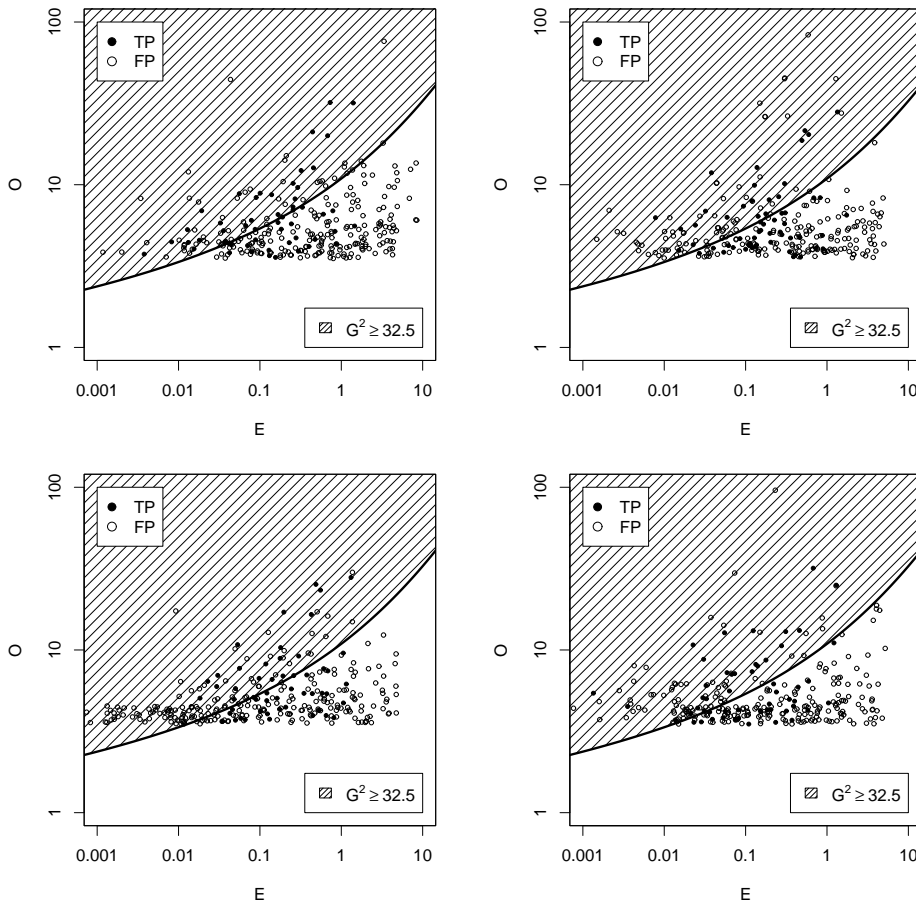


Figure 5.9: Illustration of evaluation experiment as the random selection of true and false positives from a hypothetical population.

Following the standard approach, we may now assume that P_A approximately follows a normal distribution with mean p_A and variance σ_A^2 , i.e. $P_A \sim N(p_A, \sigma_A^2)$. The mean p_A can then be interpreted as the *average precision* of the acceptance region A (obtained by averaging over many repetitions of the evaluation experiment). However, there are two problems with this assumption. First, while P_A is an unbiased estimator for p_a , the variance σ_A^2 cannot be estimated from a single experiment.⁶ Second, P_A is a discrete variable because both T_A and N_A are non-negative integers. When the number of candidates N_A is small (as it will be when we take a closer look at the differences between two measures), a continuous normal approximation for the distribution of P_A will not be valid.

It is reasonable to assume that the distribution of N_A does not depend on the average precision p_A . In this case, N_A is called an *ancillary statistic* and can be

N_A as a random variable. However, one has to keep in mind that $\gamma_g(n)$, which is determined from the candidate set C , is itself a random variable. Consequently, A is *not* a fixed acceptance region in this case and its variation counter-balances that of N_A .

⁶Sometimes, cross-validation is used to estimate the variability of evaluation results. While this method is appropriate e.g. for machine learning and classification tasks, it is not useful for the evaluation of ranking methods such as association measures. Since the cross-validation would have to be based on random samples from a single candidate set, it would not be able to tell us anything about random variation between different candidate sets.

eliminated without loss of information by conditioning on its observed value (see Lehmann (1991, 542ff) for a formal definition of ancillary statistics and the merits of conditional inference). Instead of probabilities of the form $\Pr(P_A = p)$, we will now consider the conditional probabilities $\Pr(P_A = p \mid N_A)$. Because N_A is fixed to the observed value, P_A is directly proportional to T_A and the conditional probabilities are equivalent to $\Pr(T_A = k \mid N_A)$ with $p = k/N_A$. When we select one of the N_A candidates in A at random, the probability that it is a TP (averaged over many repetitions of the experiment) should be equal to the average precision p_A . Consequently, $\Pr(T_A = k \mid N_A)$ should follow a binomial distribution with success probability p_A , i.e.

$$\Pr(T_A = k \mid N_A) = \binom{N_A}{k} \cdot (p_A)^k \cdot (1 - p_A)^{N_A - k} \quad (5.2)$$

for $k = 0, \dots, N_A$. We can now make inferences about the average precision p_A based on this binomial distribution.⁷

5.2.2 Confidence intervals and significance tests

As a second step in the search for an appropriate significance test, it is essential to understand exactly what question this test should address: What does it mean for an evaluation result (or the difference between evaluation results) to be significant? In fact, two different questions can be asked:

- A: *If we repeat an evaluation experiment under the same conditions, to what extent will the observed precision values vary?*
- B: *If we repeat an evaluation experiment under the same conditions, will association measure g_1 again perform better than association measure g_2 ?*

I will now address these two questions in turn.

Question A can be rephrased in the following way: *How much does the observed precision value for an acceptance region A differ from the true average precision p_A ?* In other words, our goal here is to make inferences about p_A , with $A = A_g(\gamma)$ for a given measure g and threshold γ . From Eq. (5.2), we obtain a binomial confidence interval for the true value p_A , given the observed values of T_A and N_A (Lehmann 1991, 89ff). At the customary 95% confidence level, p_A should be contained in the estimated interval in all but one out of twenty repetitions of the experiment. Binomial confidence intervals can easily be computed with standard software packages such as R. As an example, assume that an observed precision of $P_A = 40\%$ is based on $T_A = 200$ TPs out of $N_A = 500$ accepted candidates. Precision graphs as those in Figure 5.1 display P_A as a maximum-likelihood estimate for p_A , but its true value may range from 35.7% to 44.4% (with 95% confidence).⁸

⁷Note that some of the assumptions leading to Eq. (5.2) are far from self-evident. As an example, the equation tacitly assumes that the success probability is equal to p_A regardless of the particular value of N_A on which the distribution is conditioned, which need not be the case when the total number of collocational pair types (not the number of their incarnations) is finite and N_A happens to be particularly large. Therefore, an empirical validation of this statistical model is necessary (see Section 5.2.3).

⁸This confidence interval was computed with the R command `binom.test(200,500)`. A utility function in the UCS/R system allows direct computation of the confidence interval boundaries.

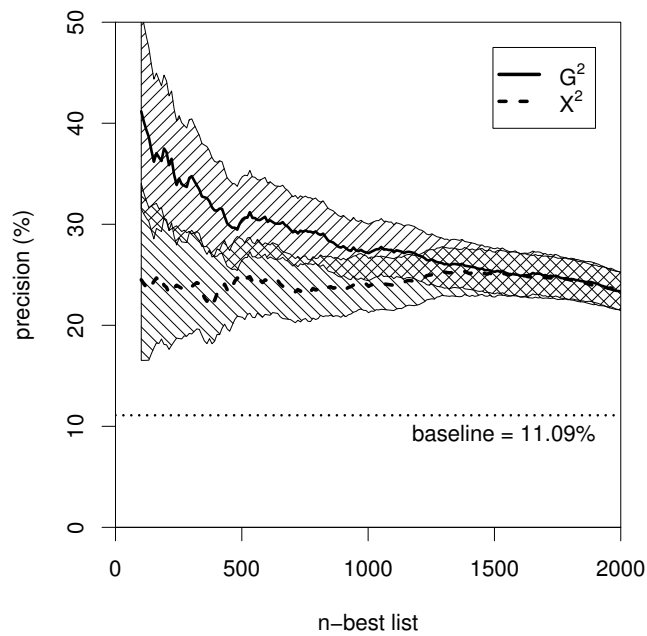


Figure 5.10: Precision graphs for G^2 and X^2 with 95% confidence intervals.

Figure 5.10 shows binomial confidence intervals for the association measures G^2 and X^2 as shaded regions around the precision graphs. It is obvious that a repetition of the evaluation experiment may lead to quite different precision values, especially for $n < 1000$. In other words, there is a considerable amount of uncertainty in the evaluation results for each individual measure. However, we can be confident that both ranking methods offer a substantial improvement over the baseline.

For an evaluation based on n -best lists, it has to be noted that the confidence intervals are estimates for the average precision p_A of a fixed γ -acceptance region $A = A_g(\gamma)$, with $\gamma = \gamma_g(n)$ computed from the observed candidate set. While this region contains exactly $N_A = n$ candidates in the current evaluation, N_A may be different from n when the experiment is repeated. Consequently, p_A is not necessarily identical to the average n -best precision across a large number of experiments.

Question B can be rephrased in the following way: *Does an association measure g_1 on average achieve higher precision than another measure g_2 ?* (This question is normally asked when g_1 performed better than g_2 in the evaluation.) In other words, our goal is to test whether $p_A > p_B$ for given acceptance regions $A = A_{g_1}(\gamma_1)$ of the measure g_1 and $B = A_{g_2}(\gamma_2)$ of the measure g_2 .

The confidence intervals around the precision graphs of two association measures g_1 and g_2 will often overlap (cf. Figure 5.10, where the confidence intervals of G^2 and X^2 overlap for all list sizes n), suggesting that there is no significant difference between the two ranking methods. Both observed precision values are consistent with an average precision $p_A = p_B$ in the region of overlap, so that the observed differences may be due to random variation in opposite directions. However, this conclusion is premature because the two rankings underlying the precision graphs are *not independent*. Therefore, the observed precision values of g_1 and g_2 will tend to vary in the same direction, the degree of correlation being determined by the

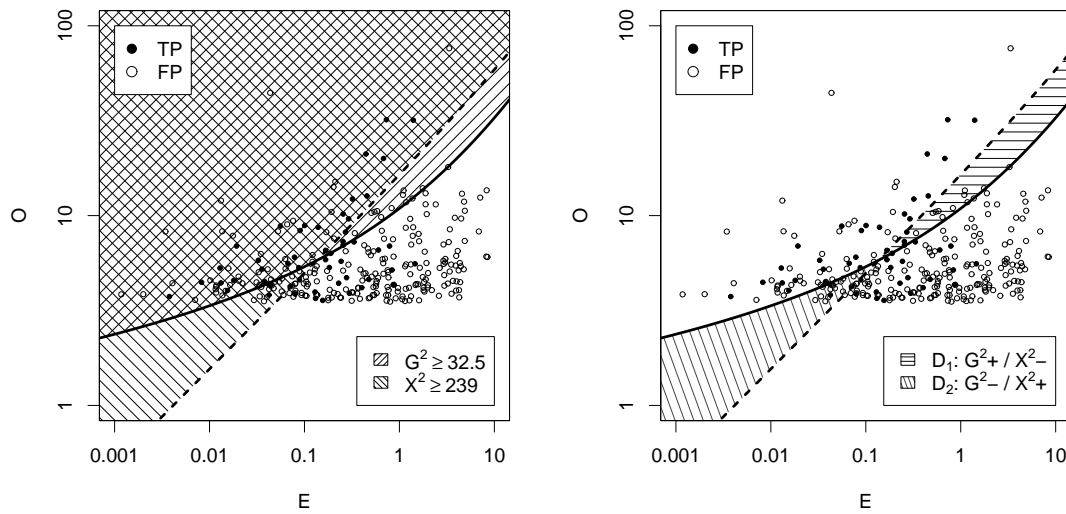


Figure 5.11: Illustration of the significance of precision differences between two association measures (here, G^2 and X^2 are compared (left panel: overlapping acceptance regions A and B ; right panel: difference regions D_1 and D_2).

amount of overlap between the two rankings. Given acceptance regions A and B as defined above, both measures make the same decision for any candidates in the intersection $A \cap B$ (both accept the candidate) and in the “complement” $\mathcal{D} \setminus (A \cup B)$ (both reject the candidate). Therefore, the performance of g_1 and g_2 can only differ in the regions $D_1 := A \setminus B$ (g_1 accepts, but g_2 rejects) and $B \setminus A$ (g_2 accepts, but g_1 rejects). Correspondingly, the counts T_A and T_B are correlated because they include the same number of TPs from the region $A \cap B$ (namely, the set $C_+ \cap A \cap B$). Figure 5.11 illustrates this situation with the measures $g_1 = G^2$ and $g_2 = X^2$ as an example. The left panel shows the overlapping acceptance regions A and B of g_1 and g_2 . All candidates in the cross-shaded region $A \cap B$ are accepted by both measures, while all candidates in the unshaded region $\mathcal{D} \setminus (A \cup B)$ are rejected by both. The right panel highlights the remaining difference regions D_1 and D_2 .

It is indisputable that g_1 is a better ranking method than g_2 iff $p_{D_1} > p_{D_2}$, and vice versa.⁹ Our goal is thus to test the null hypothesis $H_0 : p_{D_1} = p_{D_2}$ on the basis of the binomial distributions $\Pr(T_{D_1} | N_{D_1})$ and $\Pr(T_{D_2} | N_{D_2})$. Under the assumptions of Section 5.2.1, these distributions are independent because $D_1 \cap D_2 = \emptyset$. The number of candidates in the difference regions, N_{D_1} and N_{D_2} , may be small, especially for acceptance regions with large overlap (this was one of the reasons for using conditional inference rather than a normal approximation in Section 5.2.1). Therefore, it is advisable to use Fisher’s exact test (Agresti 1990, 60–66) instead of an asymptotic test that relies on large-sample approximations. The data for the application of Fisher’s test consist of a 2×2 contingency table with columns (T_{D_1}, F_{D_1}) and (T_{D_2}, F_{D_2}) . Note

⁹Note that $p_{D_1} > p_{D_2}$ does not necessarily entail $p_A > p_B$ if N_A and N_B are vastly different and $p_{A \cap B} \gg p_{D_i}$. In this case, the “winner” will always be the measure that accepts the smaller number of candidates (because the additional candidates only serve to lower the precision achieved on $A \cap B$). This example shows that it is “unfair” to compare acceptance sets of (substantially) different sizes just in terms of their overall precision. Therefore, evaluation either has to be based on n -best lists or needs to take recall into account.

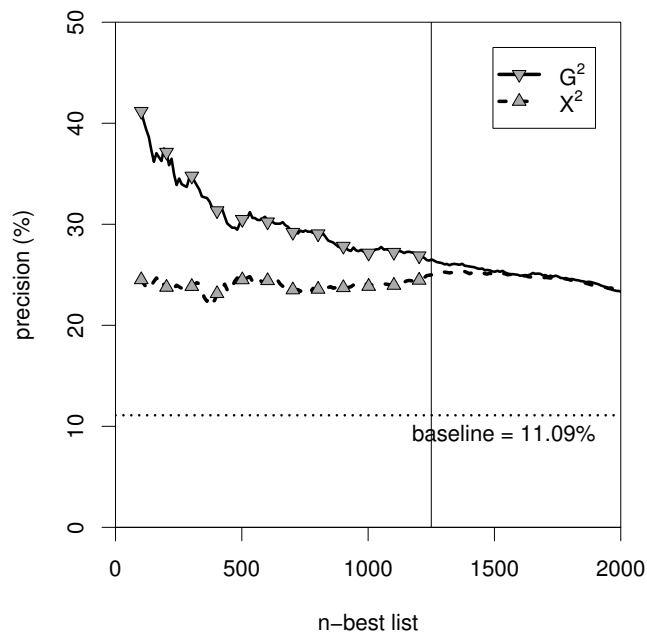


Figure 5.12: Significant differences between G^2 and X^2 at a confidence level of 95%.

that a two-sided test is called for because there is no *a priori* reason to assume that g_1 is better than g_2 (or vice versa). Although the implementation of a two-sided Fisher's test is far from trivial, it is readily available in software packages such as R.

Figure 5.12 shows the same precision graphs as Figure 5.10. Significant differences between the G^2 and X^2 measures according to Fisher's test (at 95% confidence, i.e. a significance level of $\alpha = 0.05$) are marked by grey triangles. Contrary to what the confidence intervals in Figure 5.10 suggested, the observed differences turn out to be significant for all n -best lists up to $n = 1\,250$ (marked by a thin vertical line).

Confidence intervals and significance tests for result differences are implemented in the UCS/R evaluation plot functions, which were used to create the graphs in this section.

5.2.3 Empirical validation

In order to validate the statistical model and the significance tests proposed in the previous sections, it is necessary to simulate the repetition of an evaluation experiment. Following the arguments of Section 5.2.1, the conditions should be as similar as possible for all repetitions so that the amount of purely random variation can be measured. For this purpose, I used the 80 PNV-SLICES data sets extracted from non-overlapping 500 000-word segments of the *Frankfurter Rundschau* corpus (cf. Section 2.1.3). All pair types with cooccurrence frequency $f \geq 4$ (between 223 and 369 pair types, with an average of 285) were ranked by the association measures G^2 , X^2 and t , and true positives were manually identified according to the criteria of Krenn (2000). The true average precision p_A of an acceptance set A was estimated by averaging over all 80 samples.

Both the confidence intervals and the significance tests introduced in Section 5.2.2

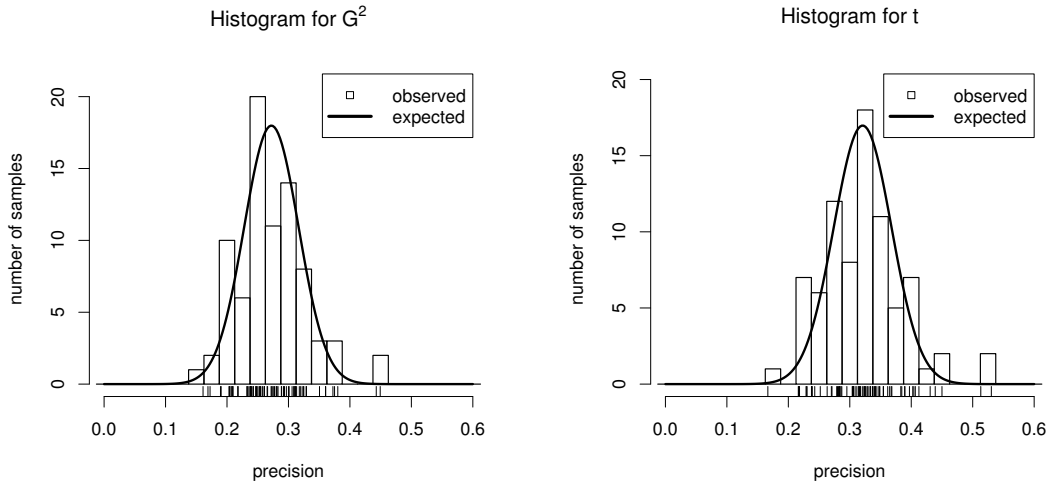


Figure 5.13: Distribution of the observed precision P_A for γ -acceptance regions of the association measures G^2 (left panel) and t (right panel). The solid curves indicate the expected distribution according to Eq. (5.2).

are based on the assumption that $\Pr(T_A | N_A)$ follows a binomial distribution as given by Eq. (5.2). Unfortunately, it is impossible to test the conditional distribution directly, which would require N_A to be the same for all samples. Therefore, I use the following approach based on the unconditional distribution $\Pr(P_A)$. If N_A is sufficiently large and (5.2) is valid, $\Pr(P_A | N_A)$ can be approximated by a normal distribution with mean $\mu = p_A$ and variance $\sigma^2 = p_A(1 - p_A)/N_A$. Since μ does not depend on N_A and the standard deviation σ is proportional to $(N_A)^{-1/2}$, it is valid to make the approximation

$$\Pr(P_A | N_A) \approx \Pr(P_A) \quad (5.3)$$

as long as N_A is relatively stable. In other words, we assume that the observed precision P_A is independent from the number of candidates N_A in the acceptance region. Eq. (5.3) allows us to pool the data from all samples, predicting that

$$\Pr(P_A) \sim N(\mu, \sigma^2) \quad (5.4)$$

with $\mu = p_A$ and $\sigma^2 = p_A(1 - p_A)/N$. Here, N stands for the *average* number of candidates in A , i.e. $N = E[N_A]$.

These predictions were tested for the measures $g_1 = G^2$ and $g_2 = t$, with cutoff thresholds $\gamma_1 = 32.5$ and $\gamma_2 = 2.09$ (chosen so that $N = 100$ candidates are accepted on average). Figure 5.13 compares the empirical distribution of P_A with the expected distribution according to Eq. (5.2). These histograms show that the theoretical model agrees quite well with the empirical results, although there is a little more variation than expected.¹⁰ The empirical standard deviation is between 20% and 40% larger than expected, with $s = 0.057$ vs. $\sigma = 0.044$ for G^2 and $s = 0.066$ vs. $\sigma = 0.047$ for t . These findings suggest that the model proposed in Section 5.2.1 may indeed represent a lower bound on the true amount of random variation. Further evidence

¹⁰The agreement is confirmed by the Kolmogorov test of goodness-of-fit (Lehmann 1991, 336), which does not reject the theoretical model in either case.

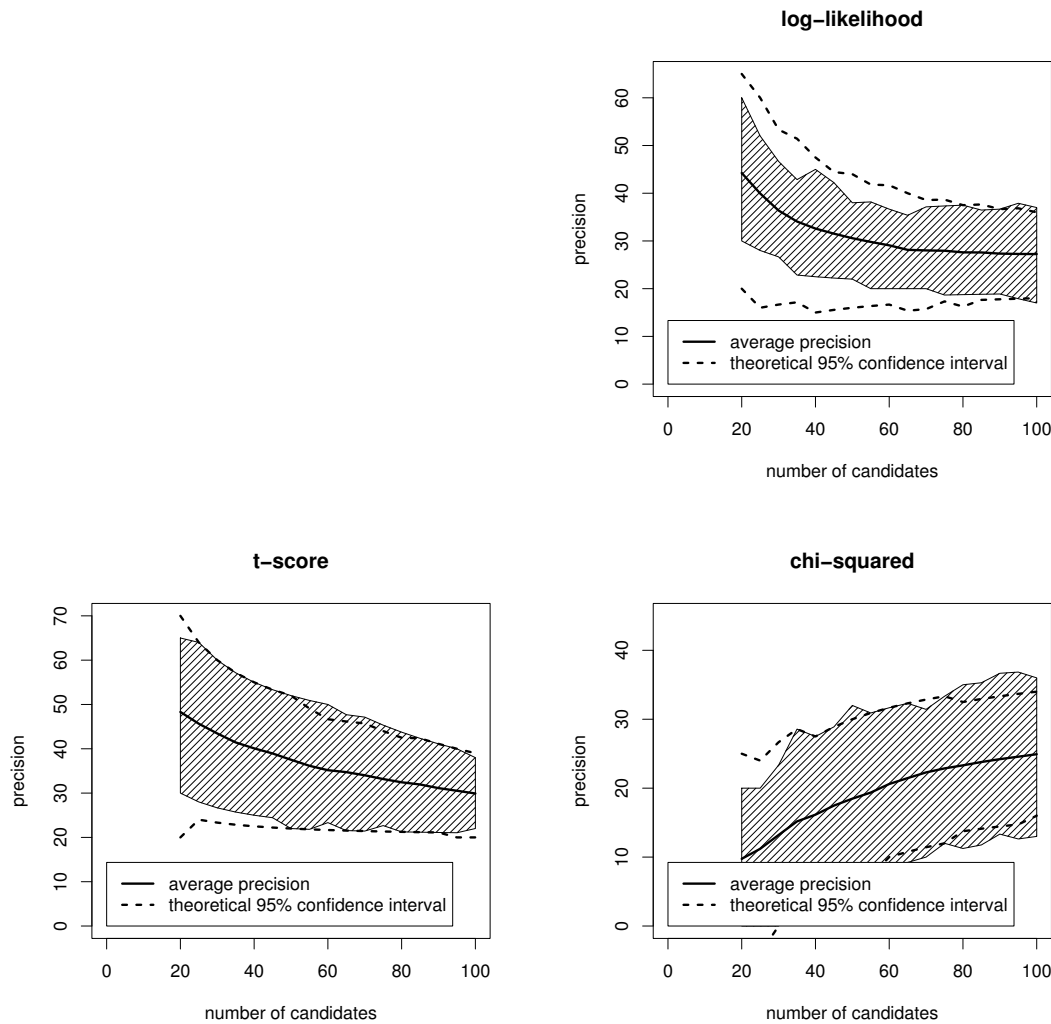


Figure 5.14: Empirical confidence intervals for the n -best precision $p_{g,n}$ of the association measures G^2 (top right panel), X^2 (bottom right panel) and t (bottom left panel).

for this conclusion comes from a direct validation of the confidence interval for p_A on a γ -acceptance region $A = A_g(\gamma)$. At 95% confidence, the true proportion p_A should fall within the confidence interval for all but 4 of the 80 samples. For G^2 (with $\gamma = 32.5$) and X^2 (with $\gamma = 239.0$), p_A was outside the confidence interval in 9 cases each (three of them very close to the boundary), while the confidence interval for t (with $\gamma = 2.09$) failed in 12 cases, which is significantly more than can be explained by chance.

I have already pointed out that the application of the confidence intervals developed in Section 5.2.2 to n -best precision may be problematic, since these estimates are based on the predicted sampling distribution of P_A for a fixed γ -acceptance region A (which happens to contain exactly n candidates in the observed sample) rather than on the sampling distribution of $P_{g,n}$. In order to test the validity of the confidence intervals, the sampling distribution of $P_{g,n}$ was estimated for $n = 20 \dots 100$ and for the association measures G^2 , X^2 and t . The shaded areas in Figure 5.14 represent

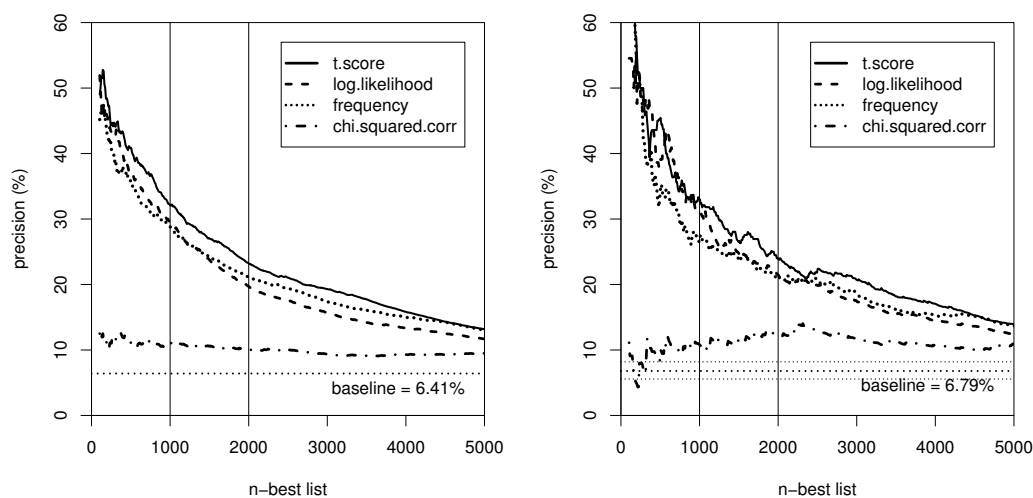


Figure 5.15: An illustration of the use of random samples for evaluation: precision graphs for the PNV-KRENN data set (left) and the corresponding estimates obtained from a 10% sample (right).

empirical 95% confidence intervals for the observed n -best precision $P_{g,n}$, given by the smallest range that contains all but the four most extreme values of $P_{g,n}$ from the 80 samples. The solid curves show the true n -best precision $p_{g,n}$, obtained by averaging over all 80 samples. The dashed lines delimit theoretical 95% confidence intervals around $p_{g,n}$ resulting from a “naive” application of the methods described in Section 5.2.2 to n -best lists. If the observed n -best precision $P_{g,n}$ falls within these intervals, it is deemed compatible with the true value $p_{g,n}$. Obviously, the theoretical and empirical values agree very well (the slight discrepancy for G^2 can be explained by random fluctuations due to the small size of the n -best lists), so that the methods of Section 5.2.2 are indeed applicable to an evaluation that is based on n -best lists.

5.3 Evaluation based on random samples

In order to reduce the amount of manual work, evaluation experiments can be based on random samples from a data set. Figure 5.15 compares evaluation results for the full data set PNV-KRENN (Krenn 2000) in the left panel, with those obtained from a 10% random sample (of the candidate types) in the right panel. Note that the values on the x -axis refer to n -best lists of the original data set. For instance, the precision values for $n = 1\,000$ in the right panel have been estimated from a sample of approx. 100 candidates. The overall impression given by the random sample evaluation is qualitatively correct: t-score emerges as the best measure, mere frequency sorting outperforms log-likelihood (at least for $n \geq 4\,000$), and chi-squared is much worse than the other measures but still clearly better than the baseline. However, the findings are much less clear-cut than for the full evaluation. The precision graphs become unstable and unreliable for $n \leq 1\,000$ where log-likelihood seems to be better than frequency and chi-squared comes close to the baseline. This is hardly surprising, considering the fact that these estimates are based on fewer than one hundred

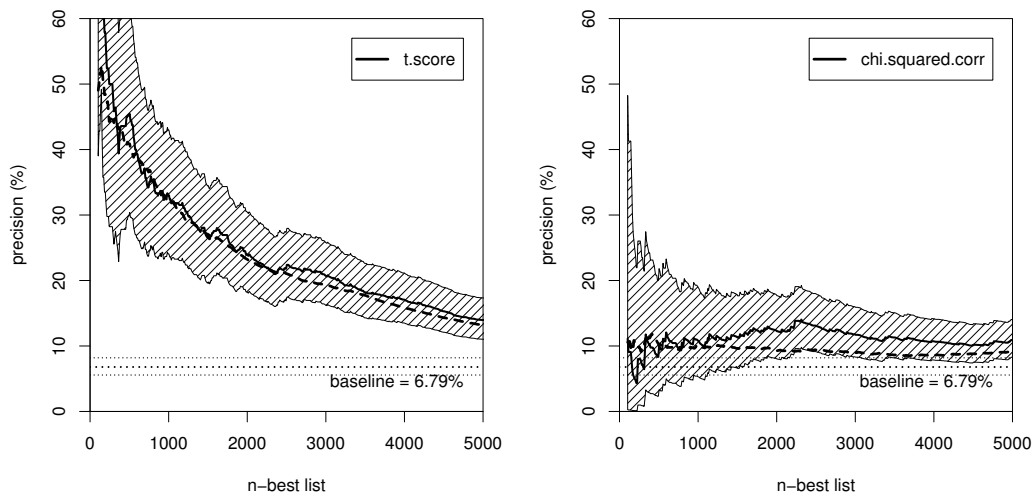


Figure 5.16: Sample estimates for the true precision with confidence intervals based on a 10% random sample. The dashed lines show the true precision computed from the full candidate set.

annotated candidates.

It is therefore particularly important to apply the significance tests of Section 5.2, in order to rule out the possibility that evaluation results are merely flukes introduced by the sampling process. Since these significance tests cast evaluation as random sampling (from a hypothetical population), no special treatment is necessary for the additional explicit sampling step (which samples from a concrete data set of collocation candidates). Its effect is simply to reduce the resulting sample size (by a factor of 10 in the current example). As a consequence, the uncertainty of the precision estimates is increased and confidence intervals become larger. In this approach, the random sample is used to generalise beyond the data set from which it was taken and draw inferences about average precision in the underlying (hypothetical) population. Evert and Krenn (2005) give a slightly different account with an explicit description of the procedures for a random sample evaluation. Their goal is to draw inferences about a specific data set based on a random sample of candidates from this set. However, after some approximations they arrive at the same statistical methods as described in Section 5.2.

Figure 5.16 shows confidence intervals for the true n -best precision estimated from the 10% sample described above, and compares them directly with precision graphs obtained from the full data set. The baseline shown in these plots has been estimated from the sample, and a confidence interval for the true baseline precision is indicated by thin dotted lines above and below the estimate. From the right panel, we can see that there is considerable uncertainty in the precision graph of chi-squared. For most of the n -best lists, the true precision might be close to the baseline or even below it. On the other hand, it might be as high as 20% for $n \leq 1000$ (where the direct estimate comes close to the baseline). The left panel, on the other hand, shows that the true n -best precision of t-score is at least 20% for $n \leq 2000$. The precision graphs obtained from the full data set have been inserted as dashed lines in both panels. For t-score the sampling variation is much smaller than predicted by the

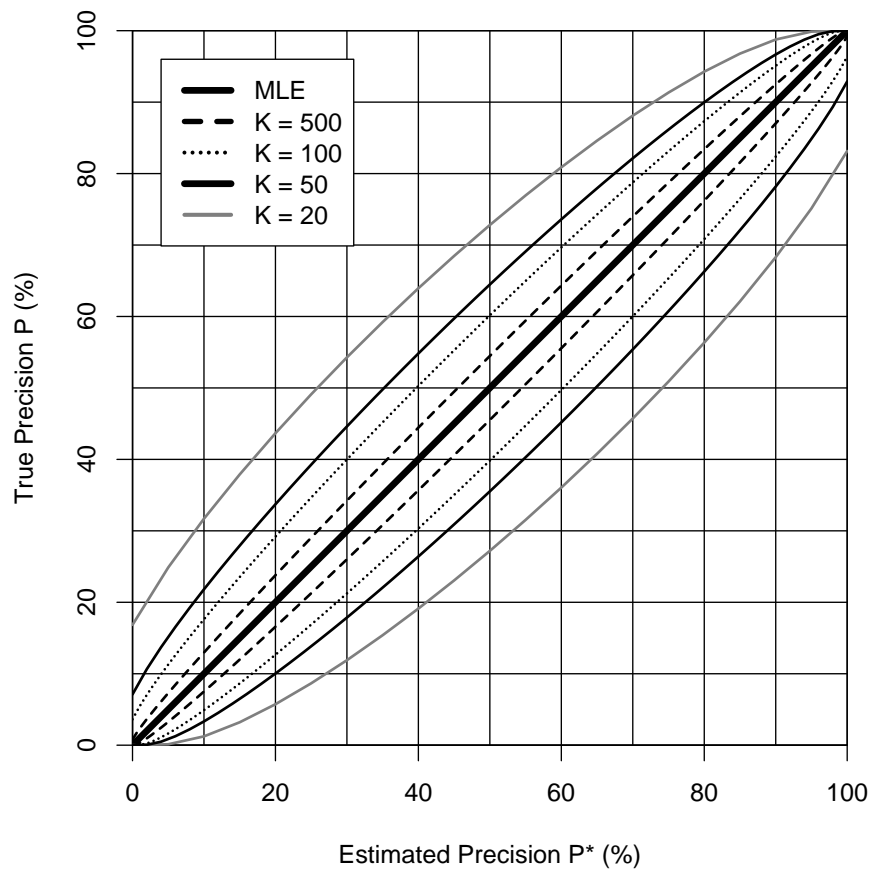


Figure 5.17: Chart of binomial confidence intervals for selected sample sizes.

significance tests. The right panel shows that our confidence intervals are not overly conservative, though: for $n \geq 2000$ the chi-squared n -best precision calculated from the full data set is close to the boundary of the confidence interval.

Obviously, it is essential to choose an appropriate sampling rate. On the one hand, the manual work should be reduced as far as possible so that more evaluation studies can be carried out under different conditions (enabling us to achieve a more complete understanding of the empirical properties of association measures). On the other hand, the confidence intervals have to be narrow enough so that we can draw meaningful conclusions from them. The width of confidence intervals depends both on the size of the sample and on the estimated precisions. Charts of binomial confidence intervals such as the one in Figure 5.17 help with this decision. For any value of estimated precision on the x -axis, 95% confidence intervals for the true precision at different sample sizes K can be read from the y -axis. For instance, when the observed precision in a sample of size $K = 100$ is 40%, the true precision can be narrowed down to the range between 30% and 50% with 95% confidence. Note that K is the absolute number of candidates used for the estimation: a 1 000-best list at a sampling rate of 10% and a 500-best list at a sampling rate of 20% lead to the same sample size $K = 100$. In other words, the lower the sampling rate, the larger n -best lists have to be so that meaningful estimates are possible.

Figure 5.18 shows another example of a random sample evaluation. Here, a 15%

sample was taken from 8 546 high-frequency adjective-noun pairs in the AN-FR data set (with $f \geq 20$) and manually annotated by professional lexicographers. The annotators accepted both collocations and typical (but less rigid) combinations as true positives, the main criterion being whether the candidates would be useful for the compilation of a large German-English dictionary.¹¹ The results of this evaluation are quite surprising in view of previous findings. Frequency-based ranking is not significantly better than the baseline, while both t-score and log-likelihood are clearly outperformed by the chi-squared measure, contradicting the arguments of Dunning (1993). For $1\,000 \leq n \leq 3\,000$, the precision of chi-squared is significantly better than that of log-likelihood, and its overestimation of the significance of association seems to have a beneficial effect. The bottom panel has an even greater surprise in store: the MS measure (Pedersen and Bruce 1996), which has never found widespread use, achieves the best results in this evaluation. It is closely followed by Dice (not shown in the graph), and both are significantly better than chi-squared. This is particularly interesting because it is one of the rare situations where the best-performing association measures are not central (or nearly central) measures. A close look at the evaluated data set and the iso-surfaces of these measures (using three-dimensional visualisation techniques from Section 3.3.3) will be necessary in order to throw light on the reasons that lie behind such unexpected results.

¹¹I would like to thank the Wörterbuchredaktion of the publishing house Langenscheidt KG, Munich for annotating this sample. The evaluation reported here emerged from a collaboration within the project TFB-32, funded at the University of Stuttgart by the DFG.

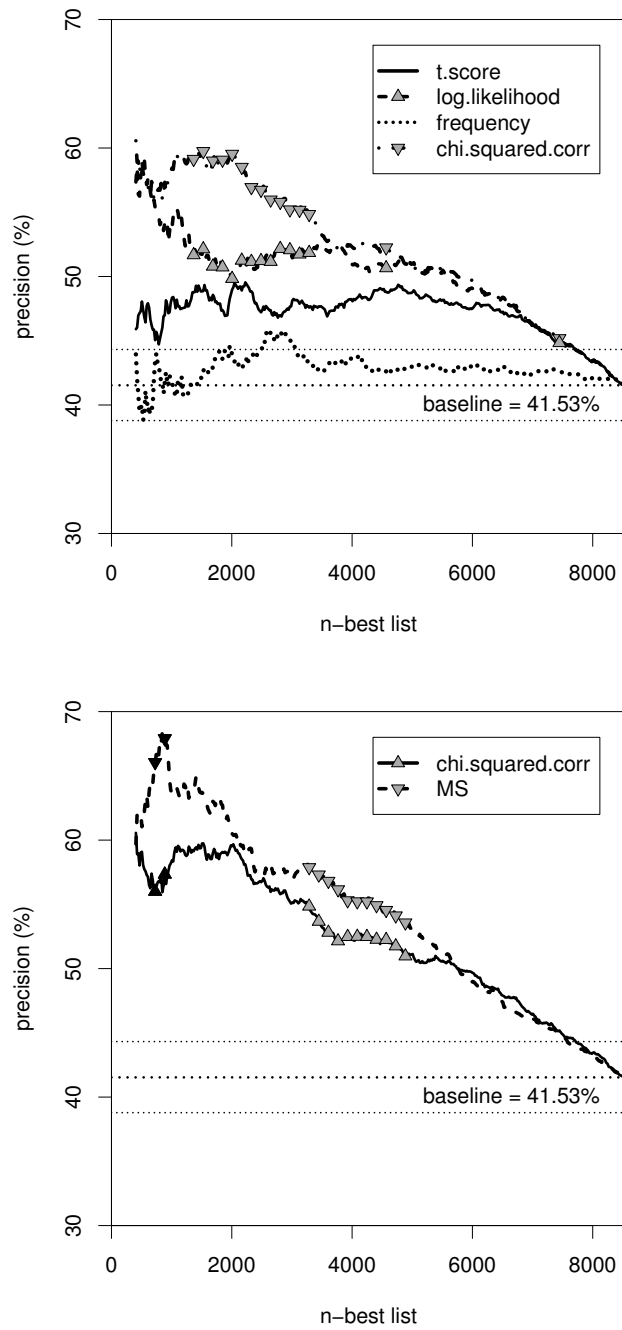


Figure 5.18: Random sample evaluation of German adjective-noun combinations.

Chapter 6

Conclusion and Future Work

Returning to the quote from Giuliano (1965b, 259) cited in Section 1.3.1, let us ask how much progress we have made towards a better understanding of the statistical association of word combinations. One thing is clear: collocation extraction is not a solved problem. More research will be needed, empirical research in particular, before we can explain puzzling results such as the evaluation experiment reported at the end of Section 5.3 (which seems to defy all established wisdom about association measures and the statistical properties of collocations). My aim in writing this book has been to provide the necessary background, research methodology and tools for such research, collected in a single volume as a handy reference.

Each of these three aspects of my work accounts for a substantial part of the text. The first, encyclopedic part extends from the beginning of Chapter 2 to Section 3.2. At first, the formal and mathematical foundations of association measures are presented: procedures for obtaining cooccurrence frequency data and statistical models for their interpretation. I make a clear distinction between relational and positional cooccurrences, which require different counting methods in order to allow for a valid statistical analysis. In Sections 2.1 and 2.4 these counting methods are formalised to the degree necessary to make them unambiguous, and they are accompanied by explicit instructions, schemata and examples that should facilitate their implementation. Section 2.2 describes the statistical model underlying the analysis of the extracted frequency data. Although this is a well-known random sample model, and it is always implicitly assumed when authors discuss or suggest association measures, its application to cooccurrence frequency data has never been given an explicit and precise definition.¹ In this section, I also address the difficult issue of how statistical association can be quantified. In Section 2.3 I discuss some problematic aspects of the random sample model, in particular the randomness assumption (which everyone knows to be untenable, but which is nonetheless rarely tested) and the issue of noise introduced by automatic processing (which everyone hopes will be filtered out by the statistical analysis, without making clear *why* the association measures should be able to achieve this).

Chapter 3 is the centrepiece of my thesis. In its first section, which still belongs to the encyclopedic part, it provides a comprehensive inventory of all association measures that I have come across during my research. The numerous measures are

¹Cooccurrence data as a random sample from what population? And what are the relevant parameters, random variables and test statistics?

organised in major and minor groups which share a common goal or theoretical background. In addition to this broad categorisation and the standard references, I take care to explain details that are usually either ignored or taken for granted. Examples are the differences between one-tailed and two-tailed measures, the application of Yates' continuity correction, and the equivalence of different versions of the chi-squared and log-likelihood measures (see Section 3.1.4 for all three examples). For each association measure, an explicit equation is given. All these equations use the same notation based on observed and expected frequencies for the cells of a contingency table. In addition, carefully designed reference implementations of the measures are available in the UCS toolkit (Section 3.2.2). There is also an on-line version of the repository at <http://www.collocations.de/AM/> with the most up-to-date information.

The second part of the book, which is concerned with research methodology, begins in Section 3.3. There, generalised association measures are introduced as arbitrary real-valued functions on contingency tables that conform to our intuitions about the fundamental properties of an association measure. This formal model leads to an intuitive geometric interpretation of cooccurrence data and association measures in a three-dimensional "parameter space", which will hopefully pave the way towards a better understanding of the characteristics of existing measures and towards the discovery of genuinely new ones. The frequency data extracted from a corpus are visualised as a point cloud in the parameter space, each point representing a single word pair. Generalised association measures can be visualised as surfaces in this space, and their properties are determined by the geometric shapes of the respective surfaces. In many cases, the parameter space can be projected to a two-dimensional plane (occasionally involving minor approximations), which simplifies visualisation and analysis considerably. As a first application, the geometric approach is used in combination with other techniques for a more detailed analysis of the major groups of association measures in Section 3.4.

Chapter 4 addresses the well-known problem of low-frequency data. Most researchers know that statistical inference from small amounts of data is problematic (to say the least).² Although Dunning (1993) suggests that the applicability of his newly introduced log-likelihood measure extends even down to the hapax legomena (word combinations that occur just once in a corpus) – and although Weeber *et al.* (2000) see opportunities to extract useful knowledge from such lowest-frequency data – most researchers silently discard rare events by setting a frequency threshold (Krenn (2000) is just one example among many). Using methods from lexical statistics, I show that reliable statistical inference is impossible *in principle* for the hapax and dis legomena ($f = 1, 2$). In this frequency range, quantisation effects and the characteristic highly skewed distribution of the cooccurrence probabilities of pair types (roughly following Zipf's law) dominate over the random variation that statistical inference normally takes into account. As a result, probability estimates are entirely unreliable unless the precise shape of the population is known. This rather negative result provides theoretical support for the application of a frequency threshold, which should at least exclude the hapax and dis legomena ($f \geq 3$). Quantisation and the shape of the population no longer play a role for $f \geq 5$, so that higher cutoff

²"Only naughty brewers deal in small samples", as Karl Pearson once put it.

thresholds are not necessary in order to ensure a reliable statistical analysis.³ A fall-out from this work is a new population model for the distribution of cooccurrence probabilities, which is analytically simple and numerically efficient. Despite its simplicity, the model compares favourably with established population models (Baayen 2001), combining better goodness-of-fit with higher robustness.

Finally, Chapter 5 addresses the relation between cooccurrences and collocations, using cooccurrence data extracted from a text corpus as candidate data for a collocation identification task. This application setting provides a framework – and a well-defined goal – for the comparative evaluation of association measures. The graphical presentation of the evaluation results, first used by Evert *et al.* (2000) and Evert and Krenn (2001), is developed further and a case study exemplifies the possibilities opened up by a fine-grained evaluation. Section 5.2 addresses the problem of testing the significance of evaluation results. An attempt is made to clear up the confusion about the choice of an appropriate significance test by introducing an explicit model for the random variation of evaluation results (which is formulated in terms of the geometric interpretation from Section 3.3). Based on this model, two procedures are suggested: (i) confidence intervals estimate the uncertainty in the evaluation results of a single association measure; and (ii) significance tests predict whether the observed differences between measures can reliably be reproduced in other experiments (under similar conditions). The model is validated on empirical data, showing that it provides a relatively tight lower bound for the true variation. Finally, the newly developed methods are applied to an evaluation procedure that reduces the amount of manual annotation work drastically by taking a random sample from the candidate set. With this new procedure, it will be possible to perform evaluation experiments under a much broader range of conditions. A first example of such an experiment, presented at the very end of Chapter 5, is already full of surprises (as has been mentioned at the beginning of this chapter).

The third, computational aspect of my research does not really belong into the text, but is an open-source software package, called the UCS toolkit, that accompanies the thesis. It provides reference implementations of all association measures listed in Chapter 3, as well as all the libraries, utilities and data sets that are needed to replicate the experiments and analyses described in the text (and even reproduce most of the graphs). This includes an R library for evaluation graphs with support for significance tests and random sample evaluation (Chapter 5), Perl utilities for carrying out the dispersion test that is used to verify the randomness assumption (Section 2.3.2), and an implementation of the new population models for the distribution of cooccurrence probabilities (Section 4.2). There is also an implementation of a number of generalised association measures and the two-dimensional visualisation procedure in the newest version of the package. With this software, which can be downloaded from <http://www.collocations.de/phd.html>, the tools needed for the study of association measures and the underlying statistical models are finally at everyone's fingertips. Appendix B contains the complete documentation of the UCS toolkit.

Although most of the discussions and examples in this thesis assume cooccur-

³There may be other reasons to apply a higher frequency threshold, of course, such as working around the problems that some association measures have with low-frequency data, or the inflation of observed frequencies through non-randomness effects.

rences of words in a text corpus, the methods that are presented and the conclusions drawn are applicable to a much broader range of phenomena, as long as they can be made to fit within the formal definitions of Chapter 2. The source of the cooccurrences need not be a text corpus, provided that it is possible to identify sets of tokens and combine them into pairs. The cooccurring items need not be words, provided that they have the characteristic skewed probability distribution of lexical data.⁴ The definition of cooccurrence may range from pairs of aligned sentences in a bilingual corpus (e.g. Church and Gale 1991) to the adjacency of nodes in a graph (Biemann *et al.* 2004).

What is it, then, that still needs to be done? The mathematical theory behind association measures and the underlying statistical models has been studied extensively, but the theoretical conclusions are all too often not borne out in practice. As an example, consider the evaluation of adjective-noun combinations in Section 5.3. The superiority of G^2 (log-likelihood) compared to X^2 (chi-squared) seemed to have been established beyond doubt, yet in this experiment X^2 achieved significantly better results. This goes to show that more empirical data needs to be collected in order to improve our understanding of cooccurrence data, statistical association and its relation to collocativity. My thesis provides the background, methods and tools for such studies: now evaluation experiments have to be carried out and old as well as new hypotheses and assumptions have to be tested under a wide range of conditions.

⁴For instance, the items might be conceptual classes in a semantic taxonomy such as WORDNET (Miller 1990). Alshawi and Carter (1994) refer to such associations as *semantic lexical collocations*, and Resnik (1997) uses cooccurrences between a predicate and the classes of its arguments for word sense disambiguation.

Appendix A

Proofs and Mathematical Background

A.1 Proofs from Chapter 2

Lemma A.1. *The maximum-likelihood estimates for the population parameters $\vec{\tau}$ under the multinomial sampling distribution (2.5) are given by $\hat{\tau}_{ij} = O_{ij}/N$. The MLEs for the alternative parameters (π, π_1, π_2) are $\hat{\pi} = O_{11}/N$, $\hat{\pi}_1 = R_1/N$ and $\hat{\pi}_2 = C_1/N$.*

Proof. The proof uses Fisher's device of representing the multinomial distribution as a conditional probability of the independent Poisson distribution (2.7). For any parameter values $\vec{\tau}$ that satisfy the multinomial condition $\sum_{ij} \tau_{ij} = 1$, we have

$$\Pr(\vec{X} = \vec{O} \mid N, \vec{\tau}) = \frac{\Pr(\vec{X} = \vec{O} \mid \nu \vec{\tau})}{\Pr(\sum_{ij} X_{ij} = N \mid \nu \vec{\tau})} \quad (\text{A.1})$$

regardless of the value chosen for ν (note that $\sum_{ij} X_{ij} = \sum_{ij} O_{ij} = N$, hence this condition does not have to be stated explicitly in the numerator). Because ν can be chosen in an arbitrary way and $\sum_{ij} X_{ij}$ has a Poisson distribution with the single parameter ν under (2.7), the denominator in (A.1) is constant and it follows that

$$\arg \max_{\vec{\tau}} \Pr(\vec{X} = \vec{O} \mid N, \vec{\tau}) = \arg \max_{\vec{\tau}} \Pr(\vec{X} = \vec{O} \mid \nu \vec{\tau}), \quad (\text{A.2})$$

again regardless of the value ν . The unconstrained probability $\Pr(\vec{X} = \vec{O} \mid \nu \vec{\tau})$ is the product of four independent Poisson distributions with parameters $\nu \tau_{ij}$,

$$\Pr(\vec{X} = \vec{O} \mid \nu \vec{\tau}) = \prod_{ij} \Pr(X_{ij} = O_{ij} \mid \nu \tau_{ij}). \quad (\text{A.3})$$

Therefore, it assumes a global maximum for the Poisson MLEs $\nu \tau_{ij} = O_{ij}$. Choosing $\nu = N$, the right-hand side of (A.2) reaches this global maximum for $\tau_{ij} = O_{ij}/N$. Since these values satisfy the summation condition $\sum_{ij} \tau_{ij} = 1$ for multinomial parameters, they also maximise the left-hand side of (A.2). Hence, they are the desired multinomial MLEs $\hat{\tau}_{ij} = O_{ij}/N$. The MLEs for (π, π_1, π_2) are obtained by direct summation: $\hat{\pi} = \hat{\tau}_{11} = O_{11}/N$, $\hat{\pi}_1 = \hat{\tau}_{11} + \hat{\tau}_{12} = R_1/N$, and $\hat{\pi}_2 = \hat{\tau}_{11} + \hat{\tau}_{21} = C_1/N$. \square

Lemma A.2. *The coefficients of association strength μ , ρ , θ , κ_{ub} , κ_{Dice} , κ_{gmean} and κ_{min} assume the values shown in Table 2.2 for the special situations listed in Table 2.1. Cases B and E are first-order approximations for $\epsilon \rightarrow 0$*

Proof. In the proofs for cases B and E, Landau notation $\mathcal{O}(\epsilon)$ is used and first-order approximations are denoted by writing \doteq instead of $=$. Only non-trivial calculations are shown (except for κ_{Jaccard}). For relative risk ρ and the odds ratio θ , the following definitions are normally used:

$$\rho = \frac{\pi(1 - \pi_2)}{\pi_2(\pi_1 - \pi)}$$

$$\theta = \frac{\pi(1 - \pi_1 - \pi_2 + \pi)}{(\pi_1 - \pi)(\pi_2 - \pi)}$$

Some proofs make use of the geometric series:

$$\frac{1}{1 - \epsilon} = 1 + \epsilon + \mathcal{O}(\epsilon^2) \quad \text{and} \quad \frac{1}{a - \epsilon b} = a^{-1} \left(1 + \epsilon \frac{b}{a} + \mathcal{O}(\epsilon^2) \right)$$

Case A: $\pi = \pi_1\pi_2$ (independence)

$$\rho = \frac{\pi_1\pi_2(1 - \pi_2)}{\pi_2(\pi_1 - \pi_1\pi_2)} = 1$$

$$\theta = \frac{\pi_1\pi_2(1 - \pi_1)(1 - \pi_2)}{(\pi_1 - \pi_1\pi_2)(\pi_2 - \pi_1\pi_2)} = 1$$

$$\kappa_{\text{min}} = \min \left\{ \frac{\pi_1\pi_2}{\pi_1}, \frac{\pi_1\pi_2}{\pi_2} \right\} = \min \{ \pi_2, \pi_1 \}$$

Case B: $\pi = (1 + \epsilon)\pi_1\pi_2$ (minimal association)

$$\begin{aligned} \rho &= \frac{(1 + \epsilon)(1 - \pi_2)}{1 - \pi_2 - \epsilon\pi_2} = (1 + \epsilon) \left(1 - \frac{\epsilon\pi_2}{1 - \pi_2} \right)^{-1} \\ &= (1 + \epsilon) \left(1 + \frac{\epsilon\pi_2}{1 - \pi_2} + \mathcal{O}(\epsilon^2) \right) = 1 + \epsilon + \epsilon \frac{\pi_2}{1 - \pi_2} + \mathcal{O}(\epsilon^2) \\ &\doteq 1 + \frac{\epsilon}{1 - \pi_2} \end{aligned}$$

The calculation for θ makes use of the abbreviation $\delta := 1 + \epsilon$ and the identities

$$(1 - \delta\pi_k)^{-1} = ((1 - \pi_k) - \epsilon\pi_k)^{-1} = (1 - \pi_k)^{-1} \left(1 + \epsilon \frac{\pi_k}{1 - \pi_k} + \mathcal{O}(\epsilon^2) \right)$$

for $k = 1, 2$. Inserting $\pi = \delta\pi_1\pi_2$ into the definition of θ yields

$$\begin{aligned}\theta &= \frac{\delta(1 - \pi_1 - \pi_2 + \delta\pi_1\pi_2)}{(1 - \delta\pi_1)(1 - \delta\pi_2)} \\ &= (1 + \epsilon) \cdot \frac{(1 - \pi_1)(1 - \pi_2) + \epsilon\pi_1\pi_2}{(1 - \pi_1)(1 - \pi_2)} \\ &\quad \cdot \left(1 + \epsilon \frac{\pi_1}{1 - \pi_1} + \mathcal{O}(\epsilon^2)\right) \cdot \left(1 + \epsilon \frac{\pi_2}{1 - \pi_2} + \mathcal{O}(\epsilon^2)\right) \\ &= (1 + \epsilon) \left(1 + \epsilon \frac{\pi_1}{1 - \pi_1} \frac{\pi_2}{1 - \pi_2}\right) \left(1 + \epsilon \frac{\pi_1}{1 - \pi_1}\right) \left(1 + \epsilon \frac{\pi_2}{1 - \pi_2}\right) + \mathcal{O}(\epsilon^2) \\ &= 1 + \epsilon \left(1 + \frac{\pi_1}{1 - \pi_1}\right) \left(1 + \frac{\pi_2}{1 - \pi_2}\right) + \mathcal{O}(\epsilon^2) \\ &\doteq 1 + \frac{\epsilon}{(1 - \pi_1)(1 - \pi_2)}\end{aligned}$$

$$\kappa_u = \frac{\epsilon\pi_1\pi_2}{\pi_2(1 - \pi_2)} = \epsilon \frac{\pi_1}{1 - \pi_2}$$

$$\kappa_{\min} = \min \left\{ \frac{(1 + \epsilon)\pi_1\pi_2}{\pi_1}, \frac{(1 + \epsilon)\pi_1\pi_2}{\pi_2} \right\} = (1 + \epsilon) \min \{\pi_2, \pi_1\}$$

Case C: $\pi = 0$ (total negative association)

$$\kappa_u = \frac{0 - \pi_1\pi_2}{\pi_2(1 - \pi_2)} = -\frac{\pi_1}{1 - \pi_2}$$

All other results for case C are trivial.

Case D: $\pi = \pi_1 = \pi_2$ (total positive association)

$$\rho = \frac{\pi(1 - \pi)}{\pi(\pi - \pi)} = \frac{1 - \pi}{0} = \infty$$

$$\theta = \frac{\pi(1 - \pi - \pi + \pi)}{(\pi - \pi)(\pi - \pi)} = \frac{\pi(1 - \pi)}{0} = \infty$$

$$\kappa_u = \frac{\pi - \pi\pi}{\pi(1 - \pi)} = 1$$

$$\kappa_{\min} = \min \left\{ \frac{\pi}{\pi}, \frac{\pi}{\pi} \right\} = 1$$

Case E: $\pi_1 = \pi_2 = (1 + \epsilon)\pi$ (nearly total association)

$$\mu = \frac{\pi}{\pi^2(1 + \epsilon)^2} = \frac{1}{\pi}(1 - 2\epsilon + \mathcal{O}(\epsilon^2)) \doteq \frac{1}{\pi} - \epsilon \frac{2}{\pi}$$

$$\rho = \frac{\pi(1 - \pi - \epsilon\pi)}{(1 + \epsilon)\pi \cdot \epsilon\pi} = \frac{1}{\epsilon} \cdot \frac{1}{\pi} \cdot \frac{1}{1 + \epsilon}(1 - \pi - \epsilon\pi)$$

$$= \frac{1}{\epsilon} \cdot \frac{1}{\pi} \cdot (1 + \mathcal{O}(\epsilon)) \cdot (1 - \pi + \mathcal{O}(\epsilon))$$

$$= \frac{1}{\epsilon} \cdot \frac{1 - \pi}{\pi} + \mathcal{O}(1)$$

$$\begin{aligned}\theta &= \frac{\pi(1-\pi-2\epsilon\pi)}{\epsilon^2\pi^2} = \frac{1}{\epsilon^2} \cdot \frac{1}{\pi} \cdot (1-\pi + \mathcal{O}(\epsilon)) \\ &= \frac{1}{\epsilon^2} \cdot \frac{1-\pi}{\pi} + \mathcal{O}\left(\frac{1}{\epsilon}\right)\end{aligned}$$

$$\begin{aligned}\kappa_u &= \frac{1-(1+\epsilon)^2\pi}{(1+\epsilon)(1-\pi-\epsilon\pi)} \\ &= \left[1-\pi-2\epsilon\pi + \mathcal{O}(\epsilon^2)\right] \cdot \left[1-\pi+\epsilon(1-2\pi) + \mathcal{O}(\epsilon^2)\right]^{-1} \\ &= \left[1-\pi-2\epsilon\pi + \mathcal{O}(\epsilon^2)\right] \cdot \frac{1}{1-\pi} \cdot \left[1-\epsilon\frac{1-2\pi}{1-\pi} + \mathcal{O}(\epsilon^2)\right] \\ &= \left[1-\epsilon\frac{2\pi}{1-\pi} + \mathcal{O}(\epsilon^2)\right] \cdot \left[1-\epsilon\frac{1-2\pi}{1-\pi} + \mathcal{O}(\epsilon^2)\right] \\ &= 1-\epsilon\frac{1}{1-\pi} + \mathcal{O}(\epsilon^2) \doteq 1-\frac{\epsilon}{1-\pi}\end{aligned}$$

$$\kappa_{\text{Dice}} = \frac{2\pi}{2 \cdot (1+\epsilon)\pi} = \frac{1}{1+\epsilon} \doteq 1-\epsilon$$

The κ_{gmean} and κ_{min} coefficients also lead to $(1+\epsilon)^{-1} \doteq 1-\epsilon$.

Case F: $\pi = \pi_1 \ll \pi_2$ (total determination, $u \rightarrow v$)

$$\begin{aligned}\rho &= \frac{\pi_1(1-\pi_2)}{\pi_2(\pi_1-\pi_1)} = \frac{\pi_1(1-\pi_2)}{0} = \infty \\ \theta &= \frac{\pi_1(1-\pi_1-\pi_2+\pi_1)}{(\pi_1-\pi_1)(\pi_2-\pi_1)} = \frac{\pi_1(1-\pi_2)}{0} = \infty \\ \kappa_u &= \frac{\pi_1-\pi_1\pi_2}{\pi_2(1-\pi_2)} = \frac{\pi_1}{\pi_2} \\ \kappa_{\text{Dice}} &= \frac{2\pi_1}{\pi_1+\pi_2} = \frac{2}{1+\pi_2/\pi_1} < 1 \\ \kappa_{\text{min}} &= \min\left\{\frac{\pi_1}{\pi_1}, \frac{\pi_1}{\pi_2}\right\} = \min\left\{1, \frac{\pi_1}{\pi_2}\right\} = \frac{\pi_1}{\pi_2}\end{aligned}$$

Case F': $\pi = \pi_2 \ll \pi_1$ (total determination, $v \rightarrow u$)

$$\begin{aligned}\rho &= \frac{\pi_2(1-\pi_2)}{\pi_2(\pi_1-\pi_2)} = \frac{1-\pi_2}{\pi_1-\pi_2} \\ \theta &= \frac{\pi_2(1-\pi_1-\pi_2+\pi_2)}{(\pi_1-\pi_2)(\pi_2-\pi_2)} = \frac{\pi_2(1-\pi_1)}{0} = \infty \\ \kappa_u &= \frac{\pi_2-\pi_1\pi_2}{\pi_2(1-\pi_2)} = \frac{1-\pi_1}{1-\pi_2}\end{aligned}$$

□

Lemma A.3. Divide a random sample of size N into K parts of S tokens each (i.e. $N = K \cdot S$). For a given type $w \in C$, let X_i stand for the local frequency of w in the i -th part, and write $\vec{X} = (X_1, \dots, X_K)$ for the vector of local frequencies. Then the total frequency of X is $f = \sum_{i=1}^K X_i$ and the dispersion of w in the sample is $D = \sum_{i=1}^K I_{[X_i > 0]}$. For any integers $d, m \in \mathbb{N}$ with $d \leq m \leq S$ and $d \leq K$, the conditional probability of observing a dispersion of d given a total frequency of m is given by

$$\Pr(D = d \mid f = m) = \binom{N}{m}^{-1} \binom{K}{d} \sum_{j=1}^d (-1)^{d-j} \binom{d}{j} \binom{S \cdot j}{m}. \quad (\text{A.4})$$

Proof. Let $\vec{m} = (m_1, \dots, m_K) \in \{0, \dots, S\}^K$ stand for a possible distribution of the instances of w across the K parts of the sample (i.e. a vector of local frequencies), $S(\vec{m}) := \sum_{i=1}^K m_i$ for the corresponding total frequency, and $D(\vec{m}) := \sum_{i=1}^K I_{[m_i > 0]}$ for the corresponding dispersion. Writing $\vec{X} = \vec{m}$ for the condition $\forall i : X_i = m_i$, the conditional probability $\Pr(D = d \mid f = m)$ expands to

$$\Pr(D = d \mid f = m) = \sum_{\substack{D(\vec{m})=d \\ S(\vec{m})=m}} \Pr(\vec{X} = \vec{m} \mid f = m), \quad (\text{A.5})$$

where $\Pr(\vec{X} = \vec{m} \mid f = m)$ is the probability of a specific distribution \vec{m} given the total frequency m . By applying Eq. (2.6) to each (independent) part of the sample, we find

$$\begin{aligned} \Pr(\vec{X} = \vec{m}) &= \prod_{i=1}^K \binom{S}{m_i} \pi^{m_i} (1 - \pi)^{S - m_i} \\ &= \pi^{S(\vec{m})} (1 - \pi)^{N - S(\vec{m})} \cdot \prod_{i=1}^K \binom{S}{m_i}, \end{aligned}$$

where π is the occurrence probability of the type w . Since

$$\Pr(f = m) = \binom{N}{m} \pi^m (1 - \pi)^{N - m},$$

we obtain

$$\Pr(\vec{X} = \vec{m} \mid f = m) = \begin{cases} \frac{\prod_{i=1}^K \binom{S}{m_i}}{\binom{N}{m}} & S(\vec{m}) = m \\ 0 & \text{otherwise} \end{cases}. \quad (\text{A.6})$$

A pair of integers $d, m \in \mathbb{N}_0$ is called *admissible* iff $1 \leq d \leq m \leq N$ and $\Pr(D = d \mid f = m) > 0$, i.e. there exists at least one local frequency vector \vec{m} with $D(\vec{m}) = d$ and $S(\vec{m}) = m$.¹ We will now determine a generating function for the probabilities in (A.4) from (A.5) and (A.6). More precisely, we are interested in the formal power series

$$f(x, y) := \sum_{d=0}^{\infty} \sum_{m=0}^{\infty} p_{d,m} y^d x^m \quad (\text{A.7})$$

¹For instance, $d = 1, m > S$ is not admissible since the $m > S$ instances of w cannot lie in a single part, which consists of S tokens only.

with

$$p_{d,m} := \begin{cases} \Pr(D = d \mid f = m) \cdot \binom{N}{m} & d, m \text{ are admissible} \\ 1 & d = m = 0 \\ 0 & \text{otherwise} \end{cases}. \quad (\text{A.8})$$

Note that only a finite number of the terms in (A.7) are non-zero. Consider the function

$$g(x, y) := 1 + y((1+x)^S - 1) = 1 + y \cdot \sum_{k=1}^S \binom{S}{k} x^k,$$

which is suggested by (A.6). We will now show that $f(x, y) = (g(x, y))^K$. In fact, each term in the expansion of this product can be represented by a distribution vector \vec{m} , picking the term $yx^{m_i} \binom{S}{m_i}$ from the i -th factor, or the constant term 1 for $m_i = 0$:

$$\begin{aligned} (g(x, y))^K &= \sum_{\vec{m}} \prod_{i=1}^K y^{I_{[m_i > 0]}} x^{m_i} \binom{S}{m_i} \\ &= \sum_{\vec{m}} y^{D(\vec{m})} x^{S(\vec{m})} \prod_{i=1}^K \binom{S}{m_i}. \end{aligned} \quad (\text{A.9})$$

The coefficient of $y^d x^m$ in the power series (A.9) is therefore

$$\begin{aligned} \sum_{\substack{D(\vec{m})=d \\ S(\vec{m})=m}} \prod_{i=1}^K \binom{S}{m_i} &\stackrel{(\text{A.6})}{=} \sum_{\substack{D(\vec{m})=d \\ S(\vec{m})=m}} \Pr(\vec{X} = \vec{m} \mid f = m) \cdot \binom{N}{m} \\ &\stackrel{(\text{A.5})}{=} \Pr(D = d \mid f = m) \cdot \binom{N}{m} \end{aligned}$$

when d, m are admissible. Otherwise, the summation is empty, except for $d = m = 0$ where we obtain the constant term 1. Comparison with (A.8) shows that

$$f(x, y) = (g(x, y))^K.$$

We can now compute $p_{d,m}$ by direct expansion of $f(x, y)$:

$$\begin{aligned} f(x, y) &= [1 + y((1+x)^S - 1)]^K \\ &= 1 + \sum_{d=1}^K \binom{K}{d} y^d [(1+x)^S - 1]^d \\ &= 1 + \sum_{d=1}^K \binom{K}{d} y^d \sum_{j=0}^d \binom{d}{j} (1+x)^{S \cdot j} (-1)^{d-j} \\ &= 1 + \sum_{d=1}^K \binom{K}{d} y^d \sum_{j=0}^d \binom{d}{j} (-1)^{d-j} \sum_{m=0}^{S \cdot j} \binom{S \cdot j}{m} x^m \\ &= 1 + \sum_{d=1}^K \binom{K}{d} y^d \sum_{\substack{0 \leq j \leq d \\ 0 \leq m \leq S \cdot d \\ m \leq S \cdot j}} x^m (-1)^{d-j} \binom{d}{j} \binom{S \cdot j}{m} \\ &= 1 + \sum_{d=1}^K \binom{K}{d} y^d \sum_{m=0}^{S \cdot d} x^m \sum_{j=\lceil m/S \rceil}^d (-1)^{d-j} \binom{d}{j} \binom{S \cdot j}{m} \end{aligned}$$

For d, m admissible with $m \leq S$, we have $\lceil m/S \rceil = 0$ and the coefficient of $y^d x^m$ is

$$p_{d,m} = \binom{K}{d} \sum_{j=1}^d (-1)^{d-j} \binom{d}{j} \binom{S \cdot j}{m},$$

which concludes the proof together with (A.8). \square

Lemma A.4. *Under the conditions of Lemma A.3, the sum in Eq. (A.4) can be computed recursively. In particular,*

$$\Pr(D = d \mid f = m) = \binom{N}{m}^{-1} \binom{K}{d} A(d, m) \quad (\text{A.10})$$

where $A(d, m)$ is defined recursively by

$$A(1, m) := \binom{S}{m} \quad (\text{A.11a})$$

$$A(d, m) := \binom{S \cdot d}{m} - \sum_{j=1}^{d-1} \binom{d}{j} A(j, m) \quad (\text{A.11b})$$

Proof. The equality

$$A(k, m) = \sum_{j=1}^k (-1)^{k-j} \binom{k}{j} \binom{S \cdot j}{m} \quad (\text{A.12})$$

can be shown by straightforward induction over d . For $d = 1$,

$$A(1, m) = \binom{S}{m} = (-1)^{1-1} \binom{1}{1} \binom{S \cdot 1}{m}.$$

Let us now assume that (A.12) has been established for $k = 1, \dots, d-1$. Inserting these terms into (A.11b), we obtain

$$\begin{aligned} A(d, m) &= \binom{S \cdot d}{m} - \sum_{j=1}^{d-1} \binom{d}{j} A(j, m) \\ &= \binom{S \cdot d}{m} - \sum_{j=1}^{d-1} \binom{d}{j} \sum_{r=1}^j (-1)^{j-r} \binom{j}{r} \binom{S \cdot r}{m} \\ &= \binom{S \cdot d}{m} - \sum_{r=1}^{d-1} \binom{S \cdot r}{m} \sum_{j=r}^{d-1} (-1)^{j-r} \binom{d}{j} \binom{j}{r} \\ &= \binom{S \cdot d}{m} - \sum_{r=1}^{d-1} \binom{S \cdot r}{m} (-1)^{d-r} \underbrace{\sum_{j=r}^{d-1} (-1)^{d-j} \binom{d}{j} \binom{j}{r}}_{\stackrel{(*)}{=} -\binom{d}{r}} \\ &= \binom{S \cdot d}{m} + \sum_{r=1}^{d-1} \binom{S \cdot r}{m} (-1)^{d-r} \binom{d}{r} \\ &= \sum_{r=1}^d (-1)^{d-r} \binom{d}{r} \binom{S \cdot r}{m}, \end{aligned}$$

proving (A.12) for $k = d$. Note that $(-1)^{d-r} \cdot (-1)^{j-r} = (-1)^{d-r} \cdot (-1)^{-(j-r)} = (-1)^{d-j}$. The equality (*) can be derived in the following way:

$$\begin{aligned} \sum_{j=r}^d (-1)^{d-j} \binom{d}{j} \binom{j}{r} &= \sum_{j=r}^d (-1)^{d-j} \binom{d}{r} \binom{d-r}{j-r} \\ &\stackrel{k:=j-r}{=} \binom{d}{r} \sum_{k=0}^{d-r} (-1)^{(d-r)-k} \binom{d-r}{k} \\ &= \binom{d}{r} (1 + (-1))^{d-r} = 0. \end{aligned}$$

□

A.2 Proofs from Chapter 3

Lemma A.5. *The association measures chi-squared_i, chi-squared, and chi-squared_h compute the same association scores:*

$$X^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = \frac{N(O_{11} - E_{11})^2}{E_{11}E_{22}} = \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{R_1R_2C_1C_2}.$$

Proof. The proof is based on the fact that the row and column sums of the expected frequencies equal those of the observed frequencies.

$$O_{i1} + O_{i2} = R_i = E_{i1} + E_{i2} \quad (\text{A.13a})$$

$$O_{1j} + O_{2j} = C_j = E_{1j} + E_{2j} \quad (\text{A.13b})$$

$$O_{11} + O_{12} + O_{21} + O_{22} = N = E_{11} + E_{12} + E_{21} + E_{22} \quad (\text{A.13c})$$

For $i = 1$, (A.13a) implies $O_{11} - E_{11} = E_{12} - O_{12}$ and hence $(O_{11} - E_{11})^2 = (O_{12} - E_{12})^2$. Together with (A.13b) for $j = 1$ and $j = 2$ we obtain

$$(O_{11} - E_{11})^2 = (O_{12} - E_{12})^2 = (O_{21} - E_{21})^2 = (O_{22} - E_{22})^2. \quad (\text{A.14})$$

Inserting (A.14) into the equation of chi-squared_i, we obtain

$$X^2 := \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = (O_{11} - E_{11})^2 \sum_{ij} \frac{1}{E_{ij}}.$$

Using the identity

$$E_{11}E_{22} = \frac{R_1R_2C_1C_2}{N^2} = E_{12}E_{21}, \quad (\text{A.15})$$

we find that

$$\sum_{ij} \frac{1}{E_{ij}} = \frac{E_{11} + E_{22}}{E_{11}E_{22}} + \frac{E_{12} + E_{21}}{E_{12}E_{21}} \stackrel{(\text{A.15})}{=} \frac{\sum_{ij} E_{ij}}{E_{11}E_{22}} \stackrel{(\text{A.13c})}{=} \frac{N}{E_{11}E_{22}}$$

and hence

$$X^2 = \frac{N(O_{11} - E_{11})^2}{E_{11}E_{22}}$$

Finally,

$$\begin{aligned} O_{11} - E_{11} &= \frac{NO_{11} - R_1C_1}{N} \\ &= \frac{(O_{11} + O_{12} + O_{21} + O_{22})O_{11} - (O_{11} + O_{12})(O_{11} + O_{21})}{N} \\ &= \frac{O_{11}O_{22} - O_{12}O_{21}}{N} \end{aligned}$$

implies together with (A.15) that

$$X^2 = \frac{N(O_{11}O_{22} - O_{12}O_{21})^2}{R_1R_2C_1C_2}.$$

□

Lemma A.6. *The identities of Lemma A.5 also hold when Yates' continuity correction is applied, i.e. when the observed frequencies O_{ij} are replaced by adjusted frequencies O'_{ij} . The continuity-corrected chi-squared statistic can be written as*

$$(X')^2 = \frac{N(|O_{11}O_{22} - O_{12}O_{21}| - N/2)^2}{R_1R_2C_1C_2}.$$

Proof. We start by noting that the continuity corrections applied to the four observed frequencies O_{ij} are not independent. If $O_{11} > E_{11}$, the equalities (A.13) imply that $O_{12} < E_{12}$, $O_{21} < E_{21}$, and $O_{22} > E_{22}$ (and vice versa for $O_{11} < E_{11}$). Therefore, the continuity correction takes the general form

$$\begin{aligned} O'_{11} &:= O_{11} + \delta & O'_{12} &:= O_{12} - \delta \\ O'_{21} &:= O_{21} - \delta & O'_{22} &:= O_{22} + \delta \end{aligned} \tag{A.16}$$

with $\delta = -1/2$ for $O_{11} > E_{11}$ and $\delta = 1/2$ for $O_{11} < E_{11}$. Inserting (A.16) into (A.13), we see that these equalities also hold for O'_{ij} instead of O_{ij} . Since the proof of Lemma A.5 is exclusively based on (A.13), it remains valid for the continuity-corrected versions of the chi-squared measures.

We can therefore compute the continuity-corrected statistic $(X')^2$ by inserting the adjusted frequencies O'_{ij} into any one of the three equivalent formulae. Starting from chi-squared_h , we find

$$\begin{aligned} O'_{11}O'_{22} - O'_{12}O'_{21} &= (O_{11} + \delta)(O_{22} + \delta) - (O_{12} - \delta)(O_{21} - \delta) \\ &= O_{11}O_{22} - O_{12}O_{21} + \delta(O_{11} + O_{12} + O_{21} + O_{22}) \\ &= (O_{11}O_{22} - O_{12}O_{21}) + \delta N \end{aligned}$$

Since (A.15) implies $O_{11}O_{22} - O_{12}O_{21} > 0 \iff O_{11} > E_{11}$, the sign of δ is opposite to that of $O_{11}O_{22} - O_{12}O_{21}$. Consequently,

$$(O'_{11}O'_{22} - O'_{12}O'_{21})^2 = (|O_{11}O_{22} - O_{12}O_{21}| - N/2)^2,$$

which completes the proof. □

Lemma A.7. For the likelihood ratio defined by

$$\lambda := \frac{\max \Pr(\vec{X} = \vec{O} \mid N, H_0)}{\max \Pr(\vec{X} = \vec{O} \mid N)}$$

with respect to the null hypothesis of independence H_0 , the following equality holds:

$$-2 \log \lambda = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

Proof. The denominator of λ is maximised by the MLE $\tau_{ij} = O_{ij}/N$ according to Eq. (2.8). Inserting these values into the multinomial distribution (2.5) for $\vec{k} = \vec{O}$, we obtain

$$\max \Pr(\vec{X} = \vec{O} \mid N) = \frac{N!}{N^N} \cdot \prod_{ij} \frac{(O_{ij})^{O_{ij}}}{O_{ij}!} \quad (\text{A.17})$$

The conditional probability under H_0 in the numerator is given by Eq. (2.12), which corresponds to the product of two independent binomial probability values: for R_1 successes out of N trials with success probability π_1 , and for C_1 successes out of N trials with success probability π_2 (except for a factor that does not depend on π_1 and π_2 and can therefore be ignored). This probability is maximised for the binomial MLE $\pi_1 = R_1/N$ and $\pi_2 = C_1/N$. Under H_0 , the parameters τ_{ij} are fully determined by π_1 and π_2 , according to Eq. (2.11):

$$\tau_{ij} = \frac{R_i C_j}{N^2} = \frac{E_{ij}}{N}$$

Inserting these values into the multinomial distribution (2.5) for $\vec{k} = \vec{O}$, the numerator of λ becomes

$$\max \Pr(\vec{X} = \vec{O} \mid N, H_0) = \frac{N!}{N^N} \cdot \prod_{ij} \frac{(E_{ij})^{O_{ij}}}{O_{ij}!} \quad (\text{A.18})$$

When we insert (A.17) and (A.18) into λ , the factorials and the factor N^N cancel out leaving

$$\lambda = \prod_{ij} \left(\frac{E_{ij}}{O_{ij}} \right)^{O_{ij}}$$

and hence

$$\begin{aligned} -2 \log \lambda &= -2 \sum_{ij} O_{ij} \cdot (\log E_{ij} - \log O_{ij}) \\ &= 2 \sum_{ij} O_{ij} \cdot (\log O_{ij} - \log E_{ij}) \\ &= 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}} \end{aligned}$$

□

A.3 Proofs from Chapter 4

Lemma A.8. Let $X \sim N(\mu_1, \sigma_1^2)$ and $Y \sim N(\mu_2, \sigma_2^2)$ be two independent, normally distributed random variables with $\mu_1, \mu_2 > 0$, $\sigma_1 \ll \mu_1$, and $\sigma_2 \ll \mu_2$. Then the ratio

$$R := \frac{X}{X + Y}$$

approximately follows a normal distribution $R \sim N(\mu_*, \sigma_*^2)$ whose mean μ_* and standard deviation σ_* are given by

$$\mu_* := \frac{\mu_1}{\mu_1 + \mu_2}, \quad \sigma_* := \frac{\sigma}{\mu} \sqrt{s(1-s) + (r-s)^2};$$

$\mu := \mu_1 + \mu_2$ and $\sigma^2 := \sigma_1^2 + \sigma_2^2$ are the mean and variance of $X + Y \sim N(\mu, \sigma^2)$, while r and s stand for the ratios $r := \mu_1 / \mu$ and $s := \sigma_1^2 / \sigma^2$.

The exact median of R is always μ_* , i.e. $\Pr(R \leq \mu_*) = 1/2$. If the condition $r = s$ holds, its exact mean is $E[R] = \mu_*$ (this is the case, in particular, when X and Y approximate Poisson distributions so that $\mu_1 = \sigma_1^2$ and $\mu_2 = \sigma_2^2$).

Proof. The following proof is based on the distribution function of R . Since $\sigma_1 \ll \mu_1$ and $\sigma_2 \ll \mu_2$, the probability that X or Y assumes a negative value is negligible, $\Pr(X \leq 0) \approx 0$ and $\Pr(Y \leq 0) \approx 0$. Therefore, the distribution of R is essentially determined by the probabilities $\Pr(R \leq a)$ for $a \in [0, 1]$. For a given value a , we have the equality

$$\begin{aligned} R \leq a &\iff \frac{X}{X + Y} \leq a \iff X \leq aX + aY \\ &\iff (1-a)X - aY \leq 0 \iff Z_a \leq 0 \end{aligned} \quad (\text{A.19})$$

where $Z_a := (1-a)X - aY$. Since X and Y are independent, $Z_a \sim N(\mu_a, \sigma_a^2)$ with $\mu_a := (1-a)\mu_1 - a\mu_2$ and $\sigma_a^2 := (1-a)^2\sigma_1^2 + a^2\sigma_2^2$. The corresponding standardised variable is $Z_a^* := (Z_a - \mu_a) / \sigma_a \sim N(0, 1)$. Using the common symbol Φ for the distribution function of the standard normal distribution,

$$\Phi(a) := \int_{-\infty}^a \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx,$$

Eq. (A.19) now implies that

$$\begin{aligned} \Pr(R \leq a) &= \Pr(Z_a \leq 0) = \Pr(\sigma_a Z_a^* + \mu_a \leq 0) \\ &= \Pr(Z_a^* \leq -\mu_a / \sigma_a) = \Phi(-\mu_a / \sigma_a) \\ &= \Phi\left(\frac{a\mu_2 - (1-a)\mu_1}{\sqrt{(1-a)^2\sigma_1^2 + a^2\sigma_2^2}}\right). \end{aligned}$$

With $\mu_1 = r\mu$, $\sigma_1^2 = s\sigma^2$ and $\sigma_2^2 = (1-s)\sigma^2$ the numerator above becomes $a(\mu_1 + \mu_2) - r\mu = \mu(a - r)$ and the square of the denominator can be rewritten as $(1-a)^2s\sigma^2 + a^2(1-s)\sigma^2$. Inserting these equalities into the expression for $\Pr(R \leq a)$ yields

$$\Pr(R \leq a) = \Phi\left(\frac{\mu}{\sigma}(a - r) \cdot [(1-a)^2s + a^2(1-s)]^{-1/2}\right) \quad (\text{A.20})$$

and thus $\Pr(R \leq \mu_*) = \Pr(R \leq r) = \Phi(0) = 1/2$, showing that μ_* is indeed the median of R . Obviously, (A.20) describes a normal distribution except for the “distortion” factor in square brackets. We will now try to estimate the amount of distortion by comparing the quantiles of R with those of a normal distribution. Let a_c be the quantile of R corresponding to C standard deviations of a normally distributed random variable, i.e. to a z -score of C . This quantile is defined by the condition $\Pr(R \leq a_c) = \Phi(C)$. If R were normally distributed, $R \sim N(\mu_*, \sigma_*^2)$, a_c would simply be given by

$$a_c = \mu_* + \sigma_* C. \quad (\text{A.21})$$

We will now compute the true quantile a_c and compare the resulting expression with (A.21). Eq. (A.20) implies that

$$\frac{\mu}{\sigma}(a_c - r) \cdot [(1 - a_c)^2 s + a_c^2(1 - s)]^{-1/2} = C. \quad (\text{A.22})$$

Taking the square of (A.22), we obtain:

$$\begin{aligned} \Pr(R \leq a_c) &= \Phi(\pm C) \\ \iff \frac{\mu^2}{\sigma^2}(a_c - r)^2 &= C^2((1 - a_c)^2 s + a_c^2(1 - s)) \\ \iff \mu^2 a_c^2 - 2\mu^2 r a_c + \mu^2 r^2 &= \sigma^2 C^2(a_c^2 - 2s a_c + s) \\ \iff (\mu^2 - \sigma^2 C^2)a_c^2 + 2(s\sigma^2 C^2 - \mu^2 r)a_c &+ (\mu^2 r^2 - s\sigma^2 C^2) = 0 \\ \iff (1 - \kappa_c^2)a_c^2 + 2(\kappa_c^2 s - r)a_c + (r^2 - \kappa_c^2 s) &= 0. \end{aligned}$$

with the abbreviation $\kappa_c := C \cdot \sigma / \mu$, so that $\kappa_c^2 = \sigma^2 C^2 / \mu^2$. Solving this quadratic equation for a_c leads to

$$\begin{aligned} a_c &= \frac{r - s\kappa_c^2 \pm \sqrt{(\kappa_c^2 s - r)^2 - (1 - \kappa_c^2)(r^2 - \kappa_c^2 s)}}{1 - \kappa_c^2} \\ &= r + \frac{\kappa_c^2(r - s)}{1 - \kappa_c^2} \pm \sqrt{\kappa_c^2} \frac{\sqrt{(1 - \kappa_c^2)s(1 - s) + (r - s)^2}}{1 - \kappa_c^2} \\ &= r + \kappa_c \frac{\sqrt{(1 - \kappa_c^2)s(1 - s) + (r - s)^2}}{1 - \kappa_c^2} + \kappa_c^2 \frac{r - s}{1 - \kappa_c^2}, \end{aligned} \quad (\text{A.23})$$

making use of the fact that the \pm -term has the same sign as C and κ_c . Assuming that $|\kappa_c| = |C \cdot \sigma / \mu|$ is small,² we have $1 - \kappa_c^2 \approx 1$ and $(1 - \kappa_c^2)^{-1} = 1 + \mathcal{O}(|\kappa_c|^2) \approx 1$. This implies that

$$\begin{aligned} \frac{\sqrt{(1 - \kappa_c^2)s(1 - s) + (r - s)^2}}{1 - \kappa_c^2} &= (1 - \kappa_c^2)^{-1} \cdot (s(1 - s) + (r - s)^2 - \kappa_c^2 s(1 - s))^{-1/2} \\ &= \left(1 + \mathcal{O}(|\kappa_c|^2)\right) \cdot \left(\sqrt{s(1 - s) + (r - s)^2} + \mathcal{O}(|\kappa_c|^2)\right) \\ &= \sqrt{s(1 - s) + (r - s)^2} + \mathcal{O}(|\kappa_c|^2) \end{aligned}$$

²Since $\sigma \ll \mu$ and the range $C \in [-5, 5]$ covers practically the entire probability mass of the distribution with $\Phi(-5) < 5 \cdot 10^{-7}$, this assumption is valid.

and hence

$$a_c = r + \kappa_c \sqrt{s(1-s) + (r-s)^2} + \kappa_c^2(r-s) + \mathcal{O}(|\kappa_c|^3). \quad (\text{A.24})$$

Inserting the definition of κ_c into (A.24), we finally obtain

$$a_c = r + \underbrace{\frac{\sigma}{\mu} \sqrt{s(1-s) + (r-s)^2} \cdot C}_{\text{normal approximation}} + \underbrace{\frac{\sigma^2}{\mu^2} (r-s) \cdot C^2}_{\text{asymmetry}} + \mathcal{O}(|C \cdot \sigma / \mu|^3). \quad (\text{A.25})$$

When we ignore the asymmetry term, which is $\mathcal{O}(|\kappa_c|^2)$ and the remaining terms of higher order, a comparison of (A.25) with (A.21) shows that the quantiles of R correspond to those of a normal distribution with mean

$$\mu_* := r$$

and standard deviation

$$\sigma_* := \frac{\sigma}{\mu} \sqrt{s(1-s) + (r-s)^2}.$$

In the special case $r = s$, (A.23) becomes

$$a_c = r + \kappa_c \frac{\sqrt{s(1-s)}}{\sqrt{(1-\kappa_c^2)}},$$

which is a symmetric function of C in the sense that $(a_{-c} - r) = -(a_c - r)$.³ Hence the distribution of R is symmetric around r and we have $E[R] = r = \mu_*$ by the standard symmetry argument. In addition, Eq. (A.25) becomes

$$a_c = r + \frac{\sigma}{\mu} \sqrt{s(1-s)} \cdot C + \mathcal{O}(|C \cdot \sigma / \mu|^3),$$

so that the normal approximation to R is highly accurate. □

A.4 Some mathematical background

The **Gamma function** $\Gamma(a)$ is a generalisation of the factorial. It can be defined by the Gamma integral

$$\Gamma(a) := \int_0^{\infty} t^{a-1} e^{-t} dt \quad (\text{A.26})$$

for $a > 0$. Its most important properties are the recurrence relation

$$\Gamma(a+1) = a \cdot \Gamma(a) \quad (\text{A.27})$$

and its relation to the factorial

$$n! = \Gamma(n+1) \quad (\text{A.28})$$

³Note that κ_c is a symmetric function of C , i.e. $\kappa_{-c} = -\kappa_c$.

for $n \in \mathbb{N}_0$ (Weisstein 1999, s.v. *Gamma Function*).

The **upper incomplete Gamma function** $\Gamma(a, x)$ is given by the partial Gamma integral

$$\Gamma(a, x) := \int_x^\infty t^{a-1} e^{-t} dt \quad (\text{A.29})$$

for $a > 0$ and $x \geq 0$. The complementary integral leads to the **lower incomplete Gamma function**

$$\gamma(a, x) := \int_0^x t^{a-1} e^{-t} dt \quad (\text{A.30})$$

for $a > 0$ and $x \geq 0$. Apart from the obvious identities

$$\Gamma(a) = \Gamma(a, 0) = \lim_{x \rightarrow \infty} \gamma(a, x) = \gamma(a, x) + \Gamma(a, x), \quad (\text{A.31})$$

the incomplete Gamma functions can be used to represent the distribution function of a Poisson distribution. For a random variable $X \sim P(\lambda)$ (i.e. X follows a Poisson distribution with parameter λ), we have

$$\Pr(X \leq n) = e^{-\lambda} \sum_{k=0}^n \frac{\lambda^k}{k!} = \frac{\Gamma(n+1, \lambda)}{\Gamma(n+1)} \quad (\text{A.32})$$

and

$$\Pr(X \geq n) = e^{-\lambda} \sum_{k=n}^{\infty} \frac{\lambda^k}{k!} = \frac{\gamma(n, \lambda)}{\Gamma(n)} \quad (\text{A.33})$$

(Weisstein 1999, s.v. *Incomplete Gamma Function*). The ratios on the right-hand side are also known as the **regularised Gamma functions** $P(a, x) := \gamma(a, x)/\Gamma(a)$ and $Q(a, x) := \Gamma(a, x)/\Gamma(a)$. For fixed a , the function $x \mapsto P(a, x)$ is the distribution function of a Gamma distribution (Weisstein 1999, s.v. *Gamma Distribution*). The incomplete and regularised Gamma functions can be computed efficiently using a power series expansion similar to (A.32) and (A.33). They are provided by many statistical software libraries through the Gamma distribution, e.g. in R:

$$\begin{aligned} \Gamma(a, x) &= \text{gamma}(a) * \text{pgamma}(x, \text{shape}=a, \text{scale}=1, \text{lower}=F) \\ \gamma(a, x) &= \text{gamma}(a) * \text{pgamma}(x, \text{shape}=a, \text{scale}=1) \end{aligned}$$

The **Beta function** $B(a, b)$ is a generalisation of the binomial coefficient. It can be defined in terms of the Gamma function by

$$B(a, b) := \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \quad (\text{A.34})$$

or by the Beta integral

$$B(a, b) = \int_0^1 t^{a-1} (1-t)^{b-1} dt \quad (\text{A.35})$$

for $a, b > 0$. Its relation to the binomial coefficient is given by

$$\binom{n}{k} = ((n+1) \cdot B(n-k+1, k+1))^{-1} \quad (\text{A.36})$$

for $n, k \in \mathbb{N}_0$ with $n \geq k$ (Weisstein 1999, s.v. *Beta Function*). The **incomplete Beta function** $B(x; a, b)$ is defined by the partial Beta integral

$$B(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1} dt \quad (\text{A.37})$$

for $a, b > 0$ and $x \in [0, 1]$, and satisfies $B(1; a, b) = B(a, b)$ (Weisstein 1999, s.v. *Incomplete Beta Function*). Again, the **regularised Beta function** is defined as $I(x; a, b) := B(x; a, b) / B(a, b)$. For fixed a and b , the function $x \mapsto I(x; a, b)$ is the distribution function of a Beta distribution (Weisstein 1999, s.v. *Beta Distribution*). In this way, it can easily be computed in R (and other statistical software libraries):

$$\begin{aligned} I(x; a, b) &= \text{pbeta}(x, \text{shape1}=a, \text{shape2}=b) \\ B(x; a, b) &= \text{pbeta}(a, b) * \text{pbeta}(x, \text{shape1}=a, \text{shape2}=b) \end{aligned}$$

An important property of the regularised Beta function is its relation to the binomial distribution. For a random variable $X \sim B(n, p)$ (i.e. X follows a binomial distribution with n trials and success probability p), we have

$$\Pr(X \geq k) = \sum_{j=k}^n \binom{n}{j} p^j (1-p)^{n-j} = I(p; k, n-k+1) \quad (\text{A.38})$$

for $k \in \{0, \dots, n\}$ (Weisstein 1999, s.v. *Binomial Distribution*). **Binomial confidence intervals** can thus be obtained from the inverse of $I(x; a, b)$ with respect to x , which is denoted I^{-1} . For instance, the upper one-sided confidence interval for the unknown success probability p at significance level α , given an observed value $X = k$, is the set

$$\begin{aligned} C &= \{p \in [0, 1] \mid \Pr(X \geq k) \geq \alpha\} \\ &= \{p \in [0, 1] \mid I(p; k, n-k+1) \geq \alpha\} \\ &= [p^*, 1] \end{aligned}$$

with $p^* = I^{-1}(\alpha; k, n-k+1)$. Lower and two-sided confidence intervals are obtained in a similar way.

The `sfunc` module of the UCS/R library provides implementations of the complete, incomplete and regularised Gamma and Beta functions, together with their logarithms, the inverse functions, and binomial confidence intervals (see Section B.2). All functions are also available to UCS/Perl scripts through the `UCS::SFunc` module (see Section B.1).

Appendix B

UCS Software Documentation

B.1 UCS/Perl

This section contains the full UCS/Perl documentation, automatically converted from POD to \LaTeX format. The formatting has been improved with some automatic transformations (based on Perl scripts). The version documented here is UCS v0.5, the official version that accompanies the thesis.

UCS/Perl documentation contents

General Documentation	186
ucsintro	186
ucsfile	188
ucsexp	193
ucsam	198
UCS/Perl Programs	202
ucsdoc	202
ucs-config	202
ucs-tool	204
ucs-list-am	205
ucs-make-tables	207
ucs-summarize	210
ucs-select	211
ucs-add	212
ucs-join	214
ucs-sort	216
ucs-info	217
ucs-print	218

UCS/Perl Modules	220
UCS	220
UCS::File	224
UCS::R	228
UCS::R::Expect	230
UCS::R::RSPerl	230
UCS::SFunc	231
UCS::Expression	237
UCS::Expression::Func	240
UCS::AM	241
UCS::AM::HTest	245
UCS::AM::Parametric	248
UCS::DS	250
UCS::DS::Stream	252
UCS::DS::Memory	256
UCS::DS::Format	265

B.1.1 General Documentation

■ ucsintro

A first introduction to UCS/Perl

INTRODUCTION

UCS is a set of libraries and tools intended for the empirical study of cooccurrence statistics. Its major uses are to *apply* such statistics, called **association measures**, to cooccurrence data obtained from a corpus, and to *evaluate* the resulting association scores and rankings against (manually annotated) reference data.

The frequency data extracted from a given corpus for a given type of cooccurrences consists of a list of **pair types** with their **frequency signatures** (i.e. joint and marginal frequencies), and is referred to as a **data set**. See (Evert 2004) for a detailed explanation of these concepts, different types of cooccurrences, and correct methods for obtaining frequency data. Data sets, stored in a special **.ds** file format, are the fundamental objects of the UCS toolkit. Most UCS programs manipulate or display such data set files.

The UCS implementation relies heavily on the programming language **Perl** (<http://www.perl.com/>) and the free statistical environment **R** (<http://www.r-project.org/>) as a library of mathematical and statistical functions. The core of UCS is written in Perl (the **UCS/Perl** part), but there is also a small library of R functions for interactive work within R (the **UCS/R** part). UCS/Perl uses R as a back-end, making the most important statistical functions available through a Perl module.

UCS/Perl is mainly a collection of Perl modules that perform the following tasks:

- read and write data set files (.ds, .ds.gz)
- manage in-memory representations of data sets
- compile UCS expressions for easy access to data set variables
- filter, annotate, sort, and analyse data sets
- provide a repository of built-in association measures
- display data sets and evaluation graphs (Perl/Tk and R) **[not implemented yet]**

Most UCS programs will be custom-built scripts, using the library of support functions provided by the UCS/Perl modules. Loading a data set, annotating it with association scores from one or more measures, and sorting it in various ways can be done with a few lines of Perl code. There are also some ready-made programs in UCS/Perl that perform such standard tasks, operating on data set files. A substantial part of the UCS/Perl functionality is thus accessible from the command-line, at the cost of some additional overhead compared to a custom script (which operates on in-memory representations).

Below, you will find a list of the general documentation files, Perl modules, and programs that are included in the UCS/Perl distribution. Manpages for all modules and programs (as well as the general documentation) are easily accessible with the **ucsdoc** program, and can also be formatted for printing.

General Documents

```
ucsdoc ucsintro      # this introduction
ucsdoc ucsfile      # description of the UCS data set file format (.ds)
ucsdoc ucsexp       # UCS expressions and wildcards
ucsdoc ucsam        # overview of built-in association measures
```

UCS/Perl MODULES

```
use UCS;              # core library
use UCS::File;        # file access utilities
use UCS::R;           # interface to UCS/R
use UCS::SFunc;       # special functions and statistical distributions

use UCS::Expression;  # Perl code interspersed with UCS variables
use UCS::Expression::Func; # utility functions available in UCS expressions

use UCS::AM;          # implementations of various association measures
use UCS::AM::HTest;   # add-on package: variants of hypothesis tests
use UCS::AM::Parametric; # add-on package: parametric association measures

use UCS::DS;          # data sets ...
use UCS::DS::Stream;  # i/o streams for data set files
use UCS::DS::Memory;  # in-memory representation of data sets
use UCS::DS::Format;  # ASCII formatter (+ other formats)
```

See the respective manpages (`ucsdoc ModuleName`) for more information.

UCS/Perl PROGRAMS

```

ucsdoc          # front-end to perldoc
ucs-config      # automatic configuration of UCS/Perl scripts
ucs-tool        # find and run user-contributed UCS/Perl scripts
ucs-list-am     # list built-in association measures & add-on packages

ucs-make-tables # compute frequency signatures from list of pair tokens
ucs-summarize   # print (statistical) summaries for selected variables

ucs-select      # select rows and/or columns from a data set file
ucs-add         # add variables to a data set file
ucs-join        # combine rows and/or columns from two data sets
ucs-sort        # sort data set file by specified attribute(s)

ucs-info        # display information from header of data set file
ucs-print       # format data set as ASCII table (for viewing and printing)

```

See the respective manpages (ucsdoc ProgramName) for more information.

TRIVIA

UCS stands for **Utilities for Cooccurrence Statistics**.

REFERENCES

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, University of Stuttgart, Germany.

On-line repository of association measures: <http://www.collocations.de/>

COPYRIGHT

Copyright (C) 2004 by Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucsfile

The UCS data set file format

INTRODUCTION

UCS data sets are stored in a simple tabular format, similar to that of a statistical table. Each row in the table corresponds to a **pair type**, and its individual fields (columns) provide various kinds of information about the pair type:

- a unique **ID number** (unique within the data set)
- the component **lexemes**
- the pair type's **frequency signature**
- [optional] contingency tables of **observed** and **expected frequencies** computed from the frequency signature
- [optional] **coordinates** computed from the frequency signature
- **association scores** and **rankings** for various association measures
- arbitrary **user-defined attributes**, especially for the manual annotation of *true positives* in an evaluation study

Following statistical terminology, the table columns are referred to as the **variables** of a data set (each of which assumes a specific value for each pair type). Columns are separated by a TAB character ("`\t`"), and the first row lists the **variable names** as table headings (see the section on VARIABLES below for naming conventions).

The actual data table may be preceded by an optional **header** of Perl-style comment lines (beginning with a `#` character). Lines with the special format

```
##:: <variable> = <value>
```

define **global variables**, which may be interpreted by some of the UCS/Perl programs (see the section on GLOBAL VARIABLES below). The variable name (*variable*) may only contain alphanumeric characters (A-Z a-z 0-9) and the period (`.`). The *value* may contain arbitrary characters, including whitespace (but leading and trailing whitespace will be ignored). Variable definitions must not span multiple lines.

UCS data set files must have the filename extension **.ds**. They may be compressed with **gzip** (and they usually are), in which case they carry the extension **.ds.gz**. UCS library functions will automatically recognise and uncompress data set files with this extension.

A special subtype of data sets are the **annotation database** files with extension **.adb** (uncompressed) or **.adb.gz** (compressed). Annotation databases omit all frequency information and association scores, listing only component lexemes and user-defined attributes. They are used as repositories of lexical information (such as manually annotated *true positives* for evaluation purposes) that applies to data sets extracted from different corpora (or with different methods).

GLOBAL VARIABLES

```
size          number of pair types in a data set
```

The only global variable that is currently supported is **size**, an integer specifying the number of pair types in a data set. Availability of the data set size in the header may give a slight performance improvement when loading data set files into memory. If **size** is set to an incorrect value, the behaviour of UCS/Perl programs and modules is undefined.

A global variable whose name is identical to that of a variable defined in the data set (i.e. a table column) is interpreted as an **explanatory note**. Such notes should typically be given for all user-defined variables, and also for user-defined association measures.

Unsupported variables will simply be ignored and will not raise errors or warnings when a data set file is parsed.

DATA TYPES

The UCS system supports four different data types:

BOOL	a logical (Boolean) value
INT	a signed integer value (≥ 32 bits)
DOUBLE	a floating-point value (IEEE double precision)
STRING	an arbitrary string (ISO-8859-1 or UTF-8)

Boolean values are represented by 1 (true) and 0 (false). **String** values may contain blanks (but no TAB characters) and are neither quoted nor escaped. Full support for Unicode strings (UTF-8) is only available within the UCS/Perl subsystem.

The UCS/R subsystem will interpret Boolean values as logical variables, and strings (except for the component lexemes) as *factor* variables with a fixed set of levels (which are automatically determined from the data).

User-defined attributes may assume the special value NA for **missing values**. (Note that the string NA will always be interpreted as a missing value rather than a literal character string!) UCS/R has built-in support for missing values, whereas UCS/Perl represents them by **undef** entries. Programs that do not support missing values may replace them by 0 (BOOL and INT), 0.0 (DOUBLE), or the empty string "" (STRING).

The **data type** of a variable is uniquely determined by the variable name, as detailed in the section on VARIABLES below.

VARIABLES

In order to be compatible with the **R** language, variable names may only contain alphanumeric characters (A-Z a-z 0-9) and periods (.), and they must begin with a letter. The main function of periods is to delimit words in complex variable names, replacing blanks, hyphens, and underscores. UCS variable names are case-sensitive.

Periods are not allowed in **Perl** variable names, but **UCS expressions** provide a special syntax for direct access to data set variables (see the `ucsexp` and `UCS::Expression` manpages). In the rare case where plain Perl variables are used to store information from a data set, periods should be replaced by underscores (underscore) in the variable names.

There are strict **naming conventions** for data set variables, which are detailed in the following subsections. Apart from a fixed list of core variables (whose names do not contain the . character), all variable names begin with a period-separated **prefix** that determines the data type of the variable.

Core Variables Core variables represent the minimal amount of information that must be present in a data set file (i.e. evidence for cooccurrences extracted from a corpus). All core variables are mandatory, except in the case of annotation database files (.adb), which omit frequency signatures (`f f1 f2 N`). For relational cooccurrences, frequency signatures can be computed with the **ucs-make-tables** utility from a stream of pair tokens (cf. the `ucs-make-tables` manpage).

```

INT    id    a numerical ID value (unique within the data set)
STRING l1   first component type of the pair
STRING l2   second component type of the pair

INT    f     cooccurrence frequency of pair type
INT    f1    marginal frequency of first component
INT    f2    marginal frequency of second component
INT    N     sample size (identical for all pair types)

```

id is a numerical ID value, which must be unique within a data set. Its intended uses are to identify pair types in subsets selected from a given data set, and to validate line numbers when attributes or association scores are computed by an external program and re-integrated into the data set file.

The **lexemes** l1 and l2 are the component (word) types that uniquely identify a pair type. Consequently, a data set file must not contain multiple rows with identical l1 and l2 values. UCS/Perl should provide reasonably good support for Unicode strings as lexemes (in UTF-8 encoding), at least when running on Perl version 5.8.0 or newer.

The quadruple f f1 f2 N is called the **frequency signature** of a pair type. It contains all the frequency information used by **association measures** and is equivalent to a contingency table. Note that the **sample size** N is identical for all pair types in a data set and is included here mainly for convenience' sake (so that association scores can be computed from the row data without reference to a global variable). See (Evert 2004) for more information on lexemes and frequency signatures.

Derived Variables Derived variables can be computed from the frequency signatures of pair types, providing different "views" of the frequency information. Normally, they are not annotated explicitly but are accessible through **UCS expressions**, which compute the required values automatically (see the ucsexp and UCS::Expression manpages).

```

INT    011   contingency table of observed frequencies
INT    012   (computed from frequency signature)
INT    021
INT    022

INT    R1    row sums in observed contingency table
INT    R2
INT    C1    column sums in observed contingency table
INT    C2

```

The variables 011 012 021 022 represent the observed **contingency table** of a pair type. Note that their frequency information is equivalent to the frequency signature of the pair type. In addition, the **row sums** (R1 R2) and **column sums** (C1 C2) of the contingency table are also made available.

```

DOUBLE E11   contingency table of expected frequencies
DOUBLE E12   under point null hypothesis
DOUBLE E21   (computed from row and column sums)
DOUBLE E22

```

The variables `E11` `E12` `E21` `E22` represent the contingency table of **expected frequencies**, i.e. the expectations of the multinomial sampling distribution under the point null hypothesis of independence. Most association measures compare observed frequencies to expected frequencies in some way.

In a **geometric interpretation** of a data set, each pair type can be interpreted as a point x in a three-dimensional **coordinate space** P . Since the sample size N is a constant parameter within the data set, the coordinates of x are given by the joint and marginal frequencies f f_1 f_2 .

```
DOUBLE lf    logarithmic coordinates
DOUBLE lf1   (base 10 logarithm)
DOUBLE lf2
```

Since the coordinates usually have a skewed distribution across several orders of magnitude, it is often more convenient to visualise them on a logarithmic scale. The variables `lf` `lf1` `lf2` give the **base ten logarithms** of the coordinate triple f f_1 f_2 .

```
DOUBLE e     ebo-coordinates
DOUBLE b     (expected, balance, observed)
DOUBLE o
```

```
DOUBLE le    logarithmic ebo-coordinates
DOUBLE lb    (base 10 logarithm)
DOUBLE lo
```

Theoretical and empirical studies of the properties of association measures will often be based on transformed coordinate systems in the coordinate space. The most useful system are the **ebo-coordinates** `e` `b` `o` (for *expected, balance, observed*). All three coordinates range from 0 to infinity (constrained by the sample size parameter N). The base 10 logarithms `le` `lb` `lo` of the ebo-coordinates are convenient for visualisation purposes. `le` and `lb` range from -infinity to +infinity, while `lo` ranges from 0 to infinity (all constrained by N).

For backward compatibility, a transformation of the coordinate system to **relative frequencies**, which were used in earlier versions of this software, is also supported. The relative cooccurrence (p) and marginal (p_1 p_2) frequencies are computed from the frequency signature according to the equations $p = f/N$, $p_1 = f_1/N$, and $p_2 = f_2/N$. Note that the logarithmic versions `lp` `lp1` `lp2` are *negative* base 10 logarithms, ranging from 0 to infinity.

Association Scores and Rankings These variables store association scores and rankings for an arbitrary number of **association measures**. Each association measure is identified by a *key*, which is appended to the respective variable name prefix (resulting in the names `am.key` and `r.key`). See the `UCS::AM` manpage (and the manpages of the add-on packages listed there) for a wide range of built-in association measures.

```
DOUBLE am.*  association scores from measure identified by *
INT    r.*   ranking for this measure (ties are allowed)
```

Rankings are often computed on the fly, but they may also be annotated in data set files. Note that the `r.*` variables should *not* break ties but report identical ranks (and skip an appropriate number of subsequent ranks). The `ucs-sort` program (cf. the `ucs-sort` manpage) can be used to resolve ties in various ways (using other association scores, lexical sort order, or randomisation).

User-Defined Variables User-defined variables may contain arbitrary information, which is typically used for filtering data sets and to determine true positives in evaluation tasks. However, some special-purpose association measures may also base their association scores on their values. In order to allow a minimal amount of automatic processing (such as sorting by user-defined attributes), the variable name prefix of a user-defined variable is used to determine its data type, according to the following list.

```

BOOL   b.*   user-defined Boolean variable
INT    n.*   user-defined integer variable (n=number)
DOUBLE x.*   user-defined floating-point variable
STRING f.*   user-defined string variable (f=factor)

```

User-defined variables with the additional prefix `ucs` (corresponding to variable names `b.ucs.*`, `n.ucs.*`, `x.ucs.*`, and `f.ucs.*`) are reserved for internal use by UCS modules and programs.

REFERENCES

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, University of Stuttgart, Germany.

COPYRIGHT

Copyright (C) 2004 by Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucsexp

Introduction to UCS expressions and wildcard patterns

INTRODUCTION

UCS expressions and wildcard patterns are two central features of the **UCS/Perl** system, which are to a large part responsible for its convenience and flexibility.

UCS **wildcard patterns** are used by most command-line tools to select data set variables with the help of shell-like wildcard characters (`?`, `*`, and `%`). A programmer interface is provided by the **UCS::Match** function from the **UCS** module (see the UCS manpage).

UCS expressions give easy access to data set variables from Perl code. With only a basic knowledge of Perl syntax, users can compute association scores and select rows from a data set (using the **ucs-add** and **ucs-select** utilities). The programmer interface is provided by the **UCS::Expression** module (see the **UCS::Expression** manpage for details). Before reading the section on UCS EXPRESSIONS, you should become familiar with the UCS data set format and variable naming conventions as described in the **ucsfile** manpage.

When used on the **command line**, wildcard patterns usually have to be quoted to keep the shell from expanding wildcards (the GNU Bash shell knows better, though, unless there happen to be matching files in the current directory). Note that when a list of variable names and patterns is passed to one of the UCS/Perl utilities, each name or wildcard pattern has to be quoted *individually*. UCS expressions (almost) always have to be quoted on the command-line. Single quotes (' . . . ') are highly recommended to avoid interpolation of variables and other meta-characters. The UCS/Perl utilities expect a UCS expression to be passed as a single argument, so the expression must be written as one string. In particular, any expression containing whitespace must be quoted.

UCS WILDCARD PATTERNS

As described in the `ucsfile` manpage, UCS **variable names** may only contain the alphanumeric characters (A-Z a-z 0-9) and the period (.), which serves as a general-purpose word delimiter. There is a fixed set of **core variables**, whose names do not contain a period. All other variable names must begin with a prefix (one of `am.` `r.` `b.` `n.` `x.` `f.`) that determines the data type of the variable. The three **wildcard** characters take the special role of the period into account. Their meanings are

```
? ... a single character, except "."
* ... a string that does NOT contain a "."
% ... an arbitrary string of characters
```

The `%` wildcard is typically used to select variable names with a specific prefix or suffix, while `*` matches the individual words (or parts of words) in a complex variable name.

Examples

- a pattern without wildcard characters corresponds to a literal variable name : `id`, `O11`, `am.log.likelihood`
- the pattern `*` matches all core variables (and nothing else); `%` matches *all* variable names
- `O*` matches the derived variables `O11`, `O12`, `O21`, and `O22`; `*11` matches `O11` and `E11`, but no complex variable names
- prefix patterns allow us to select variables by their type, e.g. `am.%` for all association scores, or `f.%` for all user-defined string variables (factors); the `*` wildcard is inappropriate here because the variable names may contain additional period after the prefix
- when variable names are chosen systematically, prefix patterns can also be used to select meaningful groups of variables: `am.chi.squared%` matches all association scores that are derived from a chi-squared test, and `am.%pv` matches all association scores that can be interpreted as probability values (see the `UCS::AM` and `UCS::AM::HTest` manpages for more information)

UCS EXPRESSIONS

An UCS expression consists of ordinary Perl code extended with a special syntax to access data set variables. This code is compiled on the fly and applied to the rows of a data set one at a time. The return value of a UCS expression is the value of the last statement executed, unless there is an explicit return statement. When the expression is used as a condition to select rows from a data set, it evaluates to true or false according to the usual Perl rules (the empty string "" and the number 0 are false, everything else is true).

Data set variables are accessed by their variable name enclosed in % characters. They evaluate to the respective value for the current row in the data set and can be used like ordinary scalar variables in Perl. Thus, %f% corresponds to the cooccurrence frequency *f* of a pair type, %l1% and %l2% to its component lexemes, and %am.log.likelihood% to an association score from the log-likelihood measure. Derived variables (see the `ucsfile` manpage) do not have to be annotated explicitly in a data set. When necessary, they are computed on the fly from a pair type's frequency signature. Variable references should be treated as read-only (they are automatically localised so that assigning a new value to a UCS variable reference does not modify the original data set).

Any temporary variables needed by the Perl code should be made lexical by declaring them with the `my` keyword. Variable names beginning with an underscore (such as `$_f` or `$_n_total`) are reserved for internal use. Please don't use global variables, which pollute the namespaces and might interfere with other parts of the program. If you feel that you absolutely need a variable to carry information from one row to the next, use a fully qualified variable name in your own namespace.

Since a UCS expression is compiled by the Perl interpreter, it offers the full power and flexibility of Perl, but it also shares its idiosyncrasies and traps for the unwary. You should have a good working knowledge of Perl in order to write UCS expressions. If you don't know the difference between `==` and `eq`, now is the time to type `perldoc perl` and start reading the Perl documentation.

Just as in Perl, data types are automatically converted as necessary. Missing values (which appear as NA in data set files) are represented by `undef` in Perl. When there may be missing values in a data set, test for definedness (e.g. with `defined(%b.colloc%)`) to avoid warning messages. UCS expression can use all standard Perl functions (described on the `perlfunc` manpage). In addition, the utility functions from `UCS::Expression::Func` (see the `UCS::Expression::Func` manpage for a detailed description) and a range of special mathematical and statistical functions defined in the `UCS::SFunc` module (see the `UCS::SFunc` manpage for a complete listing and details) are imported automatically and can be used without qualification.

UCS Expressions for Programmers The programmer interface to UCS expressions is provided by the `UCS::Expression` module (see the `UCS::Expression` manpage), with functions for compiling and evaluating UCS expressions. The `UCS::DS::Memory` module includes several methods that apply a UCS expression to the in-memory representation of a UCS data set. Note that all built-in association measures are implemented as UCS expressions (see the `UCS` and `UCS::AM` manpages for more information, or have a look at the source files).

When you want to use external functions (either defined by your own module or imported from a separate module), they must be fully qualified. For instance, you must write `Math::Trig::atan(1)` instead of just `atan(1)`. Make sure that the module is loaded (with `use Math::Trig;`) before the expression is evaluated for the first time. You can just put

the use statement in the Perl script or module where the UCS expression is defined, and it is probably also safe to include it in the expression itself (which allows you to use external libraries even in UCS expression typed on the command line).

An advanced feature of UCS expressions that is only available through the programmer interface are **parameters**. Parameters play the role of constants in UCS expressions: they can be accessed like data set variables, but their values are fixed and stored within the **UCS::Expression** object. Parameter names must be valid UCS identifiers and should be all uppercase in order to avoid conflicts with variable names. Parameters must be declared and initialised when the UCS expression is compiled. Their values can be changed with the **set_param** method. See the **UCS::Expression** manpage for more information.

Examples

- The simplest UCS expressions compare the values of a data set variable to a constant. Recall that `==` is used for numerical comparison and `eq` for string comparison in Perl. Both operands will automatically be converted into an appropriate data type.

```
%f% == 1          # hapax legomena (single occurrences)
%f% >= 5          # pair types with cooccurrence freq. >= 5
%l1% eq "black"   # first component type is "black"
```

Since UCS expressions are essentially short Perl scripts, the `#` character can be used to introduce line comments. String variables can also be matched against Perl regular expressions:

```
%l2% =~ /ness$/   # second component ends in ...ness
```

- Such simple comparisons can be combined into complex Boolean expressions. Use of the lexical operators `and`, `or`, and `not` is recommended for readability (and to avoid confusion with bit operators). Parentheses can also improve readability and help to avoid ambiguities.

```
%f% >= 5 and %f% < 10      # pair types in frequency range 5 .. 9
# pair types that are ranked high by t-score, but not by log-likelihood
(%r.t.score% <= 100) and not (%r.log.likelihood% <= 100)
```

- Missing values (NA) in a data set can be detected with Perl's **defined** operator. It may be useful to test data set variables before using them in order to avoid warning messages. The following examples assume a user-defined integer variable `n.accept`, which lists the number of annotators who have accepted a particular pair type as a collocation.

```
not defined(%n.accept%)     # selects rows where n.accept has the value NA
%n.accept% >= 1             # will print warnings for all NA values
defined(%n.accept%) and (%n.accept% >= 1) # this is safe
```

- UCS expressions may contain multiple Perl statements, which must be separated by semicolon (;) characters. In this way, a complex formula can be broken down into smaller parts. The value of the expression is determined by the last statement (or by an explicit **return** command). Temporary variables that hold intermediate values should always be declared with lexical scope (using **my**). The first example computes the minimum of two frequency ratios, using the pre-declared `min()` function from **UCS::Expression::Func**.

```
# UCS expression may also extend over multiple lines
my $ratio1 = %f% / %f1%;
my $ratio2 = %f% / %f2%;
min($ratio1, $ratio2);      # min() is pre-declared
```

The second example shows how temporary variables can be used to replace missing values with defaults. Here the integer variable `n.accept` (for the number of annotators that accepted the given pair type as a collocation) defaults to 0.

```
my $n = (defined %n.accept%) ? %n.accept% : 0;
$n >= 1;
```

The third example identifies prime numbers used as ID values.

```
foreach my $x (2 .. int(sqrt(%id%))) {
    return 0 if (%id% % $x) == 0;
}
return 1;
```

Dirty Tricks Things *not* to do ...

- Global variables can be used to carry information from one row to the next (while lexicals will be re-instantiated and possibly initialised for each row they are applied to). In order to avoid namespace pollution, put the global variable in a namespace of your own. The example below uses a global variable in a made-up namespace (`scrap`) to compute partial sums for the numerical variable `x.weight`.

```
$scrap::partial_sum += %x.weight%;
```

Of course, this expression will only work once. After that, the variable `$scrap::partial_sum` must be reset to zero. As long as the first row in the data set has an `id` value of 1, we can use the following trick (be careful when using the **UCS::DS::Memory** module, where index activation might change the order of the rows).

```
$scrap::partial_sum = 0 if %id% == 1;
$scrap::partial_sum += %x.weight%;
```

COPYRIGHT

Copyright (C) 2004 by Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ **ucsam**

Association measures in UCS/Perl

INTRODUCTION

The statistical analysis of cooccurrence data is usually based on **association measures**, mathematical formulae that compute an **association score** from the joint and marginal frequencies of a pair type (which are called a **frequency signature** in UCS). This score is a single floating-point number indicating the amount of statistical association between the components of the pair type. Association measures can often be written conveniently in terms of a **contingency table** of observed frequencies the corresponding expected frequencies under the null hypothesis that there is no association.

For instance, the word pair *black box* occurs 123 times in the *British National Corpus* (BNC), so its joint frequency is $f = 123$. The adjective *black* has a total of 13,168 occurrences, and the noun *box* has 1,810 occurrences, giving marginal frequencies of $f_1 = 13,168$ and $f_2 = 1,810$. From these data, the **MI** measure computes an association score of 1.4, while the **log.likelihood** measure computes a score of 567.72. Both scores indicate a clear positive association, but they cannot be compared directly: each measure has its own scale.

A more detailed explanation of contingency tables and association scores as well as a comprehensive inventory of association measures with equations given in terms of observed and expected frequencies can be found on-line at <http://www.collocations.de/AM/>. Also see the `ucsfile` manpage to find out how frequency signatures, contingency tables and association scores are represented in UCS **data set** files.

UCS/Perl supports more than 40 different association measures and variants. In order to keep them manageable, the measures are organised in several **packages**: a core set of widely-used "standard" measures is complemented by add-on packages for advanced users. Each package is implemented by a separate Perl module. Consult the module's manpage for a full listing of measures in the package and detailed descriptions. Listings of add-on packages, association measures, and some additional information can also be printed with the **ucs-list-am** program (see the `ucs-list-am` manpage).

Currently, there are two add-on packages in addition to the standard measures.

UCS::AM (the "standard" measures)

This core set contains all well-known association measures such as **MI**, **t-score**, and **log-likelihood** (see the listing in the Section SOME ASSOCIATION MEASURES below). These measures are also made available by various other tools (e.g. the NSP toolkit, see <http://www.d.umn.edu/~tpederse/nsp.html>) and they have often been used in applications as well as for scientific research. The **UCS::AM** package also includes several other "simple" measures that are inexpensive to compute and numerically unproblematic.

Association measures in the core set can be thought of as the "built-in" measures of UCS/Perl (although the add-on packages are also part of the distribution). They are automatically supported by tools such as **ucs-add**, while the other packages have to be loaded explicitly (see below).

See the **UCS::AM** manpage for details.

UCS::AM::HTest (measures based on hypothesis tests)

Many association measures are based on asymptotic statistical hypothesis tests. The test statistic is used as an association score and can be interpreted (i.e. translated into a **p-value**) with the help of its known limiting distribution. The **UCS::AM::HTest** package provides p-values for all such association measures as well as the "original" two-tailed versions of some tests (the core set includes only one-tailed versions).

See the **UCS::AM::HTest** manpage for details.

UCS::AM::Parametric (parametric measures)

A new approach where the equation of a parametric association measure is not completely fixed in advance. One or more parameters can be adjusted to obtain a version of the measure that is optimised for a particular task or data set. Control over the parameters is only available through the programming interface. For command-line use, special versions of these measures are provided with a pre-set parameter value, which is indicated by the name of the measure.

See the **UCS::AM::Parametric** manpage for details.

In UCS/Perl scripts both the standard measures and the add-on packages have to be loaded with **use** statements (e.g. `use UCS::AM;` for the core set). Association measures are implemented as **UCS::Expression** objects (see the **UCS::Expression** manpage). The **UCS** module maintains a registry of loaded measures with additional information and an evaluation function (see Section "ASSOCIATION MEASURE REGISTRY" in the **UCS** manpage). When one of the packages above is loaded, its measures are automatically added to this registry. Association scores can be computed more efficiently for in-memory data sets, using the **add** method in the **UCS::DS::Memory** module (see the **UCS::DS::Memory** manpage).

In the **ucs-add** program, the standard measures are pre-defined, and extension packages can be loaded with the **-x** option. Only the last part of the package name has to be specified here (e.g. **HTest** for the **UCS::AM::HTest** package). It is case-insensitive and may be abbreviated to a unique prefix (so both **-x htest** and **-x ht** work as well). See the **ucs-add** manpage for more information on how to compute association scores with the **ucs-add** program.

SOME ASSOCIATION MEASURES

This section briefly lists the most well-known association measures available in UCS/Perl, all of which are defined in the "standard" package **UCS::AM**. See the on-line resource at <http://www.collocations.de/AM/> for fully equations and the **UCS::AM** manpage for details.

MI (Mutual Information)

The mutual information (MI) measure is a maximum-likelihood for the (logarithmic) *strength of the statistical association* between the components of a pair type. It was introduced into the field of computational lexicography by Church & Hanks (1990), who derived it from the information-theoretic notion of *point-wise mutual information*. Positive values indicate positive association while negative values indicate dissociation (where the components have a tendency *not* to occur together).

Note that unlike the original version of Church & Hanks (1990), the UCS implementation computes a base 10 logarithm.

t.score (t-score)

The MI measure is prone to overestimate association strength, especially for low-frequency cooccurrences. Church *et al.* (1991) use a version of Student's *t* test (whose test statistics is called a *t-score*) to ensure that the association detected by MI is supported by a *significant* amount of evidence. Although their application of Student's test is highly questionable, the combination of MI and t.score has become a *de facto* standard in British computational lexicography.

chi.squared, chi.squared.corr (chi-squared test)

Pearson's chi-squared test is the standard test for statistical independence in a 2×2 contingency table, and is much more appropriate as a measure of the *significance of association* than t.score. Despite its central role in mathematical statistics, it has not been very widely used on cooccurrence data. In particular, t.score was found to be much more useful for the extraction of collocations from text corpora (cf. Evert & Krenn, 2001).

The "textbook" form of Pearson's chi-squared test is a two-tailed version that does not distinguish between positive and negative association. The chi.squared measure implemented in UCS/Perl has been converted to a one-sided test with the help of a heuristic decision rule. Since contingency tables often contain cells with small values, Yates' continuity correction should be applied to the test statistic (chi.squared.corr).

log.likelihood (likelihood ratio test)

Dunning (1993) showed that the disappointing performance of chi.squared in collocation extraction tasks is due to a drastic overestimation of the significance of low-frequency cooccurrences (because of a approximation to its limiting distribution). He suggested to use a likelihood ratio test instead, whose natural logarithm has the same limiting distribution as chi.squared. Under the name *log-likelihood*, this association measure has become a generally accepted standard in the field of computational linguistics.

Like the chi-squared test, the likelihood ratio test is two-sided, and the log.likelihood measure has been converted to a one-sided test with the same heuristic decision rule. Both chi.squared and log.likelihood return the value of their test statistic, which has to be interpreted in terms of the known limiting distribution. More meaningful **p-values** for both measures are available in the UCS::AM::HTest package.

Fisher.pv (Fisher's exact test)

Although log.likelihood achieves a much better approximation to its limiting distribution than chi.squared (or chi.squared.corr), it is still an asymptotic and provides only an approximate p-value. Pedersen (1996) argued in favour of Fisher's exact test for the independence of rows and columns in a contingency table, in order to remove the remaining inaccuracy of the log-likelihood ratio. A drawback of Fisher's test is that it is numerically expensive and that naive implementations can easily become unstable.

The Fisher.pv measure implements a one-sided test. It returns an exact **p-value**, which can be compared directly with the p-values of chi.squared and log.likelihood.

Dice (Dice coefficient)

The Dice coefficient is a measure from the field of information retrieval, which has been used by Smadja (1993) and others for collocation extraction. Like MI, it is a maximum-likelihood estimate of *association strength*, but its definition of "strength" differs greatly

from point-wise mutual information. It suffers from the same overestimation problem as MI, which is mitigated by its different approach to association strength, though.

References Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics* **16**(1), 22-29.

Church, K. W.; Gale, W.; Hanks, P.; Hindle, D. (1991). Using statistics in lexical analysis. In: *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, Lawrence Erlbaum, pages 115-164.

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* **19**(1), 61-74.

Evert, S. and Krenn, B. (2001). Methods for the qualitative evaluation of lexical association measures. In: *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, pages 188-195.

Pedersen, T. (1996). Fishing for exactness. In: *Proceedings of the South-Central SAS Users Group Conference*, Austin, TX.

Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics* **19**(1), 143-177.

UCS CONVENTIONS

UCS/Perl uses some conventions for the names of association measures and the computed association scores, which are described in this section. It is important to be aware of such conventions, especially when they deviate from those used by other software packages.

The **names of association measures** are taken from the on-line inventory at <http://www.collocations.de/AM/>. Hyphen characters (-) are replaced by periods (.) to conform with the UCS standards (see the `ucsfile` manpage). Capitalisation is preserved (`MI` and `Fisher.pv`, but `log.likelihood`) and subscripts are included in the name, separated by a period (`chi.squared.corr`, where `corr` is a subscript in the original name).

Association scores are always arranged so that **higher scores** indicate stronger (positive) association, applying a transformation to the original values if necessary. In the one-sided versions of two-sided tests (e.g. `chi.squared` and `log.likelihood`), negative scores indicate negative association (while positive scores indicate positive association). Scores close to zero are a sign of statistical independence. Some other measures such as `MI` also have this property, but many do not (e.g. `Fisher.pv` or `Dice`).

"Explicit" logarithms in the equation of an association measure are usually taken to the **base 10** (e.g. in the `MI` measure). This is not the case when the association score is not interpreted as a logarithm (e.g. the `log.likelihood`, which is a test statistic approximating a known limiting distribution) and the natural logarithm is required for correct interpretation. The use of base 10 logarithms is always pointed out in the documentation (see the `UCS::AM` manpage). The logarithm of infinity is represented by a large floating-point value returned by the `inf` function (from the `UCS::Expression::Func` module). Comparison with `+inf()` and `-inf()` can be used to detect a positive or negative infinite value.

The scores of association measures with the extension `.pv` represent a p-value (from an exact test or the approximate p-value of an asymptotic test). Unlike most other scores, p-values can be compared directly between different measures. They are represented as **negative base 10 logarithms**, so the association score 3.0 corresponds to a p-value of $0.001 = 1e-3$ (`+inf()` stands for zero probability, usually the result of an underflow error).

COPYRIGHT

Copyright (C) 2004 by Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

B.1.2 UCS/Perl Programs

■ **ucsdoc**

UCS front-end to perldoc

SYNOPSIS

```
ucsdoc [-tk|-ps|-t] [options] PageName | ModuleName | ProgramName
```

DESCRIPTION

ucsdoc is a front-end to the **perldoc** program, which sets the required library paths for the UCS/Perl manpages. Standard Perl documentation is available through **ucsdoc** as well.

With the **-t** option, the manpage is formatted in plain ASCII, without highlighting.

With the **-ps** option, the manpage is formatted in PostScript for printing. The PostScript code is displayed on stdout so that it can be re-directed into a file or piped into a print command.

With the **-tk** option, the manpage is displayed in a Perl/Tk window, provided that the **Tk** and **Tk::Pod** modules are installed.

Only one of the three formatting options may be specified.

All other command-line arguments are passed to the **perldoc** program. Type **perldoc -h** and **perldoc perldoc** for more information on the available options.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ **ucs-config**

Automatic configuration of UCS/Perl scripts

SYNOPSIS

```
ucs-config
```

```
ucs-config [--version | --base-dir | --perl-dir | --bin-dir | --lib-dir | --R-bin]
ucs-config [-v | --base | --perl | --bin | --lib | -R]
```

```
ucs-config ucs-script.pl ucs-script.R ...
```

```
ucs-config --run [options] one-liner.perl
ucs-config --run [options] -e '...'
ucs-config -e '...'
```

DESCRIPTION

The **ucs-config** program is used to print information about the installed UCS/Perl version and directories, as well as for the automatic configuration of UCS/Perl scripts. The program can be run in four different modes.

Invoking **ucs-config** without any arguments prints the UCS splash screen and a configuration summary.

In the second mode, the program prints one item of configuration information selected with one of the following flags. This mode is most suitable for use in shell scripts and makefiles. Note that you are not allowed to specify more than one flag at a time.

```
--version    UCS version
--base-dir   root directory of the UCS system
--perl-dir   root directory of the UCS/Perl subsystem
--bin-dir    bin/ directory of UCS/Perl (contains UCS programs)
--lib-dir    lib/ directory of UCS/Perl (contains UCS modules)
--R-bin      fully qualified filename of the R interpreter
```

The third mode is used to in-place edit Perl and R scripts so that they can load the **UCS** modules and libraries. For **Perl scripts**, **ucs-config** inserts a suitable shebang (**#!**) line, invoking the Perl interpreter for which UCS is configured together with the necessary include paths. For **R scripts** (which are recognised by their extension **.R** or **.S**), **ucs-config** looks for a line containing the command `source("../ucs.R")` in the script, and inserts the correct path there. Please make sure that this line does not contain any other commands.

The final mode, introduced by the command-line switch **-run**, invokes the Perl interpreter with the correct UCS library path and (almost) all **UCS modules** pre-loaded (including the standard association measures from **UCS::AM**, but none of the add-on packages). The remaining command-line arguments are passed through to the Perl interpreter, which is *really cool* for writing one-liners in **UCS/Perl**. The flag **-e** is an abbreviation of **-run -e**, but does not allow any options to be passed to the interpreter.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-tool

Execute UCS/Perl scripts from contrib/ tree

SYNOPSIS

```
ucs-tool --list [--category | --category=<cat>]
ucs-tool --doc <tool> [<ucsdoc options>]
ucs-tool [--category=<cat>] <tool> ...
```

DESCRIPTION

In addition to the UCS/Perl programs, which perform general tasks and will be of interest to most users, the UCS distribution includes a number of UCS/Perl scripts for more specific applications. These scripts are not directly accessible as command-line programs. They are organised into a hierarchical set of categories in the *contrib/* directory tree, and can be invoked through the **ucs-tool** program. If you want to add your own scripts to this tree, read the section on WRITING CONTRIBUTED SCRIPTS below.

LISTING CONTRIBUTED SCRIPTS When the `-list` (or `-l`) option is specified, **ucs-tool** lists all available UCS/Perl scripts from the *contrib/* tree, grouped by category. Add the option `-category` (or `-cat` or `-c`) for a listing of category names and descriptions (without the individual tools). You can also use the special short form `ucs-tool -lc` for this purpose. When an argument is given for `-category`, only scripts from the specified category are listed (the category name is case-insensitive).

Some scripts may provide manual pages in the form of embedded POD documentation. Such manual pages can be displayed with the `-doc` (or `-d`) flag, followed by the name of the script. See the section on SCRIPT INVOCATION below for details on how script names are matched. **ucs-tool** uses the **ucsdoc** program to format manual pages and accepts **ucsdoc** options (such as `-ps` and `-tk`) *after* the tool name.

SCRIPT INVOCATION In order to invoke one of the contributed UCS/Perl scripts, simply specify its name (as shown by the `-list` option), followed by command-line arguments for the selected script, e.g.

```
ucs-tool dispersion-test -m 3 -N 100000 -k 100 -V 2500
```

All contributed scripts should include a short help page that can be displayed with the `-help` (or `-h`) option. Note that this is a script option and therefore must be specified *after* the script name:

```
ucs-tool dispersion-test --help
```

Recall that full manual pages, when available, can be displayed with the `-doc` option specified *before* the script name (as described above).

Script names are case-insensitive, and it is sufficient to specify a unique prefix of the name. For instance, you can invoke the **print-documentation** script with the short name `ucs-tool print` or `ucs-tool print-doc`. It may be easier to find a unique prefix when the search space is reduced to a specific category with the `-category` (or `-c`) option.

WRITING CONTRIBUTED SCRIPTS

Contributed UCS/Perl scripts are collected in a directory tree rooted in *System/Perl/contrib/*. Each subdirectory corresponds to a script category. These categories are organised hierarchically according to the directory structure (for instance, `-list -category=Import` lists all scripts found in the directory *Import/* and its subdirectories, such as *Import/NSP/* and *Import/CWB/*). The file *CATEGORIES* contains a listing of all known categories with short descriptions (category names and descriptions must be separated by a single TAB character).

If you want to add your own UCS/Perl scripts to the repository, you should put them in the *Local/* directory (which is reserved for scripts that are not part of the UCS distribution). This is often the easiest way to make a UCS/Perl script available to all users of a UCS installation. Note that script files *must* have the extension `.perl` or `.pl`, which is not part of the script name (e.g., the script **nsp2ucs** in the category **Import/NSP** corresponds to the disk file *Import/NSP/nsp2ucs.perl* in the *contrib/* tree). You can also put your script in a different category or define your own categories (which you must add to the *CATEGORIES* file), but this will interfere with upgrading to a new UCS release. You are encouraged to share scripts with other users. To do so, please send them to the author (or maintainer) of the UCS system, indicating which category they should be included in.

Unlike ordinary UCS/Perl scripts, scripts placed in the *contrib/* tree do not have to be configured with **ucs-config**. They also do not have to be executable and start with a shebang (`#!`) line. When invoked with the **ucs-tool** program, the necessary settings are made automatically. Contributed scripts that require "private" modules (which are not installed in a public directory) can place them in a subdirectory named *lib/* (relative to the location of the script file), or in further subdirectories as required by the module's name. The *lib/* directory tree is automatically added to Perl's search path. Necessary data files should be wrapped in Perl modules and stored in the *lib/* subtree as well. For instance, assume that a script named **my-script** in the **Local** category (corresponding to the script file *Local/my-script.perl*) uses the private module **My::Functions**. This module can automatically be loaded (with use `My::Functions;`) from the file *Local/lib/My/Functions.pm* in the *contrib/* directory tree.

All contributed UCS/Perl scripts should include a short help page describing the script's function and command-line arguments, which is displayed when the script is invoked with `-help` or `-h`. Script authors are also encouraged to write full manual pages as embedded POD documentation (which can then be displayed with `ucs-tool -doc`), but these are not mandatory.

COPYRIGHT

Copyright 2004-2005 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ **ucs-list-am**

List built-in association measures and add-on packages

SYNOPSIS

```
ucs-list-am [-v | -c | -t | -f <f,f1,f2,N>]
            [-x <package> | -p <package>] [<am1> <am2> ...]

ucs-list-am --list
```

DESCRIPTION

This program is a convenient front-end to the registry of association measures maintained by the **UCS** module. It can be used to print a list of built-in association measures, add-on packages, and display additional information about the measures (where available). Detailed information about the measures can be found in the `UCS::AM` manpage and the respective manpages of the extension packages. See the `ucsam` manpage for an introduction and overview.

```
ucs-list-am --list
```

With the `-list` (or `-l`) option, **ucs-list-am** lists all available add-on packages.

```
ucs-list-am [<options>] [<am1>, <am2>, ...]
```

When **ucs-am-list** is called without arguments, it prints the names of all built-in association measures on stdout, each one followed by a short one-line description of the measure. Specific association measures can be selected by giving their names as command-line arguments. UCS wildcard patterns (see the `ucsexp` manpage) will list all matching measures.

The `-extra` (or `-x`) option can be used to load one or more add-on packages so that the association measures from these packages will be included in the listing (in addition to the built-in measures). Its argument is a comma-separated list of package names, which are case-insensitive and may be abbreviated to unique prefixes. For instance, both `-extra=HTest,Parameteric` and `-x htest,param` will load the **UCS::AM::HTest** and **UCS::AM::Parametric** packages. The special keyword `ALL` loads all available AM packages.

The `-package` (or `-p`) option is used to list the association measures from a single package (*without* the built-in measures). Again, the package name is case-insensitive and may be abbreviated to a unique prefix. Note that the `-package` option cannot be used to load multiple packages.

The amount of information provided can be controlled with the `-verbose` (or `-v`), `-code` (or `-c`), and `-terse` (or `-t`) options. In `-terse` mode, only the names of packages are printed, so that the output can be easily processed by other programs. In `-verbose` mode, the name of each association measure is immediately followed by a one-line description (in parentheses). When available, one or more lines of additional comments will also be shown. In `-code` mode, the output consists of the name of each measure, followed by its implementation (as a UCS expression), followed by a blank line. For parameteric measures, a list of parameters and their default values is shown on a separate line between the name and the implementation.

Alternatively, a frequency signature can be specified as an argument to the `-frequencies` (or `-f`) option. The expected format is a comma-separated list of four integers, representing the variables `f`, `f1`, `f2` and `N`. In this case, association scores for all selected measures are computed on the specified frequency signature. Note that it is not possible to compute scores for different frequency signatures with a single invocation of the **ucs-list-am** tool.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-make-tables

Compute contingency tables from a sequence of pair tokens

SYNOPSIS

```
... | ucs-make-tables [-v] [--sort | -s] [--sample-size=<n> | -N <n>]
      [--threshold=<n> | -f <n>] data.ds.gz

... | ucs-make-tables [-v] [-s] [-N <n>] [-f <n>]
      [--dispersion [--chunk-size=<n>] ] data.ds.gz

... | ucs-make-tables [-v] [-s] [-N <n>] [-f <n>] --segments data.ds.gz
```

DESCRIPTION

This utility computes frequency signatures and constructs a UCS data set for a stream of pair tokens (or segment-based cooccurrence data) read from STDIN. It is usually applied to the output of a cooccurrence extraction tool in a command-line pipe. The input can also be read from a file (with a < redirection), or decompressed on the fly with (`gzip -cd` or `bzip2 -cd`). The resulting data set is written to the file specified as the single mandatory argument on the command-line.

ucs-make-tables operates in two different modes for **relational** and **positional** (segment-based) cooccurrences. These two modes are described separately in the following subsections. They take the same command-line options and arguments, as described in the section **COMMAND LINE** below. Distance-based positional cooccurrences are not supported, as they usually require direct access to the source corpus in order to determine the precise window size.

Relational Cooccurrences By default, **ucs-make-tables** operates in a mode for relational cooccurrences. In this mode, the input line format is

```
<l1> TAB <l2>
```

Each such line represents a pair token with labels <l1> and <l2> (i.e. a pair token that belongs to the pair type $(l1, l2)$). For dispersion counts (see below), the input lines should preserve the order in which the corresponding pair tokens appear in the corpus. When dispersion is measured with respect to pre-annotated parts (e.g. paragraphs or documents) rather than equally-sized parts, the input must contain an extra column with unique part identifiers:

```
<l1> TAB <l2> TAB <part_id>
```

Note that all pair tokens from a given part must form an uninterrupted sequence in the input, otherwise the dispersion counts will not be correct.

Segment-based Cooccurrences The mode for segment-based cooccurrences is activated with the `-segments` (or `-S`) option. In this mode, each segment is represented by a sequence of four lines in the input stream, called a **record**:

1. `<segment_id>` [TAB `<part_id>`]
2. The labels of all tokens in the segment that can become *first* components of pairs, separated by TABs.
3. The labels of all tokens in the segment that can become *second* components of pairs, separated by TABs.
4. A blank separator line.

Duplicate strings on the second or third line will automatically be ignored. The `<segment_id>` on the first line is currently ignored. The optional `<part_id>` can be used to compute dispersion counts for pre-annotated parts. All segments that belong to a given part must appear in consecutive records, otherwise the dispersion counts will not be correct.

A prototypical example of the segment-based approach are lemmatised noun-verb cooccurrences within sentences. In this case, each record in the input stream corresponds to a sentence. The first line contains an unimportant sentence identifier. The second line contains the lemma forms of all nouns in the sentence (note that duplicates are automatically removed), and the third line contains the lemma forms of all verbs in the sentence. In order to compute the dispersion of cooccurrences across documents (i.e. *document frequencies* in the terminology of information retrieval), unique document identifiers have to be added to the first line.

COMMAND LINE

The general form of the **ucs-make-tables** command is

```
... | ucs-make-tables [--verbose | -v] [--sort | -s]
                    [--threshold=<t> | -f <t>]
                    [--sample-size=<n> | -N <n>]
                    [--dispersion [--chunk-size=<s>]]
                    [--segments]
                    data.ds.gz
```

With the `-verbose` (or `-v`) option, some progress information (including the number of pair tokens or segments, as well as the number of pair types encountered so far) is displayed while the program is running. When `-sort` (or `-s`) is specified, the resulting data set is sorted in ascending alphabetical order (on l1 first, then l2). Of course, the data set file can always be re-sorted with the **ucs-sort** utility. When a frequency threshold `<t>` is specified with the `-threshold` (or `-f`) option, only pair types with cooccurrence frequency $f \geq \langle t \rangle$ will be

saved to the data set file (but they are still included in the marginal frequency counts of relational cooccurrences, of course). This option helps keep the size of data sets extracted from large corpora manageable.

When `-sample-size` (or `-N`) is specified, only the first `<n>` pair tokens (or segment records) read from STDIN will be used, so that the sample size `N` of the resulting data set is equal to `<n>`. This option is mainly useful when computing dispersion counts on equally-sized parts (see below), but it has some other applications as well.

With the `-dispersion` (or `-d`) option, dispersion counts are added to the data set and can then be used to test the random sample assumption with a **dispersion test** (see Baayen 2001, Sec. 5.1.1). In order to do so, the token stream is divided into equally-sized **parts**, each one containing the number `<s>` of pair tokens specified with the `-chunk-size` (or `-c`) option. For segment-based cooccurrences, each part will contain cooccurrences from `<s>` segments. When the total number of pair tokens (or segments) is not an integer multiple of `<s>`, a warning message will be issued. In this case, it is recommended to adjust the number of tokens with the `-sample-size` option described above.

The dispersion count for each pair type, i.e. the number of parts in which it occurs, is stored in a variable named `n.disp` in the resulting data set file. In addition, the number of parts and the part size are recorded in the global variables `chunks` and `chunk.size`. When the part size is not specified, dispersion counts can be computed for pre-annotated parts, which must be identified in the input stream (see above). In this case, `chunk.size` is not defined as the individual parts may have different sizes. **NB:** The use of pre-annotated parts is discouraged, since the mathematics of the dispersion test assume equally-sized parts.

Examples If you have installed the IMS Corpus Workbench (CWB) as well as the CWB/Perl interface, you can easily extract relational adjective+noun cooccurrences from part-of-speech tagged CWB corpora. The `ucs-adj-n-from-cwb.perl` script supplied with the UCS system supports several tagsets for German and English corpora. It can easily be extended to other tagsets, languages, and types of cooccurrences (as long as they can be identified with the help of part-of-speech patterns).

The following example extracts adjective+noun pairs with cooccurrence frequency $f \geq 3$ from the CWB demonstration corpus DICKENS (ca. 3.4 million words), and saves them into the data set file `dickens.adj-n.ds.gz`. The shell variable `$UCS` refers to the *System/* directory of the UCS installation (as in the UCS/Perl tutorial).

```
$UCS/Perl/tools/ucs-adj-n-from-cwb.perl penn DICKENS
| ucs-make-tables --verbose --sort --threshold=3 dickens.adj-n.ds.gz
```

(Note that the command must be entered as a single line in the shell.)

Extraction from the DICKENS corpus produces approximately 122990 pair tokens. In order to apply a dispersion test with a chunk size of 1000 tokens each, the sample size has to be limited to an integer multiple of 1000:

```
$UCS/Perl/tools/ucs-adj-n-from-cwb.perl penn DICKENS
| ucs-make-tables --verbose --sort --threshold=3 --sample-size=122000
--dispersion --chunk-size=1000 dickens.disp.ds.gz
```

A dispersion test for pair types with $f \leq 5$ can then be performed with the following command, showing a significant amount of underdispersion at all levels.

```
$UCS/Perl/tools/ucs-dispersion-test.perl -v -m 5 dickens.disp.ds.gz
```

Segment-based data can be obtained from a CWB corpus with the **ucs-segment-from-cwb.perl** script. The following example extracts nouns and verbs cooccurring within sentences. A frequency threshold of 5 is applied in order to keep the amount of data (and hence the memory consumption of the **ucs-make-tables** program) manageable.

```
$UCS/Perl/tools/ucs-segment-from-cwb.perl -f 5 -t1 "VB.*" -t2 "NN.*" DICKENS s
| ucs-make-tables --verbose --segments --threshold=5 dickens.n-v.ds.gz
```

REFERENCES

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

IMS Corpus Workbench (CWB): <http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/>

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-summarize

Compute statistical summaries for variables in UCS data set

SYNOPSIS

```
ucs-summarize [-v] [-m] f f1 f2 FROM data.ds.gz
```

```
ucs-summarize [-v] [-m] am.%.pv FROM data.ds.gz
```

```
ucs-summarize [-v] [-m] data.ds.gz
```

DESCRIPTION

This program computes short statistical summaries of numerical variables in a UCS data set. The general form of the **ucs-summarize** command is

```
ucs-summarize [-v] [-m] <variables> FROM <input.ds>
```

where <variables> is a whitespace-separated list of variable names or wildcard expression, and the data set is read from the file specified as <input.ds>. Wildcard expressions may need to be quoted to avoid interpretation by the shell. When the list of variables is omitted (including the keyword FROM), summaries are generated for all variables in the data set. In

verbose mode (`-verbose` or `-v` option), some progress information is shown while computing the summary.

So far, the statistical summary includes the **minimum** (`min.`), **maximum** (`max.`), **mean** (`mean`), **empirical variance** (`var.`), and the **empirical standard deviation** (`s.d.`). In addition, the number of missing values (NA's) is reported.

When `-memory` (or `-m`) is specified, the data set will be read into memory first. In addition to the ordinary statistical summary, the **absolute minimum** (`abs.min.`, the smallest non-zero absolute value), **absolute maximum** (`abs.max.`), and **granularity** (`gran.`, smallest difference between any two unequal values) are computed in this mode.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-select

Select rows and/or columns from UCS data set

SYNOPSIS

```
ucs-select --count FROM data.ds.gz WHERE '%011% < %E11%'
ucs-select '*' 'am%.pv' FROM data.ds.gz INTO new.ds.gz
ucs-select '%' FROM data.ds.gz WHERE 'not defined %b.accept%'
```

DESCRIPTION

This program is used to select rows and/or columns from a UCS data set file, very much like a `SELECT` statement in SQL. The general form of the **ucs-select** command is

```
ucs-select [--verbose | -v] (<variables> | --count)
          [ FROM <input.ds> ] [ WHERE <condition> ] [ INTO <output.ds> ]
```

`<variables>` is a whitespace-separated list of variable names or wildcard patterns (see the `ucsexp` manpage), which are matched against the columns of the data set file `<input.ds>`. The list of variables may not be omitted: use `'%'` to select *all* columns, and `-count` to display the number of matching rows only. Note that wildcard patterns may need to be quoted individually (because they contain shell metacharacters).

`<condition>` is a UCS expression (see the `ucsexp` manpage) used to select rows from the data set for which it evaluates to a true value. When the `WHERE` clause is omitted, all rows are selected. Note that `<condition>` must be a single argument and will usually have to be quoted (single quotes are highly recommended).

The input data set file `<input.ds>` defaults to `STDIN` (when omitted). The resulting table is printed on `STDOUT` in UCS data set file format (see the `ucsfile` manpage), and can be written to a data set file `<output.ds>` with the optional `INTO` clause.

With the `-verbose` (or `-v`) option, some progress information is displayed while the program is running.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-add

Add variables (association scores) to UCS data set

SYNOPSIS

```
ucs-add [-v] [-m] am.t.score am.Fisher.pv TO data.ds.gz INTO new.ds.gz
```

```
ucs-add [-v] [-m] -x HTest am.%.pv TO data.ds.gz INTO new.ds.gz
```

```
ucs-add [-r] r.% TO data.ds.gz INTO new.ds.gz
```

DESCRIPTION

This program is used to add variables (**association scores**, **rankings**, **derived variables**, or arbitrary **UCS expressions** entered on the command line) to a UCS data set. If a variable is already defined in the data set, its values will be overwritten.

The general form of the **ucs-add** command is

```
ucs-add [--verbose | -v] [--memory | -m] [--extra=<list> | -x <list>]
      <variables> [ TO <input.ds> ] [ INTO <output.ds> ]
```

where `<variables>` is a whitespace-separated list of variable specifications (see the section on `VARIABLE SPECIFICATIONS` below for details). An additional `-randomize` option is only useful when adding rankings:

```
ucs-add [--verbose | -v] [--extra=<list> | -x <list>] [--randomize | -r]
      <variables> [ TO <input.ds> ] [ INTO <output.ds> ]
```

The data are read from the file `<input.ds>`, and the resulting data set with the new annotations is written to the file `<output.ds>`. When they are not specified, the input and output files default to `STDIN` and `STDOUT`, respectively.

Variable specifications and file names may need to be quoted individually (when they contain shell metacharacters or whitespace).

Normally, the **ucs-add** program processes the data set one row at a time, so that `<input.ds>` and `<output.ds>` must not refer to the same file. When `-memory` (or `-m`) is specified, the entire data set is read into memory, annotated, and then written back to the output file. In this case, `<input.ds>` and `<output.ds>` may be identical. This mode is automatically activated when any rankings are added to the data set.

In both modes of operation, variables are added in the order in which they are given on the command-line, so variable specifications (rankings and user-defined expressions) may refer to any of the previously introduced variables.

With the `-verbose` (or `-v`) option, some debugging and progress information is displayed while the program is running. The `-extra` (or `-x`) option loads additional built-in association measures (see the section on adding Association Scores below for details).

VARIABLE SPECIFICATIONS

Association Scores Variables representing association scores are selected by specifying their variable names (which start with the prefix `am.`). The names may be given as UCS wildcard patterns (see the `ucsexp` manpage), which will be matched against the list of all supported association measures. Examples of useful wildcard patterns are `am.%` (all measures), `am.%pv` (all measures that compute probability values), and `am.chi.squared.%` (all variants of Pearson's chi-squared test).

By default, only the basic association measures defined in `UCS::AM` are supported. Other AM packages (see the `UCS::AM` manpage for a list of add-on packages) can be loaded with the `-extra` (or `-x`) option. The argument is a comma-separated list of package names (e.g. `-extra=HTest,Parametric` to load `UCS::AM::HTest` and `UCS::AM::Parametric`), which are case-insensitive and may be abbreviated to unique prefixes (so `-x htest,param` works just as well). Use `-x ALL` to load all available AM packages.

Rankings Variables representing association score rankings are selected by specifying their variable names (which start with the prefix `r.`). In order to compute a ranking, say `r.something`, the corresponding association scores (`am.something`) must be annotated in the data set. UCS wildcard patterns are matched against all association scores in the data set (but not against other built-in association measures). Rankings can also be computed for user-defined measures, provided that their association scores are annotated. In order to compute a ranking for a built-in association measure that is not available in the data set, both the association score and the ranking variable must be specified. The example

```
ucs-add -m am.% r.% TO data.ds.gz INTO data.ds.gz
```

adds associations scores and rankings for the basic built-in association measures to the data set `data.ds.gz`.

Ties are not resolved in the rankings, so pair types with identical association scores share the same rank. The rank assigned to such a group of pair types is the lowest free rank (as in the Olympic Games) rather than the average of all ranks in the group (as is often done in statistics). With the `-random` (or `-r`) option, ties are resolved in a random fashion. When

association scores for the random measure are pre-annotated (i.e. the `am.random` variable is present in the data set), these are used for the randomization so that the ranking is reproducible.

Derived Variables Any variable names or wildcard patterns that do not match one of the built-in association measures are matched against the list of derived variables, which can be computed automatically from the frequency signatures of pair types. See the `ucsfile` manpage for a complete list of derived variables. Examples of useful patterns are `E*` (expected frequencies), `lp*` (logarithmic coordinates), and `e b m` ((e,b,m) -coordinates).

User-Defined Expressions A user-defined variable specification is a UCS expression (see the `ucsexp` manpage) of the form

```
<var> := <expression>
```

where `<var>` is the name of a user-defined variable, association score, or ranking (without surrounding `%` characters). This variable is added to the input data set if necessary and set to the values computed by the UCS expression `<expression>`. The example below computes association scores for a compound measure mixed from the rankings according to two other measures (which must both be annotated in the data set).

```
am.mixed := -max(%r.t.score%, %r.dice%)
```

Note that it isn't possible to compute the corresponding ranking `r.mixed` directly.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-join

Join rows and variables from two UCS data sets

SYNOPSIS

```
ucs-join data1.ds.gz data2.ds.gz
ucs-join [--add] [--no-overwrite] data1.ds.gz data2.ds.gz INTO new.ds.gz
ucs-join [--add] [--no-overwrite] data1.ds.gz WITH am.% FROM data2.ds.gz INTO new.ds.gz
```

DESCRIPTION

This program can be invoked in three different ways. The short form

```
ucs-join [-v] <ds1> <ds2>
```

compares two data sets <ds1> and <ds2>. In particular, the number of rows common to both data sets and the numbers of rows unique to either one of the data sets are reported. Rows are matched on the **pair types** they represent, i.e. the variables l1 and l2. Differences in the id value or any other annotations are ignored. The **coverage** is the proportion of pair types in <ds1> that are also contained in <ds2>. With the `-verbose` or `-v` switch, some progress information is displayed while the program is running.

The second form

```
ucs-join [-v] [--add] [--no-overwrite]
         <ds1> <ds2> INTO <ds3>
```

adds variables and/or rows from the data set <ds2> to <ds1>. Rows from the two data sets are matched on the l1 and l2 variables as above. For these rows, all variables from <ds2> are added to the annotations in <ds1>. Variables that are common to both data sets are overwritten with the values from <ds2>. With the `-no-overwrite` or `-n` switch, only missing values (NA) are overwritten. If `-add` or `-a` is specified, rows unique to <ds2> are added to <ds1> (with all variables that are not defined in <ds2> set to NA). The resulting data set is written to the file <ds3>.

The most general form

```
ucs-join [-v] [--add] [--no-overwrite]
         <ds1> WITH <variables> FROM <ds2> INTO <ds3>
```

adds selected variables from <ds2> only. <variables> is a whitespace-separated list of variables names and wildcard patterns, which are matched against the variables of <ds2>. Variables can be renamed with specifiers of the form `new.name=old.name` (of course, wildcard patterns cannot be used here). The `-add` switch is rarely useful with this form of the **ucs-join** command.

ANNOTATION DATABASES

The **ucs-join** program is often used to add (manual) annotations from an **annotation database** file (.adb) to a data set, and to update annotation databases. For instance, the UCS distribution includes German PP+verb pairs extracted from the *Frankfurter Rundschau* corpus (*fr-pnv.ds.gz*) and an annotation database created by Brigitte Krenn (*pnv.adb.gz*). In order to check the **coverage** of the annotation database (i.e., how many of the pair types are already contained in the database), type

```
ucs-join -v fr-pnv.ds.gz pnv.adb.gz
```

This will show a coverage of 100%. Annotations from the database can now be added to the *fr-pnv.ds.gz* data set:

```
ucs-join -v fr-pnv.ds.gz WITH 'b.*' FROM pnv.adb.gz INTO fr-pnv.annot.ds.gz
```

When an annotation database contains entries that have not been manually examined so far, these should be annotated with missing values (NA). The database can then be updated from a new file (in the same .adb format, say *new-pnv.adb*) with the following commands

```
mv pnv.adb.gz pnv.adb.BAK.gz
ucs-join -v --no-overwrite pnv.adb.BAK.gz new-pnv.adb INTO pnv.adb.gz
```

The `-no-overwrite` flag ensures that existing annotations aren't overwritten in the process. If the file *new-pnv.adb* contains additional pair types (that haven't already been entered into the database), you should also specify the `-add` flag.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-sort

Sort UCS data set by one or more variables

SYNOPSIS

```
ucs-sort [-v] [-r] [data.ds.gz] BY am.t.score [INTO new.ds.gz]
```

```
ucs-sort [-v] [-r] [data.ds.gz] BY l2+ l1- ... [INTO new.ds.gz]
```

DESCRIPTION

This program sorts the rows of UCS data by one or more variables. The general form of the `ucs-sort` command is

```
ucs-sort [--verbose | -v] [--randomize | -r]
        [<input.ds>] BY <variables> [INTO <output.ds>]
```

where `<variables>` is a whitespace-separated list of variable names. A `+` or `-` character appended to a variable name selects ascending or descending order, respectively. The default order depends on the variable type (association scores are sorted in descending order).

The data set is read from STDIN by default, or from the file `<input.ds>` when it is specified. The sorted data set is printed on STDOUT, and can be saved into the file `<output.ds>` with the optional `INTO` clause.

When `-randomize` (or `-r`) is specified, ties are broken randomly, using the `am.random` measure if it is annotated in the data set. The `-verbose` (or `-v`) option displays some (minimal) progress information.

EXAMPLES

The **ucs-sort** utility is often used in command-line pipes to sort data sets before viewing. Assuming that a data set file *candidates.ds.gz* is annotated with the necessary association scores, ranked candidate lists for the log-likelihood and t-score measures can be displayed with the following commands:

```
ucs-sort -r candidates.ds.gz BY am.log.likelihood | ucs-print -i
ucs-sort -r candidates.ds.gz BY am.t.score | ucs-print -i
```

ucs-sort can also be applied to the output of another UCS tool, e.g. **ucs-select**. The following command selects the 100 highest-ranked pair types from the data set file *candidates.ds.gz*, according to the log-likelihood measure, and displays them in alphabetical order, sorted by 12 first. (Note that the command must be entered as a single line in the shell.)

```
ucs-add -v r.log.likelihood TO candidates.ds.gz
| ucs-select -v '%' WHERE '%r.log.likelihood% <= 100'
| ucs-sort BY 12 11 | ucs-print -i
```

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-info

Display information from header of UCS data set file

SYNOPSIS

```
ucs-info [-s [-v]] [-l] data.ds.gz
```

DESCRIPTION

This small utility displays information from the header of a data set file (comment lines and global variables).

With the **-size** (or **-s**) option, the actual size of the data set (i.e. the number of pair types) is also determined, which may be different from the size reported in the header. Note that this operation has to read the entire data set file and may take some time for larger data sets (use **-verbose** or **-v** to show progress information).

With the **-list** (or **-l**) option, the data set variables are listed together with their data types and optional comments.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ ucs-print

ASCII-format UCS data set for viewing and printing

SYNOPSIS

```
ucs-print [-i] [-p <lines>] [-d <digits>] data.ds.gz
```

```
ucs-print [-o <file>] [-ps [-2] [-1]] [-p <lines>] [-d <digits>] data.ds.gz
```

```
ucs-print [<options>] '*' 'am.%.pv' FROM data.ds.gz
```

DESCRIPTION

Format data set as ASCII table for inclusion in text files, on-line viewing (in a terminal window, with `-interactive` option), and printing (in PostScript format, with `-postscript` option). The **ucs-print** utility automatically adjusts column widths and chooses an appropriate format for floating-point numbers. Boolean attributes are displayed as yes and no, while missing values are shown as NA.

In the first forms of the command (used in the first two examples above), all variables are displayed (which usually results in a very wide table). The name of the data set may be omitted, in which case data is read from STDIN.

In the second form, variables can be selected with a whitespace-separated list of UCS wildcard patterns (see the `ucsexp` manpage) or by explicitly specifying the variable names. This feature can also be used to re-order the columns or display a variable in multiple columns. The FROM clause is mandatory in this mode, but data can be read from STDIN by using `-` as the name of the data set.

Note that there may be some delay while the data set is read into memory and analysed, especially without the `-pagesize` option.

OPTIONS

- `-help, -h`
Prints short usage reminder.
- `-verbose, -v`
Prints some (minimal) progress information on STDERR.

- `-output file, -o file`

Write output to *file*, rather than printing it on `STDOUT`.

- `-postscript, -ps`

Uses the **a2ps** program (see the `a2ps(1)` manpage) to create a PostScript version of the formatted table for printing. By default, the PostScript code will be shown on `STDOUT` (and *not* be sent to a printer). It can be saved into a file with the `-output` option. If the `-pagesize` option is used, each page will contain the specified number of rows and the table will be truncated if it is too wide. If this happens, try increasing the number of rows on the page or use `-landscape`. If the table still fails to fit, split the variables into two or more groups that are printed separately.

- `-landscape, -l`

[In `-postscript` mode only.] Print pages in landscape orientation rather than portrait. Especially useful for wide tables.

- `-two-up, -2`

[In `-postscript` mode only.] Print two pages on a single sheet, same as the `-2` option in **a2ps**. This option may give a more satisfactory result for very narrow tables (e.g. when showing only the pair types).

- `-interactive, -i`

Send output to terminal pager (**less**) for interactive viewing. This option may not be used together with `-output`. The data will automatically be displayed in paged mode, with the page size adjusted to the height of the terminal window. If the screen size cannot be automatically determined, use the `-pagesize` option to activate paging explicitly. The page size should be set to the screen height (number of text lines) minus 4 for optimal results. Use `-p 0` to deactivate paging in interactive mode.

- `-pagesize n, -p n`

Split data set into smaller tables of (up to) *n* rows each, which are separated by blank lines. Use of this option may improve the formatting quality, helps to avoid excessive columns widths, and reduces the delay before (partial) results can be displayed (especially for large data sets). By default, the entire data set is formatted as a single large table (unless `-interactive` was specified).

- `-digits n, -d n`

Display floating-point numbers with a precision of approximately *n* significant digits. The actual number of digits shown may differ slightly when a fixed-point format is chosen by the formatter. The default is *n* = 8.

BUGS

The code used to determine the screen height in `-interactive` mode may not work on some platforms. It has only been tested under Linux so far. If you are using the **bash** shell, you might try `export LINES` before running the **ucs-print** tool.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

B.1.3 UCS/Perl Modules

■ UCS

Core library

SYNOPSIS

```

use UCS;

$UCS::Version;           # UCS version
$UCS::Copyright;        # UCS copyright string
$UCS::BaseDir;          # base directory of UCS system
$UCS::PerlDir;          # base directory of UCS/Perl

UCS::Die("Msg line 1", "Msg line 2", ...); # really die (even in Tk loop)
UCS::Warn("Msg line 1", "Msg line 2", ...); # warning message (may be caught by Tk)
UCS::Status("Message"); # display status message in Tk window
UCS::Splash();          # splash screen (may be shown during start-up)
UCS::Verbose = 0;       # suppress warnings
@unique_values = UCS::Unique(@list);       # remove duplicates from list

@vars = (@UCS::CoreVars, @UCS::DerivedVars); # standard variable names (core and derived)
@matches = UCS::Match($pattern, @names);    # match variable names
$ok = UCS::ValidKey($key);                  # valid identifier, e.g as AM key
$ok = UCS::ValidName($name);               # whether variable name is valid
$type = UCS::VarType($name);               # "BOOL", "INT", "DOUBLE", "STRING"
($spec, $key) = UCS::SplitName($name);     # split am.*, r.*, or user-defined variable name

@registered_AMs = UCS::AM_Keys();           # keys for built-in AMs (when loaded)
if (UCS::AM($key)) {
    $full_name = UCS::AM_Name($key);        # long descriptive name
    $description = UCS::AM_Description($key); # optional multi-line text
    $exp = UCS::AM_Expression($key);       # AM equation as compiled UCS expression
    $score = $exp->eval({f=>$f, fl=>$fl, ...}); # use UCS::Expression methods to evaluate AM
}
$score = UCS::Eval_AM($key, $arghash);     # convenient but slow

UCS::Load_AM_Package("HTest", ...);       # load built-in AM packages

$ok = UCS::Register_AM                      # register new association measure
    "t-score",                               # AM key (-> variables am.tscore and r.tscore)
    "t-score measure (Church et. al. 1991)", # long descriptive name
    '%011% - %E11%' / sqrt(%011%)',         # UCS expression (will be compiled into UCS::Expression)
    $multiline_text;                         # optional multi-line description of AM

```

DESCRIPTION

This UCS core library maintains a list of **built-in AMs** and Perl subroutines for computing their **scores** from a candidate's signatures. Utility functions perform syntax checks for **field names**, determine **field types** from the naming conventions, and **match patterns** containing UCS wildcards against field names.

CONFIGURATION VARIABLES**\$UCS::Version;**

The currently installed UCS version.

\$UCS::Copyright;

A copyright string for the UCS system. Will be displayed by some UCS/Perl scripts.

\$UCS::BaseDir;

The base directory of the UCS System installation. Compiled UCS **programs** and links to Perl scripts are installed in *\$UCS::BaseDir/bin/*, while the components of **UCS/R** can be found in *\$UCS::BaseDir/R/*.

\$UCS::PerlDir;

The base directory of the **UCS/Perl** installation. The UCS Perl modules are installed in *\$UCS::PerlDir/lib/* and its subdirectories, Perl scripts in *\$UCS::PerlDir/bin/*.

GENERAL FUNCTIONS**UCS::Die(\$message, ...);**

"Safe" replacement for Perl's built-in **die** function, which will even exit properly from a Perl/Tk loop. One or more lines of error messages are printed on STDERR (or shown in some other suitable manner).

UCS::Warn(\$message, ...);

By default, prints one or more lines of warning/error messages on STDERR like **UCS::Die**, but does not exit the script. The purpose of this replacement for the built-in **warn** function is to allow warnings to be caught and displayed in a Perl/Tk user interface. Warnings might also be redirected to a log file.

UCS::Status(\$message);

Displays a status message in a Perl/Tk interface. By default, *\$message* is appended to any previous messages. When *\$message* ends in a newline character (`\n`), the next call to **UCS::Status** will replace the current message; when it ends in a carriage return (`\r`), the next call will overwrite the current message from the start. (This is the usual effect of **printing** such control characters, and will be simulated in Perl/Tk interfaces).

UCS::Splash();

Displays a UCS splash screen with UCS version information and copyright, e.g. during the start-up phase of a larger UCS/Perl script.

\$UCS::Verbose = 0;

The variable *\$UCS::Verbose* controls whether status messages and warnings are printed on STDOUT and STDERR, respectively. Verbose output is enabled by default, and can be suppressed by setting *\$UCS::Verbose* to 0.

@unique_values = UCS::Unique(@list);

Removes duplicate values from *@list* and returns the remaining elements in the original order. Useful to avoid repetitions of variable names etc.

MANIPULATING VARIABLE NAMES

\$std_vars = (@UCS::CoreVars, @UCS::DerivedVars);

Names of **core** and **derived variables**.

\$ok = UCS::ValidKey(\$key);

Returns true iff *\$key* is a valid UCS identifier, which may be used as an AM key or in the name of a user-defined variable.

\$ok = UCS::ValidName(\$name);

Returns true iff *\$name* is a valid UCS variable name, i.e. either a standard variable (core or derived) , an association score or ranking, or a user-defined variable. See *ucsf*file for details on the UCS naming conventions.

\$type = UCS::VarType(\$name);

Determines the data type of a variable from its name *\$name*, according to the UCS naming conventions. Possible data types are **BOOL** (Boolean, 0/1), **INT** (signed integer), **DOUBLE** (double-precision floating-point), and **STRING** (string value).

(\$spec, \$key) = UCS::SplitName(\$name);

Splits the variable name *\$name* of an association score, ranking, or user-defined variable into the specifier *\$spec* and the key *\$key*. *\$spec* will be one of *am*, *r*, *b*, *f*, *n*, or *x*. If *\$name* is invalid or the name of a standard variable, (*undef*, *\$name*) is returned.

@matches = UCS::Match(\$pattern, @names);

Extract strings from *@names* that match the UCS **wildcard pattern** *\$pattern*. The pattern may contain literal characters **A-Z a-z 0-9 .** and the wildcards **?, *,** and **%**.

? ... arbitrary character
 * ... arbitrary substring without "."
 % ... arbitrary string

Thus, the pattern **%** selects all field names, ***** selects the names of core and derived fields, **am.%** all AM scores, etc. See *ucsexp* for more examples.

ASSOCIATION MEASURE REGISTRY

This **registry** maintains a list of association measures, which are automatically available to all UCS/Perl scripts. Association measures are identified by their **key**, which must be a valid UCS identifier. Association scores for a measure with the key *fisher*, for instance, will be stored in the variable *am.fisher*, and the corresponding rankings in the variable *r.fisher*. A wide range of predefined association measures can be imported from the **UCS::AM** module and several add-on packages (see the **UCS::AM** manpage).

@registered_AMs = UCS::AM_Keys();

The **UCS::AM_Keys** function returns the keys of all currently registered association measures as an unordered list. (Note that no association measures are defined unless **UCS::AM** and/or the add-on packages have been imported.)

\$ok = UCS::AM(\$key);

Returns true if an association measure is registered under *\$key*.

\$full_name = UCS::AM_Name(\$key);

Returns a long and descriptive name for the association measure identified by *\$key*. This name should be suitable for presentation to the user in a selection dialogue.

\$description = UCS::AM_Description(\$key);

An optional lengthy description of the association measure identified by *\$key*. *\$description* is a single string but will usually contain linebreaks (`\n`), which may need to be removed for automatic justification (e.g. in a Perl/Tk interface).

\$exp = UCS::AM_Expression(\$key);

Returns the equation of the association measure *\$key*, compiled into a **UCS::Expression** object. Call the **eval** or **evalloop** method of *\$exp* to compute association scores (see *UCS::Expression*). The sourcecode of this expression can be retrieved with the **string** method (which is especially useful for built-in association measures).

\$score = UCS::Eval_AM(\$key, \$arghash);

The **UCS::Eval_AM** function is a convenient and shorter alternative, and is equivalent to:

```
$exp = UCS::AM_Expression($key);
$score = $exp->eval($arghash);
```

It incurs considerable overhead when association scores are calculated for multiple pair types (because of the repeated lookup of *\$key* in the AM registry), and should be avoided in tight loops. (See *UCS::Expression* for some comments on efficiency.)

@packages = UCS::Load_AM_Package(\$name, ...);

Load one or more of the built-in AM packages as specified by the function arguments. *\$name* must match the last part of the corresponding module name, e.g. 'HTest' to load the **UCS::AM::HTest** package. *\$name* is case-insensitive and may be abbreviated to a unique prefix. The special name 'ALL' (or 'all') loads all available add-on packages, while the empty string '' loads the basic measures from **UCS::AM**. **UCS::Load_AM_Package** returns a list containing the full names of all loaded packages (with duplicates removed). If there is no match for *\$name*, an empty list is returned.

\$ok = UCS::Register_AM(\$key, \$name, \$equation [, \$description]);

The **UCS::Register_AM** function is used to register a new association measure, or overwrite an existing one with a new definition. *\$key* is the identification key of the new measure, *\$name* a descriptive name, *\$equation* the measure's equation in the form of an (uncompiled) UCS expression, and *\$description* an optional multi-line description. *\$equation* may also be an object of class **UCS::Expression** (which is cloned rather than re-compiled), enabling the use of advanced features such as parametric expressions.

The function call returns true if the new measure has been successfully registered. A false return value indicates that compilation of *\$equation* into an **UCS::Expression** object failed. The **UCS::Register_AM** function will **die** if *\$key* is not a valid UCS identifier.

The example below shows the code used to register the **t-score** measure (Church *et. al.* 1991) which has been widely used in English lexicography.

```

$ok = UCS::Register_AM "tscore",
    "t-score measure (Church et. al. 1991)",
    '(%O11% - %E11%) / sqrt(%O11%)',
    "The t-score measure applies Student's t-test to ...";
die "Syntax error in UCS expression for t-score measure"
    unless $ok;

```

SEE ALSO

Type `ucsd doc ucshintro` for an introduction to UCS/Perl and an overview of its components (in the `MODULES` and `PROGRAMS` sections).

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::File

File access utilities

SYNOPSIS

```

use UCS::File;

## open filehandle for reading or writing
# automagically compresses/decompresses files and dies on error
$fh = UCS::File::Open("> my_file.gz");
# the same without error checks (may return undefined value)
$fh = UCS::File::TryOpen("> my_file.bz2");

## temporary file objects (disk files are automatically removed)
$t1 = new UCS::File::Temp;           # picks a unique filename
$t2 = new UCS::File::Temp "mytemp";  # extends prefix to unique name
$t3 = new UCS::File::Temp "mytemp.gz"; # compressed temporary file
$filename = $t1->name;               # full pathname of temporary file
$t1->write(...);                    # works like $fh->print() ;
$t1->finish;                         # stop writing file
print $t1->status, "\n";             # WRITING/FINISHED/READING/DELETED
# main program can read or overwrite file <$filename> now
$line = $t1->read;                   # read one line (like $fh->getline())
$t1->rewind;                          # re-read from beginning of file
$line = $t1->read;                   # (reads first line again)
$t1->close;                           # stop reading and remove temporary file
# other files will be removed when objects $t2 and $t3 are destroyed

## execute shell command with error detection
$cmd = "ls -l";

```



```

$errlevel = UCS::File::ShellCmd($cmd); # dies with error message if not ok
$UCS::File::Paranoid = 1;      # more paranoid checks (-1 for less paranoid)
# $errlevel == 0 (ok), 1 (minor problems), ..., 6 (fatal error)

UCS::File::ShellCmd($cmd, \@lines); # capture standard output in array
UCS::File::ShellCmd($cmd, "file.txt"); # ... or in file (for large amounts of data)
UCS::File::ShellCmd(["ls", "-l", @files], \@lines); # bypass shell expansion

```

DESCRIPTION

This module provides some useful routines for handling files and external programs. This includes **opening files** with error checks and automagical compression/decompression, **temporary file objects** that are automatically created and deleted, and the execution of **shell commands** with extensive error checks.

OPENING FILES

\$fh = UCS::File::Open(\$name);

Open file *\$name* for reading, writing, or appending. Returns **FileHandle** object if successful, otherwise it **dies** with an error message. It is thus never necessary to check whether *\$fh* is defined.

If *\$name* starts with >, the file is opened for writing (an existing file will be overwritten). If *\$name* starts with >>, the file is opened for appending.

Files with the extensions .Z, .gz, and .bz2 are automagically compressed and decompressed, provided that the necessary tools are installed. It is also possible to append to .gz and .bz2 files.

Note that *\$name* may also be a read or write pipe ("... |" or "| ...", respectively), which is passed directly to the built-in **open** command. It is thus subject to shell expansion and does not support automagic compression and decompression.

\$fh = UCS::File::TryOpen(\$name);

Same as **UCS::File::Open**, but without the error checks. Returns **undef** if the **open()** call fails.

TEMPORARY FILES

Temporary files (implemented by **UCS::File::Temp** objects) are assigned a unique name and are automatically deleted when the script exits. The life cycle of a temporary file consists of four stages: **create**, **write**, **read** (possibly **re-read**), **delete**. This cycle corresponds to the following method calls:

```

$tf = new UCS::File::Temp; # create new temporary file in /tmp dir
$tf->write(...);          # write cycle (buffered output, like print function)
$tf->finish;               # complete write cycle (flushes buffer)
$line = $tf->read;         # read cycle (like getline method for FileHandle)
[$tf->rewind;              # optional: start re-reading temporary file ]
[$line = $tf->read;        ]
$tf->close;                # delete temporary file

```

Once the temporary file has been read from, it cannot be re-written; a new `UCS::File::Temp` object has to be created for the next cycle. When the write stage is completed (but before reading has started, i.e. after calling the `finish` method), the temporary file can be accessed and/or overwritten by external programs. Use the `name` method to obtain its full pathname. If no direct access to the temporary file is required, the `finish` method is optional. The write cycle will automatically be completed before the first `read` method call.

\$tf = new UCS::File::Temp [\$prefix ;]

Creates temporary file in `/tmp` directory. If the optional argument `$prefix` is specified, the filename will begin with `$prefix` and be extended to a unique name. If `$prefix` contains a `/` character, it is interpreted as an absolute or relative path, and the temporary file will not be created in the `/tmp` directory. To create a temporary file in the current working directory, use `./MyPrefix`.

You can add the extension `.Z`, `.gz`, or `.bz2` to `$prefix` in order to create a compressed temporary file. The actual filename (as returned by the `name` method) will have the same extension in this case.

The temporary file is immediately created and opened for writing.

\$filename = \$tf->name;

Returns the real filename of the temporary file. **NB:** direct access to this file (e.g. by external programs) is only allowed after calling `finish`, and before the first `read`.

\$tf->write(...);

Write data to the temporary file. All arguments are passed to Perl's built-in `print` function. Like `print`, this method does not automatically add newlines to its arguments.

\$tf->finish;

Stop writing to the temporary file, flush the output buffer, and close the associated file handle. After `finish` has been called, the temporary file can be accessed directly by the script or external programs, and may also be overwritten. In order to delete a file created by an external program automatically, `finish` the temporary file immediately after its creation and then allow the external tool to overwrite it:

```
$tf = new UCS::File::Temp;
$tf->finish; # temporary file has size of 0 bytes now
$filename = $tf->name;
system "$my_shell_command > $filename";
```

\$line = \$tf->read;

Read one line from temporary file (same as calling `getline` on a `FileHandle` object). Automatically invokes `finish` if called during write cycle.

\$tf->rewind;

Allows re-reading of the temporary file. The next `read` call will return the first line of the temporary file. Internally this is achieved by closing and re-opening the associated file handle.

\$tf->close;

Closes any open file handles and deletes the temporary file. This will be done automatically when the `UCS::File::Temp` object is destroyed. Use `close` to free disk space immediately.

SHELL COMMANDS

The `UCS::File::ShellCmd` function provides a convenient replacement for the built-in `system` command. Standard output and error messages produced by the invoked shell command are captured to avoid screen clutter. The collected standard output of the command can optionally be returned to the caller (similar to the backtick operator `'$shell_cmd'`). `UCS::File::ShellCmd` also checks for a variety of error conditions and returns an error level ranging from 0 (successful) to 6 (fatal error):

Error Level	Description
6	command execution failed (system error)
5	non-zero exit value or error message on STDERR
4	-- reserved for future use --
3	warning message on STDERR
2	any output on STDERR
1	error message on STDOUT

Depending on the value of `$UCS::File::Paranoid` and the error level, a warning message may be issued or the function may **die** with an error message.

`$UCS::File::Paranoid = 0;`

With the default setting of 0, `UCS::File::ShellCmd` will **die** if the error level is 5 or greater. In the **extra paranoid** setting (+1), it will almost always **die** (error level 2 or greater). In the **less paranoid** setting (-1) only an error level of 6 (i.e. failure to execute the shell command) will cause the script to abort.

`$errlvl = UCS::File::ShellCmd($cmd);`

`$errlvl = UCS::File::ShellCmd($cmd, $filename);`

`$errlvl = UCS::File::ShellCmd($cmd, \@lines);`

The first form executes `$cmd` as a shell command (through the built-in `system` function) and returns an error level. With the default setting of `$UCS::File::Paranoid`, serious errors are usually detected and cause the script to **die**, so it is not necessary to check the value of `$errlvl`.

The second form stores the standard output of the shell command in a file named `$filename`, where it can then be processed with external programs or read in by the Perl script. **NB:** Compressed files are not supported! It is recommended to use an uncompressed temporary file (`UCS::File::Temp` object).

The third form takes an array reference as its second argument, splits the standard output of `$cmd` into **chomped** lines and stores them in the array `@lines`. If there is a large amount of standard output, it is more efficient to use the second form.

Note that `$cmd` is passed to the shell for metacharacter expansion. In order to avoid this (e.g. when filename arguments may contain blanks), specify an array reference of the form `[$program, @args]` instead:

```
$errlvl = UCS::File::ShellCmd(["ls", "-l", @files], \@lines);
```

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::R

UCS/Perl interface to R

SYNOPSIS

```
use UCS::R;

UCS::R::Start();           # start R backend explicitly
UCS::R::Stop();           # terminate R backend (if possible)

@x = UCS::R::Exec($cmd);   # execute R cmd (must return numeric vector)

UCS::R::LoadVector("my.x", \@data); # load numeric vector efficiently into R
$data = UCS::R::DumpVector("my.x"); # returns arrayref

# access to special functions and statistical distributions
# through the UCS::SFunc module
```

DESCRIPTION

The **UCS::R** module provides an interface to the **R** statistical environment and the **UCS/R** libraries on an R interpreter running in the background. When available (as determined by the installation script), the **RSPerl** interface is used for efficient communication with the R interpreter. Otherwise, the system falls back on a slower but more portable solution that simulates an interactive R session through use of the **Expect** module. See the **UCS::R::RSPerl** and **UCS::R::Expect** manpages for some details about the strengths and limitations of the two backends.

The **UCS::R** interface is mainly used by the **UCS::SFunc** module to make the R implementations of **special functions** (binomial coefficients, Gamma function, Beta function) and **statistical distributions** (binomial, Poisson, normal, chi-squared, hypergeometric) available to **UCS/Perl**, without relying on an external maths library and/or compiled C code.

FUNCTIONS

UCS::R::Start();

Starts the **R** interpreter. Normally, this function does not have to be called explicitly, as the backend is automatically launched when an R command is executed for the first time. Since this will block program execution for a few seconds, some scripts may prefer to call **UCS::R::Start** at start-up time before the R process is actually needed.

UCS::R::Stop();

Terminate the **R** interpreter. Normally, this function does not have to be called explicitly, but it may be used to shut down an R process that is no longer needed and free memory resources. Note that this function is not supported by the **UCS::R::RSPerl** backend and will be silently ignored.

@x = UCS::R::Exec(\$cmd);

Executes the **R** command *\$cmd* in the server process. The command must return a vector, which is passed back to the calling script in the form of a list *@x*. When command execution fails or its return value cannot be parsed, the **UCS::R::Exec** function will **die** with an error message.

At the moment, only numeric vectors are guaranteed to work (although the **UCS::R::RSPerl** backend should support all types of vectors). It is safe to execute any command when **UCS::R::Exec** is called in void context. When using the **UCS::R::Expect** backend, complex return values should be made **invisible** for reasons of speed and robustness.

NB: This interface is not efficient for exchanging large amounts of data with R and may hang if the input/output buffers overflow. Use the **LoadVector** and **DumpVector** functions for this purpose (see below). Moreover, *\$cmd* must be a single-line command (separate multiple commands with `;`), so that it leaves a single command prompt at the beginning of a line after execution. Avoid `cat()` and any functions that prompt for user input, otherwise **UCS::R::Exec** will become confused and may hang.

UCS::R::LoadVector(\$varname, \@data);

Efficiently loads a numeric vector into **R** (making use of a temporary file and the `scan` function in R). The data *@data* are passed in as an array reference and will be stored in the **R** variable *\$varname*.

\$data = UCS::R::DumpVector(\$varname);

Efficiently reads a numeric vector from **R** (making use of a temporary file and the `write()` function). The data stored in the **R** variable *\$varname* (which must be a numeric vector) are returned as an anonymous array reference *\$data*.

SPECIAL FUNCTIONS AND STATISTICAL DISTRIBUTIONS

The special functions and statistical distributions provided through the **R** interface are not exported by this module. Use **UCS::SFunc** instead. All available functions are documented in the **UCS::SFunc** manpage. They are available under the same names in the **UCS::R** package. For instance, the R implementation of the **lgamma** function can be accessed explicitly as **UCS::R::lgamma**.

COPYRIGHT

Copyright 2004-2005 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::R::Expect

Expect-based implementation of R backend

SYNOPSIS

```
use UCS::R::Expect;
## exports Start(), Stop() and Exec() functions into current namespace
## as well as LoadVector() and DumpVector()
```

DESCRIPTION

This module should *only* be used implicitly through **UCS::R**, which loads the more efficient **UCS::R::RSPerl** implementation if available, and falls back on **UCS::R::Expect** otherwise.

LIMITATIONS

This module starts an R process in the background and communicates with it interactively through the **Expect** module. This approach has several disadvantages:

- Invoking R commands, waiting for output from the R backend, and parsing that output causes substantial overhead for R function invocations, allowing less than 1000 invocations per second even on a fast machine.
- The return value of a function call has to be printed by R, then the resulting output has to be parsed by Perl. This interfacing method is rather frail and currently supports only numeric vectors as return values.
- The interface is extremely inefficient for exchanging large amounts of data between Perl and R. It may hang if the input/output buffers used by **Expect** overflow. Use the **LoadVector** and **DumpVector** functions to pass large numeric vectors to R and back.

Because of these limitations, it is highly recommended that you install and use the **RSPerl** interface (available from <http://www.omegahat.org/>) on Unix platforms. When **RSPerl** has been installed with support for calling R from Perl, it will automatically be detected and configured for use by the UCS installation script. See *doc/install.txt* for more information and installation tips.

COPYRIGHT

Copyright (C) 2004-2005 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::R::RSPerl

RSPerl-based implementation of R backend

SYNOPSIS

```
use UCS::R::RSPerl;
## exports Start(), Stop() and Exec() functions into current namespace
## as well as LoadVector() and DumpVector()
```

DESCRIPTION

This module should *only* be used implicitly through **UCS::R**, which loads the **UCS::R::RSPerl** implementation if available, and falls back on the inefficient **UCS::R::Expect** implementation otherwise.

Note that use `UCS::R::RSPerl` will fail if RSPerl support is not available, causing the compilation of the Perl script to abort.

COPYRIGHT

Copyright (C) 2004-2005 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::SFunc

Special functions and statistical distributions

SYNOPSIS

```
use UCS::SFunc;

# special functions (all logarithms are base 10)
$c      = choose($n, $k);      # binomial coefficient
$log_c  = lchoose($n, $k);

$y      = gamma($a);          # Gamma function
$log_y  = lgamma($a);

$y      = igamma($a, $x [, $upper]); # incomplete Gamma functions
$log_y  = ligamma($a, $x [, $upper]);
$y      = rgamma($a, $x [, $upper]); # regularised Gamma functions
$log_y  = lrgamma($a, $x [, $upper]);

$x      = igamma_inv($a, $y [, $upper]); # inverse Gamma functions
$x      = ligamma_inv($a, $log_y [, $upper]);
$x      = rgamma_inv($a, $y [, $upper]);
$x      = lrgamma_inv($a, $log_y [, $upper]);

$y      = beta($a, $b);       # Beta function
$log_y  = lbeta($a, $b);

$y      = ibeta($x, $a, $b);  # incomplete Beta function
$log_y  = libeta($x, $a, $b);
$y      = rbeta($x, $a, $b);  # regularised Beta function
```

```

$log_y = lrbeta($x, $a, $b);
$x      = ibeta_inv($y, $a, $b);          # inverse Beta functions
$x      = libeta_inv($log_y, $a, $b);
$x      = rbeta_inv($y, $a, $b);
$x      = lrbeta_inv($log_y, $a, $b);

# binomial distribution (density, tail probabilities, quantiles)
$d = dbinom($k, $size, $prob);
$ld = ldbinom($k, $size, $prob);
$p = pbinom($k, $size, $prob [, $upper]);
$lp = lpbinom($k, $size, $prob [, $upper]);
$k = qbinom($p, $size, $prob [, $upper]);
$k = lqbinom($lp, $size, $prob [, $upper]);

# Poisson distribution (density, tail probabilities, quantiles)
$d = dpois($k, $lambda);
$ld = ldpois($k, $lambda);
$p = ppois($k, $lambda [, $upper]);
$lp = lppois($k, $lambda [, $upper]);
$k = qpois($p, $lambda [, $upper]);
$k = lqpois($lp, $lambda [, $upper]);

# normal distribution (density, tail probabilities, quantiles)
$d = dnorm($x, $mu, $sigma);
$ld = ldnorm($x, $mu, $sigma);
$p = pnorm($x, $mu, $sigma [, $upper]);
$lp = lpnorm($x, $mu, $sigma [, $upper]);
$x = qnorm($p, $mu, $sigma [, $upper]);
$x = lqnorm($lp, $mu, $sigma [, $upper]);

# chi-squared distribution (density, tail probabilities, quantiles)
$d = dchisq($x, $df);
$ld = ldchisq($x, $df);
$p = pchisq($x, $df [, $upper]);
$lp = lpchisq($x, $df [, $upper]);
$x = qchisq($p, $df [, $upper]);
$x = lqchisq($lp, $df [, $upper]);

# hypergeometric distribution (density and tail probabilities)
$d = dhyper($k, $R1, $R2, $C1, $C2);
$ld = ldhyper($k, $R1, $R2, $C1, $C2);
$p = phyper($k, $R1, $R2, $C1, $C2 [, $upper]);
$lp = lphyper($k, $R1, $R2, $C1, $C2 [, $upper]);

```

DESCRIPTION

This module provides **special functions** and common **statistical distributions**. Currently, all functions are imported from the UCS/R system (using the UCS::R interface).

SPECIAL FUNCTIONS

UCS::SFunc currently provides the following special mathematical functions: **binomial coefficients**, the **Gamma function**, the **incomplete Gamma functions** and their inverses, the **regularised Gamma functions** and their inverses, the **Beta function**, the **incomplete Beta**

function and its inverse, and the **regularised Beta function** and its inverse. Note that all logarithmic versions return **base 10** logarithms!

`$coef = choose($n, $k);`

`$log_coef = lchoose($n, $k);`

The **binomial coefficient** " n over k ", and its logarithm.

`$y = gamma($a);`

`$log_y = lgamma($a);`

The (complete) **Gamma function** with argument a , and its logarithm. Note that the factorial $n!$ is equal to $\text{gamma}(n+1)$.

`$y = igamma($a, $x [, $upper]);]`

`$log_y = ligamma($a, $x [, $upper]);]`

The **incomplete Gamma function** with arguments a and x , and its logarithm. If $upper$ is specified and true, the upper incomplete Gamma function is computed, otherwise the lower incomplete Gamma function. It is recommended to set $upper$ to the string constant 'upper' as a reminder of its function.

`$x = igamma_inv($a, $y [, $upper]);]`

`$x = ligamma_inv($a, $log_y [, $upper]);]`

The **inverse of the incomplete Gamma function**, as well as the inverse of its logarithm.

`$y = rgamma($a, $x [, $upper]);]`

`$log_y = lrgamma($a, $x [, $upper]);]`

The **regularised Gamma function** with arguments a and x , and its logarithm. If $upper$ is specified and true, the upper regularised Gamma function is computed, otherwise the lower regularised Gamma function. It is recommended to set $upper$ to the string constant 'upper' as a reminder of its function.

`$x = rgamma_inv($a, $y [, $upper]);]`

`$x = lrgamma_inv($a, $log_y [, $upper]);]`

The **inverse of the regularised Gamma function**, as well as the inverse of its logarithm.

`$beta = beta($a, $b);`

`$log_beta = lbeta($a, $b);`

The (complete) **Beta function** with arguments a and b , and its logarithm.

`$y = ibeta($x, $a, $b);`

`$log_y = libeta($x, $a, $b);`

The **incomplete Beta function** with arguments x , a , and b , and its logarithm.

`$x = ibeta_inv($y, $a, $b);`

`$x = libeta_inv($log_y, $a, $b);`

The **inverse** of the **incomplete Beta function**, as well as the inverse of its logarithm.

`$y = rbeta($x, $a, $b);`

`$log_y = lrbeta($x, $a, $b);`

The **regularised Beta function** with arguments x , a , and b , and its logarithm.

`$x = rbeta_inv($y, $a, $b);`

`$x = lrbeta_inv($log_y, $a, $b);`

The **inverse** of the **regularised Beta function**, as well as the inverse of its logarithm.

STATISTICAL DISTRIBUTIONS

UCS::SFunc computes **densities**, **tail probabilities** (= distribution function), and **quantiles** for the following statistical distributions: **binomial** distribution, **Poisson** distribution, **normal** distribution, **chi-squared** distribution, **hypergeometric** distribution. The function names are the common abbreviations as used e.g. in the **R** language, with additional logarithmic versions (that start with the letter **l**) (these correspond to the `log=TRUE` and `log.p=TRUE` parameters in **R**).

Note that logarithmic probabilities are always given as **negative base 10** logarithms. The logarithmic density and tail probability functions return such logarithmic p-values, and the quantile functions expect them in their first argument.

The Binomial Distribution Binomial distribution with parameters $size$ (= number of trials) and $prob$ (= success probability in single trial). $E[X] = size * prob$, $V[X] = size * prob * (1 - prob)$.

`$d = dbinom($k, $size, $prob);`

`$ld = ldbinom($k, $size, $prob);`

Density $P(X = k)$ and its negative base 10 logarithm.

`$p = pbinom($k, $size, $prob [, $upper]);`

`$lp = lpbinom($k, $size, $prob [, $upper]);`

Tail probabilities $P(X \leq k)$ and $P(X > k)$ (if $upper$ is specified and true), and their negative base 10 logarithms. It is recommended to set $upper$ to the string 'upper' as a reminder of its meaning.

The **R** implementation of binomial tail probabilities underflows for very small probabilities (even in the logarithmic version), as of **R** version 2.1. Therefore, these functions use a mixture of **R** and Perl code to compute upper tail probabilities for large samples (which are most likely to lead to underflow problems for cooccurrence data).

`$k = qbinom($p, $size, $prob [, $upper]);`

k = lqbinom(lp , $size$, $prob$ [, $upper$]);

Lower and upper quantiles. The lower quantile is the smallest value k with $P(X \leq k) \geq p$. The upper quantile (which is computed when $upper$ is specified and true) is the largest value k with $P(X > k) \geq p$. In the logarithmic version, lp must be the negative base 10 logarithm of the desired p-value.

Note that these functions use the R implementation directly without a workaround for undeflow problems. The quantiles returned for very small p-values (especially when using **lqbinom**) are therefore unreliable and should be used with caution.

The Poisson Distribution Poisson distribution with parameter λ (= expectation); $E[X] = V[X] = \lambda$.

d = dpois(k , λ);

ld = ldpois(k , λ);

Density $P(X = k)$ and its negative base 10 logarithm.

p = ppois(k , λ [, $upper$]);

lp = lppois(k , λ [, $upper$]);

Tail probabilities $P(X \leq k)$ and $P(X > k)$ (if $upper$ is specified and true), and their negative base 10 logarithms. It is recommended to set $upper$ to the string 'upper' as a reminder of its meaning.

k = qpois(p , λ [, $upper$]);

k = lqpois(lp , λ [, $upper$]);

Lower and upper quantiles. The lower quantile is the smallest value k with $P(X \leq k) \geq p$. The upper quantile (which is computed when $upper$ is specified and true) is the largest value k with $P(X > k) \geq p$. In the logarithmic version, lp must be the negative base 10 logarithm of the desired p-value.

The Normal Distribution Normal distribution with parameters μ (= expectation) and σ (= standard deviation). Unspecified parameters default to $\mu = 0$ and $\sigma = 1$. $E[X] = \mu$, $V[X] = \sigma^2$.

d = dnorm(x , μ , σ);

ld = ldnorm(x , μ , σ);

Density $P(X = x)$ and its negative base 10 logarithm.

p = pnorm(x , μ , σ [, $upper$]);

lp = lpnorm(x , μ , σ [, $upper$]);

Tail probabilities $P(X \leq x)$ and $P(X > x)$ (if $upper$ is specified and true), and their negative base 10 logarithms. It is recommended to set $upper$ to the string 'upper' as a reminder of its meaning.

x = qnorm(p , μ , σ [, $upper$]);

\$x = lqnorm(\$lp, \$mu, \$sigma [, \$upper]);

Lower and upper quantiles. The lower quantile is the smallest value x with $P(X \leq x) \geq p$. The upper quantile (which is computed when *\$upper* is specified and true) is the largest value x with $P(X \geq x) \geq p$. In the logarithmic version, *\$lp* must be the negative base 10 logarithm of the desired p-value.

The Chi-Squared Distribution Chi-squared distribution with parameter *\$df* (= degrees of freedom); $E[X] = df$, $V[X] = 2 * df$.

\$d = dchisq(\$x, \$df);

\$ld = ldchisq(\$x, \$df);

Density function $f(x)$ and its negative base 10 logarithm.

\$p = pchisq(\$x, \$df [, \$upper]);

\$lp = lpchisq(\$x, \$df [, \$upper]);

Tail probabilities $P(X \leq x)$ and $P(X \geq x)$ (if *\$upper* is specified and true), and their negative base 10 logarithms. It is recommended to set *\$upper* to the string 'upper' as a reminder of its meaning.

\$x = qchisq(\$p, \$df [, \$upper]);

\$x = lqchisq(\$lp, \$df [, \$upper);

Lower and upper quantiles. The lower quantile is the smallest value x with $P(X \leq x) \geq p$. The upper quantile (which is computed when *\$upper* is specified and true) is the largest value x with $P(X \geq x) \geq p$. In the logarithmic version, *\$lp* must be the negative base 10 logarithm of the desired p-value.

The Hypergeometric Distribution Hypergeometric distribution of the upper left-hand corner X in a 2×2 contingency table with fixed marginals $R1$, $R2$, $C1$, and $C2$, where both $R1 + R2$ and $C1 + C2$ must sum to the sample size N . k represents the observed value of X and must be in the admissible range $\max(0, R1 - C2) \leq k \leq \min(R1, C1)$, otherwise the density will be given as 0 and tail probabilities as 1 or 0, respectively. $E[X] = R1 * C1 / N$, $V[X] = R1 * R2 * C1 * C2 / (N^2 * (N-1))$.

For R versions before 2.0, the upper tail probabilities are computed with a mixture of R and Perl code to circumvent a cancellation problem in the R implementation and achieve better precision. For this reason, the functions for quantiles are currently not supported (but may be when R version 2.0 is required for the UCS toolkit).

\$d = dhyper(\$k, \$R1, \$R2, \$C1, \$C2);

\$ld = ldhyper(\$k, \$R1, \$R2, \$C1, \$C2);

Density $P(X = k)$ and its negative base 10 logarithm.

\$p = phyper(\$k, \$R1, \$R2, \$C1, \$C2 [, \$upper]);

\$lp = lphyper(\$k, \$R1, \$R2, \$C1, \$C2 [, \$upper]);

Tail probabilities $P(X \leq k)$ and $P(X > k)$ (if *\$upper* is specified and true), and their negative base 10 logarithms. It is recommended to set *\$upper* to the string 'upper' as a reminder of its meaning.

COPYRIGHT

Copyright 2004-2005 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::Expression

Compile and execute UCS expressions

SYNOPSIS

```
use UCS::Expression;

$exp = new UCS::Expression $code; # compile UCS expression
@vars = $exp->needed;             # variables needed to evaluate expression
$code = $exp->string;             # retrieve sourcecode of UCS expression
$result = $exp->eval(@args);      # evaluate UCS expression (argument list)
$result = $exp->eval($arghash);   # named arguments (UCS variable names)

$exp = new UCS::Expression $code, "MU" => 10, ...; # expression with parameters
@params = $exp->params;           # sorted list of parameter names
$value = $exp->param("MU");       # current value of parameter
$exp->set_param("MU", 1);         # set parameter value
$exp2 = $exp->copy;               # clone expression (e.g. when changing parameters)

$sub = $exp->code;                # reference to compiled Perl expression
$result = $sub->(@args);          # argument list is same as for eval()

$listref = $exp->evalloop($size, $arghash); # evaluate expression on full data set
$exp->evalloop(\@result, $size, $arghash);  # directly writes to array @result
```

DESCRIPTION

UCS expressions provide a convenient way to evaluate functions and conditions on the pair types in a data set. They consist of arbitrary Perl code with a syntax extension for direct access to data set variables: the character sequence `%varname%` (where *varname* is a legal UCS variable name) is replaced by the value of this variable (for the current pair type). See *ucsexp* for a more detailed description of UCS expressions and some cautionary remarks.

A **UCS::Expression** object represents a compiled UCS expression. The **needed** method returns a list of UCS variables that are required for evaluation of the expression. When **derived variables** are used in a UCS expression, they are automatically computed from the frequency signature.

The **eval** method is normally invoked with a (reference to a) hash of arguments, using UCS variable names as keys. It selects the variables needed to evaluate the UCS expression automatically from the hash, and ensures that all of them are present. Better performance is achieved by passing the required variables as an argument list in the correct order (as returned by **needed**).

The **evalloop** method greatly reduces overhead when a UCS expression is applied to a list of pair types (i.e. a full data set). It expects array references instead of simple variable values, and returns a reference to an array of the specified length. Optionally, **evalloop** can write directly to an existing array.

METHODS

\$exp = new UCS::Expression \$code;

Compiles the UCS expression *\$code* into a **UCS::Expression** object. If compilation fails for some reason, an **undefined** value is returned. Compiling a UCS expression involves the following steps:

- All UCS variable references in *\$code* are identified and validated.
- A list of required variables is constructed. Derived variables are implicitly computed from the frequency signature, and the necessary core variables are automatically added to the list of required variables.
- The UCS variable references are substituted with lexical Perl variables, which are initialised from the parameter list *@_*.
- The resulting Perl code is compiled into an anonymous subroutine, which is stored in the **UCS::Expression** object and can be executed through the **eval** method.

Since **UCS::Expressions** are comparatively small structures, it is usually not necessary to destroy them explicitly.

\$exp = new UCS::Expression \$code, \$param => \$value, ...;

This form of the constructor defines a UCS expression with parameters, given as pairs of parameter name *\$param* and default value *\$value*. Parameters can be used like variables in the UCS expression. Their names are simple UCS identifiers, but **must not** be valid UCS variable names. The recommended convention is to write parameter names all in upper case.

@names = \$exp->params;

Returns the names of all parameters in alphabetical order.

\$value = \$exp->param(\$name);

Returns the current value of parameter *\$name*;

\$exp->set_param(\$name, \$value);

Set the parameter *\$name* to the value *\$value*. The new value will be used by all subsequent calls to the **eval** and **evalloop** methods.

\$new_exp = \$exp->copy;

Makes a clone of the **UCS::Expression** object *\$exp*. Cloning is a fast operation and should always be used when changing the parameters of an expression shared between different modules (e.g. a registered association measure).

@vars = \$exp->needed;

The **needed** methods returns a list of UCS variable names, corresponding to the data set variables needed to evaluate *\$exp*.

\$code = \$exp->string;

Returns the original UCS expression represented by *\$exp* as a string, and can be used to modify and recompile UCS expressions (especially those of built-in association measures). Note that *\$code* is **chomped**, but may contain internal linebreaks (`\n`).

\$result = \$exp->eval(\$arghash);

The **eval** method evaluates a compiled UCS expression on the data passed in *\$arghash*, which must be a reference to a hash of variable names and the corresponding variable values. The necessary variables are extracted from *\$arghash* by name, and the method **dies** with an error message unless all required variables are present. Unused variables are silently ignored.

\$result = \$exp->eval(@args);

The second form of the **eval** method avoids the overhead of variable name lookup and error checking. Here, the argument list *@arg* consists of the values of all required variables in the order defined by the **needed** method. The list *@args* is passed directly to the compiled Perl code, so that errors will usually go undetected.

\$sub = \$exp->code;

The **code** method returns a code reference to the anonymous subroutine that resulted from compilation of the UCS expression. For an expression without parameters, the subroutine call

```
$result = $sub->(@args);
```

is equivalent to

```
$exp->eval(@args);
```

and further reduces overhead (by a small amount). It may be useful when the UCS expression is repeatedly applied, looping over a list of pair types. In most such cases, the **evalloop** method provides a better solution, though.

\$listref = \$exp->evalloop(\$size, \$arghash);

\$exp->evalloop(\@result, \$size, \$arghash);

The **evalloop** method is used to apply *\$exp* to an entire list of pair types (i.e. a data set) with a single call. Its invocation is similar to the first form of the **eval** method. The additional parameter *\$size* specifies the number of pair types to be processed. Each value in *\$arghash* must be a reference to an array of length *\$size*. The return value is a reference to an array of the same length.

The three-parameter form allows **evalloop** to write the results directly into an existing array, which may save a considerable amount of overhead when *\$size* is large.

SEE ALSO

See the `ucsexp` manpage for an introduction to UCS expressions, as well as the `UCS::SFunc` and `UCS::Expression::Func` manpages for pre-defined functions that may be used in UCS expressions.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::Expression::Func

Utility functions for UCS expressions

SYNOPSIS

```
use UCS::Expression::Func;

$min_x = min($x1, $x2, ...);    # minimum of two or more values
$max_y = max(@y);              # maximum of two or more values

$log_prob = -log10($prob);     # base 10 logarithm

$log_prob = inf()              # replace log(Infinity) = -log(0)
  if $prob == 0;               # by a very large value
```

DESCRIPTION

This module provides a collection of simple but useful functions, which are automatically imported into the **UCS::Expression** namespace so that they can be used in **UCS expressions** without full qualification.

FUNCTIONS

\$min_x = min(\$x1, \$x2, ...);

Minimum of two or more numbers. The argument could also be an array @x.

\$max_x = max(\$x1, \$x2, ...);

Maximum of two or more numbers. The argument could also be an array @x.

\$log_prob = -log10(\$prob);

Base 10 logarithm, which is used for all logarithmic scales in UCS (especially logarithmic p-values). Returns `-inf()` if `$prob` is zero or negative.

\$log_infinity = inf();

The `inf` function returns a large positive floating-point value that represents the logarithm of Infinity in UCS/Perl. Note that the logarithm of 0 should consequently be represented by `-inf()`, as does the **log10** function. In order to find out the exact value on your system, you can use the command line

```
ucs-config -e 'print inf(),"\n"'
```


COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::AM

Built-in association measures

SYNOPSIS

```
use UCS;
use UCS::AM;

@builtin_AMs = UCS::AM_Keys();

# random
# frequency
# z.score
# z.score.corr
# t.score
# chi.squared
# chi.squared.corr
# log.likelihood
# Poisson.Stirling
# Poisson.pv
# Fisher.pv
# MI
# MI2
# MI3
# relative.risk
# odds.ratio
# odds.ratio.disc
# Dice
# gmean
# MS
# Jaccard
# average.MI
# local.MI
```

DESCRIPTION

This module contains definitions for a wide range of **association measures**. When the **UCS::AM** module is imported, the built-in measures are registered with the **UCS** core library (see *UCS* for details on how to access registered association measures).

The following section gives a full listing of the built-in association measures from the **UCS::AM** module with short explanations. Please refer to <http://www.collocations.de/AM/> for the full equations and references. Further association measures can be imported from **add-on packages** (see the section on ADD-ON PACKAGES below).

Note that some association measures produce infinite values (*+inf* or *-inf*). The logarithm of infinity is represented by the return value of the built-in **inf** function (see the UCS::Expression::Func manpage). The association scores of measures with the suffix **.pv** can be interpreted as probabilities (the likelihood of the observed data or the p-value of a statistical hypothesis test). Such probabilities are given as **negative base 10 logarithms**, ranging from 0 to *+inf*. Measures with the suffix **.tt** (for *two-tailed*) are derived from two-sided statistical hypothesis tests. One-sided versions of these tests are provided under the same name, but without the suffix.

BUILT-IN ASSOCIATION MEASURES

random

Random numbers between 0 and 1 as association scores simulate random selection of pair types and are used to break ties when sorting a data set.

frequency

Cooccurrence frequency of the pair type. This association measure is used to sort data sets by frequency, but requires some systematic method for breaking ties.

z.score

A z-score for the observed cooccurrence frequency *O11* compared to the expected frequency *E11*. The value represents a standardised normal approximation of the binomial sampling distribution of *O11* under the point null hypothesis of independence.

z.score.corr

A z-score for *O11* compared to *E11* with Yates' continuity correction applied.

t.score

Church et al (1991) use Student's t-test to compare the observed cooccurrence frequency *O11* to the null expectation *E11* estimated from the sample (which is a random variate as well), applying several approximations to simplify the **t.score** equation. The computed value is a t-score with degrees of freedom roughly equal to the sample size *N*. This application of the t-test is highly questionable, though, and produces extremely conservative results.

chi.squared

One-sided version of Pearson's chi-squared test for the independence of rows and columns in a 2x2 contingency table. Positive scores indicate positive association ($O11 > E11$), and negative scores indicate negative association ($O11 < E11$). The distinction between positive and negative association is unreliable for small absolute values of the test statistic. Under the null hypothesis, the one-sided **chi.squared** statistic approximates a normal distribution (as the signed root of a chi-squared distribution with one degree of freedom).

chi.squared.corr

One-sided version of Pearson's chi-squared test for the independence of rows and columns in a 2x2 contingency table, with Yates' continuity correction applied.

log.likelihood

One-sided version of the log-likelihood statistic suggested by Dunning (1993), a likelihood ratio test for independence of rows and columns in a 2x2 contingency table (Dunning introduced the measure as a test for homogeneity of the table columns, i.e. equal success probabilities of two independent binomial distributions). Positive scores indicate positive association ($O_{11} > E_{11}$), and negative scores indicate negative association ($O_{11} < E_{11}$). The distinction between positive and negative association is unreliable for small absolute values of the test statistic. Under the null hypothesis, the one-sided **log.likelihood** statistic approximates a normal distribution (as the signed root of a chi-squared distribution with one degree of freedom).

Poisson.Stirling

Approximation of the likelihood of the observed cooccurrence frequency O_{11} under the point null hypothesis of independence (so that the expected frequency is E_{11}). The measure is derived from **Poisson.likelihood** (in the UCS::AM::HTest module) using Stirling's formula, resulting in a simple expression that can easily be evaluated. This measure was proposed by Quasthoff and Wolff (2002) and has been re-scaled to base 10 logarithms to allow a direct comparison with **Poisson.likelihood**.

Poisson.pv

Significance (one-sided p-value) of an exact Poisson test for the observed cooccurrence frequency O_{11} compared to the expected frequency E_{11} under the point null hypothesis of independence. This test is based on a Poisson approximation of the correct binomial sampling distribution of O_{11} . It is numerically and analytically much easier to handle than the binomial test.

Fisher.pv

Significance (one-sided p-value) of Fisher's exact test for independence of rows and columns in a 2x2 contingency table with fixed marginals. This test is widely accepted as the most appropriate independence test for contingency tables (cf. Yates 1984). Its use as an association measure was suggested by Pedersen (1996).

MI

Maximum-likelihood estimate of the base 10 logarithm of the μ -value, which is identical to pointwise mutual information between the events describing occurrences of a pair's components. Note that mutual information is measured in *decimal units* rather than the customary *bits*. The theoretical range is from $-\infty$ to $+\infty$, but the actual range for a given data set is restricted depending on the sample size N .

MI2

A heuristic variant of **MI** where the numerator is squared in order to discount low-frequency pairs. This measure also has some theoretical justification, being the square of the **gmean** measure.

MI3

Another heuristic variant of **MI** where the numerator is cubed, which boosts the discounting effect considerably.

relative.risk

Maximum-likelihood estimate of the logarithmic relative risk coefficient of association strength (base 10 logarithm). Ranges from $-\infty$ to $+\infty$.

odds.ratio

Maximum-likelihood estimate of the logarithmic odds ratio as a coefficient of association strength (base 10 logarithm). Ranges from *-inf* to *+inf*.

odds.ratio.disc

A "discounted" version of **odds.ratio**, adding 0.5 to each factor in the equation. This modification of the odds ratio is commonly used to avoid infinite values, but does not seem to have a theoretical foundation.

Dice

Maximum-likelihood estimate of the Dice coefficient of association strength. Ranges from 0 to 1.

Jaccard

Maximum-likelihood estimate of the Jaccard coefficient of association strength, which is equivalent to **Dice** (i.e., there is a strictly monotonic mapping between the two association scores). Ranges from 0 to 1.

MS

Maximum-likelihood estimate of the *minimum sensitivity* coefficient suggested by Pedersen and Bruce (1996). Ranges from 0 to 1.

gmean

Maximum-likelihood estimate of the *geometric mean* coefficient of association strength. Ranges from 0 to 1.

average.MI

Maximum-likelihood estimate of the average mutual information between the indicator variables X and Y marking instances of a pair type's components. This implementation uses base 10 logarithms and multiplies the mutual information value with the sample size *N* in order to obtain readable values. Interestingly, **average.MI** is identical to Dunning's log-likelihood measure (**log.likelihood** and its variants) except for a scaling factor.

local.MI

Contribution of a given pair type to the (maximum-likelihood estimate of the) average mutual information of *all* cooccurrences. Formally, this is the mutual information between the random variables U and V, which represent the component types of a pair token in the random sample.

ADD-ON PACKAGES

The **UCS::AM** module provides a basic set of useful and well-known association measures. Except for the **Poisson.pv** and **Fisher.pv**, all measures have simple equations that can be computed efficiently. Further and more specialised association measures can be imported from add-on packages. Currently, the following packages are available:

UCS::AM::HTest	variants of hypothesis tests, likelihood measures
UCS::AM::Parametric	parametric association measures

These packages are implemented as Perl modules and can simply be loaded with the **use** operator. Alternatively, the **UCS::Load_AM_Package** function provides a convenient interface, where only the last part of the package name has to be specified, is case-insensitive, and may be abbreviated to a unique prefix. For instance, the **UCS::AM::HTest** package can be loaded with the specification 'ht'. The empty string "" loads **UCS::AM**, and 'ALL' imports all available AM packages. (See the UCS manpage for details.)

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::AM::HTest

More association measures based on hypothesis tests

SYNOPSIS

```
use UCS;
use UCS::AM::HTest;

@hctest_AMs = UCS::AM::Keys();

# z.score.pv
# z.score.corr.pv
# t.score.pv
# chi.squared.tt
# chi.squared.tt.pv
# chi.squared.corr.tt
# chi.squared.corr.tt.pv
# chi.squared.pv
# chi.squared.corr.pv
# log.likelihood.tt
# log.likelihood.tt.pv
# log.likelihood.pv
# binomial.pv
# multinomial.likelihood.pv
# hypergeometric.likelihood.pv
# binomial.likelihood.pv
# Poisson.likelihood.pv
# Poisson.likelihood.Perl.pv
```

DESCRIPTION

This module contains some further **association measures** based on statistical hypothesis tests, most of which are variants of measures defined in the **UCS::AM** module. There are also several likelihood measures, which compute the probability of the observed contingency table

rather than applying a full hypothesis test. The association measures defined in this module are intended mainly for a detailed comparative study of the properties of the significance-of-association class of AMs. Casual users should stick with the variants found in the `UCS::AM` module.

The following section gives a full listing of the association measures defined in the `UCS::AM::HTest` module with short explanations. Please refer to <http://www.collocations.de/AM/> for the full equations and references. When the module is imported, the additional measures are registered with the UCS core library (see the UCS manpage for details on how to access registered association measures).

The association scores of measures with the suffix `.pv` can be interpreted as probabilities (i.e. the likelihood of the observed data or the p-value of a statistical hypothesis test). Such probabilities are given as **negative base 10 logarithms**, ranging from 0 to $+inf$ ($+inf$ is represented by the return value of the built-in `inf` function (see the `UCS::Expression::Func` manpage)). Measures with the suffix `.tt` (for *two-tailed*) are derived from two-sided statistical hypothesis tests. One-sided versions of these tests are provided under the same name without the suffix.

ASSOCIATION MEASURES

`z.score.pv`

The significance (one-sided p-value) corresponding to `z.score`, obtained from the distribution function of the standard normal distribution. (The `z.score` measure computes a z-score for the observed cooccurrence frequency O11 compared to the expected frequency E11; see the `UCS::AM` manpage for details.)

`z.score.corr.pv`

The significance (one-sided p-value) corresponding to `z.score.corr`, a z-score for O11 against E11 with Yates' continuity correction applied.

`t.score.pv`

The significance (one-sided p-value) corresponding to `t.score`, obtained from the distribution function of the standard normal distribution. Since the number of degrees of freedom is very large, the t-distribution of the test statistic is practically identical to the standard normal distribution (t-distribution with $df=inf$). (The `t.score` measure is an application of Student's t-test to the comparison of O11 against E11; see the `UCS::AM` manpage for details.)

`chi.squared.tt`

Pearson's chi-squared test for independence of rows and columns in a 2x2 contingency table. The equation used in this implementation is derived from the homogeneity version of the chi-squared test (for equality of the success probabilities of two independent binomial distributions), and is fully equivalent to that of the independence test. Note that Pearson's chi-squared test is two-sided.

`chi.squared.tt.pv`

The significance (two-sided p-value) corresponding to `chi.squared.tt`, obtained from the chi-squared distribution with one degree of freedom.

chi.squared.corr.tt

Pearson's chi-squared test for independence of rows and columns in a 2x2 contingency table, with Yates' continuity correction applied (two-sided test).

chi.squared.corr.tt.pv

The significance (two-sided p-value) corresponding to **chi.squared.corr.tt**.

chi.squared.pv

The significance (one-sided p-value) corresponding to **chi.squared**, the one-sided version of Pearson's test for the independence of rows and columns (see the UCS::AM manpage for details). The p-value is obtained from the standard normal distribution (since the signed square root of the chi-squared test statistic has a standard normal distribution).

chi.squared.corr.pv

The significance (one-sided p-value) corresponding to **chi.squared.corr**, the one-sided version of Pearson's chi-squared test with Yates' continuity correction applied. Again, the p-value is obtained from the standard normal distribution.

log.likelihood.tt

The log-likelihood statistic suggested by Dunning (1993), a likelihood ratio test for independence of rows and columns in a 2x2 contingency table. (Dunning introduced the statistic as a test for homogeneity of the table columns, i.e. equal success probabilities of two independent binomial distributions). Note that all likelihood ratio tests are two-sided tests.

log.likelihood.tt.pv

The significance (two-sided p-value) corresponding to **log.likelihood.tt**, obtained from the chi-squared distribution with one degree of freedom.

log.likelihood.pv

The significance (one-sided p-value) corresponding to **log.likelihood**, the one-sided version of Dunning's likelihood ratio test (see the UCS::AM manpage for details). The p-value is obtained from the standard normal distribution (since the signed square root of the log-likelihood statistic has a standard normal distribution.)

binomial.pv

Significance (one-sided p-value) of an exact binomial test for the observed cooccurrence frequency O11 compared to the expected frequency E11 under the point null hypothesis of independence. This test is computationally expensive and may be numerically unstable, so use with caution. (This is also the reason why it is not included in the UCS::AM module.)

multinomial.likelihood.pv

Likelihood of the observed contingency table under the point null hypothesis of independence (i.e. with expected frequencies E11, E12, E21, and E22 estimated from the observed table).

hypergeometric.likelihood.pv

Likelihood of the observed contingency table under the null hypothesis of independence of rows and columns, with all marginal frequencies fixed to the observed values.

binomial.likelihood.pv

Binomial likelihood of the observed cooccurrence frequency O_{11} under the point null hypothesis (with expected frequency E_{11} estimated from the observed table). This function is relatively slow and may be numerically unstable, so use with caution.

Poisson.likelihood.pv

Poisson approximation of the binomial likelihood **binomial.likelihood.pv**, which is numerically and analytically more manageable.

Poisson.likelihood.Perl.pv

Alternative version of **binomial.likelihood.pv**, based on a direct Perl implementation of the naive multiplicative algorithm.

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::AM::Parametric

Parametric association measures

SYNOPSIS

```
use UCS;
use UCS::AM::Parametric;

@parametric_AMs = UCS::AM::Keys();

# MI.conf
# MI.conf.<n>          [<n> = 2, 3, 5, 10, 50, 100, 1000]
# Poisson.mu.pv
# Poisson.mu.<n>.pv  [<n> = 2, 3, 5, 10, 50, 100, 1000, 10000]
```

DESCRIPTION

This module contains some parametric **association measures**, which are parametrised extensions of measures defined in the basic **UCS::AM** module. Parametric measures are a recent development in cooccurrence statistics, and the choice of appropriate parameter values is still very much a research question. Parametric measures will often be computationally expensive and may be numerically unstable, so novice users are advised to use the basic measures from the **UCS::AM** module instead.

The following section gives a full listing of the parametric association measures defined in the **UCS::AM::Parametric** module with short explanations. Please refer to <http://www.collocations.de/AM/> for the full equations and references. When the module is

imported, the additional measures are registered with the **UCS** core library (see the **UCS manpage** for details on how to access registered association measures).

The association scores of measures with the suffix **.pv** can be interpreted as probabilities (i.e. the likelihood of the observed data or the p-value of a statistical hypothesis test). Such probabilities are given as **negative base 10 logarithms**, ranging from 0 to *+inf* (*+inf* is represented by the return value of the built-in **inf** function (see the **UCS::Expression::Func manpage**)).

ASSOCIATION MEASURES

MI.conf

Conservative estimate for the base 10 logarithm of the *mu*-value (whose maximum-likelihood estimate is given by the **MI** measure). The association score computed by **MI.conf** is the lower endpoint of a two-sided confidence interval for *mu* at significance level **alpha**, which is specified by the **ALPHA** parameter (as a negative base 10 logarithm). The "usual" significance levels *.01* and *.001* correspond to **ALPHA=2** and **ALPHA=3**, respectively.

Please duplicate the **UCS::Expression** object returned by **UCS::AM_Expression("MI.conf")** before modifying the **ALPHA** parameter.

MI.conf.ALPHA

Versions of **MI.conf** with the **ALPHA** parameter pre-set to the value specified as part of the name. Available **ALPHA** values are **2, 3, 5, 10, 50, 100, and 1000**. For instance, **MI.conf.10** computes a two-sided confidence interval at significance level 1E-10.

Do not modify the **ALPHA** parameter of these association measures (in the **UCS::Expression** object returned by the **UCS::AM_Expression** function).

Poisson.mu.pv

Poisson test for *O11* under the modified point null hypothesis $p_i = p_1 * p_2 * mu$ (rather than the independence hypothesis $p_i = p_1 * p_2$ used by the **Poisson.pv** measure). The (non-logarithmic) value of *mu* is given by the **MU** parameter. For **MU=1**, the association scores computed by **Poisson.mu.pv** are identical to those of **Poisson.pv**.

Please duplicate the **UCS::Expression** object returned by **UCS::AM_Expression("Poisson.mu.pv")** before modifying the **MU** parameter.

Poisson.mu.MU.pv

Versions of **Poisson.mu.pv** with the **MU** parameter pre-set to the value specified as part of the name. Available **MU** values are **2, 3, 5, 10, 50, 100, 1000, and 10000**.

Do not modify the **MU** parameter of these association measures (in the **UCS::Expression** object returned by the **UCS::AM_Expression** function).

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::DS

Base class for data set implementations

SYNOPSIS

```

use UCS::DS;

$ds = new UCS::DS;           # "virtual" data set
$ds->add_vars($name1, $name2, ...); # append variables (= columns) in this order
$ds->delete_vars($name1, ...);   # delete variables (column 'gaps' are closed)

$type = $ds->var($name);       # check whether variable exists, returns data type
$index = $ds->var_index($name); # column index of variable
@names = $ds->vars;           # list all variables in column order

$ds->temporary($name, 1);     # mark variable as temporary (will not be saved)

@lines = $ds->comments;       # ordered list of comment lines
$ds->add_comments($line1, ...); # append comment lines
$ds->delete_comments;        # delete all comments
$ds->copy_comments($ds2);     # copy all comments from $ds2

@global_vars = $ds->globals;  # unordered list of global variable names
$value = $ds->global($var);   # return value of global variable $var
$ds->set_global($var, $value); # set value of global variable (may be new variable)
$ds->delete_global($var);     # delete global variable
$ds->copy_globals($ds2);     # copy global variables from $ds2

```

DESCRIPTION

UCS::DS acts as a base class for **data set** managers (either file streams or in-memory representations). A **UCS::DS** object manages a list of **variables** (with names according to the UCS naming conventions detailed in *ucsfile*), and maps them to the **column indices** of a data set file.

It is always ensured that the column indices of a data set span a contiguous range starting at 0. New variables will be appended to the existing columns in the order of declaration. When a variable is deleted, all columns to its right are shifted to fill the gap.

When it is available, **UCS::DS** objects also store information from the **header** of a data set file. This information includes **comment lines** and **global variables** (see *ucsfile* for details).

METHODS

\$ds = new UCS::DS;

Create a new **UCS::DS** object, with an empty list of variables. Normally, this constructor is only invoked implicitly by derived classes.

\$ds = new UCS::DS \$name1, \$name2, ...;

Creates a **UCS::DS** object with the specified variables. Same as

```
$ds = new UCS::DS;
$ds->add_vars($name1, $name2, ...);
```

\$ds->add_vars(\$name1, \$name2, ...);

Add one or more variables *\$name1*, *\$name2*, ... to the data set. Variables that are already defined will be silently ignored. New variables are appended to the existing columns in the specified order. *\$name1*, *\$name2*, ... must be valid UCS variable names.

\$ds->delete_vars(\$name1, \$name2, ...);

Delete the variables *\$name1*, *\$name2*, ... from the data set. Variables that are not defined in the data set will be silently ignored. When a variable has been deleted, all columns to its right are shifted to fill the gap. All arguments must be valid UCS variable names.

\$type = \$ds->var(\$name);

Check whether the variable *\$name* is defined in the data set *\$ds*. Returns the data type of the variable (BOOL, INT, DOUBLE, or STRING, see *ucsfile*), or **undef** if it does not exist.

\$is_temp = \$ds->temporary(\$name);

\$ds->temporary(\$name, \$val);

Mark variable *\$name* as temporary (if *\$val* is true) or permanent (if *\$val* is false). The single-argument version returns true if the variable *\$name* is temporary. Temporary variables are interpreted by in-memory representations of data sets. They may be deleted automatically and will not be written to data set files.

\$index = \$ds->var_index(\$name);

Get column index of variable *\$name*. *\$index* ranges from 0 to one less than the number of variables in the data set. Returns **undef** if the variable *\$name* does not exist in the data set. It is recommended to test this condition with the **var** method first.

@names = \$ds->vars;

Returns the names of all variables in this data set, sorted by their column indices. When saved to a data set file, the columns will appear in this order.

@lines = \$ds->comments;

Returns all comment lines as an ordered list (i.e. as they would appear in a data set file). Comment lines are **chomped** and the initial # character (followed by an optional blank) is removed.

\$ds->add_comments(\$line1, ...);

Add comment lines (which will be appended to existing comments). Like the data returned by the **comments** method, *\$line1* etc. should not begin with a # character or end in a newline.

\$ds->delete_comments;

Deletes all comment lines.

\$ds->copy_comments(\$ds2);

Copies all comment lines from *\$ds2*, which must be an object derived from **UCS::DS**. Existing comments of *\$ds* are overwritten. This command is equivalent to

```
$ds->delete_comments;
$ds->add_comments($ds2->comments);
```

@global_vars = \$ds->globals;

Returns the names of all global variables in alphabetical order. **NB:** global variable names must be valid UCS identifiers.

\$value = \$ds->global(\$var);

Returns the value of a global variable *\$var* as a character string. If the global variable *\$var* does not exist, returns **undef**.

\$ds->set_global(\$var, \$value);

Set global variable *\$var* to the string *\$value*. If *\$var* does not exist, it is automatically added to the data set.

\$ds->delete_global(\$var);

Delete a global variable. If *\$var* does not exist, the method call will be silently ignored.

\$ds->copy_globals(\$ds2);

Copies all global variables and their values from *\$ds2*, which must be an object derived from **UCS::DS**. Any existing global variables off the data set *\$ds* will be erased.

COPYRIGHT

Copyright 2003 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::DS::Stream

I/O streams for data set files

SYNOPSIS

```
use UCS::DS::Stream;

$ds = new UCS::DS::Stream::Read $filename;
die "format error" unless defined $ds;
# access variables, comments, and globals with UCS::DS methods
while ($ds->read) {
    die "read/format error"
        unless $ds->valid;                # valid row data available?
    $n = $ds->row;                          # row number
    $idx = $ds->var_index("am.log1");        # see 'ucsd doc UCS::DS'
    $log1 = $ds->columns->[$idx];           # $ds->columns returns arrayref
    $log1 = $ds->value("am.log1");         # short and safe, but slower
    $rowdata = $ds->data;                   # returns hashref (varname => value)
    $log1 = $rowdata->{"am.log1"};         # == $ds->value("am.log1")
}
```

```

}
ds->close;

$ds = new UCS::DS::Stream::Write $filename;
# set up variables, comments, and globals with UCS::DS methods
$ds->open;                # write data set header
foreach $i (1 .. $N) {
  $ds->data("id"=>$i, "l1"=>$l1, ...);# takes hashref or list of pairs
  $ds->data("am.log1"=>$log1, ...);  # may be called repeatedly to add data
  $ds->columns($i, $l1, $l2, ...);  # complete list of column data
  $ds->write;                        # write row and clear data cache
}
$ds->close;

```

DESCRIPTION

UCS data set streams are used to read and write data set files one row at a time. When an **input stream** is created, the corresponding data set file is opened immediately and its header is read in. The header information can then be accessed through **UCS::DS** methods. Each **read** method call loads a single row from the data set file into an internal representation, from which it is available to the main program.

An **output stream** creates / overwrites its associated data set file only when the **open** method is called. This allows the main program to set up variables and header data with **UCS::DS** method calls. After opening the file, the data for each row is first stored in an internal representation, and then written to disk with the **write** method.

Note that there are no objects of class **UCS::DS::Stream**. Both input and output streams inherit directly from the **UCS::DS** class.

INPUT STREAMS

Input streams are implemented as **UCS::DS::Stream::Read** objects. When an input stream is created, the header of the associated data set file is read in. Header data and information about the variables in the data set can then be accessed using **UCS::DS** methods.

The actual data set table is then loaded one row (= pair type) at a time by calling the **read** method. The row data are extracted into an internal representation where they can be accessed with various methods (some of them being safe, others more efficient).

The **na** method controls whether missing values (represented by the string **NA** in the data set file) are recognised and stored internally as **undefs**, or whether they are silently translated into 0 (**BOOL**, **INT**, and **DOUBLE** variables) and the empty string (**STRING** variables), respectively.

\$ds = new UCS::DS::Stream::Read \$filename;

Open data set file *\$filename* and read header information. Header variables and comments, as well as information about the variables in the data set can then be accessed with **UCS::DS** methods. If *\$filename* is a plain filename or a partial path (i.e., neither a full relative or absolute path starting with / or ./ nor a command pipe) and the file is not found in the current working directory, the standard UCS library is automatically searched for a data set with this name.

If there is a syntax error in the data set header, **undef** is returned. Note that the object constructor will **die** if the file *\$filename* does not exist or cannot be opened for reading.

\$ds->na(1);

Enables recognition of missing values represented by the string NA (as used by R). When enabled, missing values are represented by **undefs**. Otherwise, they will be silently translated into 0 (BOOL, INT, and DOUBLE variables) and the empty string (STRING variables), respectively. Use `$ds->na(0)`; to disable missing value support, which is by default activated.

\$ok = \$ds->read;

Read one line of data from the data set file and extract the field values into an internal representation. Returns false when the entire data set has already been processed. Typically used in a **while** loop similar to the diamond operator: `while ($ds->read) { ... }`.

\$at_end = \$ds->eof;

Returns true when the entire data set has been read, i.e. the logical complement of the value returned by the last **read** call.

\$ok = \$ds->valid;

Returns true if the internal representation contains valid row data. Currently, this only compares the number of columns in the file against the number of variables in the data set. Later on, values may also be syntax-checked and coerced into the correct data type.

\$n = \$ds->row;

Returns the current row number (of the row read in by the last **read** call, which is now stored in the internal representation).

\$value = \$ds->value(\$name);

Get value by variables name. Returns the value of variable *\$name* currently stored in the internal representation. This method is convenient and safe (because it checks that the variable *\$name* exists in the given data set), but incurs considerable overhead.

\$cols = \$ds->columns;

Return entire row data as an array reference. Individual variables have to be identified by their index, which can be obtained with the **var_index** method (`$cols->[$idx]`). Since index lookup can be moved out of the row processing loop, this access method is much more efficient than its alternatives. **NB:** the array `@$rowdata` is not reused for the next line of input and can safely be integrated into user-defined data structures.

\$rowdata = \$ds->data;

Returns hash reference containing entire row data indexed by variable names. Thus, the values of individual variables can be accessed with the expression `$rowdata->{$varname}`, similar to using the **value** method. Access with the **data** method is convenient for copying row data to an output stream. It is relatively slow, though, and should not be used in tight loops.

\$ds->close;

Close the data set file. This method is automatically invoked when the object *\$ds* is destroyed.

OUTPUT STREAMS

Output streams are implemented as **UCS::DS::Stream::Write** objects. After creating an output stream object, variables and header data are set up with the **UCS::DS** methods. The data set header is written to disk when the **open** method is called.

After that, the actual data set table is generated one row at a time. Row data is first stored in the internal presentation (using the **data** or the **columns** method), and then written to disk when the **write** method is called.

\$ds = new UCS::DS::Stream::Write \$filename;

Create output stream for data set file *\$filename*. Note that this file will only be created or overwritten when the **open** method is called (in contrast to input streams, which open the data set file immediately).

\$ds->open;

After setting up variables and header data (comment lines and global variables) with the respective **UCS::DS** methods, the **open** method opens the data set file and writes the data set header. If the file cannot be opened for writing, the **open** method will **die** with an error message.

\$ds->data(\$v1 => \$val1, \$v2 => \$val2, ...);

\$ds->data(\$hashref);

Store data for the next row to be written in an internal representation. When using the **data** method, variables are identified by name (*\$v1*, *\$v2*, ...) and can be specified in any order. The variable-value pairs can also be passed with a single hash reference. Variables that do not exist in the data set will be silently ignored. The **data** method can be called repeatedly for a single row.

\$ds->columns(\$val1, \$val2, ...);

The **columns** method provides a more efficient way to specify row data. Here, all column values are passed in a single method call, and care has to be taken to list them in the correct order (namely, the order in which the variables were set up with the **add_vars** method). **NB:** the **data** and **columns** methods cannot be mixed. It is also not possible to set up the row data incrementally with repeated **columns** calls.

\$ds->write;

Writes the row data currently stored in the internal buffer to the data set file, and resets the buffer (to **undef** values). Any **undef** values in the buffer (including the case where some variables were not specified with the **data** method) are interpreted as missing values and substituted by the string **NA**.

\$ds->close;

Completes and closes the data set file.

EXAMPLES

The recommended way of **copying rows** from one data set file to another is to use the **data** methods of both streams, so that variables are copied by name rather than column position.

It would be more efficient to pass row data directly (using the **columns** methods), but this approach is prone to lead to errors when the order of the columns is different between the input and output data sets.

The following example makes a copy of a data set file, adding an (enumerative) id variable if it is not present in the source file.

```
$in = new UCS::DS::Stream::Read $input_file;
die "$input_file: format error"
    unless defined $in;
@vars = $in->vars;
$add_id = not $in->var("id");

$out = new UCS::DS::Stream::Write $output_file;
$out->copy_comments($in);           # copy comments and
$out->copy_globals($in);           # global variables from input file
$out->add_vars("id")                # conventionally, the "id" variables
    if $add_id;                    # is in the first column
$out->add_vars(@vars);
$out->open;                          # writes header to $output_file

while ($in->read) {
    die "read/format error"
        unless $in->valid;
    $out->data($in->data);           # copy row data by field name
    $out->data("id" => $in->row)   # use row number as ID value
        if $add_id;
    $out->write;
}

$in->close;
$out->close;
```

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::DS::Memory

In-memory representation of data sets

SYNOPSIS

```
use UCS::DS::Memory;

$ds = new UCS::DS::Memory;           # empty data set
$ds = new UCS::DS::Memory $filename; # read from file (using UCS::DS::Stream)

# access & edit variables, comments, and globals with UCS::DS methods
```



```

$pairs = $ds->size;           # number of pair types
$ds->set_size($pairs);       # truncate or extend data set

$value = $ds->cell($var, $n); # read entry from data set table
$ds->set_cell($var, $n, $value); # set entry in data set table

$rowdata = $ds->row($n);     # returns hashref (varname => value)
$ds->set_row($n, $rowdata);  # set row data (ignores missing vars)
$ds->set_row($n, "f1"=>$f1, "f2"=>$f2, ...);
$ds->append_row($n, $rowdata); # append row to data set
$ds->delete_rows($from, $to); # delete a range of rows from the data set

$vector = $ds->column($var); # reference to data vector of $var
$vector->[$n] = $value;      # fast direct access to cells

$ds->eval($var, $exp)        # evaluate expression on data set & store in $var
  unless $ds->missing($exp); # check first whether all reqd. variables are available
$ds->add($var);              # auto-compute variable (derived variable or registered AM)

$stats = $ds->summary($var); # statistical summary of numerical variable

$ds->where($idx, $exp);      # define index: rows matching UCS expression
$n = $ds->count($exp);       # number of rows matching expression
$vector = $ds->index($idx);  # returns reference to array of row numbers
$ds->make_index($idx, $row1, $row2, ...); # define index: explicit list of row numbers
$ds->make_index($idx, $vector); # or array reference (will be duplicated)
$ds->activate_index($idx);   # activate index (will be used by most access methods)
$ds->activate_index();       # de-activate index
$ds->delete_index($idx);    # delete index

$ds2 = $ds->copy;            # make physical copy of data set (using index if activated)
$ds2 = $ds->copy("*", "am.%"); # copy selected variables only (in specified order)

$ds->renumber;              # renumber/add ID values as increasing sequence 1 .. size

$ds->sort($idx, $var1, $var2, ...); # sort data set on $var1, breaking ties by $var2 etc.
$ds->sort($idx, "-$var1", "+$var2"); # - = descending, + = ascending (default depends on variable type)
$ds->rank($ranking, $key1, ...);    # compute ranking (with ties) and store in data set variable $ranking

$ds->save($filename);       # save data set to file (using index if activated)

$dict = $ds->dict($var1, $var2, ...); # lookup hash for variable(s) (UCS::DS::Memory::Dict object)
($max, $average) = $dict->multiplicity; # maximum / average number of rows for each key
if ($dict->unique) { ... } # whether every key identifies a unique row
@rows = $dict->lookup($x1, $x2, ...); # look up key in dictionary, returns all matching rows
$row = $dict->lookup($x1, $x2, ...); # in scalar context, returns first matching row
@rows = $dict->lookup($other_ds, $n); # look up row $n from other data set
$n_rows = $dict->multiplicity($x1, $x2, ...); # takes same arguments as lookup()
@keys = $dict->keys;         # return unsorted list of keys entered in dictionary

```

DESCRIPTION

This module implements an in-memory representation of UCS data sets. When a data set file has been loaded into a **UCS::DS::Memory** object (or a new empty data set has been created), then **variable names**, **comments**, and **globals** can be accessed and modified with the respective **UCS::DS** methods (see the **UCS::DS** manpage).

Additional methods in the **UCS::DS::Memory** class allow the user to:

- read and write individual **cells** as well as entire **rows** or **columns**
- change the **size** of a data set
- annotate **derived variables**, **association scores**, or arbitrary **UCS expressions** in the data set

- compute statistical **summaries** of numerical variables
- **select** rows matching given UCS expression from a data set
- **sort** data sets by one or more variables and compute **rankings**
- save the data set into a data set file

The individual methods are detailed in the following sections. In all methods, columns are identified by the respective variable names, whereas rows (corresponding to pair types) are identified by row numbers. **NB:** Row numbers start with 1 (like R vectors, but unlike Perl arrays)!

GENERAL METHODS

```
$ds = new UCS::DS::Memory;
```

Create empty data set. The new data set has zero rows and no variables. Returns object of class **UCS::DS::Memory**;

```
$ds = new UCS::DS::Memory $file [, '-na' ;]
```

Reads data set file into memory and returns **UCS::DS::Memory** object. The argument *\$file* is either a string giving the name of the data set file or a **UCS::DS::Stream::Read** object (see the **UCS::DS::Stream** manpage), which has been opened but not read from. When the specified file does not exist and in the case of a read error, the constructor **dies** with an appropriate error message.

The option `'-na'` disables missing value support (which is enabled by default), so that NA values in the data set file will be replaced by 0 or the empty string, depending on the data type. Use `'+na'` to enable missing value support explicitly.

```
$V = $ds->size;
```

Returns the size of the data set, i.e. the number of rows (or pair types).

```
$ds->set_size($V);
```

Change the size of the data set to *\$V* rows. This method can both truncate and extend a data set. **NB:** Unlike the **size** method, **set_size** always applies to the real size of the data set and ignores the active row index. However, all row indices are preserved and adjusted in case of a truncation. If there is an active row index, it remains active. (See the section ROW INDEX METHODS below for more information on row indices.)

```
$value = $ds->cell($var, $n);
```

Retrieve the value of variable *\$var* for row *\$n* (i.e. the *\$n*-th pair type). This method is convenient and performs various error checks, but it involves a considerable amount of overhead. Consider the **column** method when performance is an issue.

```
$ds->set_cell($var, $n, $value);
```

Set the value of variable *\$var* for row *\$n* to *\$value*. Like **cell**, this method is convenient, but comparatively slow. Consider the **column** method when is an issue.

`$rowdata = $ds->row($n);`

Returns hash reference containing the entire data from row *\$n* indexed by variable names. This method is inefficient and mainly for convenience, e.g. when applying a UCS expression to individual rows (cf. the description of the **eval** method in the UCS::Expression manpage).

`$ds->set_row($n, $rowdata);`

`$ds->set_row($n, $var1 => $val1, $var2 => $val2, ...);`

Set the values of some or all variables for row *\$n*. The values can either be passed in a single hash reference indexed by variable names, or as *\$var => \$value* pairs. Any variables that do not exist in the data set *\$ds* are silently ignored. This method is faster than calling **set_cell** repeatedly, especially when a new row is added to the data set.

`$ds->append_row($rowdata);`

`$ds->append_row($var1 => $val1, $var2 => $val2, ...);`

Append new row to the data set and fill it with the specified values. This method is a combination of **set_size** and **set_row**. Variable values that are not specified in the argument list are set to **undef**. When there is an active row index, the new row is appended to this index, while all other indices remain unchanged (see the section on ROW INDEX METHODS below for more information on row indices).

`$ds->delete_rows($from, $to);`

Delete rows *\$from* through *\$to* from the data set. **NB:** This method always applies to the real row numbers and ignores the active row index. All existing indices are adjusted (which is an expensive operation) and an active row index remains activated. (See the section on ROW INDEX METHODS below for more information on row indices.)

`$vector = $ds->column($var);`

Returns an array reference to the data vector of variable *\$var*. *\$vector* can be used both for read and write access, so care has to be taken that the data set isn't accidentally modified (e.g. through side effects of a **map** or **grep** operation on *@\$vector*). Of course, activating a row index has no effect, since the **column** method gives direct access to the internal data structures. (See the section on ROW INDEX METHODS below for more information on row indices.)

`@missing_vars = $ds->missing($exp);`

Determines whether all variables required to evaluate the UCS expression *\$exp* (an object of class **UCS::Expression**) are defined in the data set *\$ds*. Returns an empty list if *\$exp* can be evaluated, and the names of missing variables otherwise.

`$ds->eval($var, $exp);`

Evaluate the UCS expression *\$exp* (an object of class **UCS::Expression**) on the data set *\$ds*, and store its values in the variable *\$var*. When *\$var* is a new variable, it is automatically added to the data set; Otherwise, the previous values are overwritten. This operation is *much* faster than repeatedly evaluating *\$exp* for each row. For convenience, *\$exp* can also be specified as a source string, which will be compiled on the fly. **NB:** The **eval** method always operates on the entire data set, even when a row index is activated. (See the section on ROW INDEX METHODS below for more information on row indices.)

\$ds->add(\$var);

Add a new variable to the data set and auto-compute its values, or overwrite an existing variable. *\$var* must be the name of a **derived variable** such as E11 or an **association score** such as `am.t.score` (see the `ucsfile` manpage for details).

\$stats = \$ds->summary(\$var);

Computes a statistical summary of the numerical variable *\$var* (a numerical variable is a variable of data type INT or DOUBLE). *\$stats* is a hash reference representing a data structure with the following fields:

MIN	...	minimum value
MAX	...	maximum value
ABSMIN	...	smallest non-zero absolute value
ABSMAX	...	largest absolute value
SUM	...	sum of all value
MEAN	...	mean (= average)
MEDIAN	...	median (= 50% quantile)
VAR	...	empirical variance
SD	...	empirical standard deviation (sq. root of variance)
STEP	...	smallest non-zero difference between any two values
NA	...	number of missing values (undef's)

Note that some of these fields may be **undef** if they have no meaningful value for the given data set.

\$ds2 = \$ds->copy;**\$ds2 = \$ds->copy(@variables);**

Duplicates a data set, so that *\$ds2* is completely independent from *\$ds* (whereas *\$ds2* = *\$ds*; would just give another handle on the same data set). Comments and globals are copied to *\$ds2* as well. Optionally, a list of variable names and/or wildcard patterns (see the `ucsexp` manpage) can be specified. In this case, only the selected columns will be copied. **NB:** If there is an active row index, the copy will only include the rows selected by the index, and they will be arranged in the corresponding order. However, no row indices are copied to *\$ds2*. (See the section on ROW INDEX METHODS below for more information on row indices.)

\$ds->renumber;

When rows have been deleted from a data set, or a copy has been made with an active row index, the values of the `id` variable are preserved (and can be used to match rows against the correspond entries in the original data set). When an independent numbering is desired, the **renumber** method can be used to re-compute the `id` values so that they form an uninterrupted sequence starting from 1. **NB:** The renumbering ignores an activated row index.

\$ds->save(\$filename);**\$ds->save(\$filename, @variables);**

This method saves the contents of *\$ds* to a UCS data set file *\$filename*. When an optional list of variable names and/or wildcard patterns (see the `ucsexp` manpage) is specified, only the selected columns will be saved. **NB:** If there is an active row index, only the rows selected by the index will be written to *\$filename*, and they will be

arranged in the corresponding order. The row indices themselves cannot be stored in a data set file. (See the section on ROW INDEX METHODS below for more information on row indices.) Also note that **temporary variables** will not be saved (see the UCS::DS manpage).

ROW INDEX METHODS

A **row index** is an array reference containing a list of row numbers (starting from 1, unlike Perl arrays). Row indices are used to select rows from an in-memory data set, or to represent a re-ordering of the rows (or both). They are usually created by the **where** and **sort** methods, but can also be constructed explicitly. An arbitrary number of named row indices can be stored in a UCS::DS::Memory object.

A row index can be **activated**, creating a "virtual" data set containing only the rows selected by the index, arranged in the corresponding order. Most UCS::DS::Memory methods will then operate on this virtual data set. All exceptions are marked clearly in this manpage. In particular, the **where** method selects a subset of the activated index, and **sort** can be used to reorder it. There can only be one active row index at a time. There is no way of localising the activation (so that a previously active index is restored at the end of a block), so it is highly recommended to use active indices only locally and de-activate them afterwards.

Index names must be valid UCS identifiers, i.e. they may only contain alphanumeric characters (A-Z a-z 0-9) and periods (.) (cf. VARIABLES in *ucsfile*). Note that index names beginning with a period are reserved for internal use.

\$ds->make_index(\$idx, \$row1, \$row2, ...);

\$ds->make_index(\$idx, \$vector);

Construct row index from a list of row numbers or an array reference *\$vector*, and store it under the name *\$idx* in the data set *\$ds*. In the second form, the anonymous array is duplicated, so the contents of *\$vector* can be modified or destroyed without affecting the stored row index.

\$vector = \$ds->index(\$idx);

Retrieve row index by name. Returns an array reference to the internal data, so be careful not to modify the contents of *\$vector* accidentally. In most cases, it is easier to activate *\$idx* and use the normal access methods.

\$ds->delete_index(\$idx);

Delete the row index named *\$idx*. If it happens to be activated, it will automatically de-activated.

\$ds->activate_index(\$idx);

Activate row index *\$idx*. This will clear any previous activations. Note that this operation may change the effective size of the data set as returned by the **size** method (unless *\$idx* is just a sort index).

\$ds->activate_index();

Deactivate the currently active index, re-enabling direct access to the full data set in its original order.

`$ds->where($idx, $exp);`

Construct *\$idx* selecting all rows for which the UCS expression *\$exp* (given as a **UCS::Expression** object) evaluates to true (see the `ucsexp` manpage for an introduction to UCS expression, and the `UCS::Expression` manpage for compilation instructions). It is often convenient to compile *\$exp* on the fly, especially when it is a simple condition, e.g.

```
$ds->where("high.freq", new UCS::Expression '%f% >= 10');
```

which can be shortened to

```
$ds->where("high.freq", '%f% >= 10');
```

The **where** method will automatically compile the source string passed as *\$exp* into a **UCS::Expression** object. On-the-fly compilation involves only moderate overhead. When there is an active row index, **where** will select a subset of this index, preserving its ordering.

`$n = $ds->count($exp);`

Similar to **where**, this method only counts the number of rows matching the UCS expression *\$exp*, without creating a named index. The condition *\$exp* may be given either as a **UCS::Expression** object or as a source string, which is compiled on the fly. (Internally, the rows are collected in a temporary index, which is automatically deleted when the method call returns.)

`$ds->sort($idx, $key1, $key2, ...);`

Sort data set *\$ds* by the specified sort keys. The data set is first sorted, by *\$key1*. Ties are then broken by *\$key2*, any remaining ties by *\$key3*, etc. If there are any ties left when all sort keys have been used, their ordering is undefined (and depends on the implementation of the **sort** function in Perl). The resulting ordering is stored in a row index with the name *\$idx*. When there is an active row index, **sort** will re-order the rows selected by this index.

Each **sort key** consists of a variable name, optionally preceded or followed by a + or - character to select ascending or descending sort order, respectively. The default order is *descending* for Boolean variables and association scores, and *ascending* for all other variables. The sort keys 'l1' and 'l2' sort in alphabetical order, while 'f-' puts the most frequent pair types first.

In order to break remaining ties randomly, an appropriate additional sort key has to be specified. The usual choice are the association scores of the **random** measure (see the `UCS::AM` manpage). It may be necessary to compute this measure first, which can be conveniently done with the **add** method, as shown in the example below.

```
# order pair types by frequency (descending), breaking ties randomly
if (not $ds->var("am.random")) {
  $ds->add("am.random");
  $ds->temporary("am.random", 1); # temporary, don't save to disk
}
$ds->sort("by.freq", "f-", "am.random");
```

```
$ds->rank($ranking, $key1, $key2, ...);
```

The **rank** method is similar to **sort**, but creates a ranking instead of a sort index. The ranking is stored in the integer variable *\$ranking*. Note that tied rows are assigned the same rank, which is the lowest available rank (as in the Olympic Games) rather than the average of all ranks in the group (as is often done in statistics). All other remarks about the **sort** method apply equally well to the **rank** method, especially those concerning randomisation.

DICTIONARIES (LOOKUP HASHES)

A data set **dictionary** is a **hash** structure listing all the different values that a given variable assumes in the data set (or all the different value combinations of several variables). For each value (or value combination), which is called a **key** of the dictionary, the corresponding row numbers in the data set can be retrieved (called a **lookup** of the key). In the terminology of relational databases, such a dictionary is referred to as an **index**. Be careful not to confuse this notion with the **row index** described above, which is used for subsetting and/or reordering the rows of a data set.

A dictionary can be created for any variable (or combination of variables) with the **dict** method, and is returned in the form of a **UCS::DS::Memory::Dict** object. **NB:** This dictionary is only valid as long as the data set itself is not modified (which includes activation or deactivation of a row index). Unlike a database index, the dictionary is not updated automatically. It is therefore important to keep operations on the data set under strict control while a dictionary is in use. It is always possible to add, modify, and delete variables that are not included in the dictionary, though. For the same reason (as well as to save working memory), dictionaries should be deleted when they are no longer needed.

The main purpose of a dictionary is to **look up** keys and find the matching rows in the data set efficiently (the **ucs-join** program is an example of a typical application). It is often desirable to choose variables in such a way that every key identifies a unique row in the data set (for instance, the values of 11 and 12 identify a pair type, which should have only one entry in a data set). A dictionary with this property is called **unique**. Both unique and non-unique dictionaries are supported (unique dictionaries are represented in a memory-efficient fashion). Lookup and similar operations are implemented as methods of the **UCS::DS::Memory::Dict** object.

Although mainly intended for string values, dictionaries support all data types. Boolean variables will usually be of interest only in combination with other variables (possibly also Boolean ones), and dictionaries are rarely useful for floating-point values.

```
$dict = $ds->dict($var1, ..., $varN);
```

Create a **dictionary** for the variables *\$var1*, ..., *\$varN* in the data set *\$ds*. Each **key** of this dictionary is a combination of *N* values, which must be specified in the same order as the variable names. When a row index is in effect, keys and row numbers in the dictionary are taken from the virtual data set defined by the activated index. The returned object of class **UCS::DS::Memory::Dict** is a read-only dictionary: in order to take changes in the data set *\$ds* into account (including the activation or deactivation of a row index), a new object has to be created with the **dict** method.

```
if ($dict->unique) { ... }
```

This method returns a true value iff *\$dict* is a **unique** dictionary.

```
($max, $avg) = $dict->multiplicity;
```

```
$max = $dict->multiplicity;
```

Returns the maximum (*\$max*) and average (*\$avg*) number of rows matching a key in *\$dict*. The dictionary is unique iff *\$max* equals 1.

```
@rows = $dict->lookup($x1, ..., $xN);
```

```
$row = $dict->lookup($x1, ..., $xN);
```

Look up a **key**, specified as an *N*-tuple of variable values (*\$x1*, ..., *\$xN*), in the dictionary *\$dict* and return the matching row numbers. The values *\$x1*, ..., *\$xN* must be given in the same order as the variables *\$var1*, ..., *\$varN* in the **dict** method call when the dictionary was created. When the key is not found in *\$dict*, an empty list is returned.

In scalar context, the (number of the) first matching row is returned, or **undef** if the key is not found in the dictionary.

```
@rows = $dict->lookup($ds2, $n);
```

```
$row = $dict->lookup($ds2, $n);
```

The **lookup** method can also be used to look up rows from a second data set *\$ds2*, i.e. to find rows in the dictionary's data set *\$ds* where the values of *\$var1*, ..., *\$varN* match the *\$n*-th row of *\$ds2*. For this form of invocation, the dictionary variables must be defined in *\$ds2* (otherwise, a fatal error is raised).

```
$n_rows = $dict->multiplicity($x1, ..., $xN);
```

```
$n_rows = $dict->multiplicity($ds2, $n);
```

When called with arguments, the **multiplicity** method returns the number of rows matching a specific key in *\$dict*. The key can be given in the same two ways as for the **lookup** method. (Note that calling **lookup** in scalar context returns the first matching row, *not* the total number of rows.)

```
@keys = $dict->keys;
```

```
$n_keys = $dict->keys;
```

Returns an unsorted list of all dictionary keys in the internal representation (where each key is a single string value). Such internal representations can be passed to the **lookup** and **multiplicity** methods instead of an *N*-tuple (*\$x1*, ..., *\$xN*). In scalar context, the **keys** method efficiently computes the number of keys in *\$dict*.

Examples The **keys** method and the ability to use the returned internal representations in the **lookup** method provide an easy way to compute the (empirical) **distribution** of a data set variable, i.e. a list of different values and their multiplicities. (Note that calling **lookup** in scalar context cannot be used to determine the multiplicity of a key because it returns the first matching row in this case.)

```
# frequency table for variable $v on data set $ds
$dict = $ds->dict($v);
@distribution =
  # sort values by multiplicity
  sort { $b->[1] <=> $a->[1] or $a->[0] cmp $b->[0] }
```



```

# compute multiplicity for each value
map { [$_, $dict->multiplicity($_)] }
# for a single variable $v, internal keys are simply the values
$dict->keys;
undef $dict;          # always erase dictionary after use

```

The following example is a bare-bones version of the **ucs-join** command, annotating the pair types of a data set *\$ds1* with a variable *\$var* from another data set *\$ds2* (matching rows according to the pair types they represent, i.e. using the variables l1 and l2). Typically, *\$ds2* will be an annotation database.

```

$ds1->add_variables($var); # assuming $var hasn't previously exist in $ds1
$dict = $ds2->dict($var);
$dict->unique
  or die "Not unique -- can't look up pair types.";
foreach $n (1 .. $ds1->size) {
  $row = $dict->lookup($ds1, $n);
  $ds1->set_cell($var, $n, $ds2->cell($var, $row))
    if defined $row;
}
undef $dict;

```

SEE ALSO

The `ucsfile` manpage for general information about UCS data sets and the data set file format, the `ucsexp` manpage for an introduction to UCS expressions (which are used extensively in the **UCS::DS::Memory** module) and wildcard patterns, the `UCS::Expression` manpage for information on how to compile UCS expressions, and the `UCS::DS` manpage for methods that manipulate the layout of a data set and its header information.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

■ UCS::DS::Format

ASCII-format data set or subset

SYNOPSIS

```

use UCS::DS::Memory;
use UCS::DS::Format;

$ds = new UCS::DS::Memory $filename; # needs in-memory representation

```

```

$formatter = new UCS::DS::Format $ds; # formatter object for data set $ds

$formatter->digits(6);                # number of significant digits

$formatter->mode("table");            # only mode so far

$formatter->pagelength(50);          # print in pages of 50 rows each
$formatter->pagelength(undef);       # print as single table

$formatter->vars($pattern, ...);     # select variables that will be shown

$formatter->print;                    # print formatted table on STDOUT
$formatter->print($filename);        # write to file or pipe

```

DESCRIPTION

This module provides a convenient method to format data sets as ASCII tables, which can then be used for viewing and printing. The formatter has to be applied to the in-memory representation implemented by the **UCS::DS::Memory** module. Its output is printed on STDOUT by default, but it can also be redirected to a file or pipe.

METHODS

\$formatter = new UCS::DS::Format \$ds;

Creates new formatter object for the data set *\$ds*, which must be a **UCS::DS::Memory** object. The formatter object should be used immediately after its creation and destroyed afterwards. When any changes are made in the data set *\$ds*, a new formatter has to be created.

\$formatter->digits(\$n);

Configure *\$formatter* to display approximately *\$n* significant digits for floating-point variables (data type `DOUBLE`). *\$n* must be at least 2.

\$formatter->mode("table");

The default mode `table` prints the data set in the form of a simple ASCII table with column headers. It is the only supported mode so far.

\$formatter->pagelength(\$rows);

Configure *\$formatter* to format data set in separate pages of *\$n* rows each. The individual pages are separated by a single blank line. Use of this option may improve the formatting quality, helps to avoid excessive columns widths, and reduces the delay before partial results can be displayed.

When *\$rows* is set to 0 or omitted, the entire data set is printed as a single table. This is also the default behaviour.

\$formatter->vars(\$pattern, ...);

Display only variables matching the specified wildcard patterns, in the specified order. This configuration option can also be used to change the ordering of the columns or display a variable in more than one column. Repeated calls to the **vars** method will overwrite, rather than add to, the previous selection.

\$formatter->print;

\$formatter->print(\$filename);

Format the data set with the specified options, and print the result on STDOUT. When the optional argument *\$filename* is specified, the output is redirected to this file or pipe.

SEE ALSO

See also the manpage of the PRINT utility, which is based on the **UCS::DS::Format** module.

COPYRIGHT

Copyright 2004 Stefan Evert.

This software is provided AS IS and the author makes no warranty as to its use and performance. You may use the software, redistribute and modify it under the same terms as Perl itself.

B.2 UCS/R

This section contains the full UCS/R documentation. The \LaTeX pages available within the R help system are slightly reformatted to match the layout of the thesis.

UCS/R documentation contents

Cbeta	269
Cgamma	270
EV	270
EVm	271
Ibeta	272
Igamma	273
Rbeta	274
Rgamma	275
UCS	277
VV	279
VVm	280
add.gams	281
add.jitter	282
add.ranks	283
am.key2var	284
binom.conf.interval	285
builtin.ams	286
builtin.gams	287
ds.find.am	289
eo.iso	290
eo.iso.diff	292
eo.legend	294
eo.mark	296
eo.par	297
eo.points	299
eo.setup	301
evaluation.file	303
evaluation.plot	304
evaluation.table	309
fzm	310
gam.helpers	312
gam.iso	313
gam.score	314
gamma.nbest	316
iaa.kappa	317
iaa.pta	318
lnre.goodness.of.fit	319
order.by.am	321
precision.recall	321
read.ds.gz	323
read.spectrum	324
spectrum.plot	325

ucs.library	326
ucs.par	327
write.lexstats	329
zm	330

Cbeta	<i>The Beta Function (sfunc)</i>
-------	----------------------------------

Description

Computes the (complete) Beta function and its base 10 logarithm.

Usage

```
Cbeta(a, b, log=FALSE)
```

Arguments

a, b	numeric vectors
log	if TRUE, returns the base 10 logarithm of the Beta function (default: FALSE)

Details

This is just a front-end to the built-in `beta` and `lbeta` functions, provided mainly for consistent naming. Note that the logarithmic version is scaled to base 10 logarithms, according to the UCS conventions.

Value

The Beta function with arguments (a, b), or its base 10 logarithm (if `log=TRUE`).

See Also

`beta`, `Ibeta`, `Rbeta`, `Cgamma`, `Igamma`, `Rgamma`

Examples

```
x <- 5
y <- 3
((x+y+1) * beta(x+1,y+1))^-1 # == choose(x+y, x)
```

Cgamma	<i>The Gamma Function (sfunc)</i>
--------	-----------------------------------

Description

Computes the (complete) Gamma function and its base 10 logarithm.

Usage

```
Cgamma(a, log=FALSE)
```

Arguments

a	a numeric vector
log	if TRUE, returns the base 10 logarithm of the Gamma function (default: FALSE)

Details

This is just a front-end to the built-in `gamma` and `lgamma` functions, provided mainly for consistent naming. Note that the logarithmic version is scaled to base 10 logarithms, according to the UCS conventions.

Value

The Gamma function evaluated at `a`, or its base 10 logarithm (if `log=TRUE`).

See Also

`gamma`, `Igamma`, `Rgamma`, `Cbeta`, `Ibeta`, `Rbeta`

Examples

```
Cgamma(5 + 1) # = factorial(5)
```

EV	<i>Expected Vocabulary Size of a LNRE Model (zm, fzm)</i>
----	---

Description

Computes the expected vocabulary size of a LNRE model (Baayen, 2001) at sample size N .

Usage

```
EV(model, N)
```

Arguments

<code>model</code>	an object of class "zm" or "fzm", representing a Zipf-Mandelbrot (ZM) or finite Zipf-Mandelbrot (fZM) LNRE model
<code>N</code>	a vector of positive integers, representing sample sizes

Details

The expected vocabulary size $E[V(N)]$ is the expected number of types at sample size N , according to the LNRE model `model` (see Baayen, 2001).

Value

a numeric vector of the same length as `N`

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

`zm`, `fzm`, `EVm`, `VV`, `VVm`

<code>EVm</code>	<i>Expected Frequency Spectrum of a LNRE Model (zm, fzm)</i>
------------------	--

Description

Computes the expected frequency spectrum, relative frequency spectrum, and conditional parameter distribution of a LNRE model (Baayen, 2001) at sample size N .

Usage

```
EVm(model, m, N, rho=1, relative=FALSE, ratio=FALSE, lower=TRUE)
```

Arguments

<code>model</code>	an object of class "zm" or "fzm", representing a Zipf-Mandelbrot (ZM) or finite Zipf-Mandelbrot (fZM) LNRE model
<code>m</code>	a vector of positive integers, representing frequency ranks
<code>N</code>	a vector of positive integers, representing sample sizes; either <code>m</code> or <code>N</code> should be a single number
<code>rho</code>	a vector of numbers in the range $[0, 1]$. If <code>length(rho) > 1</code> , both <code>m</code> and <code>N</code> should be single numbers. See below for details.
<code>relative</code>	if <code>TRUE</code> , computes the relative frequency spectrum (see below for details)
<code>ratio</code>	if <code>TRUE</code> , computes the ratio between consecutive elements in the expected frequency spectrum
<code>lower</code>	if <code>rho</code> is specified, controls whether the lower or upper conditional parameter distribution is computed

Details

The expected frequency spectrum consists of the numbers $E[V_m(N)]$, which stand for the expected number of types in frequency class m at sample size N , according to the LNRE model (see Baayen, 2001).

If `relative=TRUE`, the relative frequency spectrum $E[V_m(N)]/E[V(N)]$ is returned. If `ratio=TRUE`, the ratios between consecutive expected class sizes, $E[V_{m+1}(N)]/E[V_m(N)]$, are returned.

When `rho` is specified, the conditional parameter distribution $E[V_{m,\rho}(N)]$ is returned, i.e. the expected number of types in frequency class m at sample size N with probability parameter $\pi \leq \rho$. If `relative=TRUE`, the expected proportion $E[R_{m,\rho}] \approx E[V_{m,\rho}(N)]/E[V(N)]$ is returned instead. With `lower=FALSE`, computes the upper conditional parameter distribution $E[V_{m,>\rho}(N)]$ or proportion $E[R_{m,>\rho}(N)]$. See Evert (2004, Ch. 4) for details.

Value

a numeric vector of appropriate length (determined either by `m`, `N`, or `rho`)

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`zm`, `fzm`, `EVm`, `VV`, `VVm`

Ibeta

The Incomplete Beta Function (sfunc)

Description

Computes the incomplete Beta function and its inverse. The Beta value can be scaled to a base 10 logarithm.

Usage

`Ibeta(x, a, b, log=FALSE)`

`Ibeta.inv(y, a, b, log=FALSE)`

Arguments

a, b	non-negative numeric vectors, the parameters of the incomplete Beta function
x	a numeric vector with values in the range [0, 1], the point at which the incomplete Beta function is evaluated
y	a numeric vector, the values of the incomplete Beta function (or their base 10 logarithms if log=TRUE)
log	if TRUE, the Beta values are base 10 logarithms (default: FALSE)

Details

The incomplete Beta function is defined by the Beta integral

$$B(x; a, b) = \int_0^x t^{a-1} (1-t)^{b-1} dt$$

Value

Ibeta returns the incomplete Beta function with parameters (a,b) evaluated at point x.

Ibeta.inv returns the point x at which the incomplete Beta function with parameters (a,b) evaluates to y.

See Also

Cgamma, Igamma, Rgamma, Cbeta, Rbeta

Igamma

The Incomplete Gamma Function (sfunc)

Description

Computes the incomplete Gamma function and its inverse. Both the lower and the upper incomplete Gamma function are supported, and the Gamma value can be scaled to a base 10 logarithm.

Usage

```
Igamma(a, x, lower=TRUE, log=FALSE)
```

```
Igamma.inv(a, y, lower=TRUE, log=FALSE)
```

Arguments

a	a non-negative numeric vector, the parameter of the incomplete Gamma function
x	a non-negative numeric vector, the point at which the incomplete Gamma function is evaluated
y	a numeric vector, the values of the incomplete Gamma function (or their base 10 logarithms if log=TRUE)
lower	if TRUE, computes the lower incomplete Gamma function (default). Otherwise, computes the upper incomplete Gamma function.
log	if TRUE, the Gamma values are base 10 logarithms (default: FALSE)

Details

The upper incomplete Gamma function is defined by the Gamma integral

$$\Gamma(a, x) = \int_x^{\infty} t^{a-1} e^{-t} dt$$

The lower incomplete Gamma function is defined by the complementary Gamma integral

$$\gamma(a, x) = \int_0^x t^{a-1} e^{-t} dt$$

Value

Igamma returns the (lower or upper) incomplete Gamma function with parameter a evaluated at point x.

Igamma.inv returns the point x at which the (lower or upper) incomplete Gamma function with parameter a evaluates to y.

See Also

Cgamma, Rgamma, Cbeta, Ibeta, Rbeta

Rbeta

The Regularized Beta Function (sfunc)

Description

Computes the regularized Beta function and its inverse. The Beta value can be scaled to a base 10 logarithm.

Usage

Rbeta(x, a, b, log=FALSE)

Rbeta.inv(y, a, b, log=FALSE)

Arguments

a, b	non-negative numeric vectors, the parameters of the regularized Beta function
x	a numeric vector with values in the range [0, 1], the point at which the regularized Beta function is evaluated
y	a numeric vector, the values of the regularized Beta function (or their base 10 logarithms if log=TRUE)
log	if TRUE, the Beta values are base 10 logarithms (default: FALSE)

Details

The regularized Beta function scales the incomplete Beta function to the interval [0, 1], by dividing through $B(a, b)$, i.e.

$$I(x; a, b) = \frac{B(x; a, b)}{B(a, b)}$$

Value

Rbeta returns the regularized Beta function with parameters (a,b) evaluated at point x.

Rbeta.inv returns the point x at which the regularized Beta function with parameters (a,b) evaluates to y.

See Also

Cgamma, Igamma, Rgamma, Cbeta, Ibeta

Rgamma

The Regularized Gamma Function (sfunc)

Description

Computes the regularized Gamma function and its inverse. Both the lower and the upper regularized Gamma function are supported, and the Gamma value can be scaled to a base 10 logarithm.

Usage

Rgamma(a, x, lower=TRUE, log=FALSE)

Rgamma.inv(a, y, lower=TRUE, log=FALSE)

Arguments

a	a non-negative numeric vector, the parameter of the incomplete Gamma function
x	a non-negative numeric vector, the point at which the incomplete Gamma function is evaluated
y	a numeric vector, the values of the regularized Gamma function (or their base 10 logarithms if log=TRUE)
lower	if TRUE, computes the lower regularized Gamma function (default). Otherwise, computes the upper regularized Gamma function.
log	if TRUE, the Gamma values are base 10 logarithms (default: FALSE)

Details

The regularized Gamma functions scale the corresponding incomplete Gamma functions to the interval $[0, 1]$, by dividing through $\Gamma(a)$. Thus, the lower regularized Gamma function is given by

$$P(a, x) = \frac{\gamma(a, x)}{\Gamma(a)}$$

and the upper regularized Gamma function is given by

$$Q(a, x) = \frac{\Gamma(a, x)}{\Gamma(a)}$$

Value

Rgamma returns the (lower or upper) regularized Gamma function with parameter a evaluated at point x.

Rgamma.inv returns the point x at which the (lower or upper) regularized Gamma function with parameter a evaluates to y.

See Also

Cgamma, Igamma, Cbeta, Ibeta, Rbeta

Examples

```
## P(X >= k) for Poisson distribution with mean alpha
alpha <- 5
k <- 10
Rgamma(k, alpha) # == ppois(k-1, alpha, lower=FALSE)
```

Description

UCS/R consists of a set of R libraries related to the visualisation of cooccurrence data and the evaluation of association measures. The current functionality includes: evaluation graphs for association measures (in terms of precision and recall), measures for inter-annotator agreement, and two population models for word frequency distributions.

Usage

```
source("/path/to/UCS/System/R/lib/ucs.R")
ucs.library()
```

Details

UCS/R is initialised by sourcing the file ‘ucs.R’ in the ‘lib/’ subdirectory of the UCS/R directory tree. This will make the UCS/R documentation available in the R process and provide the `ucs.library` command, which is used to load individual UCS/R modules. Enter `ucs.library()` now to display a list of available modules (see the `ucs.library` manpage for details).

Currently, the following modules are available. The listing below also indicates the most important manpages for each module. Throughout the documentation, it is assumed that you are familiar with the UCS/Perl naming conventions and data set file format.

- **sfunc: Special Mathematical Functions**

Convenience interfaces to the Gamma function (`Cgamma`), the incomplete (and regularized) Gamma function and its inverse (`Igamma`, `Rgamma`), the Beta function (`Cbeta`), the incomplete (and regularized) Beta function and its inverse (`Ibeta`, `Rbeta`), and binomial confidence intervals (`binom.conf.interval`).

All these functions are computed from the `pgamma` and `pbeta` distributions (and the corresponding quantile functions) in the standard library of R.

- **base: Basic Functions for Loading and Managing UCS data sets**

This module provides functions for loading UCS data set files (`read.ds.gz`), listing annotated association measures (`ds.find.am`, `am.key2var`), ranking by association scores (`order.by.am`, `add.ranks`), and computing precision/recall tables for the evaluation of association measures (`precision.recall`).

The module also includes a listing of all built-in association measures in the UCS/Perl system, including add-on packages (`builtin.ams`).

- **plots: Evaluation Graphs for Association Measures**

This module plots precision-, recall-, and precision-by-recall graphs for the empirical evaluation of association measures (all combined in a single function, `evaluation.plot`). The graphs are highly configurable, either locally in each function call or by setting global defaults (`ucs.par`). The `evaluation.plot` function

supports confidence intervals, significance tests for result differences, and evaluation based on random samples (see Evert, 2004, Ch. 5). A simple text-mode version of the precision/recall-based evaluation is provided by the `evaluation.table` function in the base module.

- **iaa: Measures of Inter-Annotator Agreement**

Computes Cohen's kappa statistic with standard deviation (Fleiss, Cohen & Everitt, 1969) or confidence interval for proportion of true agreement (Krenn, Evert & Zinsmeister, 2004) from a 2×2 contingency table (see `iaa.kappa` and `iaa.pta`)

- **gam: Generalised association measures (GAMs)**

This module implements extensions of several association measures to continuous functions on a real-valued coordinate space (generalised association measures, GAMs). For details and terminology, please refer to Evert (2004, Sec. 3.3). The functions in this module compute GAM scores and iso-surfaces in standard or ebo-coordinates, and can add jitter to a given data set. New GAMs can easily be added with the `register.gam` function. Relevant help pages are `builtin.gams`, `gam.score`, `gam.iso`, `gamma.nbest`, `add.jitter`, `add.gams`, `add.ebo`, and `gam.helpers`.

- **eo: Visualise GAMs in the (e,o) plane**

This module implements 2-D visualisation of data sets and GAMs by plotting point clouds and iso-lines in the (e,o) plane (see Evert 2004, Sec. 3.3). The recommended starting point is the documentation of the `eo.setup` function, which initialises a new (e,o) plot. Other relevant help pages are `eo.par`, `eo.points`, `eo.iso`, `eo.iso.diff`, `eo.legend` and `eo.mark`.

- **lexstats: Utilities for lexical statistics**

This module contains miscellaneous utility functions for word frequency distributions, including: an interface to file formats used by the `lexstats` software (Baayen 2001); a range of common plots; goodness-of-fit evaluation for LNRE populations models (cf. the `zm` and `fzm` modules below). Currently, the most useful functions in this module are `read.spectrum`, `spectrum.plot`, and `lnre.goodness.of.fit`.

- **zm: The Zipf-Mandelbrot (ZM) Population Model**

This module implements a simple population model for word frequency distributions (Baayen, 2001) based on the Zipf-Mandelbrot law. See (Evert, 2004a) for details. Relevant help pages are `zm`, `EV`, `EVm`, `VV`, `VVm`, `write.lexstats`, and `lnre.goodness.of.fit`.

- **fzm: The Finite Zipf-Mandelbrot (fZM) Population Model**

This module implements the finite Zipf-Mandelbrot model, an extension of the ZM model (Evert, 2004a). Relevant help pages are `fzm`, `EV`, `EVm`, `VV`, `VVm`, `write.lexstats`, and `lnre.goodness.of.fit`.

The command `help(package=UCS)` will give you a full index of available UCS/R help pages. Use `help.search()` for full-text search.

Note

The correct source path for the file 'ucs.R' can be set automatically with the UCS/Perl tool `ucs-config`. Simply insert the statement

```
source("ucs.R")
```

on a separate line in your R script file (say, ‘my-script.R’) and run the shell command

```
ucs-config my-script.R
```

References

- Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.
- Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.
- Evert, Stefan (2004a). A simple LNRE model for random character sequences. In *Proceedings of JADT 2004*, Louvain-la-Neuve, Belgium, pages 411–422.
- Fleiss, Joseph L.; Cohen, Jacob; Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, **72**(5), 323–327.
- Krenn, Brigitte; Evert, Stefan; Zinsmeister, Heike (2004). Determining intercoder agreement for a collocation identification task. In preparation.

See Also

ucs.library, the UCS/R tutorial (‘tutorial.R’ in the ‘script’ subdirectory) and the UCS/Perl documentation.

VV	<i>Variance of the Vocabulary Size of a LNRE Model (zm, fzm)</i>
----	--

Description

Computes the variance of the vocabulary size of a LNRE model (Baayen, 2001) at sample size N .

Usage

```
VV(model, N)
```

Arguments

model	an object of class "zm" or "fzm", representing a Zipf-Mandelbrot (ZM) or finite Zipf-Mandelbrot (fZM) LNRE model
N	a vector of positive integers, representing sample sizes

Details

The variance $V[V(N)]$ is computed according to Baayen (2001, 120f). See the EV help page for some more information on the vocabulary size $V(N)$.

Value

a numeric vector of the same length as N

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

zm, fzm, VVm, EV, EVm

VVm	<i>Variances of the Frequency Spectrum of a LNRE Model (zm, fzm)</i>
-----	--

Description

Computes the variances of the frequency spectrum and conditional parameter distribution of a LNRE model (Baayen, 2001) at sample size N .

Usage

```
VVm(model, m, N, rho=1, relative=FALSE, lower=TRUE)
```

Arguments

model	an object of class "zm" or "fzm", representing a Zipf-Mandelbrot (ZM) or finite Zipf-Mandelbrot (fZM) LNRE model
m	a vector of positive integers, representing frequency ranks
N	a vector of positive integers, representing sample sizes; either m or N should be a single number
rho	a vector of numbers in the range [0, 1]. If <code>length(rho) > 1</code> , both m and N should be single numbers. See below for details.
relative	if TRUE, computes variances for the relative conditional parameter distribution (see below for details). May only be used when rho is specified.
lower	if rho is specified, controls whether variances are computed for the lower or for the upper conditional parameter distribution

Details

The variance $V[V_m(N)]$ is computed according to Baayen (2001, 120f).

When rho is specified, the variances of the conditional parameter distribution $V[V_{m,\rho}(N)]$ or the corresponding proportions $V[R_{m,\rho}(N)]$ are returned, depending on the value of `relative`. With `lower=FALSE`, computes variances for the upper conditional parameter distribution $V[V_{m,>\rho}(N)]$ or proportion $V[R_{m,>\rho}(N)]$. See Evert (2004, Ch. 4) for details.

The EVm help page provides more information about $V_m(N)$, $V_{m,\rho}(N)$, $R_{m,\rho}(N)$, $V_{m,>\rho}(N)$ and $R_{m,>\rho}(N)$.

Note that this function does *not* compute variances for the relative frequency spectrum ($V[V_m(N)/V(N)]$) or the ratio between consecutive spectrum elements ($V[V_{m+1}(N)/V_m(N)]$).

Value

a numeric vector of appropriate length (determined either by `m`, `N`, or `rho`)

References

- Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.
- Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`zm`, `fzm`, `VVm`, `EV`, `EVm`

`add.gams`

Annotate Data Set with GAM Scores (gam)

Description

Annotates data set with GAM scores, possibly overwriting existing scores of a standard AM. Optionally, *jitter* annotated in the data set can be taken into account when computing the scores.

Usage

```
add.gams(ds, names, jitter=FALSE)
```

Arguments

- | | |
|-------------------------|---|
| <code>ds</code> | a UCS data set object |
| <code>name</code> | a character vector specifying the names of generalised association measures to be annotated in the data set |
| <code>add.jitter</code> | if TRUE, random jitter (which must be annotated in the data set) is added to the frequency signatures before computing GAM scores (see details below) |

Details

The `add.gams` function uses the standard variable names for AM scores (e.g. `am.t.score` for the `t.score` measure), so that existing scores for the respective standard AMs in the data set will be overwritten. Rankings for the GAM scores can then be computed in the normal way using the `add.ranks` function.

With `jitter=TRUE`, a small amount of random jitter is added to the frequency signatures in order to avoid ties in the rankings and facilitate visualisation of the data set. The necessary jitter vectors have to be stored in special variables in the data set first, which is most easily achieved with the `add.jitter` function.

Value

a copy of the data set `ds` annotated with GAM scores for the specified measures

See Also

`gam.score`, `gam.iso`, `builtin.gams`, `add.ranks`, `add.jitter`

Examples

```
ds <- add.ranks(add.gams(ds, c("t.score", "chi.squared.corr")))

ds <- add.jitter(ds)
gam.names <- ds.find.am(ds)
gam.names <- gam.names[ is.builtin.gam(gam.names) ]
ds <- add.gams(ds, gam.names, jitter=TRUE)
ds <- add.ranks(ds, gam.names, randomise=FALSE, overwrite=TRUE)
```

<code>add.jitter</code>	<i>Random Jitter for Frequency Signatures in Data Set (gam)</i>
-------------------------	---

Description

Add random *jitter* to the frequency signatures in a data set, in order to avoid ties in rankings according to GAM scores and to facilitate visualisation of the data set with `eo` and `ebo` plots. The `add.ebo` function is used to re-compute `ebo`-coordinates from the jittered frequency signatures.

Usage

```
add.jitter(ds, amount=0.5, overwrite=FALSE)
```

```
has.jitter(ds, fail=FALSE)
```

```
add.ebo(ds, jitter=FALSE)
```

Arguments

<code>ds</code>	a UCS data set object
<code>amount</code>	amount of jitter to be added; the jitter vector for each coordinate (<code>f</code> , <code>f1</code> , <code>f2</code>) has a uniform distribution over the range <code>[-amount, +amount]</code>
<code>overwrite</code>	if TRUE, overwrite existing jitter vectors in the data set
<code>fail</code>	if TRUE, abort with an error message unless the data set contains jitter vectors
<code>jitter</code>	if TRUE, use the jittered frequency signatures to compute <code>ebo</code> -coordinates (default: unjittered integer frequencies)

Details

The `add.jitter` function adds jitter vectors for the joint and marginal frequencies (`f`, `f1`, `f2`) to a data set, i.e. uniformly distributed random numbers in the range `[-amount, +amount]`. These vectors are stored in variables `x.jitter.f`, `x.jitter.f1` and `x.jitter.f2`, where they can be used by `add.ebo`, `add.gams` and other functions. `has.jitter` tests for the presence of these variables.

`add.ebo` computes ebo-coordinates from the frequency signatures and stores them in the standard variables `e`, `b`, `o`. Unlike the values computed with UCS/Perl tools, `add.ebo` uses jitter vectors in this computation when the option `jitter=TRUE` is passed.

Value

`add.jitter` and `add.ebo` return a copy of the data set `ds` with the request variables added. `has.jitter` returns `TRUE` if the jitter variables are present in `ds`, and `FALSE` otherwise.

See Also

`add.gams`, `gamma.nbest`

Examples

```
ds <- add.jitter(ds, amount=0.2)

ds <- add.ebo(ds, jitter=TRUE) # recompute ebo coordinates with jitter
```

<code>add.ranks</code>	<i>Compute Rankings for Annotated Association Measures (base)</i>
------------------------	---

Description

Add rankings (with or without ties) for specified association measures to a data set object.

Usage

```
add.ranks(ds, keys=ds.find.am(ds), randomise=TRUE, overwrite=TRUE)
```

Arguments

<code>ds</code>	a UCS data set object
<code>keys</code>	a character vector giving the names of one or more association measures. When it is omitted, rankings are computed for all annotated measures.
<code>randomise</code>	if <code>TRUE</code> , ties are broken randomly (default). Otherwise, tied rows are assigned the same rank, which is the first free one (as in the Olympic Games). See below for prerequisites.
<code>overwrite</code>	if <code>TRUE</code> , existing rankings are overwritten (default). Otherwise, association measures for which ranks are already annotated are silently skipped. If you modify association scores within R, be sure to call <code>add.ranks</code> with <code>overwrite=TRUE</code> .

Details

Since `add.ranks` is based on the `order.by.am` function, the prerequisites are the same: the data set must contain association scores for the random measure if `randomise=TRUE` and an `id` variable if `randomise=FALSE`. See the `order.by.am` manpage for further information.

Value

Invisibly returns a copy of `ds` annotated with the requested rankings. The rankings are stored in variables `r.*`, where `*` stands for the name of an association measure (according to the UCS naming conventions, cf. the `am.key2var` manpage).

See Also

`order.by.am`, `am.key2var`, `ds.find.am`, `read.ds.gz`

Examples

```
## from the UCS/R tutorial
GLAW <- read.ds.gz("glaw.scores.ds.gz")
GLAW <- add.ranks(GLAW)

## combine into single command
GLAW <- add.ranks(read.ds.gz("glaw.scores.ds.gz"))
```

`am.key2var`

UCS Variable Names for Association Scores and Rankings (base)

Description

These functions implement the UCS naming conventions for variables storing association scores and the corresponding ranking. `is.valid.key` checks whether a given string is valid as a name for an association measure. `am.key2var` translates a valid AM name into the corresponding variables (for scores or ranking), and `am.var2key` extracts the AM name from such a variable.

Usage

```
is.valid.key(key, warn=FALSE)

am.key2var(key, rank=FALSE)

am.var2key(var)
```

Arguments

<code>key</code>	a character vector, giving the names of one or more association measures
<code>var</code>	a character vector of variable names, which must be either association scores or rankings (but both types can be mixed in the vector)
<code>warn</code>	if TRUE, issues a warning if the vector <code>key</code> contains invalid AM names. All invalid entries are listed in the warning message.
<code>rank</code>	if TRUE, return names of the ranking variables corresponding to the specified association measures. otherwise, return names of variables for association scores.

Value

`is.valid.key` returns a logical vector, `am.var2key` returns a list of AM names (“keys”), and `am.key2var` returns a list of variable names (either for association scores or rankings, depending on the `rank` parameter).

See Also

`builtin.ams` for information about built-in association measures, and the `ucsfile` manpage in UCS/Perl for a description of the UCS naming conventions (enter the shell command `ucsdoc ucsfile`).

Examples

```
am.key2var(c("t.score", "MI"), rank=TRUE)
am.var2key(c("am.t.score", "r.MI"))
```

`binom.conf.interval`

Binomial Confidence Intervals

Description

Computes confidence intervals for the success probability of a binomial distribution efficiently. Unlike `binom.test`, this function can be applied to vectors.

Usage

```
binom.conf.interval(k, size, limit=c("lower","upper"),
  conf.level=0.05, one.sided=FALSE)
```

Arguments

<code>k</code>	a vector of non-negative integers. Each element represents the number of successes out of <code>size</code> trials, i.e. the observed value of a random variable with binomial distribution.
<code>size</code>	a vector of positive integers. Each element represents the number of trials of a binomial distribution.
<code>limit</code>	if "upper", the upper boundaries of the confidence intervals are returned. If "lower", the lower boundaries are returned. Note that this works both for one-sided and for two-sided confidence intervals.
<code>conf.level</code>	the required confidence level, or rather the significance level of the corresponding binomial test (note that this behaviour differs from the built-in <code>binom.test</code> function). The default <code>conf.level=0.05</code> stands for 95% confidence.
<code>one.sided</code>	if TRUE, computes one-sided confidence interval (either lower or upper, depending on the value of <code>limit</code>). If FALSE, a two-sided confidence interval is computed (default).

Details

If `one.sided=TRUE`, the underlying test is one-sided (with alternative "less" or "greater", depending on the `limit` parameter), and the non-trivial boundary of the confidence interval is returned.

If `one.sided=FALSE`, the underlying test is two-sided and the requested boundary of the two-sided confidence interval is returned. For efficiency reasons, the `binom.conf.interval` function cheats a little and computes one-sided confidence intervals with significance level `conf.level / 2`.

Value

A numeric vector with the requested boundary of confidence intervals for the unknown success probabilities of binomial variables.

See Also

`binom.test`

`builtin.ams`

UCS/Perl Built-in Association Measures (base)

Description

`builtin.ams` returns a character vector listing the built-in association measures of the UCS/Perl system (including the standard add-on packages), `is.builtin.am` checks whether a specified measure belongs to this set, and `am.key2desc` returns a short description of the specified measure.

Usage

```

builtin.ams()

is.builtin.am(key)

am.key2desc(key)

```

Arguments

`key` a character vector specifying the names of one or more association measures

Value

`builtin.ams` returns a character vector containing the names of all built-in association measures, `is.builtin.am` returns a logical vector, and `am.key2desc` returns a character vector with a short description of each of the measures in `key`.

See Also

The information provided by these functions is obtained from the UCS/Perl tool `ucs-list-am`. See the `ucsam` manpage in UCS/Perl for further information about built-in association measures (using the shell command `ucsd doc ucsam`).

Examples

```

print(builtin.ams())
am.key2desc("chi.squared.corr")

```

<code>builtin.gams</code>	<i>Built-in Generalised Association Measures (gam)</i>
---------------------------	--

Description

List available GAMs (generalised association measures) that can be computed with functions such as `gam.score`, `add.gams` and `gam.iso`, or test whether a specific GAM is available. Additional GAMs can be defined with the `register.gam` function.

Usage

```

builtin.gams()

is.builtin.gam(names)

register.gam(name, equation, iso.equation=NULL)

```

Arguments

<code>names</code>	a character vector specifying the names of GAMs whose availability is tested
<code>name</code>	a single character string specifying the name of a GAM that is defined or re-defined
<code>equation</code>	a function that computes GAM scores from standard or ebo-coordinates (see below for details)
<code>iso.equation</code>	an optional function that computes iso-surfaces in standard or ebo-coordinates (see below for details)

Details

The names of built-in GAMs are identical to those of the corresponding standard AMs (e.g. `t.score` and `chi.squared.corr`).

The `equation` argument of `register.gam`, i.e. the equation defining a new GAM), must be a function with the signature `(o, e, b, f, f1, f2, N)`. This function can compute GAM scores either from the ebo-coordinates `e, b, o` or from the standard coordinates `f, f1, f2, N`. It is always invoked with all seven arguments, which are guaranteed to be vectors of the same length, and must return a vector of corresponding GAM scores.

When an explicit equation for iso-surfaces $\{g = \gamma\}$ exists, it can be made available through the optional argument `iso.equation`, which expects a function with the signature `(gamma, e, b, f1, f2, N)`. Again, all six arguments are guaranteed to be vectors of the same length, and the function must return the corresponding `o` (or `f`) coordinates that satisfy the condition $g(o, e, b) = \gamma$ (or $g(f, f_1, f_2, N) = \gamma$). When the `iso.equation` function is available for a GAM, it will be used by `gam.iso` for greater speed and accuracy. Otherwise, the iso surface is determined by a binary search algorithm (which has a unique solution for any semi-sound GAM).

The signatures of the `equation` and `iso.equation` functions are checked by `register.gam`, which will abort with an error message if they are not correct.

Value

`builtin.gams` returns a character vector listing the names of available GAMs. `is.builtin.gam` returns a logical vector indicating which of the GAMs in the vector `names` are available.

See Also

`builtin.ams`, `gam.score`, `add.gams`, `gam.iso`, `gam.helpers`

Examples

```
print(builtin.gams())

all(is.builtin.gam(c("MI", "t.score", "chi.squared")))

register.gam("MI5",
  eq = function (o, e, b, f, f1, f2, N) { log10(o^5 / e) },
  iso = function (gamma, e, b, f1, f2, N) { 10^(gamma/5) * e^(1/5) })
```

<code>ds.find.am</code>	<i>List Association Scores and Rankings in Data Set (base)</i>
-------------------------	--

Description

`am.in.ds` tests whether a specified association measure is annotated in a data set, `ds.find.am` lists all annotated association measures, and `ds.match.am` searches the data set for AMs whose names may be abbreviated to a unique prefix. All three functions look either for association scores or for rankings.

Usage

```
am.in.ds(ds, keys, rank=FALSE, fail=FALSE)
```

```
ds.find.am(ds, rank=FALSE)
```

```
ds.match.am(ds, abbrevs, rank=FALSE)
```

Arguments

<code>ds</code>	a UCS data set, read from a data set file with the <code>read.ds.gz</code> function
<code>keys</code>	a character vector of AM names
<code>abbrevs</code>	a character vector of AM names, each of which may be abbreviated to a unique prefix (within the data set)
<code>rank</code>	if TRUE, the functions look for annotated rankings; otherwise, they look for annotated association scores (default)
<code>fail</code>	if TRUE, the function aborts with an error message unless all specified AMs are annotated in the data set

Details

If any of the `abbrevs` do not have a unique match in the data set, `ds.match.am` aborts with an error message (listing all strings that failed to match uniquely).

Value

`am.in.ds` returns a logical vector of the same length as `keys`. `ds.find.am` and `ds.match.am` return a character vector containing the names of the annotated association measures.

See Also

`read.ds.gz`, `am.var2key`

Examples

```
GLAW <- read.ds.gz("glaw.scores.ds.gz")
print(ds.find.am(GLAW))
```

`eo.iso`*Draw Iso-Line of a GAM in the (e,o) Plane (eo)*

Description

Draw an iso-line of a generalised association measure (GAM) in the (e,o) plane, either for a specified cutoff threshold γ or an n-best iso-line for a given data set `ds`. Optionally, the corresponding acceptance region can be shaded or filled with solid colour.

Usage

```
eo.iso(gam, gamma=0, b=1, N=1e6, n.best=NULL, ds=NULL,
       style=1, fill=solid, solid=FALSE,
       steps=eo.par("steps"), jitter=eo.par("jitter"), bw=bw,
       col=eo.par("col"), lty=eo.par("lty"), lwd=eo.par("lwd"),
       angle=eo.par("angle"), density=eo.par("density"),
       solid.col=eo.par("solid"))
```

Arguments

- | | |
|---------------------------------------|---|
| <code>gam</code> | a character string giving the name of a generalised association measure (GAM). Use the function <code>builtin.gams</code> from the <code>gam</code> module to obtain a list of available GAMs. |
| <code>gamma</code> | a cutoff threshold that determines the iso-line to be drawn (by the implicit equation $\{g = \gamma\}$). Use the <code>n.best</code> and <code>ds</code> parameters instead of <code>gamma</code> in order to obtain an n-best iso-line for the data set <code>ds</code> . |
| <code>b</code> , <code>N</code> | optional balance (<code>b</code>) and sample size (<code>N</code>) parameters for GAMs that are not central or size-invariant, respectively. The default <code>b=1</code> yields the centralised version of a non-central GAM (for details, see Evert 2004, Sec. 3.3) |
| <code>n.best</code> , <code>ds</code> | When these parameters are specified, the cutoff threshold <code>gamma</code> will automatically be determined so as to yield an n-best acceptance region for the data set <code>ds</code> . |
| <code>jitter</code> | If <code>TRUE</code> , use jittered coordinates for computing the n-best cutoff threshold (see above). In this case, the data set has to be annotated with the <code>add.jitter</code> function first. |
| <code>style</code> | an integer specifying the style (colour, line type and width) in which iso-lines will be drawn. The number of styles available depends on the global parameter settings (<code>eo.par</code>). The "factory settings" define 5 different styles for iso-lines. |
| <code>fill</code> | If <code>TRUE</code> , fill in the acceptance region bounded by the given iso-line with shading lines, according to the chosen <code>style</code> and <code>bw</code> mode. See <code>eo.par</code> for details on shading styles. |

<code>solid</code>	If TRUE, fill the acceptance region with solid colour rather than shading lines, also according to the chosen style and bw mode. Setting <code>solid=TRUE</code> implies <code>fill=TRUE</code> .
<code>steps</code>	an integer specifying how many equidistant steps are used for drawing iso-lines. The default value is set with <code>eo.par</code> .
<code>bw</code>	If TRUE, the iso-lines are drawn in B/W mode, otherwise in colour mode. This parameter defaults to the state specified with the initial <code>eo.setup</code> call, but can be overridden manually.
<code>col, lty, lwd</code>	can be used to override the default style parameters for iso-lines, which are determined automatically from the global settings (<code>eo.par</code>) according to the selected style and bw mode.
<code>angle, density</code>	can be used to override the default style parameters for shaded acceptance region, which are determined automatically from the global settings (<code>eo.par</code>) according to the selected style and bw mode.
<code>solid.col</code>	can be used to override the default colour for solid filled acceptance regions, which is determined automatically from the global settings (<code>eo.par</code>) according to the selected style and bw mode.

Details

See the `eo.setup` help page for a description of the general procedure used to create (e,o) plots. This help page also has links to other (e,o) plotting functions. The "factory setting" styles are described on the `eo.par` help page.

The cutoff threshold γ can either be specified explicitly (with the `gamma` parameter) or implicitly as an n-best threshold (with `n.best`, `ds`, and optional `jitter`). The latter method produces the same result as

```
gam.iso(gam, gamma=gamma.nbest(ds, gam, n.best, jitter), ...)
```

Visualisation by (e,o) iso-lines is most suitable for GAMs that are both central and size-invariant (see Evert 2004, Sec. 3.3). For non-central measures, the `eo.iso` function uses a balance value of $b = 1$, yielding a centralised version of the GAM. Note that many non-central GAMs (especially those based on statistical tests, such as `log.likelihood` and `chi.squared`) have only a weak dependency on the balance b , so that their centralised iso-surfaces (i.e. extrusions of the iso-lines along the b-axis) are very similar to the original iso-surfaces. Other GAMs (most notably Dice and similar measures) are highly dependent on b , though. For measures that are not size-invariant, the sample size is arbitrarily set to $N = 10^6$, which is in a realistic range for real-life data sets. You may wish to modify the default value in order to match a data set shown in the plot (this is *not* done automatically when the `ds` parameter is specified), or to demonstrate the dependency of iso-lines on N .

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`eo.par`, `eo.setup`, `eo.iso.diff`

Examples

```
## an example can be found on the "eo.setup" help page
```

<code>eo.iso.diff</code>	<i>Highlight Differences between Two Acceptance Regions in the (e,o) Plane (eo)</i>
--------------------------	---

Description

Compare the acceptance regions of two GAMs by shading the two difference sets (cf. Evert 2004, Sec. 5.2.2) in different fill styles. This function should be followed by two `eo.iso` calls to draw the iso-lines bounding the difference regions.

Usage

```
eo.iso.diff(gam1, gam2, gamma1=0, gamma2=0, b=1, N=1e6,
            n.best1=NULL, n.best2=NULL, ds=NULL,
            style1=4, style2=5, solid=FALSE, bw=bw,
            steps=eo.par("steps"), jitter=eo.par("jitter"),
            col1=eo.par("col"), angle1=eo.par("angle"),
            density1=eo.par("density"), solid.col1=eo.par("solid"),
            col2=eo.par("col"), angle2=eo.par("angle"),
            density2=eo.par("density"), solid.col2=eo.par("solid"))
```

Arguments

- `gam1`, `gam2` character strings giving the names of two generalised association measures (GAMs). Use the function `builtin.gams` from the `gam` module to obtain a list of available GAMs.
- `gamma2`, `gamma1` cutoff thresholds that determines the two acceptance regions ($\{g_1 = \gamma_1\}$ and $\{g_2 = \gamma_2\}$) to be compared. You can use `n.best` and `ds` parameters (see below) to compute n-best thresholds automatically.
- `b`, `N` optional balance (`b`) and sample size (`N`) parameters for GAMs that are not central or size-invariant, respectively. The default `b=1` yields the centralised version of a non-central GAM (for details, see Evert 2004, Sec. 3.3). Note that the same values are used for both GAMs.

<code>n.best1, n.best2, ds</code>	When <code>n.best1</code> is specified, the cutoff threshold <code>gamma1</code> will automatically be determined so as to yield an n-best acceptance region for the data set <code>ds</code> . In the same way, <code>n.best2</code> computes <code>gamma2</code> as an n-best acceptance threshold. Note that the data set <code>ds</code> is used for both n-best thresholds.
<code>jitter</code>	If <code>TRUE</code> , use jittered coordinates for computing n-best cutoff thresholds (see above). In this case, the data set has to be annotated with the <code>add.jitter</code> function first.
<code>style1, style2</code>	integer values specifying fill styles for the two difference regions. <code>style1</code> is used for the region D_1 of the (e,o) plane accepted by <code>gam1</code> but not <code>gam2</code> , and <code>style2</code> for the region D_2 accepted by <code>gam2</code> but not <code>gam1</code> . Style parameters include the colour, angle and density of shading lines, or the solid fill colour if <code>solid=TRUE</code> . See the <code>eo.par</code> help page for more information about available fill styles.
<code>solid</code>	If <code>TRUE</code> , fill the difference regions with solid colour rather than shading lines, also according to the chosen styles and bw mode.
<code>bw</code>	If <code>TRUE</code> , the regions are drawn in B/W mode, otherwise in colour mode. This parameter defaults to the state specified with the initial <code>eo.setup</code> call, but can be overridden manually.
<code>steps</code>	an integer specifying how many equidistant steps are used for the (combined) boundaries of the difference regions. The default value is set with <code>eo.par</code> .
<code>col1, col2</code>	can be used to override the default colours for shading lines, which are determined automatically from the global settings (<code>eo.par</code>) according to the selected styles and bw mode.
<code>angle1, angle2</code>	can be used to override the default angles of shading lines, which are determined automatically from the global settings (<code>eo.par</code>) according to the selected styles and bw mode.
<code>density1, density2</code>	can be used to override the default densities of shading lines, which are determined automatically from the global settings (<code>eo.par</code>) according to the selected styles and bw mode.
<code>solid.col1, solid.col2</code>	can be used to override the default solid fill colours (with <code>solid=TRUE</code>), which are determined automatically from the global settings (<code>eo.par</code>) according to the selected styles and bw mode.

Details

See the `eo.setup` help page for a description of the general procedure used to create (e,o) plots. This help page also has links to other (e,o) plotting functions. The "factory setting" styles are described on the `eo.par` help page.

See the `eo.iso` help page for details about iso-lines, acceptance regions and n-best cutoff thresholds.

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`eo.par`, `eo.setup`, `eo.iso`

Examples

```
## setup code (see "eo.setup" example for a detailed explanation)
ucs.library("eo")
ds <- add.jitter(read.ds.gz("dickens.ds.gz"))
select <- rbinom(nrow(ds), 1, .1) == 1
ds <- ds[select,]

## comparison of 300-best acceptance regions for Poisson and MI measures
eo.setup(xlim=c(-3,2), ylim=c(0,2), aspect=FALSE)
eo.iso.diff("Poisson.pv", "MI", n.best1=300, n.best2=300, ds=ds,
           solid=TRUE, jitter=TRUE)
eo.points(ds, style=1, jitter=TRUE)
eo.iso("Poisson.pv", n.best=300, ds=ds, style=4)
eo.iso("MI", n.best=300, ds=ds, style=5)
eo.legend.diff(3, c("Poisson+ / MI-", "Poisson- / MI+"), solid=TRUE)
eo.close()
```

<code>eo.legend</code>	<i>Draw Legend Box for Point Cloud or Iso-Lines (eo)</i>
------------------------	--

Description

Draw a legend box in one of the corners of the active (e,o) plot, showing labels for one or more styles of data set points, iso-lines or shaded/filled acceptance regions.

Usage

```
eo.legend.points(corner, legend, styles, bw=bw, cex.mul=2.5, ...)

eo.legend.iso(corner, legend, styles, bw=bw, fill=solid, solid=FALSE,
             lw.add=0, density.mul=2, ...)

eo.legend.diff(corner, legend, style1=4, style2=5,
             bw=bw, solid=FALSE, density.mul=2, ...)
```

Arguments

<code>corner</code>	an integer specifying the corner of the plot where the legend box will be drawn (1 = top left, 2 = top right, 3 = bottom right, 4 = bottom left)
<code>legend</code>	a character vector specifying labels for the legend box. For the <code>eo.legend.diff</code> function, it must have length 2 (labels for the difference regions D_1 and D_2).
<code>styles</code>	an integer vector specifying display styles for the items in the legend box (see the <code>eo.par</code> help page for more information about display styles). Note that <code>styles</code> must have exactly the same length as <code>legend</code>
<code>style1, style2</code>	display styles for the first and second difference region (D_1 and D_2). The defaults are set to match those of <code>eo.iso.diff</code> .
<code>bw</code>	If TRUE, the points, lines or shading/colour boxes in the legend are drawn in B/W mode; otherwise, they are drawn in colour mode. This parameter defaults to the state specified with the initial <code>eo.setup</code> call, but can be overridden manually.
<code>fill</code>	If TRUE, show the shadings of acceptance regions instead of iso-line styles in the legend.
<code>solid</code>	If TRUE, show solid colours instead of shadings for acceptance regions in the legend. Setting <code>solid=TRUE</code> implies <code>fill=TRUE</code> .
<code>cex.mul</code>	numeric factor by which plot symbols are scaled in the legend box (with respect to their size in the plot)
<code>lw.add</code>	numeric value added to line widths in the legend box. Only needed when widths of iso-lines are too thin to be clearly visible in the legend box.
<code>density.mul</code>	numeric factor by which the density of shading lines is multiplied in the legend box in order to improve visibility of the shading style
<code>...</code>	Any additional parameters are passed through to the legend function used to draw the legend box.

Details

See the `eo.setup` help page for a description of the general procedure used to create (e,o) plots. This help page also has links to other (e,o) plotting functions. The "factory setting" styles are described on the `eo.par` help page.

`eo.legend.points` displays a legend box for point clouds plotted with `eo.points`; `eo.legend.iso` a legend box for iso-lines or acceptance regions drawn with `eo.iso`; and `eo.legend.diff` a legend box for differences between two acceptance regions that have been highlighted with `eo.iso.diff` (this is just a convenience wrapper around `eo.legend.iso`).

Note that legend boxes can only be created for the default styles set with `eo.par` since it is not possible to override the style parameters manually.

See Also

`eo.par`, `eo.setup`, `eo.points`, `eo.iso`, `eo.iso.diff`

Examples

```
## an example can be found on the "eo.setup" help page
```

eo.mark	<i>Mark Individual Pair Types in Point Cloud (eo)</i>
---------	---

Description

Mark individual pair types from a data set in a point cloud plotted with the `eo.points` function.

Usage

```
eo.mark(ds, select, style=1, bw=bw, cex=1.5, lwd=3,
        jitter=eo.par("jitter"))
```

Arguments

ds	a data set containing pair types that have been plotted as a point cloud, some or all of which will be marked
select	an expression that will be evaluated on the data set <code>ds</code> to determine the pair types that will be marked. In order to mark the point representing the word pair <i>black box</i> , e.g., specify <code>select=(l1 == "black" & l2 == "box")</code> .
style	an integer specifying the style from which the colour of the markers is taken. Note that the symbol (a thick ring) and its size are hard-coded in the function and cannot be changed globally.
bw	If <code>TRUE</code> , the markers are drawn in B/W mode, otherwise in colour mode. This parameter only affects the colour of the marker rings. It defaults to the state specified with the initial <code>eo.setup</code> call, but can be overridden manually.
cex, lwd	size and thickness of the marker rings. The default values are suitable for the "factory setting" styles used for data set points (see <code>eo.points</code>).
jitter	If <code>TRUE</code> , the coordinates of pair types are jittered for the plot. This parameter must have the same value as in the <code>eo.points</code> call that was used to plot the point cloud, otherwise marker placement will be incorrect. When <code>jitter=TRUE</code> , the data set has to be annotated with the <code>add.jitter</code> function first. The default value is set with <code>eo.par</code> .

Details

See the `eo.setup` help page for a description of the general procedure used to create (e,o) plots. This help page also has links to other (e,o) plotting functions. The "factory setting" styles are described on the `eo.par` help page.

See Also

`eo.par`, `eo.setup`, `eo.points`

`eo.par`

Graphics Parameters for (e,o) Plots (eo)

Description

Set default graphics parameters for (e,o) plots, similar to `ucs.par` in the `plots` module and `par` for general graphics parameters. Parameter values can be set by specifying them as arguments in `name=value` form, or by passing a single list of named values. The current values can be queried by giving their names as character strings.

Usage

```
eo.par(...)
```

```
.eo.PAR
```

Arguments

... either character strings (or vectors) specifying the names of parameters to be queried, or parameters to be set in `name=value` form, or a single list of named values. Valid parameter names are described below.

Details

The current default parameters are stored in the global variable `.eo.PAR`. They can be queried by giving their names as one or more character vectors to `eo.par`. `eo.par()` (no arguments) returns all `eo` graphics parameters.

Parameters are set by specifying their names and the new values as `name=value` pairs. Such a list can also be passed as a single argument to `eo.par`, which is typically used to restore previous parameter values (that have been saved in a list variable).

In order to restore the "factory settings", reload the module with the command `ucs.library("eo", reload=TRUE)`.

Value

When parameters are set, their former values are returned in an invisible named list. Such a list can be passed as a single argument to `eo.par` to restore the previous settings.

When a single parameter is queried, its value is returned directly. When two or more parameters are queried, the result is a named list.

Note the inconsistency, which is the same as for `par`: setting one parameter returns a list, but querying one parameter returns a vector (or a scalar, i.e. a vector of length 1).

Graphics Parameters for (e,o) Plots

- bw** If TRUE, (e,o) plots are created in B/W mode by default.
- xlim, ylim** Integer vectors of length 2, specifying default ranges for the e-axis (xlim) and o-axis (ylim) in orders of magnitude (i.e., base 10 logarithms: -2 corresponds to .01, 0 corresponds to 1, and 3 corresponds to 1000). When the default values are not set, every call to the `eo.setup` function must either specify xlim and ylim values or a data set, from which suitable ranges are computed.
- aspect** If TRUE, an aspect ratio of 1:1 is enforced for every (e,o) plot, i.e. the axis ranges are extended as necessary (assuming a square plotting region). The factory setting is TRUE.
- log.marks** If TRUE, tick marks on the axes are labelled in logarithmic units, i.e. orders of magnitude. Otherwise, absolute numbers are used. The factory setting is FALSE. (Note that (e,o) plots are always drawn in logarithmic scale.)
- steps** An integer specifying the number of equidistant steps used for drawing iso-lines. The factory setting is 100.
- jitter** If TRUE, always uses jittered coordinates for plotting data sets and computing n-best thresholds. Note that all data sets must be annotated with the `add.jitter` function first. The factory setting is FALSE.
- cex** Overall character expansion factor (for tick marks, axis labels and legends). The factory setting is 1.3.
- col** A character or integer vector specifying line colours for the different styles of iso-lines in colour mode (see the `par` manpage for details on colour specification). Values are recycled to match the length of the `lty` and `lwd` parameters when necessary. The factory setting defines 5 styles in black, blue, red, magenta and cyan.
- lty** A character or integer vector specifying line types for the different styles of iso-lines in colour mode (see the `par` manpage for details). Values are recycled to match the length of the `col` and `lwd` parameters when necessary.
- lwd** A numeric vector specifying line widths for the different styles of iso-lines in colour mode. Values are recycled to match the length of the `col` and `lty` parameters when necessary.
- angle, density** Numeric vectors specifying the angle and density of shading lines when the acceptance region bounded by a given iso-line is filled. These vectors should support as many styles as `col`, `lty` and `lwd` above. Details on shading lines can be found on the `polygon` help page.
- solid** A character or integer vector specifying background colours for the different styles of iso-lines when the acceptance region is filled with solid colour (rather than shading lines).
- bw.col, bw.lty, bw.lwd** Colour, line type and line width for iso-lines in B/W mode (corresponding to `col`, `lty` and `lwd` in colour mode). The factory setting defines 5 styles with solid, dashed, grey, dotted and dark grey dot-dash lines.
- bw.angle, bw.density, bw.solid** Angle and density of shading lines, as well as solid colour, for filled acceptance regions in B/W mode (corresponding to `angle`, `density` and `solid` in colour mode)
- pt.pch** A character or integer vector specifying plot symbols for the different styles of data set points in colour mode (see the `points` help page for a full list of available

plot symbols). Values are recycled to match the length of the `pt.cex` and `pt.col` parameters when necessary. The factory setting defines 5 styles with black, green, red, yellow and orange dots.

pt.cex A numeric vector specifying character expansion factors for the different styles of data set points in colour mode. Values are recycled to match the length of the `pt.pch` and `pt.col` parameters when necessary.

pt.col A character or integer vector specifying colours for the different styles of data set points in colour mode (see the `par` help page for details on colour specification). Values are recycled to match the length of the `pt.pch` and `pt.cex` parameters when necessary.

bw.pt.pch, bw.pt.cex, bw.pt.col Plot symbol, character expansion and colour for data set points in B/W mode (corresponding to `pt.pch`, `pt.cex` and `pt.col` in colour mode). The factory setting defines 5 styles with black dots, circles, + crosses, triangles and x crosses.

See Also

`eo.setup`, `eo.iso`, `eo.iso.diff`, `eo.points`, `eo.legend`, `ucs.par`, `par`

Examples

```
print(names(ucs.eo()))           # list available parameters

eo.par("col", "lty", "lwd")     # the default styles for iso-lines
eo.par(c("col", "lty", "lwd")) # works as well

## temporary changes to graphics paramters:
par.save <- eo.par(bw=TRUE, steps=200)
## (e,o) plots use the modified parameters here
eo.par(par.save)                # restore previous values

ucs.library("eo", reload=TRUE) # reload module for factory defaults
```

<code>eo.points</code>	<i>Draw Data Set as Point Cloud in (e,o) Plane (eo)</i>
------------------------	---

Description

Plot (selected) pair types from a data set as a point cloud in the (e,o) plane. Points can be drawn in any of the styles defined in the global defaults (`eo.par`), as determined by the `style` parameter.

Usage

```
eo.points(ds, style=1, select=NULL, bw=bw, jitter=eo.par("jitter"),
          pch=par("pt.pch"), cex=par("pt.cex"), col=par("pt.col"), ...)
```

Arguments

<code>ds</code>	a data set containing the pair types to be plotted as a point cloud
<code>style</code>	an integer specifying the style (shape, size and colour) in which points will be drawn. The number of styles available depends on the global parameter settings (<code>eo.par</code>). The "factory settings" define 5 different styles for points.
<code>select</code>	an optional expression, which is evaluated on the data set <code>ds</code> to select a subset of the pair types for plotting (e.g. <code>select=(f <= 10 & b.TP)</code> to display pair types with joint frequency $f \leq 10$ that are marked as true positives).
<code>bw</code>	If TRUE, the points are drawn in B/W mode, otherwise in colour mode. This parameter defaults to the state specified with the initial <code>eo.setup</code> call, but can be overridden manually.
<code>jitter</code>	If TRUE, the coordinates of pair types are jittered for the plot, i.e. a small random displacement is added to each point so that the point cloud has a more homogeneous appearance. In order to use this option, the data set has to be annotated with the <code>add.jitter</code> function first. The default value is set with <code>eo.par</code> .
<code>pch, cex, col</code>	The style parameters for points are determined automatically from the global settings (<code>eo.par</code>), according to the selected style and <code>bw</code> mode. They can be overridden by specifying explicit values in the function call.
<code>...</code>	Any additional parameters are passed through to the <code>points</code> function that draws the point cloud.

Details

See the `eo.setup` help page for a description of the general procedure used to create (e,o) plots. This help page also has links to other (e,o) plotting functions. The "factory setting" styles are described on the `eo.par` help page.

See Also

`eo.par`, `eo.setup`

Examples

an example can be found on the "eo.setup" help page

`eo.setup`*Initialise and Finalise an (e,o) Plot (eo)*

Description

`eo.setup` initialises a new (e,o) plot window, which can then be drawn into with calls to `eo.iso`, `eo.points` and similar functions. The plot has to be finalised with `eo.close` before a new plot can be generated.

A detailed explanation of (e,o) plots and their interpretation can be found in Section 3.3 of Evert (2004).

Usage

```
eo.setup(xlim=eo.par("xlim"), ylim=eo.par("ylim"), ds=NULL,
        bw=eo.par("bw"), file=NULL,
        aspect=eo.par("aspect"), log.marks=eo.par("log.marks"),
        cex=eo.par("cex"), ...)

eo.close()
```

Arguments

- | | |
|-------------------------|---|
| <code>xlim, ylim</code> | integer vectors of length 2, specifying ranges for the e-axis (<code>xlim</code>) and o-axis (<code>ylim</code>) in orders of magnitude (i.e., base 10 logarithms: -2 corresponds to .01, 0 corresponds to 1, and 3 corresponds to 1000). If <code>xlim</code> and <code>ylim</code> are not given and no default values have been set with <code>eo.par</code> , the <code>ds</code> parameter has to be specified. Note that (e,o) plots are always drawn in logarithmic scale. |
| <code>ds</code> | A data set from which suitable ranges for the e-axis and o-axis are computed. The automatically determined values are overridden by explicit <code>xlim</code> and <code>ylim</code> parameters. |
| <code>bw</code> | If <code>TRUE</code> , the (e,o) plot is drawn in B/W mode, otherwise in colour mode. The default value is set with <code>eo.par</code> . |
| <code>file</code> | a character string giving the name of a PostScript file. If specified, the (e,o) plot is saved to <code>file</code> in EPS format rather than displayed on screen. Note that this file will only be written after <code>eo.close</code> has been called. |
| <code>aspect</code> | If <code>TRUE</code> , an aspect ratio of 1:1 is enforced by extending the axis ranges as necessary (assuming that the plotting region is square). The default value is set with <code>eo.par</code> . |
| <code>log.marks</code> | If <code>TRUE</code> , tick marks on the axes are labelled in logarithmic units, i.e. orders of magnitude. Otherwise, absolute numbers are used. The default value is set with <code>eo.par</code> . (Recall that (e,o) plots are always drawn in logarithmic scale.) |

<code>cex</code>	overall character expansion factor (for tick marks, axis labels and legends). The default value is set with <code>eo.par</code> .
<code>...</code>	Any additional parameters are passed through to the <code>plot</code> function used to set up the plot region and axes.

Details

An (e,o) plot is typically created in four stages:

- Set up the plot with `eo.setup`, defining suitable ranges for the e-axis. These ranges and some other state information (e.g. whether the plot is drawn in colour or B/W mode) are recorded in the global variable `.eo.STATE`.
- Draw data sets as point clouds with `eo.points` and iso-lines for GAMs with `eo.iso`. Differences between two acceptance regions can be highlighted with `eo.iso.diff`. The `eo.mark` function can be used to mark individual points with circles.
- Draw legend boxes in the corners of the plot with `eo.legend.points`, `eo.legend.iso` and `eo.legend.diff`.
- Finalise the plot with `eo.close`. When a `file` argument has been specified in the `eo.setup` call, the plot will be saved to a PostScript file at this stage.

Default values for `xlim`, `ylim`, `bw`, `aspect`, `log.marks` and `cex` can be set with the `eo.par` function. See the `eo.par` help page for "factory settings" of these parameters, as well as default line and point styles in colour and B/W mode.

Note that (e,o) plots are always drawn in logarithmic scale and tick marks are shown for orders of magnitude (full powers of ten). The `log.marks` parameter only determines whether the labels on these tick marks show linear (.1, 1, 10, 100, ...) or logarithmic (-1, 0, 1, 2, ...) values.

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`eo.par`, `eo.points`, `eo.iso`, `eo.iso.diff`, `eo.mark`, `eo.legend`

Examples

```
ucs.library("eo")

## load data set file, add jitter, and reduce to random 10
ds <- add.jitter(read.ds.gz("dickens.ds.gz"))
select <- rbinom(nrow(ds), 1, .1) == 1
ds <- ds[select,]

## 1) set up new (e,o) plot with suitable axis ranges
eo.setup(ds=ds) # note that y axis is extend to enforce 1:1 aspect
```

```
## 2) add data set as point cloud and three iso-lines
eo.points(ds, style=5, jitter=TRUE)
eo.iso("Poisson.pv", 3, style=1) # p-value = 1e-3
eo.iso("z.score", 3.09, style=2) # corresponding one-sided z-score
eo.iso("t.score", 3.09, style=3) # same as t-score with df=Inf

## 3) add legend boxes in top right (2) and bottom right (3) corner
eo.legend.points(2, "pair type", 5)
eo.legend.iso(3, c("Poisson", "z-score", "t-score"), 1:3)

## 4) finalise the (e,o) plot
eo.close()
```

`evaluation.file` *Evaluation Graphs for Association Measures (plots)*

Description

The `evaluation.plot` function is often invoked twice with the same parameter settings, once for on-screen display, and once for saving the plot to a PostScript file. `evaluation.file` automates this process, automatically switching between colour mode for the screen version and B/W mode for the PostScript version.

Usage

```
evaluation.file(ds, keys, file, bw=NULL, ...)
```

Arguments

<code>ds</code>	a UCS data set object (passed to <code>evaluation.plot</code>)
<code>keys</code>	a character vector specifying the names of association measures to be evaluated (passed to <code>evaluation.plot</code>)
<code>file</code>	a character string giving the name of a file to which the PostScript version of the plot will be saved
<code>bw</code>	if <code>TRUE</code> , both versions will be in B/W; if <code>FALSE</code> , both versions will be in colour. If unspecified, <code>evaluation.file</code> switches automatically from colour mode (for the screen version) to B/W mode (for the PostScript file), which is the most common use.

Details

PostScript versions can be suppressed by setting

```
ucs.par(do.file=FALSE)
```

In this case, `evaluation.file` will only draw the screen versions of the graphs, which is convenient when experimenting and while fine-tuning the plots.

See Also

`evaluation.plot`, `ucs.par`, and the tutorial script ‘`tutorial.R`’ in the ‘`script/`’ directory.

`evaluation.plot` *Evaluation Graphs for Association Measures (plots)*

Description

An implementation of evaluation graphs for the empirical evaluation of association measures in terms of precision and recall, as described in (Evert, 2004, Ch. 5). Graphs of precision, recall and local precision for n-best lists, as well as precision-by-recall graphs are all provided by a single function `evaluation.plot`.

Usage

```
evaluation.plot(ds, keys, tp=ds$b.TP,
               x.min=0, x.max=100, y.min=0, y.max=100,
               x.axis=c("n.best", "proportion", "recall"),
               y.axis=c("precision", "local.precision", "recall"),
               n.first=ucs.par("n.first"), n.step=ucs.par("n.step"),
               cut=NULL, window=400,
               show.baseline=TRUE, show.nbest=NULL, show.npair=NULL,
               conf=FALSE, conf.am=NULL, conf.am2=NULL,
               test=FALSE, test.am1=NULL, test.am2=NULL,
               test.step=ucs.par("test.step"), test.relevant=0,
               usercode=NULL,
               file=NULL, aspect=1, plot.width=6, plot.height=6,
               cex=ucs.par("cex"), lex=ucs.par("lex"), bw=FALSE,
               legend=NULL, bottom.legend=FALSE,
               title=NULL, ...)
```

Arguments

- | | |
|-------------------|---|
| <code>ds</code> | a UCS data set object, read in from a data set file with the <code>read.ds.gz</code> function. <code>ds</code> must contain rankings for the association measures listed in the <code>keys</code> parameter (use <code>add.ranks</code> to add such rankings to a data set object). |
| <code>keys</code> | a character vector naming up to 10 association measures to be evaluated. Each name may be abbreviated to prefix that must be unique within the measures annotated in <code>ds</code> . Use the <code>ds.find.am</code> function to obtain a list of measures annotated in the data set, and see the <code>ucsam</code> manpage in UCS/Perl for detailed information about the association measures supported by the UCS system (with the shell command <code>ucsd doc ucsam</code>). |
| <code>tp</code> | a logical vector indicating true positives, parallel to the rows of the data set <code>ds</code> . If <code>tp</code> is not specified, the data set must contain a variable named <code>b.TP</code> which is used instead. |

<code>x.min, x.max</code>	the limits of the x-axis in the plot, used to “zoom in” to an interesting region. The interpretation of the values depends on the <code>x.axis</code> parameter below. For <code>x.axis="n.best"</code> (the default case), <code>x.min</code> and <code>x.max</code> refer to n-best lists. Otherwise, they refer to percentages ranging from 0 to 100. By default, the full data set is shown.
<code>y.min, y.max</code>	the limits of the y-axis in the plot, used to “zoom in” to an interesting region. The values are always interpreted as percentages, ranging from 0 to 100. By default, <code>y.max</code> is fitted to the evaluation graphs (unless <code>y.axis="recall"</code> , where <code>y.max</code> is always set to 100).
<code>x.axis</code>	select variable shown on x-axis. Available choices are the n-best list size <code>n</code> (" <code>n.best</code> ", the default), the same as a proportion of the full data set (" <code>proportion</code> "), and the recall as a percentage (" <code>recall</code> "). The latter produces precision-by-recall graphs. Unless you are silly enough to specify <code>y.axis="recall"</code> at the same time, that is.
<code>y.axis</code>	select variable shown on x-axis. Available choices are the precision (" <code>precision</code> ", the default), an estimate for local precision (" <code>local.precision</code> ", see details below), and the recall (" <code>recall</code> "). All three variables are shown as percentages ranging from 0 to 100.
<code>n.first</code>	the smallest n-best list to be evaluated. Shorter n-best lists typically lead to highly unstable evaluation graphs. The standard setting is 100, but a higher value may be necessary for random sample evaluation (see details below). If <code>n.first</code> is not specified, the default supplied by <code>ucs.par</code> is used.
<code>n.step</code>	the step width for n-best lists in the evaluation graphs. Initially, precision and recall are computed for all n-best lists, but only every <code>n.step</code> -th one is plotted, yielding graphs that look less jagged and reducing the size of generated PostScript files (see the <code>file</code> parameter below). If <code>n.step</code> is not specified, the default supplied by <code>ucs.par</code> is used.
<code>cut</code>	for each association measure, pretend that the data set consists only of the <code>cut</code> highest-ranked candidates according to this measure. This trick can be used to perform an evaluation of n-best lists without having to annotate the full data set. The candidates from all relevant n-best lists are combined into a single data set file and <code>cut</code> is set to <code>n</code> .
<code>window</code>	number of candidates to consider when estimating local precision (default: 400), i.e. with the option <code>y.axis="local"</code> . Values below 400 or above 1000 are rarely useful. See below for details.
<code>show.baseline</code>	if TRUE, show baseline precision as dotted horizontal line with label (this is the default). Not available when <code>y.axis="recall"</code> .
<code>show.nbest</code>	integer vector of n-best lists that will be indicated as thin vertical lines in the plot. When <code>x.axis="recall"</code> , the n-best lists are shown as diagonal lines.
<code>show.npair</code>	when <code>x.axis="proportion"</code> , the total number of candidates in <code>ds</code> is shown in the x-axis label. Set <code>show.npair=NULL</code> to suppress this, or set it to an integer value in order to lie about the number of candidates (rarely useful).

<code>conf</code>	if <code>TRUE</code> , confidence intervals are shown as coloured or shaded regions around one or two precision graphs. In this case, the parameter <code>conf.am</code> must also be specified. Alternatively, <code>conf</code> can be set to a number indicating the significance level to be used for the confidence intervals (default: 0.05, corresponding to 95% confidence). See below for details. Note that <code>conf</code> is only available when <code>y.axis="precision"</code> .
<code>conf.am</code>	name of the association measure for which confidence intervals are displayed (may be abbreviated to a prefix that is unique within keys)
<code>conf.am2</code>	optional second association measure, for which confidence intervals will also be shown
<code>test</code>	if <code>TRUE</code> , significance tests are carried out for the differences between the evaluation results of two association measures, given as <code>test.am1</code> and <code>test.am2</code> below. Alternatively, <code>test</code> can be set to a number indicating the significance level to be used for the tests (default: 0.05). <code>n</code> -best lists where the result difference is significant are indicated by arrows between the respective evaluation graphs (when <code>x.axis="recall"</code>) or by coloured triangles (otherwise). See details below. Note that <code>test</code> is <i>not</i> available when <code>y.axis="local"</code> .
<code>test.am1</code>	the first association measure for significance tests (may be abbreviated to a prefix that is unique within keys). Usually, this is the measure that achieves better performance (but tests are always two-sided).
<code>test.am2</code>	the second association measure for significance tests (may be abbreviated to a prefix that is unique within keys)
<code>test.step</code>	the step width for <code>n</code> -best lists where significance tests are carried out, as a multiple of <code>n.step</code> . The standard setting is 10 since the significance tests are based on the computationally expensive <code>fisher.test</code> function and since the triangles or arrows shown in the plot are fairly large. If <code>test.step</code> is not specified, the default supplied by <code>ucs.par</code> is used.
<code>test.relevant</code>	a positive number, indicating the estimated precision differences that are considered “relevant” and that are marked by dark triangles or arrows in the plot. See below for details.
<code>usercode</code>	a callback function that is invoked when the plot has been completed, but before the legend box is drawn. This feature is mainly used to add something to a plot that is written to a PostScript file. The <code>usercode</code> function is invoked with parameters <code>region=c(x.min,x.max,y.min,y.max)</code> and <code>pr</code> , a list of precision/recall tables (as returned by <code>precision.recall</code>) for each of the measures in keys.
<code>file</code>	a character string giving the name of a PostScript file. If specified, the evaluation plot will be saved to <code>file</code> rather than displayed on screen. See <code>evaluation.file</code> for a function that combines both operations.
<code>aspect</code>	a positive number specifying the desired aspect of the plot region (only available for PostScript files). In the default case <code>x.axis="n.best"</code> , <code>aspect</code> refers to the absolute size of the plot region. Otherwise, it specifies the size ratio between percentage points on the x-axis and the y-axis. Setting <code>aspect</code> modifies the height of the plot (<code>plot.height</code>).

<code>plot.width</code> , <code>plot.height</code>	the width and height of a plot that is written to a PostScript file, measured in inches. <code>plot.height</code> may be overridden by the <code>aspect</code> parameter, even if it is set explicitly.
<code>cex</code>	character expansion factor for labels, annotations, and symbols in the plot (see <code>par</code> for details). If <code>cex</code> is not specified, the default supplied by <code>ucs.par</code> is used.
<code>lex</code>	added to the line widths of evaluation graphs and some decorations (note that this is not an expansion factor). If <code>lex</code> is not specified, the default supplied by <code>ucs.par</code> is used.
<code>bw</code>	if <code>TRUE</code> , the evaluation plot is drawn in black and white, which is mostly used in conjunction with <code>file</code> to produce figures for articles (defaults to <code>FALSE</code>). See below for details.
<code>legend</code>	a vector of character strings or expressions, used as labels in the legend of the plot (e.g. to show mathematical symbols instead of the names of association measures). Use <code>legend=NULL</code> to suppress the display of a legend box.
<code>bottom.legend</code>	if <code>TRUE</code> , draw legend box in bottom right corner of plot (default is top right corner).
<code>title</code>	a character vector or expression to be used as the main title of the plot (optional)
<code>...</code>	any other arguments are set as local graphics parameters (using <code>par</code>) before the evaluation plot is drawn

Details

When `y.axis="local.precision"`, the evaluation graphs show **local precision**, i.e. an estimate for the density of true positives around the *n*-th rank according to the respective association measure. Local precision is smoothed using a kernel density estimate with a Gaussian kernel (from the density function), based on a symmetric window covering approximately `window` candidates (default: 400). Consequently, the resulting values do not have a clear-cut interpretation and should not be used to evaluate the performance of association measures. They are rather a means of exploratory data analysis, helping to visualise the relation between association scores and the true positives in a data set (see Evert, 2004, Sec. 5.2 for an example).

In order to generalise evaluation results beyond the specific data set on which they were obtained, it is necessary to compute confidence intervals for the observed precision values and to test whether the observed result differences are significant. See (Evert, 2004, Sec. 5.3) for the methods used and the interpretation of their results.

Confidence intervals are computed by setting `conf=TRUE` and selecting an association measure with the `conf.am` parameter. The confidence intervals are displayed as a coloured or shaded region around the precision graph of this measure (confidence intervals are not available for graphs of recall or local precision). The default confidence level of 95% will rarely need to be changed. Optionally, a second confidence region can be displayed for a measure selected with the `conf.am2` parameter.

Significance tests for the result differences are activated by setting `test=TRUE` (not available for graphs of local precision). The evaluation results of two association measures

(specified with `test.am1` and `test.am2`) are compared for selected n-best lists, and significant differences are marked by coloured triangles or arrows (when `x.axis="recall"`). The default significance level of 0.05 will rarely need to be changed. Use the `test.step` parameter to control the spacing of the triangles or arrows.

A significant difference indicates that measure A is truly better than measure B, rather than just as a coincidence in a single evaluation experiment. Formally, this “true performance” can be defined as the average precision of a measure, obtained by averaging over many similar evaluation experiments. Thus, a significant difference means that the average precision of A is higher than that of B, but it does not indicate how great the difference is. A tiny difference (say, of half a percent point) is hardly **relevant** for an application, even if there is significant evidence for it. If the `test.relevant` parameter is set, the `evaluation.plot` function attempts to estimate whether there is significant evidence for a relevant difference (of at least a many percent points as given by the value of `test.relevant`), and marks such cases by darker triangles or arrows. This feature should be considered experimental and used with caution, as the computation involves many approximations and guesses (exact statistical inference for the difference in true precision not being available).

It goes without saying that confidence regions and significance tests do not allow evaluation results to be generalised to a different extraction task (i.e. another type of cooccurrences or another definition of true positives), or even to the same task under different conditions (such as a source corpus from a different domain, register, time, or a corpus of different size). The unpredictability of the performance of association measures for different extraction tasks or under different conditions has been confirmed by various evaluation studies.

Generally, evaluation plots can be drawn in two modes: **colour** (`bw=FALSE`, the default) or **black and white** (`bw=TRUE`). The styles of evaluation graphs are controlled by the respective settings in `ucs.par`, while the appearance of various other elements is hard-coded in the `evaluation.plot` function. In particular, confidence regions are either filled with a light background colour (colour mode) or shaded with diagonal lines (B/W mode). The triangles or arrows used to mark significant differences are yellow or red (indicating relevance) in colour mode, and light grey or dark grey (indicating relevance) in B/W mode. B/W mode is mainly used to produce PostScript files to be included as figures in articles, but can also be displayed on-screen for testing purposes.

The `evaluation.plot` function supports **evaluation based on random samples**, or RSE for short (Evert, 2004, Sec. 5.4). Missing values (NA) in the `tp` vector (or the `b.TP` variable in `ds`) are interpreted as unannotated candidates. In this case, precision, recall and local precision are computed as maximum-likelihood estimates based on the annotated candidates. Confidence intervals and significance tests, which should not be absent from any RSE, are adjusted accordingly. A confidence interval for the baseline precision is automatically shown (by thin dotted lines) when RSE is detected. Note that n-best lists (as shown on the x-axis) still refer to the full data set, not just to the number of annotated candidates.

Note

The following functions are provided for compatibility with earlier versions of UCS/R: `precision.plot`, `recall.plot`, and `recall.precision.plot`. They are simple front-ends to `evaluation.plot` with the implicit parameter settings `y.axis="recall"` and `y.axis="precision"`, `x.axis="recall"` for the latter two.

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`ucs.par`, `evaluation.file`, `read.ds.gz`, and `precision.recall`. The R script ‘tutorial.R’ in the ‘script/’ directory provides a gentle introduction to the wide range of possibilities offered by the `evaluation.plot` function.

<code>evaluation.table</code>	<i>Precision/Recall Tables for the Evaluation of Association Measures (base)</i>
-------------------------------	--

Description

A simple text-mode version of the precision/recall-based evaluation provided by the `plots` module. Returns a table of precision or recall values for a selected range of association measures on selected n-best lists. **This is a preliminary version of the function - both interface and functionality may change in future releases.**

Usage

```
evaluation.table(ds, keys, n, tp=ds$b.TP, recall=FALSE)
```

Arguments

<code>ds</code>	a UCS data set object, read in from a data set file with the <code>read.ds.gz</code> function. <code>ds</code> must contain rankings for the association measures listed in the <code>keys</code> parameter (use <code>add.ranks</code> to add such rankings to a data set object).
<code>keys</code>	a character vector specifying the names of association measures to be evaluated. Each name may be abbreviated to prefix that must be unique within the measures annotated in <code>ds</code> . Use the <code>ds.find.am</code> function to obtain a list of measures annotated in the data set, and see the <code>ucsam</code> manpage in UCS/Perl for detailed information about the association measures supported by the UCS system (with the shell command <code>ucsd doc ucsam</code>).
<code>n</code>	a vector of n-best sizes for which precision or recall values are computed
<code>tp</code>	a logical vector indicating true positives, parallel to the rows of the data set <code>ds</code> . If <code>tp</code> is not specified, the data set must contain a variable named <code>b.TP</code> which is used instead.
<code>recall</code>	if <code>TRUE</code> , returns table of recall values, otherwise table of precision values (default)

Value

A data frame whose rows correspond to n-best lists. In addition to the column labelled `n`, which gives the n-best lists for which the evaluation was carried out, there is one column for each selected association measure. The column is labelled with the name of the measure and lists the corresponding precision or recall values, depending on the `recall` parameter.

See Also

`evaluation.plot`, `precision.recall`

`fzm`

The Finite Zipf-Mandelbrot LNRE Model (fzm)

Description

Object constructor for a finite Zipf-Mandelbrot (fZM) LNRE model with parameters α , A and B (Evert, 2004a). Either the parameters are specified explicitly, or one or more of them can be estimated from an observed frequency spectrum.

Usage

```
fzm(alpha, A, B)
```

```
fzm(alpha, A, N, V)
```

```
fzm(alpha, N, V, spc, m.max=15, stepmax=10, debug=FALSE)
```

```
fzm(N, V, spc, m.max=15, stepmax=10, debug=FALSE)
```

Arguments

<code>alpha</code>	a number in the range (0, 1), the shape parameter α of the fZM model. <code>alpha</code> can automatically be estimated from <code>N</code> , <code>V</code> , and <code>spc</code> .
<code>A</code>	a small positive number $A \ll 1$, the parameter A of the fZM model. <code>A</code> can automatically be estimated from <code>N</code> , <code>V</code> , and <code>spc</code> .
<code>B</code>	a large positive number $B \gg 1$, the parameter B of the fZM model. <code>B</code> can automatically be estimated from <code>N</code> and <code>V</code> .
<code>N</code>	the sample size, i.e. number of observed tokens
<code>V</code>	the vocabulary size, i.e. the number of observed types
<code>spc</code>	a vector of non-negative integers representing the class sizes V_m of the observed frequency spectrum. The vector is usually read from a file in <code>lexstats</code> format with the <code>read.spectrum</code> function.
<code>m.max</code>	the number of ranks from <code>spc</code> that will be used to estimate the α parameter

stepmax	maximal step size of the <code>nlm</code> function used for parameter estimation. It should not be necessary to change the default value.
debug	if TRUE, print debugging information during the parameter estimation process. This feature can be useful to find out why parameter estimation fails.

Details

The fZM model with parameters $\alpha \in (0, 1)$ and $C > 0$ is defined by the type density function

$$g(\pi) := C \cdot \pi^{-\alpha-1}$$

for $A \leq \pi \leq B$. The normalisation constant C is determined from the other parameters by the condition

$$\int_A^B \pi \cdot g(\pi) d\pi = 1$$

The parameters α and A are estimated simultaneously by nonlinear minimisation (`nlm`) of a multinomial chi-squared statistic for the observed against the expected frequency spectrum. Note that this is different from the multivariate chi-squared test used to measure the goodness-of-fit of the final model (Baayen, 2001, Sec. 3.3).

See Evert (2004, Ch. 4) for further mathematical details, especially concerning the expected vocabulary size, frequency spectrum and conditional parameter distribution, as well as their variances.

Value

An object of class "fzm" with the following components:

alpha	value of the α parameter
A	value of the A parameter
B	value of the B parameter
C	value of the normalisation constant C
C	population size S predicted by the model
N	number of observed tokens (if specified)
V	number of observed types (if specified)
spc	observed frequency spectrum (if specified)

This object prints a short summary, including the population size S and a comparison of the first ranks of the observed and expected frequency spectrum (if available).

References

- Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.
- Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.
- Evert, Stefan (2004a). A simple LNRE model for random character sequences. In *Proceedings of JADT 2004*, Louvain-la-Neuve, Belgium, pages 411–422.

See Also

zm, EV, EVm, VV, VVm, write.lexstats, lnre.goodness.of.fit, read.spectrum, and spectrum.plot

gam.helpers

Helper Functions for GAM Equations (gam)

Description

gam.yates and gam.yates.inv implement an invertible version of the discounting function used by Yates' correction. signed.sqrt, b.star, b.norm and e.bar are standard abbreviations used in the definition of generalised association measures in terms of ebo-coordinates.

Usage

gam.yates(d)

gam.yates.inv(d.corr)

signed.sqrt(x)

b.star(b)

b.norm(b)

e.bar(e, b, N)

Arguments

d	difference between observed and expected frequency, to which the generalised Yates' correction is applied
d.corr	difference between observed and expected frequency with generalised Yates' correction applied, from which the original difference can uniquely be reconstructed
x	a vector of positive or negative real numbers
b	a vector of <i>balance</i> (<i>b</i>) values in the ebo coordinate system
e	a vector of <i>expectation</i> (<i>e</i>) values in the ebo coordinate system
N	sample size <i>N</i>

Details

The standard discounting function for Yates' correction is $d^* := d - 1/2$ for $d \geq 0$ and $d^* := d + 1/2$ for $d < 0$, where d is the difference between observed and expected frequency. This definition does not lead to a continuous and invertible function of d , so a GAM with Yates' correction applied does not satisfy the soundness conditions. The generalised Yates' correction implemented by gam.yates and gam.yates.inv is a monotonic (and hence invertible) function that is identical to the standard discounting function for $|d| \geq 1$ and uses linear interpolation for $-1 < d < 1$.

The functions `signed.sqrt`, `b.star`, `b.norm` and `e.bar` compute the standard abbreviation $\pm\sqrt{x}$, b^* , $\|b\|$ and \bar{e} (“e bar”) used by Evert (2004) for the definition of GAMs in terms of ebo-coordinates.

Value

all functions return a vector of real numbers

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

Examples

```
d <- runif(20, -2, 2)
d.corr <- gam.yates(d)
all(d == gam.yates.inv(d.corr))

signed.sqrt(-4:4)
```

<code>gam.iso</code>	<i>Compute Iso-Surfaces for GAMs (gam)</i>
----------------------	--

Description

Computes iso-surfaces for a generalised association measure (GAM) in standard or ebo-coordinates.

Usage

```
gam.iso(name, gamma, f1, f2, N, bsearch.min=NULL, bsearch.max=NULL)
gam.iso(name, gamme, e, b=1, N=1e6, bsearch.min=NULL, bsearch.max=NULL)
```

Arguments

<code>name</code>	name of a generalised association measure (GAM)
<code>gamma</code>	a numerical constant that determines the desired iso-surface $\{g = \gamma\}$
<code>f1</code> , <code>f2</code> , <code>N</code>	numerical vectors specifying the <code>f1</code> and <code>f2</code> coordinates of points in the standard coordinate space, as well as the sample size <code>N</code>
<code>e</code> , <code>b</code>	numerical vectors specifying the <code>e</code> and <code>b</code> coordinates of points in the ebo-coordinate space (if the <i>balance</i> <code>b</code> is not specified, it defaults to 1)
<code>N</code>	optional numerical vector specifying the sample size <code>N</code> when computing iso-surfaces for a GAM that is not size-invariant in ebo-coordinates (defaults to 1e6)
<code>bsearch.min</code>	initial lower boundary for binary search algorithm, when no explicit equation for the iso-surface is available
<code>bsearch.max</code>	initial upper boundary for the binary search algorithm

Details

Note that all function arguments except for `name` must be passed explicitly by name in order to distinguish the two operating modes of `gam.iso` (standard vs. ebo-coordinates).

When ebo-coordinates are used, the argument `N` (*sample size*) can safely be omitted for any size-invariant GAM (in ebo-coordinates). For other GAMs, a default value of 1e6 will be used, corresponding to the typical size of a co-occurrence data set. The argument `b` (*balance*) can be omitted for any central GAMs. Otherwise, it defaults to a value of 1, corresponding to the centralized version of the respective GAM.

Use `gamma.nbest` to compute a suitable γ values for n-best surfaces.

When no explicit equation for the iso-surface of a GAM is available, the `gam.iso` function uses a binary search algorithm to solve the implicit equation $\{g = \gamma\}$. Since some GAMs are only defined for valid frequency signatures (where all four cells of the contingency table are non-negative), the binary search for the `o` coordinate is confined to the range from 0 to $\min\{f_1, f_2\}$. When no solution can be found in this range, `gam.iso` returns `NA` for the corresponding points. For GAMs where it is safe to search a larger range (notably `Poisson.pv` and `log.likelihood`), the boundaries of the search interval can be adjusted with the `bsearch.min` and `bsearch.max` parameters. Note that most other GAMs have explicit iso-equations, so these parameters are rarely needed.

Value

a vector of real numbers representing the `f` or `o` coordinates of the respective iso-surface; these are the values of `f` or `o` that solve the implicit equation $\{g = \gamma\}$ for the specified values of `f1`, `f2`, `N` or `e`, `b` (and `N`); this vector may contain missing values (`NA`) for points where no solution is found (see "Details" for more information)

See Also

`gam.score`, `builtin.gams`, `gamma.nbest`

Examples

```
e <- 10^seq(-2, 1, .1)          # compute iso-line on logarithmic scale
o <- gam.iso("t.score", 2, e=e)

x <- 10^seq(0, 2, .1)          # compute iso-surface over rectangular grid
g <- expand.grid(f1=x, f2=x)
g$f <- gam.iso("t.score", 2, f1=g$f1, f2=g$f2, N=1000)
library(lattice)
wireframe(f ~ f1 * f2, log(g))
```

`gam.score`

Compute GAM Scores in Standard or EBO-Coordinates (gam)

Description

Computes scores of a generalised association measure (GAM) in standard or ebo-coordinates.

Usage

```
gam.score(name, f, f1, f2, N)
gam.score(name, o, e, b=1, N=1e6)
```

Arguments

<code>name</code>	name of a generalised association measure (GAM)
<code>f, f1, f2, N</code>	numerical vectors specifying the (generalised) frequency signatures of candidates
<code>o, e, b</code>	numerical vectors specifying the ebo-coordinates of candidates (if the <i>balance</i> <code>b</code> is not specified, it defaults to 1)
<code>N</code>	optional numerical vector specifying the sample size N when computing scores of a GAM that is not size-invariant in ebo-coordinates (defaults to 1e6)

Details

Note that all function arguments except for `name` must be passed explicitly by name in order to distinguish the two operating modes of `gam.score` (standard vs. ebo-coordinates).

The components of the generalised frequency signature (`f, f1, f2, N`) can be arbitrary positive real numbers.

When ebo-coordinates are used, the argument `N` (*sample size*) can safely be omitted for any size-invariant GAM (in ebo-coordinates). For other GAMs, a default value of 1e6 will be used, corresponding to the typical size of a co-occurrence data set. The argument `b` (*balance*) can be omitted for any central GAMs. Otherwise, it defaults to a value of 1, corresponding to the centralized version of the respective GAM.

The `gam.score` function automatically converts between standard and ebo-coordinates, depending on the requirements of the GAM implementation.

Value

a vector of real numbers representing generalised association scores

See Also

`add.gams`, `gam.iso`, `builtin.gams`

Examples

```
gam.score("t.score", f=1:10, f1=(1:10)*5, f2=100, N=1000)
gam.score("t.score", o=1:10, e=(1:10)/2)
```

<code>gamma.nbest</code>	<i>Compute Gamma Threshold for N-Best Acceptance Region (gam)</i>
--------------------------	---

Description

Computes a suitable value of γ such that the acceptance region $\{g \geq \gamma\}$ contains exactly n candidates from a given data set.

Usage

```
gamma.nbest(ds, name, n, jitter=FALSE)
```

Arguments

<code>ds</code>	a UCS data set object
<code>name</code>	name of a generalised association measure (GAM)
<code>n</code>	an integer, specifying the number of candidates to be included in the acceptance region
<code>jitter</code>	if TRUE, random jitter is added to the coordinates of candidates for computation of the n-best threshold

Details

When `jitter=TRUE`, the data set `ds` must contain jitter vectors stored in special variables. Such jitter variables can easily be added with the `add.jitter` function.

Value

a real number specifying a suitable threshold γ , i.e. the data set `ds` contains exactly n candidates with a GAM score $g \geq \gamma$ (for the specified measure name)

See Also

`add.jitter`, `gam.score`, `add.gams`, `gam.iso`, `builtin.gams`

Examples

```
e <- 10^seq(-2, 1, .1)          # 100-best iso-line for UCS data set ds
gamma <- gamma.nbest(ds, "t.score", 100)
o <- gam.iso("t.score", gamma, e=e)
```

`iaa.kappa`*Inter-Annotator Agreement: Cohen's Kappa (iaa)*

Description

Compute the kappa statistic (Cohen, 1960) as a measure of intercoder agreement on a binary variable between two annotators, as well as a confidence interval according to Fleiss, Cohen & Everitt (1969). The data can either be given in the form of a 2×2 contingency table or as two parallel annotation vectors.

Usage

```
iaa.kappa(x, y=NULL, conf.level=0.95)
```

Arguments

<code>x</code>	either a 2×2 contingency table in matrix form, or a vector of logicals
<code>y</code>	a vector of logicals; ignored if <code>x</code> is a matrix
<code>conf.level</code>	confidence level of the returned confidence interval (default: 0.95, corresponding to 95% confidence)

Value

A data frame with a single row and the following variables:

<code>kappa</code>	sample estimate for the kappa statistic
<code>sd</code>	sample estimate for the standard deviation of the kappa statistic
<code>kappa.min</code> , <code>kappa.max</code>	two-sided asymptotic confidence interval for the “true” kappa, based on normal approximation with estimated variance

The single-row data frame was chosen as a return structure because it prints nicely, and results from different comparisons can easily be combined with `rbind`.

References

Cohen, Jacob (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, **20**, 37–46.

Fleiss, Joseph L.; Cohen, Jacob; Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, **72**(5), 323–327.

See Also

`iaa.pta`

Examples

```
## kappa should be close to zero for random codings
p <- 0.1 # proportion of true positives
x <- runif(1000) < p # 1000 candidates annotated randomly
y <- runif(1000) < p
iaa.kappa(x, y)
```

<code>iaa.pta</code>	<i>Inter-Annotator Agreement: Estimates for the Proportion of True Agreement (iaa)</i>
----------------------	--

Description

Compute confidence interval estimates for the proportion of true agreement between two annotators on a binary variable, as described by Krenn, Evert & Zinsmeister (2004). `iaa.pta.conservative` computes a conservative estimate that is rarely useful, while `iaa.pta.homogeneous` relies on additional assumptions. The data can either be given in the form of a 2×2 contingency table or as two parallel annotation vectors.

Usage

```
iaa.pta.conservative(x, y=NULL, conf.level=0.95, debug=FALSE)
```

```
iaa.pta.homogeneous(x, y=NULL, conf.level=0.95, debug=FALSE)
```

Arguments

<code>x</code>	either a 2×2 contingency table in matrix form, or a vector of logicals
<code>y</code>	a vector of logicals; ignored if <code>x</code> is a matrix
<code>conf.level</code>	confidence level of the returned confidence interval (default: 0.95, corresponding to 95% confidence)
<code>debug</code>	if TRUE, show which divisions of the data are considered when computing the confidence interval (see Krenn, Evert & Zinsmeister, 2004)

Details

This approach to measuring intercoder agreement is based on the assumption that the observed **surface agreement** in the data can be divided into **true agreement** (i.e. candidates where both annotators make the same choice *for the same reasons*) and **chance agreement** (i.e. candidates on which the annotators agree purely by coincidence). The goal is to estimate the proportion of candidates for which there is true agreement between the annotators, referred to as PTA.

The two functions differ in how they compute this estimate. `iaa.pta.conservative` considers all possible divisions of the observed data into true and chance agreement, leading to a conservative confidence interval. This interval is almost always too large to be of any practical value.

`iaa.pta.homogeneous` makes the additional assumption that the average proportion of true positives is the same for the part of the data where the annotators reach true agreement and for the part where they agree only by chance. Note that there is no *a priori* reason why this should be the case. Interestingly, the confidence intervals obtained in this way for the PTA correspond closely to those for Cohen's kappa statistic (`iaa.kappa`).

Value

A numeric vector giving the lower and upper bound of a confidence interval for the proportion of true agreement (both in the range $[0, 1]$).

Note

`iaa.pta.conservative` is a computationally expensive operation based on Fisher's exact test. (It doesn't use `fisher.test`, though. If it did, it would be even slower than it is now.) In most circumstances, you will want to use `iaa.pta.homogeneous` instead.

References

Krenn, Brigitte; Evert, Stefan; Zinsmeister, Heike (2004). Determining intercoder agreement for a collocation identification task. In preparation.

See Also

`iaa.kappa`

Examples

```
## how well do the confidence intervals match the true PTA?
true.agreement <- 700          # 700 cases of true agreement
chance <- 300                 # 300 cases where annotations are independent
p <- 0.1                      # average proportion of true positives
z <- runif(true.agreement) < p # candidates with true agreement
x.r <- runif(chance) < p      # randomly annotated candidates
y.r <- runif(chance) < p
x <- c(z, x.r)
y <- c(z, y.r)
cat("True PTA =", true.agreement / (true.agreement + chance), "\n")
iaa.pta.conservative(x, y)    # conservative estimate
iaa.pta.homogeneous(x, y)    # estimate with homogeneity assumption
```

`lnre.goodness.of.fit`

Perform Goodness-of-Fit Evaluation of LNRE Model

Description

Evaluate the goodness-of-fit of a LNRE model with a multivariate chi-squared test (Baayen, 2001, Sec. 3.3).

Usage

```
lnre.goodness.of.fit(model, m.max=15)
```

Arguments

<code>model</code>	an object representing a LNRE model whose parameters have been estimated from observed word frequency data. Currently, the Zipf-Mandelbrot (ZM, class "zm") and the finite Zipf-Mandelbrot (fZM, class "fzm") models are supported.
<code>m.max</code>	highest frequency rank to be included in the evaluation (limited by the number of ranks stored in the model object).

Details

This function performs a multivariate chi-squared test to evaluate the goodness-of-fit of an LNRE model (Baayen 2001, p. 119-122).

All LNRE models that follow the UCS/R conventions are supported. In particular, they must specify the number of parameters estimated from the observed data (in the `n.param` component), and they must provide appropriate implementations of the `EV`, `EVm`, and `VV` methods. Currently available LNRE models are objects of class "zm" or "fzm". The `model` object must include observed frequency data (in components `N`, `V`, and `spc`), which is usually achieved by estimating the model parameters from the observed frequency spectrum.

Value

A data frame with one row and three columns:

<code>X2</code>	the value of the multi-variate χ^2 test statistic
<code>df</code>	the degrees of freedom of the approximate χ^2 distribution of the test statistic under the null hypothesis
<code>p</code>	the p-value for the test

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

`zm`, `fzm`

<code>order.by.am</code>	<i>Sort Rows of a Data Set by Association Scores (base)</i>
--------------------------	---

Description

Sort the rows of a data set according to the annotated scores of an association measure (in descending order). Ties in the ordering are broken randomly by default, using the random association measure to yield a reproducible ordering.

Usage

```
order.by.am(ds, am, randomise=TRUE)
```

Details

With `randomise=TRUE`, the data set must contain a variable named `am.random`, which is used to break ties in the ordering. Otherwise, tied rows are arranged according to their ID values, and the corresponding `id` variable must be annotated in the data set.

The random association measure is used for breaking ties (rather than random numbers generated on the fly) in order to ensure that the ordering is reproducible. If this measure has not been annotated in a data set file, you can easily add the required variable to a data set `ds` with the command

```
ds$am.random <- runif(nrow(ds))
```

You should probably use `set.seed` to ensure a reproducible ordering.

Value

an integer vector of row numbers, which can be used as a row index for the data set object

See Also

`read.ds.gz`, `add.ranks`

<code>precision.recall</code>	<i>Compute Precision and Recall for N-Best Lists (base)</i>
-------------------------------	---

Description

Computes precision and recall of n-best lists for a UCS data set annotated with true positives and rankings (based on association scores). This function forms the basis for the evaluation graphs in the `plots` packages.

Usage

```
precision.recall(ds, am, tp=ds$b.TP, step=1, first=1, cut=0, window=0)
```

Arguments

<code>ds</code>	a UCS data set object
<code>am</code>	a character string giving the name of an association measure. The corresponding ranking must be annotated in the data set (usually with the <code>add.ranks</code> function).
<code>tp</code>	a logical vector, which must be parallel to the rows of the data set. TRUE values indicate true positives (see details below for the use of missing values). If <code>tp</code> is omitted, the data set must contain a Boolean variable <code>b.TP</code> which is used instead.
<code>step</code>	step width for n-best lists considered, i.e. precision and recall are computed for every <code>step</code> -th value of <code>n</code> only (default: 1)
<code>first</code>	smallest n-best list for which precision and recall are computed (default: 1)
<code>cut</code>	pretend that data set consists only of the first <code>cut</code> rows in the ranking, i.e. treat <code>cut</code> -best list as full data set (for percentage and recall).
<code>window</code>	if specified, local precision is estimated, considering a window of approximately the given size around each value of <code>n</code> (uses the <code>density</code> function for smoothing). Useful window sizes range from 400 to 1000.

Details

The `precision.recall` function supports evaluation based on random samples (cf. Evert, 2004, Sec. 5.4). Any NA values in the `tp` parameter (or the `b.TP` variable) are interpreted as unannotated candidates. Precision and recall values are computed from the annotated candidates only (as are the `tp`, `fp`, and `lp` variables in the returned data frame). For a random sample evaluation, confidence intervals should always be supplied with the raw precision values, and result differences should be tested for significance. Such tests are implemented by the `evaluation.plot` function, for instance.

Value

An invisible data frame with rows corresponding to n-best lists and the following variables:

<code>n</code>	the number of candidates in the n-best list
<code>perc</code>	the same as a percentage of the full data set (or the <code>cut</code> highest-ranking candidates if specified)
<code>tp</code>	the number of true positives in the n-best list
<code>fp</code>	the number of false positives in the n-best list
<code>precision</code>	the precision of the n-best list, i.e. the number of TPs divided by <code>n</code>
<code>recall</code>	the recall of the n-best list, i.e. the number of TPs divided by the total number of TPs in the data set
<code>lp</code>	if <code>window</code> is specified, an estimate for the <i>local precision</i> , i.e. the density of TPs in the vicinity of the n-th rank. Averages over a symmetric window of approximately the specified total size by convolution with a Gaussian kernel (using the <code>density</code> function).

References

Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.

See Also

`add.ranks`, `read.ds.gz`, `evaluation.plot`

<code>read.ds.gz</code>	<i>Load UCS data set file (base)</i>
-------------------------	--------------------------------------

Description

Load a UCS data set file, which is uncompressed on the fly if necessary.

Usage

```
read.ds.gz(filename)
```

Arguments

`filename` name, partial or full path of the data set file to be loaded.

Details

When the specified file is not found in the current directory, it is automatically searched in the standard UCS data library (the 'DataSet' directory and its subdirectories). Should there be multiple matches, a warning is issued and the first match is used. You may specify partial paths to identify the desired file unambiguously (e.g. "Distrib/dickens.ds.gz"). The automatic search facility is suppressed when `filename` is an explicit absolute or relative path (starting with / or ./).

gzip-compressed data set files, whose name must end in `.gz`, are automatically decompressed.

Value

A data frame with column names (i.e. variables) corresponding to those in the data set file. 11 and 12 are read as character vectors, all other string variables (`f.*`) are converted into factors, and Boolean variables (`b.*`) are converted into logicals.

Any comments and global variables in the file header are discarded.

Examples

```
## load GLAW data set from UCS distribution
GLAW <- read.ds.gz("glaw.ds.gz")
```

<code>read.spectrum</code>	<i>Read Frequency Spectrum File (lexstats)</i>
----------------------------	--

Description

Read a word frequency spectrum from a `.spc` file in `lexstats` format (see Baayen, 2001). Returns spectrum as integer vector, possibly including zeroes, whose m -th element gives the number of types V_m with frequency rank m . Also computes sample size N and vocabulary size V .

Usage

```
read.spectrum(file, m.max=Inf, expected=FALSE)
```

Arguments

<code>file</code>	a character string giving the name of a frequency spectrum file in <code>lexstats</code> format (usually with the extension <code>.spc</code>)
<code>m.max</code>	maximum length of frequency spectrum, i.e. frequency ranks $m > m_{\max}$ are discarded. Setting <code>m.max</code> is a good idea if there are high-frequency types, so that the spectrum is sparse. For most applications, only the first 10 to 100 ranks are of interest.
<code>expected</code>	if <code>TRUE</code> , reads expected class sizes (in the <code>EV_m</code> column) rather than the observed ones (in the <code>V_m</code> column). This is only possible when the <code>.spc</code> file was generated by a LNRE model, of course.

Value

A list with the following components:

<code>spc</code>	an integer vector containing the class sizes V_m
<code>N</code>	the sample size computed from the spectrum
<code>V</code>	the vocabulary size computed from the spectrum

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

`spectrum.plot`, `zm`, `fzm`

spectrum.plot *Comparative Plot of Word Frequency Spectra (lexstats)*

Description

Comparative plot of up to five word frequency spectra (see Baayen, 2001), either as a side-by-side barplot or as points and lines on a logarithmic scale.

Usage

```
spectrum.plot(spc, m.max=Inf, log=FALSE, y.min=100, y.max=0,
              xlab="m", ylab="V_m / E[V_m]",
              legend=NULL,
              pch=c(1, 3, 15, 2, 20),
              lwd=1,
              lty=c("solid", "dashed", "dotdash", "dotted", "twodash"),
              col=if (log) c("black") else c("black", "grey50", ...))
```

Arguments

spc	a list containing up to five frequency spectrum vectors. Such spectrum vectors can be read in from a file in <code>lexstats</code> format with <code>read.spectrum</code> or generated by a ZM or fZM model with the EVM method.
m.max	number of frequency ranks to be shown in plot. If unspecified, it is determined by the shortest spectrum vector in <code>spc</code> .
log	if TRUE, display frequency spectra as points and lines on a logarithmic scale. If FALSE, display spectra as side-by-side barplot on a linear scale (default). The latter is only useful when <code>m.max</code> is comparatively small.
y.min, y.max	range of y-axis. <code>y.max</code> is automatically computed to fit the data in <code>spc</code> . <code>y.min</code> is only used when <code>log=TRUE</code> and defaults to 100.
legend	a vector of character strings or expressions specifying the labels to be shown in a legend box. If <code>legend</code> is missing, no legend box will be displayed.
xlab, ylab	character strings giving labels for the x-axis and y-axis
pch, lwd, lty	vectors of plot symbols, line widths, and line types (only used if <code>log=TRUE</code>). Values are recycled if necessary. See the <code>par</code> manpage for possible ways of specifying these attributes.
col	a vector of colours for the lines (<code>log=TRUE</code>) or bars (<code>log=FALSE</code>) in the plot. Values are recycled if necessary. Colours are specified as described in the <code>par</code> manpage.

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

read.spectrum, zm, fzm, EVm

ucs.library *Load UCS/R Modules*

Description

Since the UCS/R functions are imported into the global namespace, they are collected in various modules that can be loaded separately on demand. `ucs.library` loads a specified module. When called without arguments, it prints a listing of available modules.

Usage

```
ucs.library(name, all=FALSE, reload=FALSE)
```

Arguments

<code>name</code>	a character string giving the name of a <i>single</i> UCS/R module to be loaded. If omitted, a list of all available modules is displayed (see below).
<code>all</code>	if TRUE, all available modules are loaded
<code>reload</code>	if TRUE, force module to be loaded even if it has already been imported (useful when developing UCS/R modules)

Details

Like the `library` and `package` functions, `ucs.library(module)` checks whether the requested module has already been loaded by a previous `ucs.library` call. Set `reload=TRUE` in order to skip this test and force re-loading a module (especially while developing or debugging module code).

Value

Calling the `ucs.library` function without arguments returns a list of all available UCS/R modules as an object of class "UCSLibList", which prints as a nicely formatted listing including one-line descriptions. Use `names(ucs.library())` to obtain a plain vector of module names.

See Also

UCS for an overview of the UCS/R modules

Examples

```
print(ucs.library()) # list of available modules

ucs.library("base") # load and manage UCS data sets
ucs.library("plots") # evaluation graphs

ucs.library(all=TRUE) # load all modules
```

`ucs.par`*Graphics Parameters for Evaluation Graphs (plots)*

Description

Set default graphics parameters for the `evaluation.plot` function, similar to `par` for general graphics parameters. The current parameter values are queried by giving their names as character strings. The values can be set by specifying them as arguments in `name=value` form, or by passing a single list of named values.

Usage

```
ucs.par(...)
```

```
.ucs.PAR
```

Arguments

... either character strings (or vectors) specifying the names of parameters to be queried, or parameters to be set in `name=value` form, or a single list of named values. Valid parameter names are described below.

Details

The current default parameters are stored in the global variable `.ucs.PAR`. They can be queried by giving their names as one or more character vectors to `ucs.par`. `ucs.par()` (no arguments) returns all UCS graphics parameters.

Parameters are set by specifying their names and the new values as `name=value` pairs. Such a list can also be passed as a single argument to `ucs.par`, which is typically used to restore previous parameter values (that have been saved in a list variable).

In order to restore the "factory settings", reload the module with the command `ucs.library("plots", reload=TRUE)`.

Value

When parameters are set, their former values are returned in an invisible named list. Such a list can be passed as a single argument to `ucs.par` to restore the parameter values.

When a single parameter is queried, its value is returned directly. When two or more parameters are queried, the result is a named list.

Note the inconsistency, which is the same as for `par`: setting one parameter returns a list, but querying one parameter returns a vector (or a scalar, i.e. a vector of length 1).

UCS Graphics Parameters

- col** A character or integer vector specifying line colours for up to 10 evaluation graphs (see the `par` manpage for details). Values are recycled if necessary.
- lty** A character or integer vector specifying line styles for up to 10 evaluation graphs (see the `par` manpage for details). Values are recycled if necessary.
- lwd** A numeric vector specifying line widths for up to 10 evaluation graphs (see the `par` manpage for details). Values are recycled if necessary.
- bw.col** The line colours used in B/W mode (see the `evaluation.plot` manpage for details).
- bw.lty** The line styles used in B/W mode.
- bw.lwd** The line widths in B/W mode.
- n.first** The smallest n-best list to be evaluated (default: 100). Shorter n-best lists typically lead to highly unstable evaluation graphs. It may be necessary to set `n.first` to a higher value for evaluation based on random samples (cf. `evaluation.plot`).
- n.step** The step width for n-best lists in evaluation graphs (default: 1). The default setting evaluates all possible n-best lists. Higher values speed up computation, make graphs look less jagged, and reduce the size of PostScript files. A useful range is 5...20, depending on the size of the data set file.
- test.step** Step width for n-best lists where significance tests for result differences are applied, as a multiple of `n.step` (default: 10). Since these tests are time-consuming and significant differences are indicated by fairly large symbols in the plot, values below 5 are rarely useful.
- cex** A character expansion factor for labels, annotations, and symbols in evaluation plots (see `par` for details).
- lex** This parameter can be used to increase the line widths of evaluation graphs and some decorations. Note that `lex` is not an expansion factor, but is simply *added* to all line widths in the plot.
- do.file** If FALSE, `evaluation.file` will not generate PostScript files, which is useful while testing and fine-tuning plots (default: TRUE).

See Also

`evaluation.plot`, `evaluation.file`, `par`

Examples

```
print(names(ucs.par()))          # list available parameters

ucs.par("col", "lty", "lwd")    # the default line styles
ucs.par(c("col", "lty", "lwd")) # works as well

## temporary changes to graphics paramters:
par.save <- ucs.par(n.first=200, n.step=5)
## plots use the modified parameters here
ucs.par(par.save)               # restore previous values

ucs.library("plots", reload=TRUE) # reload module for factory defaults
```

<code>write.lexstats</code>	<i>Write Data Files for Goodness-of-Fit Evaluation of LNRE Model (zm, fzm)</i>
-----------------------------	--

Description

Creates three data files in `lexstats` format, which can be used to compare and LNRE model with other models from the `lexstats` package and evaluate its goodness-of-fit by a multivariate chi-squared test (Baayen, 2001, Sec. 3.3), using the `lnreChi2` program (Baayen, 2001).

Usage

```
write.lexstats(model, file)
```

Arguments

<code>model</code>	an object of class "zm" or "fzm", representing a Zipf-Mandelbrot (ZM) or finite Zipf-Mandelbrot (fZM) LNRE model. The object must include observed word frequency data (in components <code>N</code> , <code>V</code> , and <code>spc</code>), usually because the model parameters have been estimated from the observed frequency spectrum.
<code>file</code>	a character string giving the basename of the files that will be created

Details

This functions creates files in `lexstats` format with the extensions `.spc`, `.sp2`, and `.ev2`, which are required by the `lnreChi2` tool (Baayen, 2001, 270).

In addition, the basename `file` is extended with the string `"_bZM"` (for a ZM model) or `"_bfZM"` (for a fZM model), so that the `lnreChi2` tool can correctly identify the number of degrees of freedom (reduced by two estimated parameters for the ZM model, and three estimated parameters for the fZM model).

Value

The full basename of the created files (obtained by adding a model-specific suffix to the `file` parameter).

Note

The combination of `write.lexstats` and the external `lnreChi2` program to evaluate the goodness-of-fit of a LNRE model has been superseded by the built-in `lnre.goodness.of.fit` function (in the `lexstats` module). This function implements the multivariate chi-squared test as described by Baayen (2001, Sec. 3.3) in R without relying on external software.

References

Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.

See Also

zm, fzm, EV, EVm, lnre.goodness.of.fit

zm	<i>The Zipf-Mandelbrot LNRE Model (zm)</i>
----	--

Description

Object constructor for a Zipf-Mandelbrot (ZM) LNRE model with parameters α and C (Evert, 2004a). Either the parameters are specified explicitly, or one or both of them can be estimated from an observed frequency spectrum.

Usage

```
zm(alpha, C)
```

```
zm(alpha, N, V)
```

```
zm(N, V, spc, m.max=15, stepmax=10, debug=FALSE)
```

Arguments

alpha	a number in the range (0, 1), the shape parameter α of the ZM model. alpha can automatically be estimated from N, V, and spc.
C	a positive number, the parameter C of the ZM model. C can automatically be estimated from N and V.
N	the sample size, i.e. number of observed tokens
V	the vocabulary size, i.e. the number of observed types
spc	a vector of non-negative integers representing the class sizes V_m of the observed frequency spectrum. The vector is usually read from a file in lexstats format with the read.spectrum function.
m.max	the number of ranks from spc that will be used to estimate the α parameter
stepmax	maximal step size of the nlm function used for parameter estimation. It should not be necessary to change the default value.
debug	if TRUE, print debugging information during the parameter estimation process. This feature can be useful to find out why parameter estimation fails.

Details

The ZM model with parameters $\alpha \in (0, 1)$ and $C > 0$ is defined by the type density function

$$g(\pi) := C \cdot \pi^{-\alpha-1}$$

for $0 \leq \pi \leq B$, where the upper bound B is determined from C by the normalisation condition

$$\int_0^{\infty} \pi \cdot g(\pi) d\pi = 1$$

The parameter α is estimated by nonlinear minimisation (nlm) of a multinomial chi-squared statistic for the observed against the expected frequency spectrum. Note that this is different from the multivariate chi-squared test used to measure the goodness-of-fit of the final model (Baayen, 2001, Sec. 3.3).

See Evert (2004, Ch. 4) for further mathematical details, especially concerning the expected vocabulary size, frequency spectrum and conditional parameter distribution, as well as their variances.

Value

An object of class "zm" with the following components:

alpha	value of the α parameter
B	value of the upper bound B (a normalisation device)
C	value of the C parameter
N	number of observed tokens (if specified)
V	number of observed types (if specified)
spc	observed frequency spectrum (if specified)

This object prints a short summary, including a comparison of the first ranks of the observed and expected frequency spectrum (if available).

References

- Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer, Dordrecht.
- Evert, Stefan (2004). *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. PhD Thesis, IMS, University of Stuttgart.
- Evert, Stefan (2004a). A simple LNRE model for random character sequences. In *Proceedings of JADT 2004*, Louvain-la-Neuve, Belgium, pages 411–422.

See Also

fzm, EV, EVm, VV, VVm, write.lexstats, lnre.goodness.of.fit, read.spectrum, and spectrum.plot

Zusammenfassung

Das gemeinsame Vorkommen von Wörtern in natürlicher Sprache – sei es in unmittelbarer Nachbarschaft, innerhalb desselben Satzes oder in einer bestimmten syntaktischen Relation – stellt eine zentrale Wissensquelle für die maschinelle Sprachverarbeitung dar. Frequenzdaten für derartige Kookkurrenzen (*cooccurrences*) können leicht aus Textkorpora gewonnen werden, wobei in den meisten Fällen zunächst eine linguistische Vorverarbeitung erfolgt (diese besteht traditionell aus Wortartenannotation und Lemmatisierung, und wird heutzutage oft durch eine partielle syntaktische Analyse ergänzt). Eine mathematische Auswertung erlaubt dann, diese Ergebnisse über das spezifische Extraktionskorpus hinaus zu verallgemeinern und auf statistische Assoziationen zwischen dem Vorkommen der beteiligten Wörter in der Sprache insgesamt (oder zumindest in einer Teilsprache) zu schließen.¹ Das gebräuchlichste Verfahren hierfür sind sogenannte Assoziationsmaße (*association measures*), die ausgehend von der im Korpus ermittelten Frequenzinformation eine Bewertungszahl (*association score*) errechnen: je höher dieser Wert, desto stärker ist die mutmaßliche Assoziation. Dabei stützt sich das Maß lediglich auf die Kookkurrenzhäufigkeit (*cooccurrence frequency*) und auf die Häufigkeiten der einzelnen Wörter (*marginal frequencies*).

Die so gewonnene Information läßt sich in vielfältiger Weise anwenden, unter anderem zur Desambiguierung von syntaktischen Analysen, zur Identifikation von Satz- und Phrasengrenzen, zur Verbesserung von stochastischen Sprachmodellen, zur Lesartendesambiguierung und anderen Klassifikationsaufgaben, sowie zur Bestimmung von semantischen Ähnlichkeiten zwischen Wörtern wie Synonymie und Hyponymie (siehe Abschnitt 1.2.1). Andererseits bieten statistische Assoziationen einen wichtigen Anhaltspunkt für die Identifikation lexikalischer Wortverbindungen, sogenannter Kollokationen (*collocations*).² Das gebräuchlichste Verfahren zur Extraktion von Kollokationen aus Textkorpora wird in Abschnitt 1.2.2 dargestellt und dient später auch als Grundlage für wissenschaftliche Untersuchungen.

Bereits zur Zeit der ersten computerlinguistischen Arbeiten mit Kookkurrenzdaten und Kollokationen stand eine nahezu unüberschaubare Vielfalt von Assoziationsmaßen zur Verfügung: man bediente sich bei diversen Fachgebieten, allen voran

¹Der Begriff *Assoziation* wird in der vorliegenden Arbeit stets in seiner statistischen Bedeutung gebraucht: “the tendency of two events to occur together” (Porkess 1991, s.v. *association*) und ist nicht mit dem gleichlautenden psycholinguistischen Begriff zu verwechseln.

²Ich schließe mich damit einer in der Computerlinguistik gebräuchlichen Verwendung von „Kollokation“ als Sammelbegriff für verschiedene Arten lexikalischer Wortverbindungen an. In anderen Fachrichtungen werden Kollokationen speziell als semikompositionelle Kombinationen aufgefaßt oder sogar mit statistischer Assoziation gleichgesetzt. Eine ausführliche und ansprechende Diskussion verschiedener Kollokationsbegriffe findet sich bei Bartsch (2004, 27–64). Darüberhinaus werden linguistische Definitionsansätze beschrieben (Bartsch 2004, Ch. 3, 65–78).

natürlich der mathematischen Statistik. Schon im Jahr 1964 zog Vincent Giuliano nach dem *Washington Symposium on Statistical Association Methods for Mechanized Documentation* das Fazit:

It soon becomes evident that at least a dozen somewhat different procedures and formulae for associations are suggested. . . . One thing which is badly needed is a better understanding of the boundary conditions under which the various techniques are applicable and the expected gains to be achieved through using one or the other of them. This advance would primarily be one in theory, not in abstract statistical theory but in a problem-oriented branch of statistical theory. (Giuliano 1965b, 259)

Giuliano wünscht sich hier eine Art Enzyklopädie der Assoziationsmaße, die neben einer Zusammenstellung der mathematischen Grundlagen und einer reinen Auflistung von Formeln Beziehungen zwischen den Maßen knüpfen und Unterschiede deutlich machen soll, sowohl auf einer theoretischen Ebene als auch im Hinblick auf Anwendungen.

Seit Giulianos Fazit sind nunmehr vierzig Jahre vergangen, doch die Situation ist im wesentlichen die gleiche geblieben – wenn nicht sogar noch verwirrender geworden, da ständig neue Assoziationsmaße hinzukommen (und manchmal auch alte wiederentdeckt werden). Jeder neue Vorschlag wird mit wahrscheinlichkeitstheoretisch, philosophischen oder einfach pragmatischen Argumenten untermauert; Kookkurrenzdaten und Assoziationsmaße werden mit wechselndem Erfolg in zahlreichen Anwendungen eingesetzt; hin und wieder werden Fallstudien durchgeführt, die über die Eigenschaften verschiedener Maße Aufschluß geben sollen; und es gibt eine Handvoll ernstzunehmender Evaluationen, die eine größere Anzahl von Assoziationsmaßen hinsichtlich ihres Nutzens für die Kollokationsextraktion vergleichen. Giulianos Wunsch nach einer Enzyklopädie kommt wohl das 5. Kapitel von Manning and Schütze (1999) am nächsten. Dort werden vier weitverbreitete Assoziationsmaße beschrieben, und die ihnen üblicherweise nachgesagten Eigenschaften werden durch kurze Listen „interessanter Bigramme“ illustriert. Was jedoch nach wie vor fehlt, ist eine umfassende Zusammenstellung des bekannten Wissens, die theoretische und empirische Aspekte berücksichtigt und miteinander verknüpft. Meine Dissertation soll einen Beitrag dazu leisten, diese Lücke endlich zu schließen.

In Kapitel 2 werden zunächst die wichtigsten theoretischen Grundlagen zusammengestellt, beginnend mit geeigneten Zählverfahren für Kookkurrenzhäufigkeiten. Dabei lege ich Wert auf eine klare Trennung zwischen einem auf syntaktischen Relationen basierenden Kookkurrenz begriff (*relational cooccurrences*) einerseits und einem auf dem Abstand zwischen Wörtern bzw. dem gemeinsamen Vorkommen in einer textstrukturellen Einheit basierenden Begriff (*positional cooccurrences*) andererseits. Die beiden Arten von Kookkurrenzen erfordern unterschiedliche Zählmethoden und angepaßte statistische Modelle. Letztlich können sie dann jedoch mit denselben Assoziationsmaßen bewertet werden. Ein großer Teil des Kapitels widmet sich den statistischen Modellen für die Analyse solcher Frequenzdaten, wobei alle relevanten Formeln explizit und in einem einheitlichen Formalismus dargestellt werden. Dazu gehört auch eine Diskussion der verschiedenen mathematischen Ansätze, Assoziation zu quantifizieren. Schließlich wird auch die oft vernachlässigte Frage nach der Anwendbarkeit der statistischen Methoden auf korpuslinguistische Daten angespro-

chen, und die Auswirkungen von Vorverarbeitungs- bzw. Extraktionsfehlern wird diskutiert.

Kapitel 3 setzt den enzyklopädischen Teil mit einer umfassenden Sammlung bekannter Assoziationsmaße fort, die anhand ihres theoretischen Hintergrundes in Gruppen eingeteilt werden. Dabei stellt sich heraus, daß es zahlreiche Ähnlichkeiten und Verwandtschaftsbeziehungen auch zwischen Maßen aus verschiedenen Gruppen gibt. Auf diese Weise wird die zunächst unüberschaubare Vielfalt von Maßen reduziert und weiter strukturiert. Besonderer Wert wird auf eine explizite Darstellung aller Formeln und einheitliche Notation gelegt, so daß sämtliche Assoziationsmaße leicht auf dem Computer umgesetzt werden können. Wo nötig wird auf potentielle Probleme und leicht zu übersehende Details hingewiesen. Darüber hinaus ist eine Referenzimplementierung aller Maße verfügbar, die numerische Genauigkeit und korrektes Verhalten unter Randbedingungen sicherstellen soll.

Der zweite Teil des Kapitels schlägt einen neuen Weg zur Erforschung von Assoziationsmaßen ein, der sich von rein theoretischen Diskussionen abwendet und stattdessen empirische Untersuchungen und ein intuitives Verständnis der Eigenschaften verschiedener Maße in den Mittelpunkt stellt. Schlüssel hierzu ist ein allgemeines Modell für Assoziationsmaße (sogenannte *generalised association measures*), das eine geometrische Interpretation der Formeln ermöglicht. Dabei werden die aus einem Korpus gewonnenen Frequenzdaten mit Punkten in einem dreidimensionalen Raum gleichgesetzt – jeder Punkt entspricht einem Wortpaar. Assoziationsmaße lassen sich dann als Flächen in diesem “Frequenzraum” veranschaulichen. Die Eigenschaften eines Maßes werden durch die Form der zugehörigen Flächen bestimmt, und durch ihren Vergleich werden die Unterschiede bzw. Gemeinsamkeiten verschiedener Maße deutlich. Mit Hilfe der neu gewonnenen Methoden wird schließlich die Untersuchung und Klassifikation der eingeführten Assoziationsmaße fortgesetzt.

Kapitel 4 wendet sich wieder dem statistischen Modell für Frequenzdaten zu und beschäftigt sich mit der Genauigkeit und Zuverlässigkeit von statistischen Tests und Schätzwerten. Im Gegensatz zur gängigen mathematischen Theorie wird dabei die typische ungleichmäßige Verteilung von Worthäufigkeiten berücksichtigt, bei der einer kleinen Menge häufiger Wörter eine riesige Anzahl von extrem seltenen Wörtern gegenübersteht (als „Zipfsches Gesetz“ bekannt). Mit Hilfe von Methoden aus dem Gebiet der Lexikostatistik kann nachgewiesen werden, daß herkömmliche statistische Schlußfolgerungen und Schätzwerte aufgrund der großen Zahl seltener Wörter grundsätzlich unzuverlässig sind, besonders wenn sie sich auf lediglich ein oder zwei Vorkommen eines Wortes stützen. Erst bei fünf oder mehr Vorkommen spielt die Verteilung der Worthäufigkeiten keine wesentliche Rolle mehr. Dieses Resultat liefert eine theoretische Begründung für die weitverbreitete Praxis, nur diejenigen Ereignisse zu berücksichtigen, deren Häufigkeit einen gewissen Schwellwert überschreitet. Als Nebenprodukt dieser Untersuchungen ist ein einfaches und effizientes Modell für Wortfrequenzverteilungen entstanden, das dennoch zumindest für die Beschreibung großer Korpora gleichwertig zu anderen bekannten Modellen (siehe Baayen 2001) ist oder diese sogar übertrifft.

In Kapitel 5 wird schließlich eine Verbindung zwischen Kookkurrenzen und Kollokationen hergestellt, indem Assoziationsmaße als Werkzeug zur Extraktion von Kollokationen aus Textkorpora eingesetzt und im Rahmen dieser Anwendung evaluiert werden. Ich argumentiere dabei für eine manuelle Evaluation, bei der alle aus ei-

nem Korpus gewonnenen Kollokationskandidaten von Experten geprüft und annotiert werden. Durch den Vergleich mit nach Assoziationswerten sortierten Listen läßt sich dann jedem Maß eine Güte zuordnen und z.B. durch die *precision* (d.h. den Anteil "echter" Kollokationen unter einer gewissen Anzahl von Kandidaten mit den höchsten Assoziationswerten) quantitativ messen. Verschiedene graphische Darstellungen ermöglichen einen anschaulichen und aussagekräftigen Vergleich der Assoziationsmaße. Anhand einer Fallstudie wird gezeigt, wie die Kombination verschiedener Methoden zu neuen Erkenntnissen führt. Der erhebliche Arbeitsaufwand für die manuelle Prüfung der Kollokationskandidaten läßt sich deutlich reduzieren, indem nur zufällig ausgewählte Stichproben annotiert werden. Durch Anwendung geeigneter statistischer Signifikanztests ist sichergestellt, daß keine irrtümlichen Schlußfolgerungen aus rein zufälligen Vorkommnissen gezogen werden. Bei der Formulierung dieser Signifikanztests spielt das geometrische Modell aus Kapitel 3 wieder eine wesentliche Rolle.

Sowohl die Gültigkeit der statistischen Modelle als auch die Ergebnisse einer Evaluation von Assoziationsmaßen hängen von zahlreichen Faktoren ab: neben Textsorte und Größe des Extraktionskorpus spielen die Qualität der linguistischen Vorverarbeitung, die betrachtete Art von Kookkurrenzen und besonders die genaue Ausprägung des Kollokationsbegriffs eine entscheidende Rolle. Dies hat zur Folge, daß sich Ergebnisse empirischer Untersuchungen nur in sehr beschränktem Maße auf andere Situationen übertragen lassen. Um ein besseres Verständnis der statistischen Eigenschaften von Kollokationen zu erreichen ist es daher erforderlich, zahlreiche Experimente unter den verschiedensten Bedingungen durchzuführen. Die vorliegende Arbeit stellt das notwendige Handwerkszeug bereit, was durchaus nicht nur im übertragenen Sinn gemeint ist: wesentlicher Bestandteil der Dissertation ist ein umfangreiches und vollständig dokumentiertes Softwarepaket (das UCS-Toolkit), mit dem sich alle beschriebenen Experimente leicht nachvollziehen lassen (die dazu notwendigen Daten, Programme und Beschreibungen sind in dem Softwarepaket enthalten). Die vollständige Dokumentation des UCS-Toolkit ist in Anhang B abgedruckt.

Summary

In natural language, words are not combined randomly into phrases and sentences, constrained only by the rules of syntax. They have a tendency to appear in certain recurrent combinations, prompting Firth (1957) to coin his famous slogan: *You shall know a word by the company it keeps!*. Indeed, such *cooccurrences* – whether they are immediately adjacent words, stand in a particular syntactic relation or just tend to be used in the same sentence – are a goldmine of information for linguistics and natural language processing. They include compound nouns (*black box*), fixed idioms (*kick the bucket*), lexically determined combinations (*heavy smoker*) and formulaic expressions (*have a nice day*). They can often tell us something about the meaning of a word (think of combinations like *dark night* and *bright day*), an idea that has inspired latent semantic analysis and similar vector space models of word meaning.

With modern computers it is easy to extract evidence for recurrent word pairs from huge text corpora, often aided by linguistic pre-processing and annotation (so that specific combinations, e.g. noun+verb can be targeted). However, the raw data – in the form of frequency counts for word pairs – are often not very meaningful as a measure for the amount of “glue” between two words. Provided that both words are sufficiently frequent, their cooccurrences might be pure coincidence. Therefore, a statistical interpretation of the frequency data is necessary, which determines the degree of statistical association between the words and attempts to factor out the effects of chance. The most widely used method is the application of so-called *association measures*, which assign a score to each word pair based on the observed frequency data. The higher this score is, the stronger and more certain the association between the two words.

The earliest reports of the application of association measures to language data go back to Stevens *et al.* (1965). Even at that time, an enormous range of different measures was available, borrowed from mathematical statistics and related fields. With so many options, but little guidance as to which measure to choose, Giuliano (1965b, 259) reflected: “One suspects that each has its own possible merits and disadvantages, but the line between the profound and the trivial often appears blurred. One thing which is badly needed is a better understanding of the boundary conditions under which the various techniques are applicable and the expected gains to be achieved through using one or the other of them. . . . it is my feeling that the time is now ripe to conduct carefully controlled experiments of an evaluative nature, . . .”

It is amazing to see how little progress has been made in the understanding of word cooccurrences and association measures in the forty years that have passed since these words were written. The reference work that Giuliano felt was so urgently needed – a compendium that lists, explains and compares the multitude of available association measures – has never seen the light of day. My thesis aims to fill this

gap, providing both a comprehensive reference and a methodology for the kind of research Giuliano envisaged.

Chapter 2 collects the foundations of association measures: procedures for obtaining cooccurrence frequency data and statistical models for their interpretation. I make a clear distinction between relational cooccurrences (which are usually head-modifier combinations) and positional cooccurrences (which are words that occur close to each other but need not be in a direct relation). The two types of cooccurrences require different counting methods in order to allow for a sound statistical analysis. In Sections 2.1 and 2.4 these counting methods are formalised to the degree necessary to give an unambiguous account, and they are accompanied by explicit instructions, schemata and examples to facilitate their implementation. Section 2.2 describes the statistical model underlying the analysis of the extracted frequency data. Although this is a well-known random sample model, and it is always implicitly assumed when authors discuss or suggest association measures, its application to cooccurrence frequency data has never been given an explicit and precise definition.³ In Section 2.3 I discuss some problematic aspects of the random sample model, in particular the randomness assumption and the issue of noise introduced by automatic processing.

Chapter 3 is the centrepiece of my thesis. Continuing the encyclopaedic part, it provides a comprehensive inventory of all association measures that I have come across during my research. The numerous measures are organised in major and minor groups which share a common goal or theoretical background. In addition to this broad categorisation and the standard references, I take care to explain details that are often ignored or taken for granted. Examples are the application of Yates' continuity correction, the difference between one-sided and two-sided measures, and the existence of several equivalent versions of the chi-squared and log-likelihood measures (see Section 3.1.4 for all three examples). For each association measure, an explicit equation is given, using the same notation with observed and expected frequencies to facilitate implementation. Carefully designed reference implementations are available in the UCS toolkit (Section 3.2.2). There is also an online version of the collection at <http://www.collocations.de/AM/> with the most up-to-date information.

In the second part of this chapter, Section 3.3 introduces generalised association measures as arbitrary real-valued functions on contingency tables that conform to our intuitions about fundamental properties of association measures. This formal model allows an intuitive geometric interpretation of cooccurrence data and association measures in a three-dimensional "parameter space". The frequency data are represented as a set of points in this space, each point corresponding to a word pair. Generalised association measures can then be understood as surfaces, their properties being determined by the specific shape of each surface. This visual approach will hopefully pave the way towards a better understanding of the characteristics of existing measures and towards the discovery of genuinely new ones. In Section 3.4 it is used to learn more about the different groups of association measures, and about differences between the measures in each group.

Chapter 4 addresses the well-known problem of low-frequency data. Most re-

³Cooccurrence data as a random sample from what population? And what are the relevant parameters, random variables and test statistics?

searchers know that statistical inference from small amounts of data is problematic (to say the least). Although Dunning (1993) suggests that the applicability of his newly introduced log-likelihood measure extends even down to the hapax legomena (word combinations that occur just once in a corpus) – and although Weeber *et al.* (2000) see opportunities to extract useful knowledge from such lowest-frequency data – most researchers silently discard rare events by setting a frequency threshold (Krenn (2000) is just one example among many). Using methods from lexical statistics, I show that reliable statistical inference is impossible *in principle* for the hapax and dis legomena ($f = 1, 2$). In this frequency range, quantisation effects and the characteristic highly skewed distribution of the cooccurrence probabilities of pair types (roughly following Zipf's law) dominate over the random variation that statistical inference normally takes into account. As a result, probability estimates are entirely unreliable unless the precise shape of the population is known. This rather negative result provides theoretical support for the application of a frequency threshold, which should at least exclude the hapax and dis legomena ($f \geq 3$). Quantisation and the shape of the population no longer play a role for $f \geq 5$, so that higher cutoff thresholds are not necessary in order to ensure a reliable statistical analysis.⁴ A fall-out from this work is a new population model for the distribution of cooccurrence probabilities, which is analytically simple and numerically efficient. Despite its simplicity, the model compares favourably with established population models (Baayen 2001), combining better goodness-of-fit with higher robustness.

Finally, Chapter 5 addresses the relation between statistical association and linguistic phenomena, using cooccurrence data extracted from a text corpus as candidate data for a collocation identification task. This application setting provides a framework – and a well-defined goal – for the comparative evaluation of association measures. The graphical presentation of the evaluation results, first used by Evert *et al.* (2000) and Evert and Krenn (2001), is developed further and a case study exemplifies the possibilities opened up by a fine-grained evaluation. Section 5.2 addresses the problem of testing the significance of evaluation results. An attempt is made to clear up the confusion about the choice of an appropriate significance test by introducing an explicit model for the random variation of evaluation results, which also makes use of the geometric interpretation introduced in Section 3.3. Based on this model, two procedures are suggested: (i) confidence intervals estimate the uncertainty in the evaluation results of a single association measure; and (ii) significance tests predict whether the observed differences between measures can reliably be reproduced in other experiments (under similar conditions). The model is validated on empirical data, showing that it provides a relatively tight lower bound for the true variation. Finally, the newly developed methods are applied to an evaluation procedure that reduces the amount of manual annotation work drastically by taking a random sample from the candidate set. With this new procedure, it will be possible to perform evaluation experiments under a much broader range of conditions.

⁴There may be other reasons to apply a higher frequency threshold, of course, such as working around the problems that certain association measures have with low-frequency data or inflated frequencies caused by violations of the randomness assumption.

Bibliography

- Agresti, Alan (1990). *Categorical Data Analysis*. John Wiley & Sons, New York.
- Agresti, Alan (1992). A survey of exact inference for contingency tables. *Statistical Science*, 7(1), 131–153.
- Alshawi, Hiyan and Carter, David (1994). Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20(4), 635–648.
- Aston, Guy and Burnard, Lou (1998). *The BNC Handbook*. Edinburgh University Press, Edinburgh. See also the BNC homepage at <http://www.natcorp.ox.ac.uk/>.
- Baayen, R. Harald (1996). The randomness assumption in word frequency statistics. In G. Perissinotto (ed.), *Research in Humanities Computing* 5, pages 17–31. Oxford University Press, Oxford.
- Baayen, R. Harald (2001). *Word Frequency Distributions*. Kluwer Academic Publishers, Dordrecht.
- Baayen, R. Harald and Renouf, Antoinette (1996). Chronicling the Times: Productive lexical innovations in an English newspaper. *Language*, 72(1), 69–96.
- Bannard, Colin; Baldwin, Timothy; Lascarides, Alex (2003). A statistical approach to the semantics of verb-particles. In *Proceedings of the ACL Workshop on Multiword Expressions*, pages 65–72, Sapporo, Japan.
- Barlow, Michael and Kemmer, Suzanne (eds.) (2000). *Usage-based Models of Language*. CSLI Publications, Stanford.
- Barnard, G. A. (1947). Significance tests for 2×2 tables. *Biometrika*, 34(1/2), 123–138.
- Baroni, Marco; Matiasek, Johannes; Trost, Harald (2002). Unsupervised discovery of morphologically related words based orthographic and semantic similarity. In *Proceedings of the ACL Workshop on Morphological and Phonological Learning*, pages 48–57.
- Bartsch, Sabine (2004). *Structural and Functional Properties of Collocations in English*. Narr, Tübingen.
- Beeferman, Doug; Berger, Adam; Lafferty, John (1997). A model of lexical attraction and repulsion. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)*, pages 373–380.

- Berry-Rogghe, Godelieve L. M. (1973). The computation of collocations and their relevance to lexical studies. In A. J. Aitken, R. W. Bailey, and N. Hamilton-Smith (eds.), *The Computer and Literary Studies*, pages 103–112. Edinburgh.
- Biber, Douglas (1993). Co-occurrence patterns among collocations: A tool for corpus-based lexical knowledge acquisition. *Computational Linguistics*, **19**(3), 549–556.
- Biemann, Christian; Bordag, Stefan; Quasthoff, Uwe (2004). Automatic acquisition of paradigmatic relations using iterated co-occurrences. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 967–970, Lisbon, Portugal.
- Blaheta, Don and Johnson, Mark (2001). Unsupervised learning of multi-word verbs. In *Proceedings of the ACL Workshop on Collocations*, pages 54–60, Toulouse, France.
- Breidt, Elisabeth (1993). Extraction of N-V-collocations from text corpora: A feasibility study for German. In *Proceedings of the 1st ACL Workshop on Very Large Corpora*, Columbus, Ohio. (a revised version is available from <http://arxiv.org/abs/cmp-1g/9603006>).
- Brent, Michael R. (1993). From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, **19**(2), 243–262.
- Burger, Harald; Buhofer, Annelies; Sialm, Ambros (1982). *Handbuch der Phraseologie*. de Gruyter, Berlin, New York.
- Bußmann, Hadumod (1990). *Lexikon der Sprachwissenschaft*. Kröner, Stuttgart, 2nd edition.
- Carletta, Jean (1996). Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, **22**(2), 249–254.
- Choueka, Yaacov (1988). Looking for needles in a haystack. In *Proceedings of RIAO '88*, pages 609–623.
- Choueka, Yaacov; Klein, Shmuel T.; Neuwitz, E. (1983). Automatic retrieval of frequent idiomatic and collocational expressions in a large corpus. *Journal of the Association for Literary and Linguistic Computing (ALLC)*, **4**.
- Church, Kenneth; Gale, William A.; Hanks, Patrick; Hindle, Donald (1991). Using statistics in lexical analysis. In *Lexical Acquisition: Using On-line Resources to Build a Lexicon*, pages 115–164. Lawrence Erlbaum.
- Church, Kenneth W. (2000). Empirical estimates of adaptation: The chance of two Noriegas is closer to $p/2$ than p^2 . In *Proceedings of COLING 2000*, pages 173–179, Saarbrücken, Germany.
- Church, Kenneth W. and Gale, William A. (1991). Concordances for parallel text. In *Proceedings of the 7th Annual Conference of the UW Center for the New OED and Text Research*, Oxford, UK.

- Church, Kenneth W. and Hanks, Patrick (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, **16**(1), 22–29.
- Cox, D. R. (1970). The continuity correction. *Biometrika*, **57**(1), 217–219.
- Daille, Béatrice (1994). *Approche mixte pour l'extraction automatique de terminologie : statistiques lexicales et filtres linguistiques*. Ph.D. thesis, Université Paris 7.
- Daille, Béatrice (1996). Study and implementation of combined techniques for automatic extraction of terminology. In J. L. Klavans and P. Resnik (eds.), *The Balancing Act*, chapter 3, pages 49–66. MIT Press, Cambridge, MA.
- DeGroot, Morris H. and Schervish, Mark J. (2002). *Probability and Statistics*. Addison Wesley, Boston, 3 edition.
- Dennis, Sally F. (1965). The construction of a thesaurus automatically from a sample of text. In M. E. Stevens, V. E. Giuliano, and L. B. Heilprin (eds.), *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, volume 269 of *National Bureau of Standards Miscellaneous Publication*, pages 61–148, Washington, DC.
- Dias, Gaël (2003). Multiword unit hybrid extraction. In *Proceedings of the ACL Workshop on Multiword Expressions*, Sapporo, Japan.
- Dias, Gaël; Guilloire, Sylvie; Lopes, José G. P. (1999). Language independent automatic acquisition of rigid multiword units from unrestricted text corpora. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Cargèse, France.
- Dorow, Beate and Widdows, Dominic (2003). Discovering corpus-specific word senses. In *Companion Volume to the Proceedings of the 10th Conference of The European Chapter of the Association for Computational Linguistics*, pages 79–82.
- Dunning, Ted E. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, **19**(1), 61–74.
- Dunning, Ted E. (1998). *Finding Structure in Text, Genome and Other Symbolic Sequences*. Ph.D. thesis, Department of Computer Science, University of Sheffield.
- Edmonds, Philip (1997). Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 8th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1997)*, pages 507–509, Madrid, Spain.
- Edmundson, H. P. (1965). A correlation coefficient for attributes or events. In M. E. Stevens, V. E. Giuliano, and L. B. Heilprin (eds.), *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, volume 269 of *National Bureau of Standards Miscellaneous Publication*, pages 41–44, Washington, DC.
- Eisele, Andreas (1999). *Representation and stochastic resolution of ambiguity in constraint-based parsing*. Ph.D. thesis, IMS, University of Stuttgart.

- Erbach, Gregor and Krenn, Brigitte (1993). Idioms and support verb constructions in HPSG. CLAUS-Report 28, Universität des Saarlandes, Saarbrücken.
- Evert, Stefan (2004a). Significance tests for the evaluation of ranking methods. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling 2004)*, Geneva, Switzerland.
- Evert, Stefan (2004b). A simple LNRE model for random character sequences. In *Proceedings of the 7èmes Journées Internationales d'Analyse Statistique des Données Textuelles*, pages 411–422, Louvain-la-Neuve, Belgium.
- Evert, Stefan and Kermes, Hannah (2003). Experiments on candidate data for collocation extraction. In *Companion Volume to the Proceedings of the 10th Conference of The European Chapter of the Association for Computational Linguistics*, pages 83–86.
- Evert, Stefan and Krenn, Brigitte (2001). Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 188–195, Toulouse, France.
- Evert, Stefan and Krenn, Brigitte (2005). Using small random samples for the manual evaluation of statistical association measures. *Computer Speech and Language*, **19**(4), 450–466.
- Evert, Stefan; Heid, Ulrich; Lezius, Wolfgang (2000). Methoden zum Vergleich von Signifikanzmaßen zur Kollokationsidentifikation. In W. Zühlke and E. G. Schukat-Talamazzini (eds.), *KONVENS-2000 Sprachkommunikation*, pages 215 – 220. VDE-Verlag.
- Fano, Robert M. (1961). *Transmission of information; a statistical theory of communications*. MIT Press, New York.
- Ferret, Olivier (2002). Using collocations for topic segmentation and link detection. In *Proceedings of COLING 2002*, Taipei, Taiwan.
- Firth, J. R. (1957). A synopsis of linguistic theory 1930–55. In *Studies in linguistic analysis*, pages 1–32. The Philological Society, Oxford.
- Fisher, R. A. (1922). On the interpretation of χ^2 from contingency tables and the calculation of P . *Journal of the Royal Statistical Society*, **85**(1), 87–94.
- Fisher, R. A. (1934). *Statistical Methods for Research Workers*. Oliver & Boyd, Edinburgh, 2nd edition.
- Fisher, R. A. (1935). The logic of inductive inference. *Journal of the Royal Statistical Society Series A*, **98**, 39–54.
- Giuliano, Vincent E. (1965a). The interpretation of word associations. In M. E. Stevens, V. E. Giuliano, and L. B. Heilprin (eds.), *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, volume 269 of *National Bureau of Standards Miscellaneous Publication*, pages 25–32, Washington, DC.

- Giuliano, Vincent E. (1965b). Postscript: A personal reaction to reading the conference manuscripts. In M. E. Stevens, V. E. Giuliano, and L. B. Heilprin (eds.), *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, volume 269 of *National Bureau of Standards Miscellaneous Publication*, pages 259–260, Washington, DC.
- Goldberg, David (1991). What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, **23**(1), 5–48.
- Goldman, Jean-Philippe; Nerima, Luka; Wehrli, Eric (2001). Collocation extraction using a syntactic parser. In *Proceedings of the ACL Workshop on Collocations*, pages 61–66, Toulouse, France.
- Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, **40**(3/4), 237–264.
- Good, I. J.; Gover, T. N.; Mitchell, G. J. (1970). Exact distributions for X^2 and for the likelihood-ratio statistic for the equiprobable multinomial distribution. *Journal of the American Statistical Association*, **65**, 267–283.
- Greenbaum, Sidney (1970). *Verb-Intensifier Collocations in English. An experimental approach*, volume 86 of *Janua linguarum. Series minor*. Den Haag, Paris.
- Grossmann, Francis and Tutin, Agnès (2003). Quelques pistes pour le traitement des collocations. In F. Grossmann and A. Tutin (eds.), *Les Collocations: analyse et traitement*, pages 5–21. De Werelt, Amsterdam.
- Ha, Le Quan; Sicilia-Garcia, E. I.; Ming, Ji; Smith, F. J. (2002). Extension of Zipf's law to words and phrases. In *Proceedings of COLING 2002*, Taipei, Taiwan.
- Haberman, Shelby J. (1988). A warning on the use of chi-squared statistics with frequency tables with small expected cell counts. *Journal of the American Statistical Association*, **83**, 555–560.
- Hausmann, Franz Josef (1989). Le dictionnaire de collocations. In *Wörterbücher, Dictionaries, Dictionnaires. Ein internationales Handbuch*, pages 1010–1019. de Gruyter, Berlin.
- Hausmann, Franz Josef (2004). Was sind eigentlich Kollokationen? In K. Steyer (ed.), *Wortverbindungen – mehr oder weniger fest*, Jahrbuch des Instituts für Deutsche Sprache 2003, pages 309–334. de Gruyter, Berlin.
- Heaps, H. S. (1978). *Information Retrieval – Computational and Theoretical Aspects*. Academic Press.
- Heid, Ulrich (2004). On the presentation of collocations in monolingual dictionaries. In *Proceedings of the 11th Euralex International Congress*, pages 729–738, Lorient, France.

- Heid, Ulrich; Evert, Stefan; Docherty, Vincent; Worsch, Wolfgang; Wermke, Matthias (2000). A data collection for semi-automatic corpus-based updating of dictionaries. In U. Heid, S. Evert, E. Lehmann, and C. Rohrer (eds.), *Proceedings of the 9th EURALEX International Congress*, pages 183 – 195.
- Herdan, Gustav (1964). *Quantitative Linguistics*. Butterworths, London.
- Heyer, Gerhard; Läuter, Martin; Quasthoff, Uwe; Wittig, Thomas; Wolff, Christian (2001). Learning relations using collocations. In *Proceedings of the IJCAI Workshop on Ontology Learning*, pages 19–24, Seattle, WA.
- Hindle, Donald and Rooth, Mats (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, **19**(1), 103–120.
- Hisamitsu, Toru and Niwa, Yoshiki (2001). Extracting useful terms from parenthetical expressions by combining simple rules and statistical measures. In D. Bouri-gault, C. Jacquemin, and M.-C. L'Homme (eds.), *Recent Advances in Computational Terminology*, chapter 10, pages 209–224. John Benjamins, Amsterdam.
- Holgate, P. (1969). Species frequency distributions. *Biometrika*, **56**(3), 651–660.
- Hollander, Myles and Wolfe, Douglas A. (1999). *Nonparametric Statistical Methods*. Wiley, New York, 2nd edition.
- Johnson, Mark (2001). Trading recall for precision with confidence sets. Unpublished technical report.
- Justeson, John S. and Katz, Slava (1995a). Technical terminology: Some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, **1**, 9–27.
- Justeson, John S. and Katz, Slava M. (1991). Co-occurrences of antonymous adjectives and their contexts. *Computational Linguistics*, **17**(1), 1–19.
- Justeson, John S. and Katz, Slava M. (1995b). Principled disambiguation: Discriminating adjective senses with modified nouns. *Computational Linguistics*, **21**(1), 1–27.
- Kaalep, Heiki-Jaan and Muischnek, Kadri (2003). Inconsistent selectional criteria in semi-automatic multi-word unit extraction. In *Proceedings of the 7th Conference on Computational Lexicography and Text Research (COMPLEX 2003)*, pages 27–36, Budapest, Hungary.
- Kageura, Kyo and Umino, Bin (1996). Methods of automatic term recognition. *Terminology*, **3**(2), 259–289.
- Kahane, Sylvain and Polguère, Alain (2001). Formal foundation of lexical functions. In *Proceedings of the ACL Workshop on Collocations*, pages 8–15, Toulouse, France.
- Katz, Slava M. (1996). Distribution of content words and phrases in text and language modelling. *Natural Language Engineering*, **2**(2), 15–59.

- Keller, Frank and Lapata, Mirella (2003). Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, **29**(3), 459–484.
- Kermes, Hannah (2003). *Off-line (and On-line) Text Analysis for Computational Lexicography*. Ph.D. thesis, IMS, University of Stuttgart. Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS), volume 9, number 3.
- Kermes, Hannah and Heid, Ulrich (2003). Using chunked corpora for the acquisition of collocations and idiomatic expressions. In *Proceedings of the 7th Conference on Computational Lexicography and Text Research (COMPLEX 2003)*, pages 37–46, Budapest, Hungary.
- Khmaladze, E. V. (1987). The statistical analysis of large number of rare events. Technical Report MS-R8804, Department of Mathematical Statistics, CWI, Amsterdam, Netherlands.
- Kilgarriff, Adam (2001). Comparing corpora. *International Journal of Corpus Linguistics*, **6**(1), 1–37.
- Kiss, G. R.; Armstrong, C.; Milroy, R.; Piper, J. (1973). An associative thesaurus of English and its computer analysis. In A. Aitken, R. Beiley, and N. Hamilton-Smith (eds.), *The Computer and Literary Studies*. Edinburgh University Press, Edinburgh.
- Kiss, Tibor and Strunk, Jan (2002a). Scaled log likelihood ratios for the detection of abbreviations in text corpora. In T. Shu-Chuan (ed.), *Proceedings of COLING 2002*, pages 1228–1232, Taipei, Taiwan.
- Kiss, Tibor and Strunk, Jan (2002b). Viewing sentence boundary detection as collocation identification. In S. Busemann (ed.), *Tagungsband der 6. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2002)*, pages 75–82, Saarbrücken, Germany. DFKI.
- Kita, Kenji; Kato, Yasuhiko; Omoto, Takashi; Yano, Yoneo (1994). A comparative study of automatic extraction of collocations from corpora: Mutual information vs. cost criteria. *Journal of Natural Language Processing*, **1**(1), 21–33.
- Krenn, Brigitte (2000). *The Usual Suspects: Data-Oriented Models for the Identification and Representation of Lexical Collocations*, volume 7 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. DFKI & Universität des Saarlandes, Saarbrücken, Germany.
- Krenn, Brigitte and Evert, Stefan (2001). Can we do better than frequency? A case study on extracting PP-verb collocations. In *Proceedings of the ACL Workshop on Collocations*, pages 39–46, Toulouse, France.
- Krenn, Brigitte; Evert, Stefan; Zinsmeister, Heike (2004). Determining intercoder agreement for a collocation identification task. In *Proceedings of KONVENS 2004*, Vienna, Austria.

- Kuhns, J. L. (1965). The continuum of coefficients of association. In M. E. Stevens, V. E. Giuliano, and L. B. Heilprin (eds.), *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, volume 269 of *National Bureau of Standards Miscellaneous Publication*, pages 33–39, Washington, DC.
- Landauer, Thomas K. and Dumais, Susan T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, **104**(2), 211–240.
- Lapata, Maria; McDonald, Scott; Keller, Frank (1999). Determinants of adjective-noun plausibility. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999)*, pages 30–36, Bergen, Norway.
- Läuter, Martin and Quasthoff, Uwe (1999). Kollokationen und semantisches Clustering. In *11. Jahrestagung der GLDV*.
- Lehmann, Erich Leo (1991). *Testing Statistical Hypotheses*. Wadsworth, 2nd edition.
- Lehr, Andrea (1996). *Kollokationen und maschinenlesbare Korpora*, volume 168 of *Germanistische Linguistik*. Niemeyer, Tübingen.
- Lemnitzer, Lothar (1998). Komplexe lexikalische Einheiten in Text und Lexikon. In G. Heyer and C. Wolff (eds.), *Linguistik und neue Medien*, pages 85–92. DUV, Wiesbaden.
- Lezius, Wolfgang (1999). Automatische Extrahierung idiomatischer Bigramme aus Textkorpora. In *Tagungsband des 34. Linguistischen Kolloquiums*, Germersheim, Germany.
- Lezius, Wolfgang; Dipper, Stefanie; Fitschen, Arne (2000). IMSLex – representing morphological and syntactical information in a relational database. In U. Heid, S. Evert, E. Lehmann, and C. Rohrer (eds.), *Proceedings of the 9th EURALEX International Congress*, pages 133–139, Stuttgart, Germany.
- Li, Wentian (1992). Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, **38**(6), 1842–1845.
- Liddell, Douglas (1976). Practical tests of 2×2 contingency tables. *The Statistician*, **25**(4), 295–304.
- Lin, Dekang (1998). Extracting collocations from text corpora. In *Proceedings of the First Workshop on Computational Terminology*, pages 57–63, Montreal, Canada.
- Magerman, David M. and Marcus, Mitchell P. (1990). Parsing a natural language using mutual information statistics. In *8th National Conference on Artificial Intelligence (AAAI 90)*, pages 984–989, Boston, MA.
- Mandelbrot, Benoit (1962). On the theory of word frequencies and on related Markovian models of discourse. In R. Jakobson (ed.), *Structure of Language and its Mathematical Aspects*, pages 190–219. American Mathematical Society, Providence, RI.

- Manning, Christopher D. and Schütze, Hinrich (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.
- McEnery, Tony and Wilson, Andrew (2001). *Corpus Linguistics*. Edinburgh University Press, 2nd edition.
- Meľčuk, Igor A. (2003). Collocations: définition, rôle et utilité. In F. Grossmann and A. Tutin (eds.), *Les Collocations: analyse et traitement*, pages 23–31. De Werelt, Amsterdam.
- Miller, George A. (1957). Some effects of intermittent silence. *The American Journal of Psychology*, **52**, 311–314.
- Miller, George A. (1990). WordNet: An on-line lexical database. *International Journal of Lexicography*, **3**(4).
- Monaghan, James (1979). *The Neo-Firthian Tradition and its Contribution to General Linguistics*, volume 73 of *Linguistische Arbeiten*. Niemeyer, Tübingen.
- Motulsky, Harvey (1995). *Intuitive Biostatistics*. Oxford University Press, New York.
- Nerima, Luka; Seretan, Violeta; Wehrli, Eric (2003). Creating a multilingual collocation dictionary from large text corpora. In *Companion Volume to the Proceedings of the 10th Conference of The European Chapter of the Association for Computational Linguistics*, pages 131–134.
- Pantel, Patrick and Lin, Dekang (2002). Discovering word senses from text. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton, Canada.
- Pearce, Darren (2002). A comparative evaluation of collocation extraction techniques. In *Third International Conference on Language Resources and Evaluation (LREC)*, pages 1530–1536, Las Palmas, Spain.
- Pearsall, Judy and Hanks, Patrick (eds.) (1998). *The New Oxford Dictionary of English*. Oxford University Press, Oxford.
- Pedersen, Ted (1996). Fishing for exactness. In *Proceedings of the South-Central SAS Users Group Conference*, Austin, TX.
- Pedersen, Ted (2001). A decision tree of bigrams is an accurate predictor of word sense. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01)*, Pittsburgh, PA.
- Pedersen, Ted and Bruce, Rebecca (1996). What to infer from a description. Technical Report 96-CSE-04, Southern Methodist University, Dallas, TX.
- Porkess, Roger (1991). *The HarperCollins Dictionary of Statistics*. HarperCollins, New York.
- Powers, David M. W. (1998). Applications and explanations of Zipf's law. In D. M. W. Powers (ed.), *Proceedings of New Methods in Language Processing and Computational Natural Language Learning*, pages 151–160. ACL.

- Quasthoff, Uwe (1998). Deutscher Wortschatz im Internet. *LDV-Forum*, 15(2), 4–23.
- Quasthoff, Uwe and Wolff, Christian (2002). The Poisson collocation measure and its application. In *Workshop on Computational Approaches to Collocations*, Vienna, Austria.
- R Development Core Team (2003). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3. See also <http://www.r-project.org/>.
- Rapp, Reinhard (1999). Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Maryland.
- Rapp, Reinhard (2002). The computation of word associations: Comparing syntagmatic and paradigmatic approaches. In *Proceedings of COLING 2002*, Taipei, Taiwan.
- Rapp, Reinhard (2003). Discovering the meanings of an ambiguous word by searching for sense descriptors with complementary context patterns. In *Proceedings of the 5èmes Rencontres Terminologie et Intelligence Artificielle (TIA-2003)*, Strasbourg, France.
- Rapp, Reinhard (2004). Utilizing the one-sense-per-discourse constraint for fully unsupervised word sense induction and disambiguation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 951–954, Lisbon, Portugal.
- Resnik, Philip (1997). Selectional preferences and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, Washington, D.C.
- Rosenfeld, Ronald (1996). A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10, 187–228.
- Rouault, Alain (1978). Lois de Zipf et sources markoviennes. *Annales de l'Institut H. Poincaré (B)*, 14, 169–188.
- Schiffman, Barry; Mani, Inderjeet; Concepcion, Kristian J. (2001). Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.
- Schmid, Helmut (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, pages 44–49.
- Schone, Patrick and Jurafsky, Daniel (2001). Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108, Pittsburgh, PA.

- Sichel, H. S. (1975). On a distribution law for word frequencies. *Journal of the American Statistical Association*, **70**, 542–547.
- Siegel, Sidney (1956). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill Kogakusha, Tokyo.
- Sinclair, John (1965). When is a poem like a sunset? *A Review of English Literature*, **6**(2), 76–91.
- Sinclair, John (1991). *Corpus, Concordance, Collocation*. Oxford University Press, Oxford.
- Sinclair, John; Jones, Susan; Daley, Robert; Krishnamurthy, Ramesh (2004). *English Collocation Studies: The OSTI Report*. Continuum Books, London and New York. Originally written in 1970 (unpublished).
- Skut, Wojciech; Brants, Thorsten; Krenn, Brigitte; Uszkoreit, Hans (1998). A linguistically interpreted corpus of German newspaper texts. In *Proceedings of the ESLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany. See also <http://www.coli.uni-sb.de/sfb378/negra-corpus/>.
- Smadja, Frank (1991). From n-grams to collocations: An evaluation of Xtract. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 279–284, Berkeley, CA.
- Smadja, Frank (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, **19**(1), 143–177.
- Smadja, Frank; McKeown, Kathleen R.; Hatzivassiloglou, Vasileios (1996). Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, **22**(1), 1–38.
- Stevens, Mary Elizabeth; Giuliano, Vincent E.; Heilprin, Laurence B. (eds.) (1965). *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation, Washington 1964*, volume 269 of *National Bureau of Standards Miscellaneous Publication*.
- Stone, Matthew and Doran, Christine (1996). Paying heed to collocations. In *Proceedings of the International Language Generation Workshop (INLG '96)*, pages 91–100, Herstmonceux Castle, Sussex, UK.
- Stubbs, Michael (1995). Collocations and semantic profiles: On the cause of the trouble with quantitative studies. *Functions of Language*, **1**, 23–55.
- Tamir, Raz and Rapp, Reinhard (2003). Mining the web to discover the meanings of an ambiguous word. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 645–648, Melbourne, FL.
- Tan, Pang-Ning; Kumar, Vipin; Srivastava, Jaideep (2002). Selecting the right interestingness measure for association patterns. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–41, Edmonton, Canada.

- Terra, Egidio and Clarke, Charles L. A. (2003). Frequency estimates for statistical word similarity measures. In *Proceedings of HLT-NAACL 2003*, pages 244–251, Edmonton, Alberta.
- Terra, Egidio and Clarke, Charles L. A. (2004). Fast computation of lexical affinity models. In *Proceedings of COLING 2004*, Geneva, Switzerland.
- Turney, Peter D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In L. De Raedt and P. Flach (eds.), *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, pages 491–502, Freiburg, Germany.
- Upton, G. J. G. (1982). A comparison of alternative tests for the 2×2 comparative trial. *Journal of the Royal Statistical Society, Series A*, **145**, 86–105.
- Venables, W. N. and Ripley, B. D. (1999). *Modern Applied Statistics with S-PLUS*. Springer, New York, 3rd edition.
- Vivaldi, Jorge and Rodríguez, Horacio (2001). Improving term extraction by combining different techniques. *Terminology*, **7**(1), 31–48.
- Volk, Martin (2002). Combining unsupervised and supervised methods for pp attachment disambiguation. In *Proceedings of COLING 2002*, Taipei, Taiwan.
- Wall, Larry; Christiansen, Tom; Schwartz, Randal L. (1996). *Programming Perl*. O'Reilly, 2nd edition.
- Weeber, Marc; Vos, Rein; Baayen, R. Harald (2000). Extracting the lowest-frequency words: Pitfalls and possibilities. *Computational Linguistics*, **26**(3), 301–317.
- Weisstein, Eric W. (1999). *Eric Weisstein's World of Mathematics*. Wolfram Inc. An on-line encyclopedia. <http://mathworld.wolfram.com/>.
- Wiebe, Janyce; Wilson, Theresa; Bell, Matthew (2001). Identifying collocations for recognizing opinions. In *Proceedings of the ACL Workshop on Collocations*, pages 24–31, Toulouse, France.
- Wilks, S. S. (1935). The likelihood test of independence in contingency tables. *The Annals of Mathematical Statistics*, **6**(4), 190–196.
- Williams, Geoffrey (2003). Les collocations et l'école contextualiste britannique. In F. Grossmann and A. Tutin (eds.), *Les Collocations: analyse et traitement*, pages 33–44. De Werelt, Amsterdam.
- Yates, F. (1934). Contingency tables involving small numbers and the χ^2 test. *Supplement to the Journal of the Royal Statistical Society*, **1**, 217–235.
- Yates, F. (1984). Tests of significance for 2×2 contingency tables. *Journal of the Royal Statistical Society, Series A*, **147**(3), 426–463.
- Yeh, Alexander (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany.

- Yoon, Juntae; Choi, Key-Sun; Song, Mansuk (2001). A corpus-based approach for korean nominal compound analysis based on linguistic and statistical information. *Natural Language Engineering*, 7(3), 251–270.
- Zinsmeister, Heike and Heid, Ulrich (2004). Collocations of complex nouns: Evidence for lexicalisation. In *Proceedings of the 11th Euralex International Congress*, Lorient, France.
- Zipf, George Kingsley (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA.