

Institute for Visualization and Interactive Systems

Masterarbeit Nr. 25

# **Developing and Evaluating Input Methods for Uncertain Data**

Marius Kleiner

**Course of Study:** Computer Science

**Examiner:** Prof. Dr. Albrecht Schmidt

**Supervisor:** Dipl.-Inf. Miriam Greis,  
Jun.-Prof. Niels Henze

**Commenced:** January 15, 2015

**Completed:** July 17, 2015

**CR-Classification:** H.5.2



# Acknowledgements

At this point I would like to gracefully thank everyone who helped me with the creation and elaboration of this master thesis and who supported me for the time I worked on this thesis. This includes the participants of the online survey and the user study. A special thank goes to my family, housemates, and friends who supported me with patience, motivation and trust. Especially my sister deserves my exceptional gratitude. I appreciate the understanding of my friends that I could not show up often while writing this thesis and that they, nevertheless, did not forget about me in this period of time. Further thanks goes to Dipl.-Inf. Miriam Greis, Jun.-Prof. Niels Henze, and Prof. Dr. Albrecht Schmidt for offering me the possibility to write this thesis in the context of human computer interaction and for the provision of great help in various aspects. In addition to that I want to thank many others, who are not named here, for their help, advices, suggestions, and the (sometimes necessary) distraction.

## Kurzfassung

Die Welt, in der wir leben, ist voller Unsicherheiten, welche sich beispielsweise bei Wettervorhersagen bemerkbar machen. Solche Vorhersagen werden mit der Hilfe von Simulationen getroffen. Simulationen liefern nicht zwingend nur unsichere Ausgaben, sondern können auch die Eingabe von Daten mit Unsicherheiten fordern. Die Zahl der Simulationstools, die auch von Nicht-Wissenschaftlern bedient werden können, nimmt stetig zu. Daher bedarf es an Eingabemethoden die sowohl von Experten, als auch von Laien genutzt werden können. Diese Masterarbeit befasst sich mit der Entwicklung und Evaluation solcher Eingabemethoden. Wir stellen eine Auswahl von in diesem Kontext entwickelten Eingabemethoden vor und erörtern eine Vorauswahl und die Weiterentwicklung der vielversprechendsten Prototypen. Diese Eingabemethoden wurden zunächst mit Hilfe einer Online Survey evaluiert und in einem zweiten Schritt wurde eine User Study durchgeführt um, aufbauend auf der Survey, detailliertere Ergebnisse zu erhalten. Die Auswertung der Ergebnisse und die daraus resultierenden Schlüsse geben einen Einblick in mögliche Verbesserungen oder Veränderungen und in welchen Situationen die Eingabemethoden jeweils angewendet werden können. Darüber hinaus stellen wir ein web-basiertes Simulationstool, das während dieser Masterarbeit entstand und in welches die entwickelten Eingabemethoden integriert sind, vor.

## Abstract

The world, we are living in, is full of uncertainties, which we can see for an instance in weather forecasts. Such forecasts are made with the help of simulations. Simulations do not only sometimes produce uncertain outputs, but may also require the input of uncertain data. Since there are more and more simulation tools which are supposed to be used by non-scientists, suitable input methods for uncertain data, which can be used by both, experts and lay persons, are required. This master thesis deals with the development and evaluation of such input methods. We present a selection of developed input methods and discuss a preselection and the enhancements of the most promising prototypes. These input methods were tested and evaluated in an online survey first and in a second step we conducted a user study to receive more detailed results. These results are discussed and a conclusion and outlook gives insights on how these input methods can be further improved or altered and in which situations they can be applied. A web-based simulation tool, that includes the developed input methods, has been created in the context of this thesis and is also briefly described.



# Contents

1. Introduction	1
1.1. Scope of this Thesis . . . . .	2
1.2. Outline . . . . .	2
2. Related Work	5
2.1. Background . . . . .	5
2.2. Uncertainty . . . . .	12
2.3. Uncertainty and Visualization . . . . .	16
2.4. Uncertainty and Input Methods . . . . .	18
2.5. Input Methods . . . . .	19
2.6. Visualization and Input Methods . . . . .	20
2.7. Simulation Tools . . . . .	21
2.8. Summary and Discussion . . . . .	22
3. Inspecting Uncertainty and Input Methods	23
3.1. Humans Dealing with Uncertainty . . . . .	23
3.2. Uncertainty Occurrences when Dealing with Simulations . . . . .	25
3.3. Uncertainties of Interest for this Thesis . . . . .	29
3.4. Classification of Uncertainty Representations . . . . .	30
3.5. Input Method Metrics . . . . .	40
3.6. Summary and Discussion . . . . .	42
4. Input Method Prototypes	45
4.1. Assembly and Classification of Input Methods . . . . .	45
4.2. Expressiveness vs. Usability Estimation . . . . .	64
4.3. Discussion and Preselection . . . . .	66
4.4. Upgrading most promising Input Methods . . . . .	67
4.5. Input Method Implementation . . . . .	71
4.6. Summary and Discussion . . . . .	71
5. Web-Based Simulation Tool	73
5.1. Django and Bootstrap . . . . .	73
5.2. Concept and Development . . . . .	74
5.3. Example Simulation . . . . .	76
5.4. Summary and Discussion . . . . .	76

6. Online Survey	81
6.1. Survey Composition . . . . .	81
6.2. Questionnaire . . . . .	82
6.3. Participants . . . . .	84
6.4. Results . . . . .	86
6.5. Discussion . . . . .	98
6.6. Summary . . . . .	100
7. User Study	101
7.1. Study Composition . . . . .	101
7.2. Tasks and Questionnaire . . . . .	103
7.3. Participants . . . . .	104
7.4. Results . . . . .	104
7.5. Discussion . . . . .	108
7.6. Summary . . . . .	110
8. Further Steps on the Basis of Survey and Study Results	113
8.1. Further Enhancements and Alternatives . . . . .	113
8.2. Summary and Discussion . . . . .	116
9. Summary and Conclusions	117
9.1. Summary . . . . .	117
9.2. Conclusions . . . . .	117
9.3. Suitability for different Scenarios . . . . .	118
9.4. Outlook . . . . .	119
A. Appendix	121
Bibliography	135



# List of Figures

---

2.1. Overview of three main subject areas of this thesis . . . . .	6
2.2. Classes of the Uncertainty Ontology [LL08] . . . . .	7
2.3. <b>Left:</b> Normal Distribution Example[Ind08], <b>Right:</b> Fair dice probability distribution[Ole07] . . . . .	10
2.4. Quality in use after Nigel Bevan [Bev08] . . . . .	20
2.5. <b>Left:</b> NetLogo’s user interface, with model Diffusion Limited Aggregation[TW04], <b>Right:</b> Forio Epicenter example: Investment Portfolio Simulation[For15] . . . . .	21
3.1. Educational Attainment of the Population in Germany from 2013 (Highest level of general education completed) from Federal Statistical Office of Germany[Fed13] (MS Office 2013) . . . . .	24
3.2. Educational Attainment of the Population in Germany from 2013 (Vocational qualification attained) from Federal Statistical Office of Germany[Fed13] (MS Office 2013) . . . . .	24
3.3. Effectiveness Example, <b>Left:</b> Low Expressiveness / Estimate or Guess, <b>Right:</b> Higher Expressiveness / Probability Distribution . . . . .	27
3.4. Uncertainties in Simulations Overview . . . . .	28
3.5. Accuracy and Precision in a Probability Density Function [Pek07] . . . . .	33
3.6. Example Input Methods <b>Left:</b> Expressiveness Level 3, <b>Right:</b> Expressiveness Level 4 . . . . .	37
3.7. Different Forms of Distributions <b>Left:</b> Skew [Rod08], <b>Right:</b> High and Low Kurtosis [Chr09] . . . . .	38
3.8. Expressiveness vs. Usability Estimate before Survey and Study . . . . .	39
3.9. Factors contributing to System Usability and User Experience [Bev08] . . . . .	41
4.1. <i>Simple Slider</i> (Percentage) Example . . . . .	45
4.2. <i>Ratio Selector</i> Example . . . . .	46
4.3. <i>Extended Ratio Selector</i> Example . . . . .	47
4.4. <i>Ratio Range Selector</i> Example . . . . .	48
4.5. <i>2D Scaler</i> Examples . . . . .	49
4.6. <i>2D Range Scaler</i> Example . . . . .	50
4.7. <i>2D Range Best Estimate Scaler</i> Example . . . . .	51
4.8. <i>Ratio Drag And Drop</i> Examples . . . . .	52
4.9. <i>Ratio Range Drag And Drop</i> Example . . . . .	53

4.10. <i>Ratio Range Best Estimate Drag And Drop</i> Example . . . . .	54
4.11. <i>Term Selector</i> Example . . . . .	55
4.12. <i>Term Range Selector</i> Example . . . . .	56
4.13. <i>Quality Thumb Rotator</i> Example . . . . .	57
4.14. <i>Quality Thumb Range Rotator</i> Example . . . . .	58
4.15. <i>Simple Location Range Picker</i> Example . . . . .	59
4.16. <i>Location Range Picker</i> Example . . . . .	60
4.17. <i>Location Likely Range Picker</i> Example . . . . .	61
4.18. <i>Bar Slider</i> Example . . . . .	61
4.19. <i>Fixed Range Bar Slider</i> Example . . . . .	62
4.20. <i>Dynamic Range Bar Slider</i> Example . . . . .	63
4.21. <i>Dynamic Range Bar Best Estimate Slider</i> Example . . . . .	63
4.22. Expressiveness vs. Usability Estimation for presented Input Methods after brief Testing . . . . .	66
4.23. Overhauled Input Methods . . . . .	68
4.24. Colour Gradient Transition Examples . . . . .	70
4.25. <i>Dynamic Range Bar Slider</i> with predefined Distribution Types (from left to right: Triangular Distribution, Normal Distribution, Wigner Semicircle Distribution, Laplace Distribution) . . . . .	71
5.1. Simulation Tool Client Server Communication Overview . . . . .	75
5.2. <i>Hitting the Bull's-eye</i> Test Simulation Front-End Screenshot . . . . .	79
6.1. Participants' average Familiarity with different Web-Based Input Methods as Likert Scale Rating with Standard Error Bars . . . . .	85
6.2. Amount of Participants' with average prior Knowledge about Statistics, Probabi- lity Theory and Stochastics in General . . . . .	86
6.3. Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Effectivity for the five preselected Input Methods and an applied Bonferroni correction, resulting in a Significance Level of $p < 0,005$ . Statistically significant Values are highlighted with green Colour. . . . .	87
6.4. Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Efficiency for the five preselected Input Methods and an applied Bonferroni correction, resulting in a Significance Level of $p < 0,005$ . Statistically significant Values are highlighted with green Colour. . . . .	88
6.5. Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Confidence for the five preselected Input Methods and an applied Bonferroni correction, resulting in a Significance Level of $p < 0,005$ . Statistically significant Values are highlighted with green Colour. . . . .	89

6.6. Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Satisfaction for the five preselected Input Methods and an applied Bonferroni correction, resulting in a Significance Level of $p < 0,005$ . Statistically significant Values are highlighted with green Colour. . . . .	90
6.7. Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Learnability for the five preselected Input Methods and an applied Bonferroni correction, resulting in a Significance Level of $p < 0,005$ . Statistically significant Values are highlighted with green Colour. . . . .	91
6.8. Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the <i>Bar Slider</i> Input Method . . . . .	92
6.9. Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the <i>Fixed Range Bar Slider</i> Input Method . . . . .	92
6.10. Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the <i>Dynamic Range Bar Slider</i> Input Method . . . . .	93
6.11. Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the <i>Dynamic Range Bar Best Estimate Slider</i> Input Method . . . . .	93
6.12. Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the <i>Advanced Dynamic Range Best Estimate Slider</i> Input Method . .	94
6.13. Input Methods Likability Ranking . . . . .	94
6.14. Input Methods Usefulness Ranking . . . . .	95
6.15. Average Likert Scale Values with Standard Error Bars for the Measures Understandability, Likability and Usefulness for the three supportive graphical elements . . . . .	96
6.16. Expressiveness vs. Usability Estimation for preselected Input Methods after Online Survey, incorporating initial Testings and Survey Results. . . . .	99
7.1. Average System Usability Scale Results with Standard Error Bars for the the five preselected Input Methods . . . . .	105
7.2. Average Input Time and Effectivity Results for the five preselected Input Methods	106
7.3. Average Help Reading Time and Confidence for five preselected Input Methods	107
7.4. Average Help Reading Time and Confidence for the three Task Blocks of the User Study . . . . .	107
7.5. Average total Clicks per Input Element for the five preselected Input Methods with Standard Error Bars . . . . .	108
7.6. Expressiveness vs. Usability Estimation for preselected Input Methods after Online Survey and User Study. Usability Values are a Combination of the SUS Results, the Online Survey Results and the initial Testings. . . . .	110
7.7. Updated Expressiveness vs. Usability Pattern Estimate, based on the Results from the Online Survey and the User Study. . . . .	111

8.1. <i>Dynamic Range Bar Best Estimate Slider</i> with futher Enhancements based on the Results of the Online Survey and User Study . . . . .	114
8.2. Precise Slider Improvement by Vertical Dragging and alternative Approach for <i>Advanced Dynamic Range Best Estimate Slider</i> to modify Kurtosis . . . . .	115
8.3. Slider Node moving its Neighbour Node when Dragging over its Borders . . . . .	115
A.1. Screenshot from User Study, Task 2, <i>Dynamic Range Bar Best Estimate Slider</i> . . . . .	121

## List of Tables

---

3.1. Overview of Classification Techniques for Input Methods dealing with Uncertainty(Randomness) . . . . .	37
4.1. Classification of <i>Simple Slider</i> Input Method . . . . .	46
4.2. Classification of <i>Ratio Selector</i> Input Method . . . . .	47
4.3. Classification of <i>Extended Ratio Selector</i> Input Method . . . . .	47
4.4. Classification of <i>Ratio Range Selector</i> Input Method . . . . .	48
4.5. Classification of <i>2D Scaler</i> Input Method . . . . .	49
4.6. Classification of <i>2D Range Scaler</i> Input Method . . . . .	50
4.7. Classification of <i>2D Range Best Estimate Scaler</i> Input Method . . . . .	51
4.8. Classification of <i>Ratio Drag And Drop</i> Input Method . . . . .	52
4.9. Classification of <i>Ratio Range Drag And Drop</i> Input Method . . . . .	53
4.10. Classification of <i>Ratio Range Best Estimate Drag And Drop</i> Input Method . . . . .	54
4.11. Classification of <i>Term Selector</i> Input Method . . . . .	55
4.12. Classification of <i>Term Range Selector</i> Input Method . . . . .	56
4.13. Classification of <i>Quality Thumb Rotator</i> Input Method . . . . .	57
4.14. Classification of <i>Quality Thumb Range Rotator</i> Input Method . . . . .	58
4.15. Classification of <i>Simple Location Range Picker</i> Input Method . . . . .	59
4.16. Classification of <i>Location Range Picker</i> Input Method . . . . .	60
4.17. Classification of <i>Location Likely Range Picker</i> Input Method . . . . .	61
4.18. Classification of <i>Bar Slider</i> Input Method . . . . .	62
4.19. Classification of <i>Fixed Range Bar Slider</i> Input Method . . . . .	62
4.20. Classification of <i>Dynamic Range Bar Slider</i> Input Method . . . . .	63
4.21. Classification of <i>Dynamic Range Bar Best Estimate Slider</i> Input Method . . . . .	64
4.22. Input Methods Expressiveness vs. Usability Estimation . . . . .	65
4.23. Classification of <i>Advanced Dynamic Range Best Estimate Slider</i> Input Method . . . . .	69
6.1. Friedman Test Results for the five Usability Measures in the Online Survey . . . . .	86

# List of Listings

---

5.1. Django Template Tag Example (for-loop, dictionary and filters) . . . . .	74
5.2. <i>Hitting the Bull's-eye</i> Test Simulation Python Code – Part 1 . . . . .	77
5.3. <i>Hitting the Bull's-eye</i> Test Simulation Python Code – Part 2 . . . . .	78
A.1. Test Script Django View Function . . . . .	122
A.2. Python Function to calculate Square . . . . .	123
A.3. Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 1 . . . . .	124
A.4. Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 2 . . . . .	125
A.5. Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 3 . . . . .	126
A.6. Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 4 . . . . .	127
A.7. Python Simulation Register File . . . . .	128
A.8. Client-Side of Test Script (JavaScript) . . . . .	129
A.9. HTML Snippet for the Inclusion of Input Method Templates with the Help of Django Template Tags . . . . .	129
A.10. Javascript Simulation Run Snippet . . . . .	130
A.11. Example for the Javascript Part of the Input Method Implementation (Bar Slider) Part 1 . . . . .	131
A.12. Example for the Javascript Part of the Input Method Implementation (Bar Slider) Part 2 . . . . .	132
A.13. Example for the Javascript Part of the Input Method Implementation (Bar Slider) Part 3 . . . . .	133
A.14. Example for the Javascript Part of the Input Method Implementation (Bar Slider) Part 4 . . . . .	134



# 1. Introduction

Nowadays, people encounter uncertainty very often, in many cases in the form of probabilities, without even being aware of that. Uncertainty is all around us and we can encounter it in very different ways. If one can not decide if a colour is more green or blue we talk about vagueness, which is one form of uncertainty. There are many more examples for uncertainty occurrences in everyday life, such as weather forecasts, casino games and medical decisions, such as the chance of a cancer screening test to show a false positive result. Frequently, people have problems with the interpretation of uncertainty representations. One example for this is that the term „a 30% chance of rain“ is interpreted differently, namely as „it will rain on 30% of the surface“, as „it will rain 30% of the time“ and as „in 3 of 10 cases it will rain on such days“. The interpretation becomes even harder when data become more complex, for instance if the data is presented as a probability distribution. Working with uncertainty can be a very hard task and quite confusing. There are also numerous uncertainty occurrences when dealing with simulations. The output often contains uncertainties, in many cases represented as probability distributions, due to the usage of approaches to run multiple parallel simulations. Simulations can help to solve problems where real tests do not come into question for several reasons, such as high costs and risks for humans and the environment. With the increasing processing power of home computers and even mobile devices simple simulations became accessible for everyone. With the advent of cloud computing, even resource consuming simulations can be conducted from a small computing device, while the computing power is provided by the cloud. This makes simulations accessible for a wide range of people. Often not only the output, but also the input contains uncertainty. These data may contain epistemic uncertainty, which means uncertainty due to a lack of knowledge, and aleatoric uncertainty, which describes the intrinsic uncertainty that lies in the nature of the data. Simulations can provide many useful and valuable information, not only for scientists, but also for laymen. The simulation of the development of the own fuel consumption within the next year or the determination of the ecological footprint are examples for simulations a non-scientist might want to conduct. Now that also non-experts have access to simulation tools, appropriate input methods are required to allow the input of data with epistemic and aleatoric uncertainty and which are understandable for laymen. Since people often have problems with even simple outputs for uncertain data, this puts high demands on input methods for this purpose, too. More advanced approaches than simple text and number inputs, which are enhanced with visual support and interactive elements, promise to help users to achieve the specification of the desired input values. Some of such input methods may be more flexible while others provide better results for more specialized application areas. Different contexts have put different demands

on such input methods and sometimes users want to put the focus more on the accuracy and expressiveness while in other cases it is more important to provide a very high usability. In this thesis we examine the subject of input methods for uncertain data and inspect how to satisfy both, the demand for a high usability and the need for expressiveness and accuracy, especially in the context of web-based simulation tools.

### 1.1. Scope of this Thesis

This work focuses on the development and evaluation of input methods for uncertain data. This includes the task of inspecting and outlining related work and background information. The scope of this thesis includes the design and development of prototypes for input methods for the input of uncertain data. A selection of these input methods has to be implemented for the usage in a web-based context. In addition to that a web-based simulation tool has to be developed and implemented, which integrates the input methods for uncertain data. This tool should make it easy for developers to create new simulations and leave the creation of the simulation front-end to the simulation framework. A simple sample simulation is to be developed and implemented to prove the tools functionality. An online survey is conducted for a first evaluation of preselection of the most promising input methods. From those input methods the most appropriate candidates, in general and in particular contexts, has to be determined. A user study is conducted to inspect the input methods more in detail and to confirm or disprove the findings and conclusions of the online survey. The analysis and discussion of the results completes the assignment of this thesis.

### 1.2. Outline

We first have a look at related work in **Chapter 2 – Related Work**. Since there is not much related work that deals with input methods for uncertain data we often fall back on work which deals with the sub-topics uncertainty, input methods and visualization or a combination of those. We pick up the information from **Chapter 2 – Related Work** in **Chapter 3 – Inspecting Uncertainty and Input Methods** to draw some conclusions and to make some assumptions, which we want to confirm or disprove in this thesis. In this context we discuss, amongst other topics, uncertainty occurrences when dealing with simulations and techniques for the classification of uncertainty representations. In **Chapter 4 – Input Method Prototypes** we present a selection of input methods that we developed for this thesis. We chose the selected input methods in a way that they show at least one representative for the most classification types, which we defined in *Section 3.4 – Classification of Uncertainty Representations*. In this chapter we also describe one central assumption of this thesis, namely the relationship between expressiveness and usability of input methods for uncertain data. We discuss the preselection of the most promising input methods and describe what kind of improvements we



implemented to enhance these input methods. Before evaluating the input methods, **Chapter 5 – Web-Based Simulation Tool** introduces the simulation tool which we created for this thesis and presents the creation of a simple sample simulation. For the evaluation of the input methods, presented in **Chapter 4 – Input Method Prototypes**, we first conducted an online survey, outlined in **Chapter 6 – Online Survey**. In this survey we chose to stick with the subjective measurement of the input methods usability by questionnaire. We included objective measurement techniques in the user study that we conducted as a second step of evaluation. This study is described in **Chapter 7 – User Study**. Based on the results of the online survey and the user study we discuss some possible alterations and enhancements for the tested input methods and make suggestions on their application for different users and tasks in **Chapter 8 – Further Steps on the Basis of Survey and Study Results**. The conclusions are presented in **Chapter 9 – Summary and Conclusions**, where we also provide a brief outlook for further improvements and alternatives for the input methods.



## 2. Related Work

In this chapter, we present background information, relevant papers, and other sources that are relevant for our work. The various papers have different focus areas, of which we expose not all, but the most relevant. In *Section 2.1 –Background* we show the three main subject areas in *Figure 2.1*. Each related work can be assigned to one of the main topics or even to a cut-set of two topics. Some of the relationships between the outlined content and this thesis might not be obvious at first sight, but we refer to them in following chapters, which should point out the connections.

### 2.1. Background

In this section we want to look into some background information. There are three main topics that play a major role in this thesis:

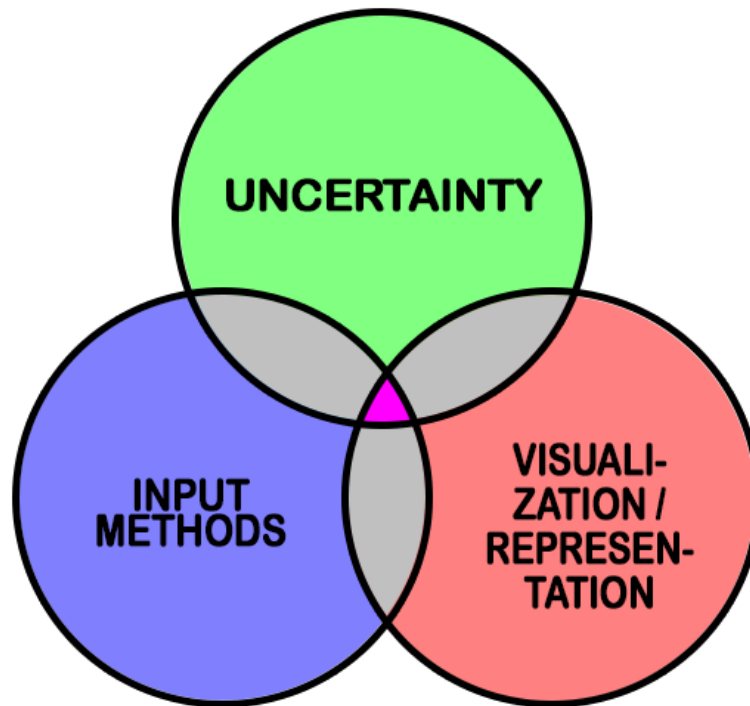
- Uncertainty
- Visualizations
- Input Method(s)

The subject of this thesis can be put at where those three main topics overlap (see *Figure 2.1*). Here we discuss some required basic information which we complement later by looking into related work in *Chapter 2 – Related Work*. Since the areas where two or more topics overlap go beyond basic knowledge they are mainly covered by related work. The collected knowledge is then worked up in *Chapter 3 – Inspecting Uncertainty and Input Methods*.

#### 2.1.1. Uncertainty

The term uncertainty can cause quite some confusion due to its various meanings and application areas. Therefore we first try to clarify what is meant by the term *uncertainty*. Alex T. Pang et al. describe uncertainty as follows:

„We define uncertainty to include statistical variations or spread, errors and differences, minimum- and maximum range values, noisy, or missing data.“ [PWL97]



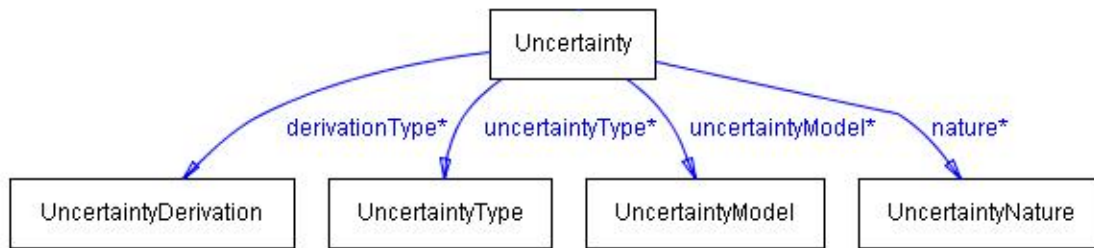
**Figure 2.1.:** Overview of three Main Subject Areas of this Thesis

He also notes that data quality has an inverse relationship with data uncertainty. Consequently uncertainty in data can be regarded as the absence of data quality. The description of Alex T. Pang et al. covers a broad range of sources of uncertainty in data. That's why we look into types and sources of uncertainty more deeply in *Section 2.1.1.1 – Uncertainty Types* and *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*.

Douglas Hubbard gives the following description:

„The lack of certainty. A state of having limited knowledge where it is impossible to exactly describe the existing state, a future outcome, or more than one possible outcome.“ [Hub10]

The *Uncertainty Reasoning for the World Wide Web Incubator Group Charter* [LL08] addresses the problem of defining the challenge of reasoning with and representing uncertain information. K. Laskey et al. developed a simple ontology which can be used to classify use cases related to uncertainty reasoning. *Figure 2.2* shows the classes of this uncertainty ontology.



**Figure 2.2.:** Classes of the Uncertainty Ontology [LL08]

Following this ontology there are four ways to categorize uncertainty: *Uncertainty Derivation*, *Uncertainty Type*, *Uncertainty Model* and *Uncertainty Nature*. In terms of *Uncertainty Derivation* there is a distinction between objective and subjective uncertainty, whereas the latter means a subjective judgement or a guess and objective uncertainty is derived in a formal way. The nature of uncertainty tells us whether the uncertainty is aleatoric or epistemic. We discuss this in *Section 2.1.1.2 – Aleatoric and Epistemic Uncertainty*. There are several mathematical theories that can be applied to uncertainty. Uncertainty can be expressed and is included in different mathematical models. The *w3.org* website mentions specific types of theories. Besides probability theory, which is the most common theory applied for modeling uncertainty, there are a number of other theories that come into question, including fuzzy sets, belief functions, random sets, rough sets and more. We take up this subject again in *Section 2.1.2 – Probability Distributions*. K. Laskey et al. also distinguish between different kinds of uncertainty with the help of five different *Uncertainty Types*. We look into that first, in *Section 2.1.1.1 – Uncertainty Types*. Note, that there are numerous other ways to distinguish between different types of uncertainty.

#### 2.1.1.1. Uncertainty Types

The *Uncertainty Types* listed by K. Laskey et al. are the following:

- Ambiguity
- Empirical
- Vagueness
- Inconsistency
- Incompleteness

## 2. Related Work

---

Ambiguity can be defined as the possibility of interpreting an expression in two or more distinct ways [Dic15b]. This type of uncertainty can occur if the context or reference classes are partly or entirely undefined. Everybody has already encountered ambiguity in natural everyday language. The *Uncertainty Reasoning for the World Wide Web Incubator Group Charter* defines the empirical *Uncertainty Type* as follows: „a sentence about a world (an event) is either satisfied or not satisfied in each world, but it is not known in which worlds it is satisfied; this can be resolved by obtaining additional information (e.g., an experiment).“ This type includes randomness which is relating to a type of circumstance or event that is described by a probability distribution [Dic15a]. Note the reference to probability distributions in this definition. We are talking about vagueness if there is no precise correspondence between some artifacts and their referents in the world. K. Laskey et al. call uncertainty to be of the inconsistency *Uncertainty Type* if it contains a contradiction and is therefore not satisfiable. Incompleteness means that some information is missing or incomplete.

### 2.1.1.2. Aleatoric and Epistemic Uncertainty

What K. Laskey et al. name as the *Uncertainty Nature* is the commonly used way to distinguish between aleatoric and epistemic uncertainty. Aleatoric uncertainty, also known as statistical uncertainty, can be described as inherent uncertainty due to probabilistic variability [CD13]. Similarly K. Laskey et al. designate this kind of uncertainty to be an inherent property of the world. An example for this is uncertainty in weather forecasts. This uncertainty is irreducible due to our inability to measure the required information in sufficient accuracy. However, this does not mean that this information does not exist. There is just no way for us to collect it. In contrast to that epistemic uncertainty, also known as systematic uncertainty, can be eliminated. We talk about epistemic uncertainty if some required information could in principle be known but in practice is not. There are several possible reasons for this lack of information such as insufficient accuracy of our methods of quantity measurement. Another possibility is that we just did not catch up on the required information. K. Chowdhary and P. Dupuis describe this in short as „limited knowledge of the system (might not even be random)“ [CD13].

### 2.1.1.3. Propagation of Uncertainty

The term propagation of uncertainty refers to the overall effect of the uncertainty of variables on the uncertainty of a function, which is based on these variables. The uncertainty of the entirety of input parameters can be calculated with the help of a predefined mathematical combination of the uncertainties of all single input parameters. This uncertainty propagates through the calculation of the function or a simulation and effects the outputs. If the uncertainty or error is not too large, the impact on the outcome can be determined. Mathematically expressed this means that a function  $y(x_1, x_2, \dots)$  with multiple independent variables  $x_i$  that contain a small uncertainty or error  $\delta x_i$  also contains an uncertainty or error  $\delta y$  which can be determined. In

this context we talk about errors if the error is, in principle, determinable in value and sign. With the help of this information, called the systematic error of input variables, the systematic error of the output variables can be calculated. In contrast to that, the value and sign of the random error is not clearly identifiable. The term random error is used when there are multiple values for the same input variable, which is the case if there are for example multiple measurements for an input value. This kind of uncertainty leads to an uncertainty in the output variables. This can be calculated analogously. For calculation it is necessary to consider if the input variables  $x_i$  are independent or not, and whether they affect the function  $y(x_1, x_2, \dots)$  sufficiently linear or not, in order to choose the correct calculation method.

### 2.1.2. Probability Distributions

Uncertainty can often be expressed in probabilities and therefore we shortly recall the basics. If you throw a fair dice the outcome is uncertain and the chance for each side to be on top is, expressed as probability,  $1/n$ , whereas  $n$  is the amount of dice edges. Considering all the possible outcomes of a dice throw, we have to work with probability distributions. We need to distinguish between discrete and continuous random variables. We talk about discrete distributions if the range of values is limited to a finite or countable quantity of numbers. Discrete distributions can be described by probability mass functions. The probability mass function  $f_x : A \rightarrow [0, 1]$  for a discrete random variable  $X : S \rightarrow A (A \subseteq \mathbf{R})$  defined on space  $S$  is defined as  $f_x(x) = Pr(X = x) = Pr(s \in S : X(s) = x)$ . An example for a discrete probability function is shown on the right of *Figure 2.3*. If the range of values is uncountable or not finite it is a continuous distribution. The equivalent of the probability mass function for the continuous case is the probability density function. A function  $f : \mathbf{R} \rightarrow [0, \infty]$  is called probability density function of the distribution  $\mu$  of the random variable  $X$  if  $P(a \leq X \leq b) = \mu([a, b]) = \int_a^b f(x)dx$ . The integral of the function  $f$  always equals 1:  $\int_{-\infty}^{\infty} f(x)dx = 1$ .

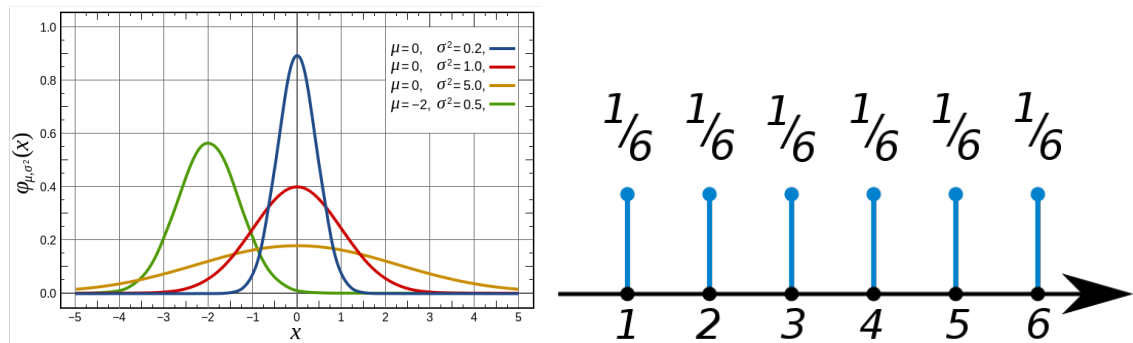
There are also other mathematical models that can be applied to this subject, but probability theory is the most frequently used.

#### 2.1.2.1. Normal Distribution

One important type of continuous probability density functions is the normal distribution. This distribution is of particular importance, because, according to the central limit theorem and under some special conditions, the arithmetic mean of a sufficient large number of literates of independent random variables are approximately normal distributed. This is true regardless of the underlying distribution for independent random variables with a well-defined expected value and well-defined variance. That is why the normal distribution is applicable for describing many random events. On the left of *Figure 2.3* shows some examples of normal distributions.

The probability density function of the normal distribution is:

## 2. Related Work



**Figure 2.3.:** Left: Normal Distribution Example[Ind08], Right: Fair Dice Probability Distribution[Ole07]

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$\mu \in \mathbf{R}$  is the mean,  $\sigma$  is the standard variation and the variance is  $\sigma^2 > 0$ . The red curve on the left of *Figure 2.3* is a special case of the normal distribution, namely a standard normal distribution. This is the simplest case of a normal distribution with  $\mu = 0$  and  $\sigma = 1$ . Since the interval  $\pm 3\sigma$  from the mean contains 99.73 of all measured values one often only considers this interval when working with normal distributions.

### 2.1.3. Input Method Utility, Usability, and Likeability

In order to create a user interface that is accepted by users, Brian Shaker [Sha91] states that the user has to consider it sufficiently useful, usable and likeable in relation to what it costs the user. Therefore a good user interface provides a high *utility*, *usability* and *likeability*. Since input methods are a part of user interfaces this also applies to them. The utility tells if an input method does what is needed functionally. Likeability determines to what degree the users consider a user interface to be suitable and if they liked working with the user interface. Usability determines if the users can accomplish their work successfully. B. Shaker measures usability on four dimensions:

- effectiveness
- learnability
- flexibility
- attitude



The term effectiveness refers to the performance in learning, relearning and carrying out a representative range of operations. Learnability describes the degree of required learning to accomplish a task and the effort required to access utility. Flexibility measures the range of tasks for which a user interface or input method is useful. The attitude tells about the users' satisfaction with the system.

### 2.1.3.1. Usability

Jakob Nielsen defines Usability as follows:

„Usability is a quality attribute that assesses how easy user interfaces are to use. The word 'usability' also refers to methods for improving ease-of-use during the design process.“ [Nie03]

According to J. Nielsen [Nie03] usability has five quality components, which he describes as follows:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- **Memorability:** When users return to the design after a period of not using it, how easily can they reestablish proficiency?
- **Errors:** How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction:** How pleasant is it to use the design?

Alan Dix [Dix09] lists some principles to support usability, split up in principles for learnability, flexibility and robustness. He indicates a list of five principles supporting learnability: predictability, synthesizability, familiarity, generalizability and consistency. Predictability is described as support for the user to determine the effect of future actions based on the past interaction history. Synthesizability is the users' ability to assess the effect of past operations on the current state. If users have already worked with a user interface they have some knowledge and experience in working with it. The extent of this knowledge and experience is the familiarity principle. The support for the user to use this knowledge of a specific interaction within and across applications to other similar situations is called generalizability. If the input-output behaviour is alike in similar situations and task objectives, we talk about consistency. Flexibility is supported by dialog initiative, multi-threading, task migatibility, substitutivity, and customizability. If a user is free to initiate any action towards the system the dialog is user pre-emptive. A users' high ability to pre-empt the system and a low systems ability to pre-empt the user is desired. That is meant with dialog initiative. A system that supports multi-threading supports users to work on multiple tasks at a time. Task migratability refers to the possibility to pass control for task execution between user and system. Substitutivity

means that equivalent values should be interchangeable for each other. For instance, the value seven could be substituted by the calculation  $2 + 5$ . We talk about customizability if the user can modify the user interface to match his desires. Observability, recoverability, responsiveness and task conformance support robustness. If the user can evaluate the internal state of the system from its perceivable representation the system is observable. Giving a user the ability to take corrective action once an error has been recognized is understood by recoverability. The rate of communication between the system and the user is called its responsiveness. If a system enables the users to perform all of the tasks they wish, in the desired way, it fulfills the principle of task conformance.

Ben Shneiderman [Shn92] introduces some principles for designing user interfaces. He suggests to recognize user diversity, prevent errors, and follow the *Eight Golden Rules*:

1. Strive for consistency
2. Enable frequent users to use shortcuts
3. Offer informative feedback
4. Design dialog to yield closure
5. Offer simple error handling
6. Permit easy reversal of actions
7. Support internal locus of control
8. Reduce short-term memory load

### 2.2. Uncertainty

One of the three central topics of this thesis is uncertainty. Since the term *uncertainty* is rather vague and is used in different ways in a number of fields we clarify our understanding of this term. Uncertainty may have different meanings depending on the context. In *Section 2.1.1 – Uncertainty* we have discussed uncertainty terminology and its association to statistics and decision theory.

When dealing with simulations, there are several kinds of uncertainty to consider, outlined in *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*. Because our subject is to develop input methods, we focus on the measurement of uncertainty and variability of in- and output data. However, there is few related work about uncertainty related to data input. Therefore we first present work about data output, its interpretation, and understanding by humans.

Most publications in this subject area explore uncertainty in the context of weather forecasts. This is because among the general public, weather forecasts are probably the most visible and commonly used type of scientific prediction [RBKSC06].

Kenneth R. Mylne [Myl02] makes a comparison between deterministic and probabilistic weather forecasts and argues about the decision making of users with the help of such forecasts. This is important for us because we want to create input methods where people have to cope with uncertainty. Therefore it is interesting to know if they get along better with deterministic representations or with probability distributions. This becomes relevant in *Chapter 4 – Input Method Prototypes*. In addition to that, it is also mentioned that forecast uncertainty can be expressed qualitatively or quantitatively, which might also matter when developing input methods. Kenneth R. Mylne describes a method of estimating the economic value of weather forecasts for decision-making. Its purpose is to aid forecast users in optimising their decision-making from probability forecasts. Kenneth R. Mylne calculates the value in the same way for both, probability weather forecast and deterministic weather forecasts. This way their value can be compared. Over the years several objective methods of generating probability forecasts have been developed to take account of uncertainty and deterministic model forecasts have been augmented with a probability distribution based on a previously observed distribution of model errors. There is a number of sources of uncertainty in numerical weather forecasts, such as uncertainty due to approximation by parameterization in models, and uncertainty in the analysis of the initial state. This is of relevance for this thesis, since we have to deal with uncertainty of input data of a simulation. Concerning these sources of uncertainty, Kenneth R. Mylne mentions ensemble prediction systems that have been developed in an attempt to estimate the probability density function of forecast solutions by sampling the uncertainty in the analysis and running a number of forecasts from perturbed analyses. This makes sense only if a probability forecast provides more information than a deterministic forecast. Indeed, Kenneth R. Mylne calculated a higher forecast value for probability forecasts. However, this article mentions that users are familiar with making decisions from deterministic forecasts and are uncertain how to respond to probability forecasts. This article shows the potential benefits of augmenting deterministic approaches with probability distributions, when dealing with uncertainty, but it also outlines the increased requirements placed on the user.

Covering a similar topic Rebecca E. Morss et al. [MLD10] also respond to different ways to convey uncertainty „as a range of possible values or a percentage chance of exceeding a damage threshold“ . Decision scenario questions were asked in a nationwide US survey, as this article addresses the issue of the hydro meteorological community having limited understanding of how people interpret forecast information. However, the results suggest that many people were able to interpret probabilistic forecasts well enough to use them in decision questions. Similar to the work of Kenneth R. Mylne the questions cover both, deterministic forecasts and such forecasts that convey uncertainty. For this thesis it is of relevance, that the results provide insight into the respondents' interpretations of deterministic and uncertainty forecasts. Rebecca E. Morss et al. state that the findings suggest possible presentation issues in the interpretation of range forecasts and therefore we have to assume that not all individuals

might be able to understand uncertainty represented in a probabilistic way. Another article from Rebecca E. Morss et al [MDL08] covers related topics such as if people infer uncertainty into deterministic forecasts or what formats people prefer for receiving forecast uncertainty information.

Knowing that people in general have trouble working even with basic probability representations, it is of interest to know how they deal with these problems in more detail. Isaac M Lipkus et al. [LSR01] examined how highly educated participants performed on a general and expanded numeracy scale. In contrast to the previously mentioned work, this one focuses on different numerical expressions, namely probabilities, percentages and frequencies. Although usually basic numeracy on average people is assumed, only about 60% of the sample could convert percentage to proportion or even solve a basic probability problem. Results were even worse for converting proportions to percentages and poorest performance was in converting probability to proportion. Overall, people performed slightly better when dealing with percentages than with proportions.

Ulrich Hoffrage et al. [Hof00] also discuss the varying performance of humans in interpreting uncertain data, depending on their representation as probabilities, percentages, or absolute frequencies. As well as many other articles, this one mentions the bad performance and difficult understanding of most people when using or combining statistical information. This especially applies when those information is provided in the form of percentages. Participants of a study answered questions about a cancer screening test tremendously worse when the probabilities were communicated as percentages compared to the answers when information was provided with natural frequencies. Since natural frequencies carry implicit information about base rates and they reduce the number of computations required to determine the positive predictive value of a test, Hoffrage et al. [Hof00] state that they facilitate inferences. As a response to this article B. Butterworth [But99] even suggests that natural frequencies facilitate reasoning because people are born with a specialized capacity for processing collections of discrete objects, rather than probabilities and fractions.

Returning to the different possibilities in expressing uncertainty, mentioned by Kenneth R. Mylne [Myl02], namely qualitative and quantitative, Mark S. Roulston [RBKSC06] provides more insight into user benefits when provided with quantitative information. It provides experimental evidence that, when provided with quantitative information about weather forecast uncertainty, non-specialists can make better decisions. A standard tool of operational weather forecasting is *Ensemble Forecasting*. Multiple simulations of the atmosphere are made to reflect uncertainty and this allows estimates of the forecast standard errors. Mark S. Roulston [RBKSC06] states that, so far, mainstream media provide little information about uncertainty, with few exceptions. Similar to the findings of Kenneth R. Mylne [Myl02], the result shows that people actually do better when provided with additional uncertainty information in contrast to those without such information. Another interesting outcome is that the quantitative uncertainty information of the standard error yielded most of the improvement in decision

making, whereas additionally stating the probability most relevant to the task did not turn out in a significant improvement.

Quantitative probabilities are assumed to provide more information to the public than qualitative risk statements [GHv<sup>+</sup>05]. This is the reason for risk experts to promote quantitative probabilities. Both groups, Allan H. Murphy et al. [MLFW80] and G. Gigerenzer et al. [GHv<sup>+</sup>05] state that the problems people have with interpreting the weather forecasts in public media do not originate in the lack of understanding quantitative probabilities, but in the missing specification of the class of events it refers to. Although quantitative probabilities are assessed as an appropriate way to communicate uncertainty because of its ability to express uncertainty inherent in forecasts in a precise, unambiguous manner [MLFW80], they are actually misunderstood by many. The probability of rain, for instance, can be understood as referring to days, regions or time. Gigerenzer et al. [GHv<sup>+</sup>05] state that this problem of possible miscommunication also applies to other domains. They present a possibility to overcome this problem through an example of a newspaper report. In addition to a verbal explanation of the probability of rain a table with expected chances and expected amount of rain is provided.

This leads to the subject of verbal probability terms, which is another alternative to represent uncertainty. Thomas S. Wallsten et al. [WBR<sup>+</sup>86] investigate how people interpret such terms. They point out that in most empirical work on the meaning of probability terms the various, commonly used probability terms have been classified by asking subjects to give them numerical equivalent. Thomas S. Wallsten et al. explain that a probability term can be expressed as membership functions over the  $[0, 1]$  probability interval. For weather forecasters there are two main reasons to prefer non-numerical terms over probability numbers: Weather forecasts are not precise and thus a numerical representation would be misleading, and the second reason is that most people are more confident when working with probability terms instead of numerical representations, since people more naturally think about uncertainty in a verbal manner. A major drawback, however, is that people have very different interpretations for the various probability terms and therefore these expressions have generally vague meaning. When developing text-based input methods including verbal probability terms, the tables presented by Thomas S. Wallsten et al. [WBR<sup>+</sup>86] provide useful information about how people interpret non-numerical terms in probability numbers.

Related work in this field gives valuable information about how people interpret and use probability information. People seem to cope better with discrete forecasts and quantitative information than they do with qualitative and continuous representations of probabilities. Although people seem to perform poor in using weather forecasts, there is evidence that this is not because of a misunderstanding of distributions, but the lack of information about the reference class.

### 2.3. Uncertainty and Visualization

For the combination of the main topics **Uncertainty** and **Visualization** there is less related work. However, there are some publications that explore various types of representations for different types of uncertainty representations.

Although Ulrich Hoffrage et al. [Hof00] imply that people can work more easily with collections of discrete objects and most weather forecasts stick with discrete probabilities there was evidence in the presented work that people do benefit from extended representations such as continuous variables. Harald Ibrenk et al. [IM87] mention that most of the literature in this area mainly covers discrete probabilities and explicitly extends to the consideration of continuous variables. A questionnaire on nine alternative graphical representations, which contain, amongst others, point estimate with error bar, various visualizations of a probability density function, discretised versions thereof, and a conventional cumulative distribution function, has been carried out. Although the task of the participants was to give their best estimate of the mean, providing limited information about the overall performance only, these results give an interesting insight into how well these various visualizations communicate probability information. Another intriguing finding is that, when using displays that did not explicitly mark the mean, people showed a tendency to select the mode rather than the mean. Harald Ibrenk et al. noted, that until future work suggests otherwise they use a combination of a cumulative distribution function and a probability density function plot.

As we already mentioned, the term uncertainty can mean many things and therefore we have to differ between several types of uncertainty. Olston and Mackinlay [OM02] distinguish between two forms of uncertainty: statistical uncertainty and bounded uncertainty. In this regard they refer to a report by the US Department of Commerce National Institute of Standards and Technology [Tay09]. While statistical uncertainty contains information about the distribution of possible values there is no such information with bounded uncertainty and so no assumption can be made about the probability distribution of possible values inside the interval. Olston and Mackinlay describe a technique for conveying bounded uncertainty in visualizations and show how it can be applied systematically to the common display of abstract charts and graphs. They explain that data values in abstract charts and graphs are graphically encoded in the positions of graphical elements or in the size of elements, be it in one or more dimensions. They also suggest to clearly indicate both, the presence and the form of uncertainty when the underlying data is uncertain and not only the presence of uncertainty, which is the case for most common visualizations so far. To overcome the problem of using the same graphical representation for both kinds of uncertainty two alternative methods for conveying uncertainty in the positions or extents of graphical representations of data has been advocated: error bars for statistical uncertainty and ambiguities for bounded uncertainty. In the bounded uncertainty case the ambiguous region between possible boundaries can be drawn as graphical fuzz. Olston and Mackinlay state that these techniques could also be applied to a broad range of displays that use position and extent to encode data.

Besides the possibility to enhance existing visualization methods with graphical elements that add uncertainty information, new and more advanced visualizations can be developed. While such visualizations have the advantage that they are explicitly created for the visualization of data with uncertainty and therefore the presence of uncertainty is obvious and well represented, the disadvantage is that people are not as familiar in interpreting these visualizations than they are when working with extended existing visualizations. Alex T. Pang et al. [PWL97] survey techniques for presenting data together with uncertainty. They state that some of the reasons for most visualization research to ignore or separate the presentation of uncertainty from data is the lack of methods that present uncertainty and data and the inherent difficulty in defining, characterizing and controlling the uncertainty in the visualization process. However, there are some existing displays that visualize data and uncertainty. To be able to classify those existing visualizations and those to be developed, Alex T. Pang et al. describe a way to categorize uncertainty displays. Therefore they first briefly mention uncertainty visualization categorizations in other work like for example a more general classification from Keller and Keller [KK93], where a taxonomy of visualization goals is used. Other mentioned methods are classification by data-ink maximization [Tuf83], by using fundamental parameters like length, area and ratios, by using a lattice arrangement of data [BHG89] and a classification based on the dimensionality of the entity that is being visualized. Brodlie uses characteristics to classify techniques as follows [BCE<sup>+</sup>92]:

- **underlying field:** groups as dependent or independent variables and differs between having ordinal or nominal values
- **view:** the kind of graphical representation

Hesselink et al. [HPv94] use the following:

- **order of data:** scalar, vector, tensor
- **spatial domain of visualization objects:** point, line, surface, volume
- **information level of data:** elementary, local, global

Partly based on the works, mentioned above, a classification with five characteristics is proposed in order to incorporate uncertainty information into visualization schemes:

- **value:** scalar, vector, tensor, multivariate
- **location:** dimensionality of space, time, ...
- **data extent:** discrete or continuous, corresponds to the range, interval or period over which data is valid
- **visualization extent:** using a discrete visualization, indicating individual datums, or a continuous range of data

- **axes mapping:** experiential rendering, replicating the viewers' experience with the according visualized phenomenon, or abstract rendering, plotting the data in a non-experience-based mapping

We look at those schemes more detailed in *Section 3.4 – Classification of Uncertainty Representations*. It should be noted that Alex T. Pang et al. differ between visualizations for the discrete and continuous case. That is what they refer to when using the term data extent. Using these schemes, some existing uncertainty visualization techniques, such as side-by-side, difference image, pseudo-colour, contour lines, blinking and scatter plot, are categorized. In the same way new developed visualization techniques have been categorized. They are more advanced than most of the existing displays with respect to uncertainty visualization. They use techniques like adding glyphs, adding geometry, modifying attributes, modifying geometry, animation, sonification, and psycho-visuals. In *Chapter 4 – Input Method Prototypes* we discuss how valuable these approaches are for our work.

We have seen that there are already some approaches to visualize data with uncertainty. However, Andrej Cdeilnik et al. [CR00] point out that uncertainty should not overshadow the data values. A. Cedilnik et al. state that many methods can distract the data when addressing the problem of visualizing data with uncertainty. Therefore they developed a method with minimal conceal of the data by using procedurally generated annotations which are deformed according to the uncertainty information. These annotations can inherit the quality contribution so that the viewer is able to focus on the data, but use the annotations for information about uncertainty. Andrej Cdeilnik et al. describe four techniques for distorting annotation lines: intensity width variation, exponential sharpness, noise, and amplitude modulated distortion. Their proposed method is the encoding of uncertainty information in the annotation lines of the grid which overlays the data.

There is a broad range of proposed uncertainty visualizations, such as spatial uncertainty as they relate to kinds of cartographic symbolization [Mac92], glyphs for visualizing uncertainty in vector fields [WPL96], graphs and pictograms [ZFFU08], multiple realizations of a continuous surface [DK97] and various ways of cartographic visualisation of uncertainty [VVG98].

### 2.4. Uncertainty and Input Methods

When combining the topics **Uncertainty** and **Input Methods** there is little related work to be found. In this subject area it is important to distinguish between different types of uncertainty that may occur when using input methods. In *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations* we have a closer look at the different types of uncertainty that may occur in the process of conducting a simulation.

The topic of data input and uncertainty is not completely ignored by experts. Julia Schwarz et al. [SHMW] address the problem of uncertainty that occurs when using upcoming input



technologies, such as touch- and recognition-based input. Julia Schwarz et al. present a framework for handling input that entails uncertainty. In contrast to input handling in most modern interface toolkits, this framework takes ambiguity and input errors into account. They describe the problem that if the uncertainty of an input is too large, e.g., a touch screen has been pressed between two closely located UI elements, there is a chance to determine the desired action by considering several categories of information for the input: type, value, time, location and context. This framework supports existing interaction methods and enables new interactions. The framework addresses the problem of uncertainty on data input, but with respect to the uncertainty occurring in the action of conducting the input and in the interaction with the input method. In this work, we want to deal with the uncertainty in the data, which is to be entered, itself. Still this is a valuable article for us, because we also consider other sources of uncertainty, so that we can observe and determine borders of the different kinds of uncertainty that we have to deal with.

## 2.5. Input Methods

Once a prototype has been created, it has to be evaluated. Taking the properties and characteristics of good input methods into account early improves the overall results and saves time in the process of developing input methods. In this thesis, we refer to a framework for classifying usability and user experience measures, which is proposed by Nigel Bevan [Bev08]. He states that the most common reason for measuring usability is to obtain a more complete understanding of users' needs and to improve the product in order to provide a better user experience. He also mentions that it is important to establish criteria on an early stage of design. Two types of user experience and usability measurement are distinguished here: One type is to measure the result of using the whole system. This is like a black-box view, as it examines *what* is achieved, rather than *how*. Nigel Bevan splits up overall quality in use, as you can see in the *Figure 2.4*.

The other type is measuring the quality of the user interface. This one is more appropriate for our case, because we want to evaluate the input methods alone at first, without the context of an embracing system. When evaluating the integration of the input methods into the simulation tool the black box approach might come in handy. When talking about user interface usability Nigel Bevan refers to views of usability in Human Computer Interaction (see *Section 2.1 – Background*). Putting together the information gathered in this article Nigel Bevan presents a table showing the factors contributing to system usability and user experience, which we use in *Section 3.5 – Input Method Metrics* and *Chapter 6 – Online Survey*.

## 2. Related Work

---

1. Usability in use
  - a) Effectiveness in use
  - b) Productivity in use
  - c) Satisfaction in use
    - i. Likeability
    - ii. Pleasure
    - iii. Comfort
    - iv. Trust
2. Flexibility in use
- a) Context conformity in use
- b) Context extensibility use
- c) Accessibility in use
3. Safety
  - a) Operator health and safety
  - b) Public health and safety
  - c) Environmental harm in use
  - d) Commercial damage in use

**Figure 2.4.:** Quality in Use after Nigel Bevan [Bev08]

## 2.6. Visualization and Input Methods

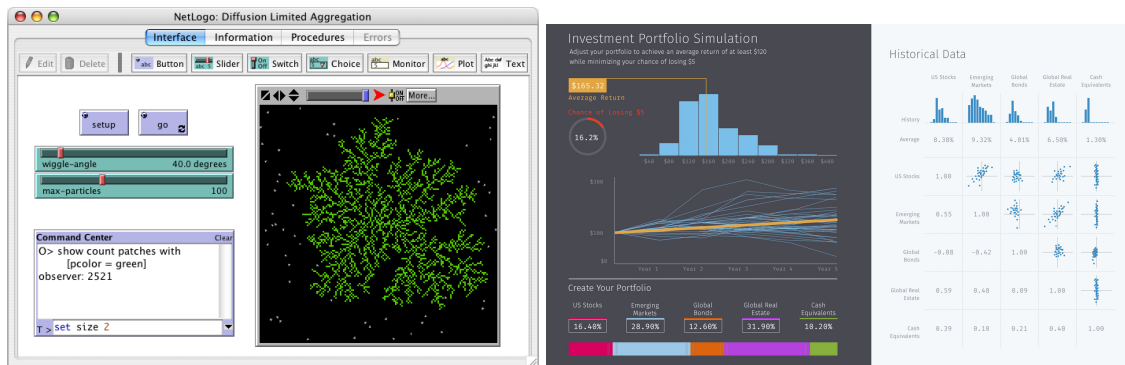
The combination of the topics **Visualization** and **Input Methods** primarily deals with the interactivity of input methods or with graphical supportive interface elements that derive from the according visualization itself. Therefore there is few related work that explicitly pertains to this section and reveals valuable information for this thesis.

An interesting concept is to increase the effectiveness of sliders with respect to data visualization. To show selected values in relation to the data and its distribution Stephen G. Eic [Eic] uses the space inside the sliders as follows:

1. interactive colour scale,
2. bar plot for discrete data, and
3. density plot for continuous data

Data sliders are suitable for this purpose because it is an established mechanism that is used in pretty much any user interface system. He mentions different possible applications of sliders: double-edged sliders with upper and lower thresholds, categorical sliders for selecting an arbitrary subset of data and an alpha slider for selecting a single string from a sorted list. He points out that the idea uniting these applications is filtering. Stephen G. Eick [Eic] presents four data visualization sliders, two for both continuous and discrete data enabling the user to specify an arbitrary number of disconnected intervals. They are, compared to ordinary sliders, improved in three ways:

1. The space inside the slider is used as colour scale
2. The data values are shown as tick marks in a „rug plot“



**Figure 2.5.:** Left: NetLogo’s User Interface, with Model Diffusion Limited Aggregation[TW04], Right: Forio Epicenter Example: Investment Portfolio Simulation[For15]

### 3. The distribution of the data is shown as a density plot

The sliders A and B combine a visualization colour, scale and a slider with interaction techniques, which B extends by a smoothed distribution. C and D show categorical data with 50 discrete values mapped to one colour-coded bar. We want to highlight the approach of the slider B, as it contains a density plot that shows a smoothed distribution. We refer back to this when developing and extending our input methods in *Chapter 4 – Input Method Prototypes*.

## 2.7. Simulation Tools

As we want to create a simulation tool that makes use of the developed input methods we briefly look at some existing input tools.

NetLogo is a multi-agent programming language and modelling environment for simulating natural and social phenomena, presented by Seth Tisue et al. [TW04]. It is implemented in Java for two major reasons: It is cross-platform and it has a relatively high performance. It contains an API for extensions so that users can add new elements to the language by implementing them directly in Java. These tools enables users to explore the behaviour of existing simulations under various conditions and tries to provide an easy way to create new models, in addition to the more than 140 pre-built simulations, even without professional programming knowledge. The left side of *Figure 2.5* shows NetLogo’s user interface after opening and running a model. On the left you can see the model controls. There is a web-based tool which provides two-way links between NetLogo and its browser page, the Behaviour Composer [Dr.14]. It can help to build, share and discuss agent-based computer models.

Forio Epicenter is more than just a simulation tool and is described as a computational platform for hosting server-side models, creating interactive interfaces and sharing insights [For15]. It allows quick building of web-based simulations. In addition to the drag-and-drop User Interface

Builder tool it comes with the possibility to create an own user interface code using JavaScript or REST API. A screenshot of an example project is provided on the right side of *Figure 2.5*. This screenshot shows various interesting inputs and visual outputs and an appealing user interface. Models can be written by using various languages: Forio SimLang, Julia, Python, R or Vensim. Since Epicenter has the option to use Node.js, the logic can be executed server-side, but this feature is only accessible for those who at least pay for a Medium Team subscription plan. Unfortunately, it is free for personal use only.

### 2.8. Summary and Discussion

We presented related work about the various topics that intersect with the topic of this thesis. While some articles argue against the extended usage of probability representations for the public, other articles reveal the possible benefits and suggest to even include probability distributions [Myl02] [MDL08]. When confronting users with probabilities it makes a difference how they are presented and we saw arguments for preferring natural frequencies over percentages [Hof00]. For the case of visual representations there are reasons suggesting to make use of probability density functions [Hof00] [IM87]. One argument for the provision of probability density information is the suggestion to enhance bounded probability representations with probability density information. We have also inspected techniques for the classification of uncertainty visualizations which we use in this thesis [BHG89] [BCE<sup>+</sup>92] [HPv94]. We have seen that when dealing with input methods there are more sources of uncertainty than only the intrinsic uncertainty in the input data [SHMW] and we examine this further in *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*. Furthermore we have seen usability measurement techniques that we use to evaluate the developed input methods [Bev08]. The proposals to visually enhance sliders are useful for the development of visually appealing and informative input methods [Eic]. Since we plan to create a simulation tool, we also took a look at some existing simulation tools.

## 3. Inspecting Uncertainty and Input Methods

In this chapter, we gather up the information presented in *Chapter 2 – Related Work* to highlight the important facts and to draw useful conclusions. The knowledge worked out in this chapter provides the basis for further reflections on developing input methods.

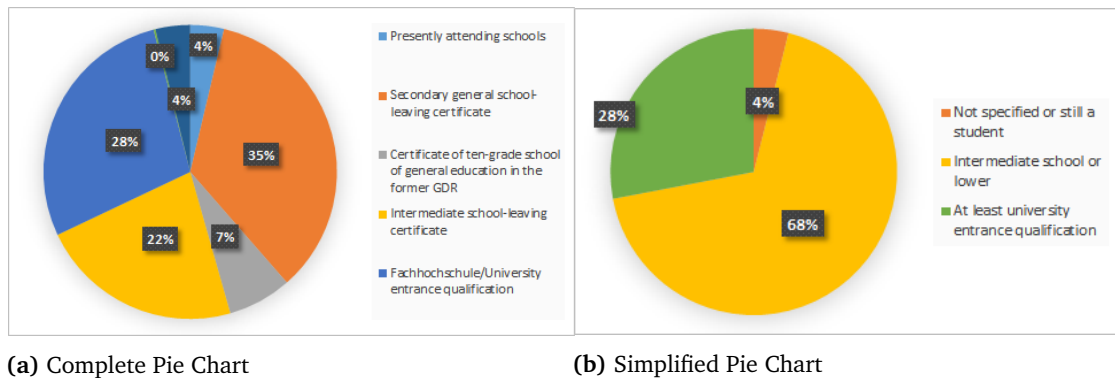
### 3.1. Humans Dealing with Uncertainty

As we have already seen in *Chapter 2 – Related Work*, humans often perform badly when dealing with uncertainty representations, such as weather forecasts, in everyday life. However, we have to inspect the articles precisely to filter those references to weather forecasts that do not represent a type of uncertainty of our interest.

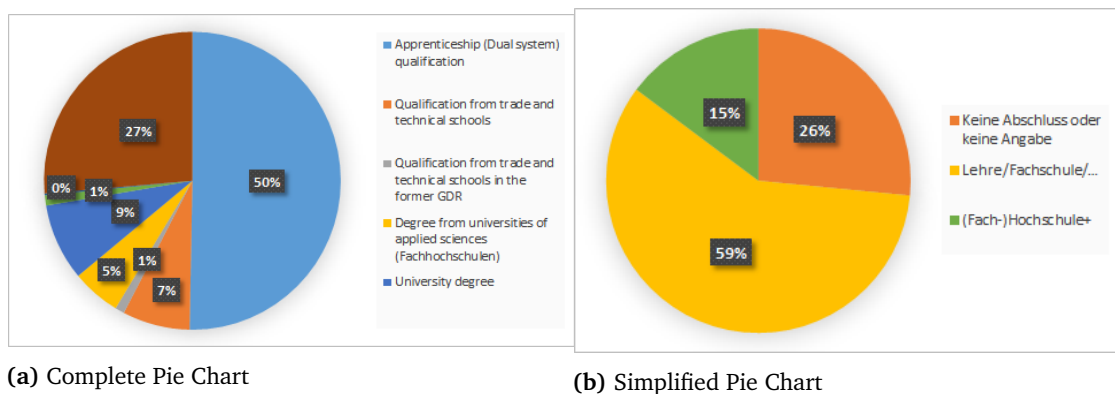
Although weather simulations do not provide explicit results, but a set of several events that occur to a certain probability, there are both, deterministic and probabilistic representations. Although probabilistic weather forecasts provide more information Kenneth R. Mylne [Myl02] states that humans cope better with deterministic approaches. This suggests that the more information a representation provides the less well humans get along with it and the less they are able to interpret and use its information. If this is true and whether this also applies to input methods, this has to be reviewed. It might be reasonably assumed that if this turns out to be the case for visual outputs it also applies to visual input methods which lean in design towards the according output visualization. In this case one approach would be to supply users with an input method that suits their level of education, so that only highly educated people work with a probabilistic approach. Yet, Rebecca E. Morss et al. [MLD10] and Isaac M Lipkus et al. [LSR01] found indications that humans generally, and even highly educated people, have trouble working with probabilities.

At this point we have a look into some statistics about the education of the population in Germany. The Federal Statistical Office of Germany (ger. Statistisches Bundesamt Deutschland) provides statistics of educational attainment [Fed13]. *Figure 3.1* and *Figure 3.2* show the latest available data from 2013. We first look at the level of general education (see *Figure 3.1a*). Assuming that only those who acquired a university entrance qualification, or similar, benefit from their knowledge obtained in school when dealing with uncertainties only 27.9% of the population in Germany might be able to work with advanced uncertainty representations, leaving 68.4% with little chance getting around with such tasks (see *Figure 3.1b*). If we expect

### 3. Inspecting Uncertainty and Input Methods



**Figure 3.1.:** Educational Attainment of the Population in Germany from 2013 (Highest Level of general Education completed) from Federal Statistical Office of Germany[Fed13]



**Figure 3.2.:** Educational Attainment of the Population in Germany from 2013 (Vocational Qualification attained) from Federal Statistical Office of Germany[Fed13]

people with an intermediate school-leaving certificate (22.3%) to be able to work with advanced uncertainty approaches there are 49.8% of the population left. When we take the attained vocational qualifications as information basis more than 50% of the population will likely experience problems. If we emanate that a completed general education does not provide any advantage, in this case only 14.7% are likely to be prepared to cope with advanced approaches, assuming that only a degree from universities of applied sciences (5.3%), university degrees (8.3%) and doctor's degrees (1.1%) provides the required knowledge (see Figure 3.2b). If we take into account that we want to develop input methods for a broad range of users, especially non-experts, we have to investigate this issue.

Note that this is a theory only and it remains to be seen how big the influence of the educational degree actually is, since it does not directly indicate knowledge about statistics, stochastics, probability theory, or even probability distribution.

Gigerenzer et al. [GHv<sup>+</sup>05] state that the problems people have when interpreting weather forecasts originate in the missing specification of the class of events it refers to. This suggests that we could even confront low educated people with probabilistic approaches. This has to be reviewed. When developing input methods for uncertain data, it is also important to specify the reference class.

When we come to design input methods for data with uncertainty we should consider how well people generally deal with different ways of uncertainty representation, like mentioned by Isaac M Lipkus et al. [LSR01]. They conclude that natural frequencies can be understood easier than percentages. In our work, we should therefore also consider approaches based on natural frequencies.

### 3.2. Uncertainty Occurrences when Dealing with Simulations

While there are many occurrences of uncertainty in science and everyday life we are particularly interested in uncertainties in simulations. Since the work of this thesis included the development of a web-based simulation tool which integrates the developed input methods, we should know about the various uncertainties that play a role in the simulation process and interaction.

*Figure 3.4* shows an overview of uncertainties one faces during a simulation process. First, we briefly go through this figure and then we provide more details. Before *users* can start to enter data into a simulation tool they need to gather the information (*facts*) required to conduct the simulation. Once the required data is available to users they can start to enter the data with the help of the provided *input* methods. The *input* is passed to the *simulation implementation*, which runs on a hardware system that provides the required resources to conduct the calculations. This simulation in the form of a computer program is the *implementation* of a *model* that describes the attributes or behaviours of the world we live in. After the simulation calculations are finished the user can access the results by collecting the data of the output. By interpreting the simulation *output* data the *user* receives the *results*.

The red, numbered dots in *Figure 3.4* denote possible sources of uncertainty. It is striking that this happens when information is passed from one entity to another. In the following we step through *Figure 3.4* to inspect these sources of uncertainty.

**(1) Facts → User:** The data which serve as input for the simulation usually base on facts about entities in the real world that we live in. Some of those information might not be available to the user. One possible reason is a simple lack of knowledge. Users might not have access to the required information or they did not inform themselves well enough. This is epistemic uncertainty, that we have already discussed in *Section 2.1 – Background*. If the requested information are the quarterly numbers of a company and the user can only estimate the correct numbers, it is due to lack of knowledge, since the information are in principle available. This

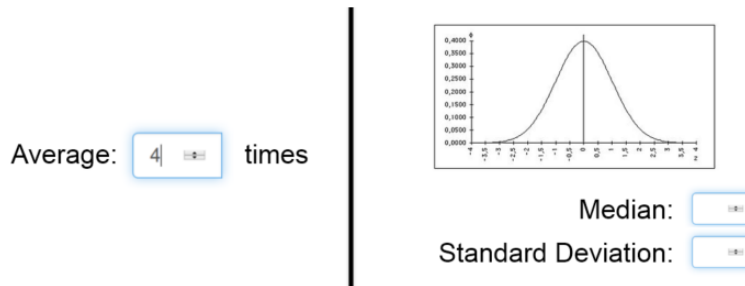
is related to external uncertainty described by Xiaoping et al. [DCG99]. Another possibility is that the information can not be obtained. Some physical quantities can not be determined due to limitations of measuring methods. The information is there, but there is no way for us to get them, at least not now. The limitations of the measuring methods and our limited knowledge about the world restrict our possibilities to obtain the desired information. Pang et al. [PWL97] refers to this as *uncertainty in acquisition*. Consequently uncertainty can also arise because we have no information about the extent of missing or wrong information due to lack of knowledge or imprecise measurement.

**(2) User → Input:** The *user* interacts with the provided *input methods* to input the required data, but there are some obstacles for the *user*. The *input* design might arise problems in the *users* understanding and therefore complicate the correct *input* of the data. This leads to some uncertainty about the correctness of the *input*. The *user* might enter different data as intended due to this misunderstanding. More uncertainties arise due to input method limitations. A horizontal value slider with the width of 200 pixels and an input data extent of 1000 cuts the amount of possible input values. *Users* who try to enter the value 3 will be disappointed since the slider can be dragged with pixel precision only. So they will have to choose 0 or 5 instead. Another example is a number field that takes integer values only, but a user wants to enter a real number, such as 2.5. In this case the user will have to choose between 2 and 3. We can not tell which value in the interval  $[2, 3]$  the *user* tried to enter and therefore this leads to more uncertainty.

**(3) Input → Implementation:** Once the required data have been entered it might be necessary to transform them. In this context Pang et al. [PWL97] talk about *uncertainty in transformation*. This can be a simple transformation such as the change of the unity from kilometer to meter or a more complex transformation such as the combination of multiple parameters to calculate another desired value. It might also be necessary to rescale, resample or quantize the data. Since these transformations alter the form of the data in its original form they potentially introduce uncertainties. Another source of uncertainty is what we call the expressiveness of the *input method*. Although a *user* might have entered the data as expected for the according input method there might be some uncertainty due to limited information value the input method allows to enter. Consider *users* who are asked to enter how often they go to work by car per month. One possibility is to provide a single number field to enter the average value. Alternatively we can ask for a normal distribution with the help of two number fields, the variance and the mean. This provides much more information than a simple average value and we can now tell something about the possibility of the user to take the car for a certain amount of times in a month, for example. *Figure 3.3* illustrates this. If the *input* method does not allow to enter the required information in a way that provides sufficient information value, the missing information has to be guessed or interpolated, which leads to uncertainty. For the example above one would have to guess the variance of the distribution.

**(4) World → Model:** A *conceptual model* is a composition of concepts that help us understand subjects, attributes and behaviours. It always refers to a set universe, usually to the real





**Figure 3.3.:** Effectiveness Example, **Left:** Low Expressiveness / Estimate or Guess, **Right:** Higher Expressiveness / Probability Distribution

*world* we live in. Since conceptual models are only the attempt to explain or describe a subject, attribute or behaviour, the according conceptual *model* is not necessarily completely correct. Sometimes the concepts are too complicated to easily understand them and therefore a simplified *model* is preferred over a complicated one. In many cases we simply do not know better and the according *model* might be only the best explanation so far or an approximation. The free-fall model, for an instance, models the process of falling objects. It does not consider air friction and therefore is inaccurate. Consequently there are uncertainties, which are called structural uncertainty. We call this structural uncertainty.

**(5) Model → Implementation:** Some aspects of models can not be modelled in programming languages, like for example uncountably infinite sets. If the *models* are not expressed in a mathematical way they first have to be transformed, if possible, where some aspects might have to be simplified. Due to limited resources of simulation hardware it is necessary to make an accuracy vs. speed trade-off. If the available resources are low one will have to accept a longer runtime and/or a lower accuracy, due to the usage of approximation techniques. It's also possible that inappropriate datatypes are chosen or there might even be errors in the implementation. The numerical errors and numerical approximations are also called algorithmic or numerical uncertainty.

**(6) Implementation → Output:** The *output* suffers similar problems as input methods. If the representation type is not chosen appropriately there is a chance that the user misinterprets the data. Finding the correct representation or visualization can be a hard task. Pang et al. [PWL97] refer to the part of this problem that deals with visual representations. They mention several sources of uncertainty in visualization, such as interpolation, different integration methods in flow visualization and different approaches in volume rendering. Such errors and inaccuracies can also occur in other forms of *output* representations and therefore lead to uncertainty.

**(7) Output → User:** Analogous to the *users'* interaction with the input method, we consider two sources of uncertainty in the users' interaction with the *output*: *output* limitations and

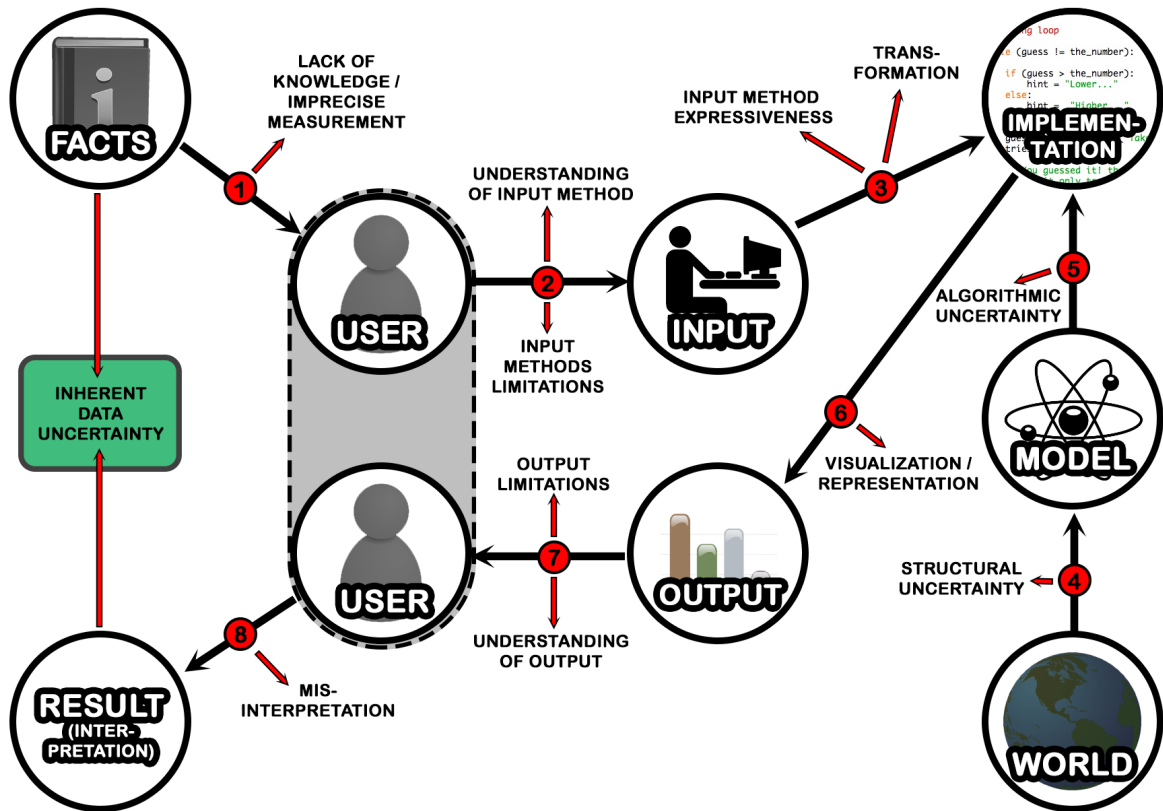


Figure 3.4.: Uncertainties in Simulations Overview

understanding of visualization or *output*. This user is not necessarily the same *user* that inputs the data. In weather forecasts, for example, the *users* who input the data are weather forecast experts, while the *users* who read the output are people watching the weather forecast in TV or reading it in the newspaper. If in contrast to that a *user* enters data in a simulation tool the input and *output user* is the same person.

**(8) User → Result:** Independent from the uncertainties that may arise when *users* try to read the data from the simulations outputs, they might draw a wrong conclusion. In *Chapter 2 – Related Work* many presented articles deal with decision finding of users that use uncertainty visualizations.

We have seen many sources of uncertainties in simulations that may occur in different stages of the simulation process which we have to consider when dealing with simulations.

### 3.3. Uncertainties of Interest for this Thesis

In *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations* we have seen many uncertainties in simulations that influence the results, but they do not belong to the uncertainty that we are interested in. The inherent uncertainty in the facts is the one we want to work with. These uncertainties propagate through the whole simulation and lead to results that contain uncertainty. Also the uncertainty in *Figure 3.4 – (1)* might be of interest. Sometimes the desired information is the amount of information a user has about a certain topic considering the epistemic uncertainty. However, the main focus lies on the facts' inherent uncertainties and therefore one has to consider the existence of the other uncertainties and find ways to handle, monitor or minimize them.

#### 3.3.1. Managing other Uncertainties

We have to think about strategies to handle other uncertainties in order to be able to verify input methods in the context of simulations. Also for those who develop simulations should know and take care about the sources of uncertainties in *Figure 3.4*.

**(1) Facts → User:** It is hard to respond to the limited knowledge of the users while designing input methods or developing a simulation. We can hardly do anything to minimize this epistemic uncertainty but we may try to monitor and take it into consideration in our calculations. The user might not be aware of the extent of the missing knowledge and therefore we would hardly be able to measure this. One could enable to user to specify the degree of certainty of his input. Since we want to focus on the uncertainty that is inherent in the facts we drop this approach and try to work with data that are very well known by the users so that we can make sure the epistemic uncertainty is as small as possible. The problem of insufficiently accurate measurement methods can not be solved generally. In some cases it is possible to determine the extent of inaccuracy of a measuring method. This information can be used to determine the extent of uncertainty that arises due to these inaccuracies. For the validation of our input methods we simply use data that can be measured accurately.

**(2) User → Input:** One main focus lies on the the uncertainties that occur when users interact with input methods. The input methods limitation in precision can be overcome by a good input method design and the right choice of input parameters. We therefore have to choose the input method parameters wisely. The other source of uncertainty occurs if the user does not completely understand the usage of the input method. If the user has problems working with the input method this is an indicator for bad usability. We have to take care for a good usability when we create the input methods. This, and more, is evaluated in a online survey and a user study in *Chapter 6 – Online Survey* and *Chapter 7 – User Study*.

**(3) Input → Implementation:** Most transformations such as the change from meters to kilometers do not lead to a loss of information, but sometimes more complex transformations

can not be avoided. If such transformations are necessary one has to accept the associated uncertainty. Since we know what the transformation does, we have knowledge about the possible extent of uncertainty. The input method expressiveness may lead to a more simple and understandable input method at the expense of its certainty. See *Section 3.4.2.1 – Expressiveness vs. Usability* for more information about input method expressiveness.

The other sources of uncertainty can be bypassed in evaluation by not testing the user input in the context of a whole simulation but only in data entering. Therefore, minimizing the other uncertainty sources is not that important for this thesis, but we briefly look into it.

**(4) World → Model:** Choosing an appropriate and well-founded model is the best way to minimize this uncertainty until new and better models are introduced.

**(5) Model → Implementation:** The more the development in computing power proceeds the less approximations have to be accepted and the more accurate the model implementations become. Putting the focus more on accuracy than on performance can help to keep this uncertainty low.

**(6) Implementation → Output:** Choosing the right way of data representation prevents unnecessary inaccuracies. However, problems, such as different 3D rendering methods, are harder to overcome and if there are no better representation methods we have limited options.

**(7) Output → User:** When using output methods that are easy to understand few misinterpretations should occur. Providing an additional description or training the user may even increase the understanding. Uncertainties due to output limitations should occur seldom if an output method with sufficient expressiveness is chosen.

**(8) User → Result:** Like for the case of epistemic uncertainty of the users limited knowledge about the facts we can not do much about it. We could make sure that only such users with sufficient knowledge about the simulations topic use it or provide additional background information to help the user interpret the data correctly.

## 3.4. Classification of Uncertainty Representations

In this section we work out a way to classify the input methods we plan to develop. In *Section 2.3 – Uncertainty and Visualization* we have already seen some related work about uncertainty, presenting approaches to classify visualization methods. The approach for the classification of visualizations by using a taxonomy of visualization goals from Keller and Keller [KK93] does not work for us, because, in contrast to Keller and Keller, we have to create input methods for general data input purpose and therefore we can not respond to the type key information in the entered data. Brodlie [BCE<sup>+</sup>92] used two ways for classification, one of which is the distinction between ordinal or nominal data. We inspected if this way of

categorization also makes sense for the classification of input methods. Uncertain data are usually data in the form of numerical probability expressed as a single probability value or a probability distribution, which can be a purely ordinal construct. The percentages referring to the possibility of the occurrence of an event can easily be brought into an order. But there are also cases where data is to be treated nominal. If you can not tell if a colour is more green or blue this can not be expressed in an ordinal way. This is uncertainty in the form of vagueness. The distinction between ordinal and nominal is worth considering. L. Hesselink et al. [HPv94] use three characteristics to classify techniques. What they call the *order of data* is taken up and extended by Pang et al. [PWL97]. They call this *value of data* and we look into this shortly. As a second classification they use the *spatial domain of visualization objects*. Since many input methods do not contain a visual part we can not use this way of classification. But we may consider a more general approach and use the *type of representation* for classification. The *information level* of the data (elementary, local, global) is the third classification, which is also very closely related to visualizations and is therefore not suitable for general input method classification. Since Pang et al. took, amongst others, the classification techniques which we have just seen to extend them with respect to visualization in uncertainty, they are quite interesting for our purpose to categorize input methods. We investigate which of these techniques, that Pang et al. suggest, also work for input methods. First they define the *value of datum* and its associated value uncertainty, which extends the order of the data, introduced by Hesselink et al. [HPv94] and corresponds to the dependent variables of Brodlie's underlying field [BCE<sup>+</sup>92]. The *data value* can be characterized as a scalar, a vector, a tensor or a multivariate, which is also appropriate for the classification of input methods, because the input method design should fit the data value. The concept of classification by location of datum and its associated positional uncertainty specifies the dimensionality of the space in which the data resides and time. This corresponds to the independent variables of Brodlie's underlying field and specifies how many independent variables are used to describe each data value. This does also apply to input methods. The extent of datum location and value corresponds to the distribution, range, interval or period over which data is valid. When conducting a data input the data extent is usually known, but the input method itself mostly supports big and variable data extents. This classification makes sense for input methods, but we consider the possible data extent of the input method instead of the extent of a certain use case. The fourth way of classifying visualizations is the *visualization extent*. This determines if data is visualized in a deterministic or in a continuous way. The indication of individual datums is not restricted to visual representations and does also work for input methods without graphical elements. The last suggested classification is the kind of axe mapping. This is related too much to visualization and therefore we drop this.

Although the investigated classification techniques were originally designed for data visualizations some of them also work for the classification of input methods. Let's sum up the classification techniques that we can also use as a basis to develop classification techniques for input methods:

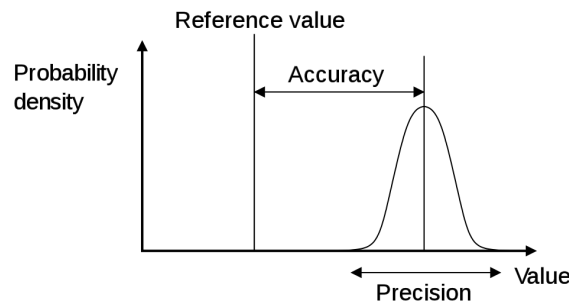
- ordinal or nominal
- deterministic or continuous
- data value (scalar, vector, tensor, multivariate)
- data location (dimensionality and time)
- data extent

When creating techniques for the classification of input methods for data with uncertainty we considered the techniques listed above to extend or adapt them. We step through the classifications we use in the following.

#### 3.4.1. Classification Techniques

**Absolute or Comparative:** Sometimes it is not necessary to know the precise probability value. It may be enough to know whether an event is more or less likely to occur than another one. An input method might give the ability to order events in ascending or descending order of chance of occurrence. Another possibility is to specify how much bigger or smaller the precise chance of an event occurrence is compared to a second event, which is even possible if the probability for the second event is not known. Therefore it is valid to consider input methods for the *comparative* input of uncertainty, at least for probabilities. It might also work for other forms of uncertainty, but this problem can be addressed when working on input methods design and purpose. We chose the expression *comparative* over *relative*, since we wanted to avoid the interpretation as a distinction between absolute values and percentages (relative). Input methods for absolute specification are the common form we usually encounter. This is also the form we want to focus on, since the direct specification values is usually easier and more flexible to process and to work with.

**Input Value Precision:** Looking at the example of ordering probabilities of event occurrences it is apparent that this way of probability specification is less accurate than entering percentage values, for example. Therefore input methods may be distinguished by their input precision. Precision must not be confused with accuracy. Even though we accept lower precision we still expect the input method to allow accurate inputs. To clarify this we briefly look at the difference between accuracy and precision. One definition for accuracy is the following: „The degree of agreement between a measured or computed value of a physical quantity and the standard or accepted value for that quantity“ [acc91]. This is not necessarily limited to physical quantities. Consider this definition for precision: „The degree to which the correctness of a quantity is expressed“ [pre10]. *Figure 3.5* helps to clarify this fact. We consider three levels of precision: *precise*, *close*, *imprecise*. *Precise* means that the exact value, or at least as close as possible, can be entered. If the input method allows an roughly precise value, which is still likely to be close to the correct value we call it *close*. If it is only possible to enter whole numbers, but the correct value is 1,5 we accept 2 to be close to the correct value, as it is the



**Figure 3.5.:** Accuracy and Precision in a Probability Density Function [Pek07]

result of rounding up 1, 5. If the distance between possible values is too high to either equal the values by rounding or the deviation is greater than 1% of the total data extent we consider the precision to be *imprecise*. The input value precision refers to the maximum degree of reachable precision for entered data. Since the used data extent and value interval is a parameter chosen by the user interface designer and not only limited by the design of the input method it is possible that an input method classified as *precise* delivers imprecise data if the input method configuration parameters have not been chosen wisely.

**Representation Type:** The *spatial domain of visualization objects*, introduced by Hesselink et al. [HPv94] references too much to the classification of visualizations to be applicable to classify input methods. But we generalize this to refer to the overall representation of input data and interactive elements. This becomes clear if we look at the three types of representation we consider here: *graphical*, *numerical*, *descriptive*. We have seen many visualizations for data with uncertainty in *Chapter 2 – Related Work* and some of them are also suitable to adapt for data input or to support the input with a supportive graphical visualization. There are numerous possibilities for *graphical* representation of data, especially when dealing with data referring to probabilities or statistics. The common way of visualizing a probability distribution is the probability density function plot. An interactive plot with support for mouse dragging and scaling can enable users to enter a probability distribution, for instance for a normal distribution, in a *graphical* way. But the common way of communicating and entering such data is in the form of numbers and values, what we call numerical representation. For instance a normal distribution can be specified by the input of two simple numbers, the mean and the variance. Such an input method is the *numerical* equivalent of the graphical probability distribution input method explained above. As we are often confronted with uncertainty in natural language and people often use expressions such as „quite possible“ or „likely“ we should also consider this way of entering uncertainties as worded expression, in a *descriptive* way. In addition to the representation of a single kind it is possible to combine representations. The specification of a normal distribution by the input of the mean and variance in a numerical way may be supported by a non-interactive probability density plot. So there is usually a main- and a supportive representation. When specifying the representation type we separate

representation parts of an input method of equal value by a forward slash and put supportive representations in brackets.

**Data Value Quantity:** The classification by value of datum, explained by Pang et al. [PWL97] is not suitable for the general classification of input methods. We can not accept the four characterizations *scalar*, *vector*, *tensor* and *multivariate*, since this can not be applied to all kinds of uncertainty. Let's look at an example of vagueness: The attempt of classifying a colour as green or blue can not be assigned to one of the four characterizations proposed by Pang et al. We therefore use a more general approach and make a distinction between *single value* and *multi-value*.

**Data Dimensionality:** Most values reside in the one dimensional space. When counting the amount of artifacts or defining the length of an object it requires one variable only. In contrast to that it requires two independent variables to specify the location on a surface or the position of a coordinate. These dimensionality often refer to spatial dimensions. However time is also an important factor which is often modelled as one extra dimension in addition to the spatial dimensions. While a datum of any finite dimensionality describes its location in the according  $N$ -dimensional space we usually correspond to the space (and time) of the real universe, in which we live. In mathematics and science other dimensional spaces occur, such as parameter spaces, which consists of all possible value combinations for the different parameters in a mathematical model. This may refer to a collection of independent attributes (parameters), each describing the property a certain artifact. For classification we use the amount of dimensions of the space the datum resides in.

**Value Set:** Pang et al. [PWL97] introduce the *visualization extent* as classification technique. The distinction between discrete and continuous has its justification, but it refers more to sets of data than to input values. If the input method corresponds to statistical data input and there is a corresponding visualization a classification as continuous or discrete makes sense. But for other application scenarios, such as the input of a single probability value, this does not fit. We could specify if we pick a value from a discrete or a continuous set. More precise is the specification of the set of possible input values. For numbers one can specify the number type, e.g. natural numbers  $\mathbb{N}$  or real numbers  $\mathbb{R}$ . For colours one can define an own set to specify the possible values.

**Data Extent:** The *extent of datum location and value*, described by Hesselink et al. [HPv94], corresponds to the distribution, range, interval or period over which data is valid. For example, if an input method serves the purpose to enter a probability value, and it is not restricted due to input method limitations, the according *data extent* interval is  $[0, 1]$ . According to the input method configuration parameters, chosen by the user interface developer, the *data extent* of each individual instance of an input method may vary. Therefore we define both, location and value of *data extent* individually. For the example above we define the *data extent* value as  $(0, 1]$  and the *data extent* location as  $[0, 1]$ . Since the interval can not be of zero size we define its *data extent* value interval minimum to be open. The defined intervals for data extent location and value mean that the input datum can reside in an interval with a size lying in the  $(0, 1]$



interval (*data extent* value) and with minimum and maximum value lying in the  $[0, 1]$  interval (*data extent* location). The specification of the data extent is obvious for numbers, but what about values of non-number sets? For this case a description in colloquial language should be used if another mathematical description is impossible or too cumbersome.

**Reference Class:** In *Section 3.1 – Humans Dealing with Uncertainty* we concluded due to the statement of Gigerenzer et al. [GHv<sup>+</sup>05], that when the interpretation of weather forecasts is imprecise, we have to take care to clearly specify the class the input references to. Therefore we classify the input methods depending on their reference class, so we can determine for which kinds of input values they fit. There is a great number of possible reference classes and we mention some, but not all options:

- Probability
  - Number between 0 and 1
  - Ratio
  - Percentage
- Generic Value
- Time
- Distance/Size/Length/...
- Quality
- Speed
- Colour
- Location
- ...

The *Probability* and *Generic Value* reference classes are used most often, because many cases of uncertainty belong to probabilities and *generic Single Value* inputs are very flexible in usage, e.g. they can be used to enter probabilities and percentages, too. Therefore *Generic Value* implies suitability for *Probability* values. Most reference classes, such as *Time*, *Quality* and *Colour* are very specific and probably end up in a very specialized input method. There are more possibilities for reference classes and the classification could be more finely, but the options presented above should cover the most application scenarios.

**Expressiveness:** Expressiveness is a classification measure that is related to effectivity. In *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations* we have already mentioned an example: The input of how often someone goes to work by car per month can be done as an average estimate or as a distribution (see *Figure 3.3*). The distribution provides more information and therefore is more expressive. We define expressiveness as a component of

effectivity. It describes the amount of information a user interface can provide and consequently influences the input uncertainty due to input limitations that lead to ambiguity and inaccuracy. For the case of uncertainty in the form of probability the expressiveness levels describe to what extent a distribution can be entered. For each uncertainty type the expressiveness levels have to be specified separately. We focus on the uncertainty type randomness and therefore defined expressiveness levels for distributions with one or multiple peaks. This covers important distributions, such as the normal distribution, which has one peak. The expressiveness levels for randomness uncertainty in the form of distribution are as follows:

1. Level: Single Value / Peak / Mode / Best Estimate / Average
2. Level: Range of fixed size
3. Level: Range of dynamic size (minimum and maximum)
4. Level: Range of dynamic size + peak(s)
5. Level: Range of dynamic size + peak(s) + 2 nodes per peak

This classification has been created explicitly for the comparison of input methods for randomness uncertainty type. Expressiveness levels for the other uncertainty types and reference classes are yet to be created. The higher the level the more precise a distribution can be specified. This is discussed more precisely in *Section 3.4.2 – Expressiveness*. If the input method is restricted to a certain amount of peaks this is specified in brackets behind the expressiveness level. For those input methods that do not allow a general input of distributions we alternatively specify the type of distribution which can be entered, e.g. normal distribution. Expressiveness must not be mistaken with utility, which measures how well a user can use this input method to achieve an input goal in practical use rather than the maximum possible value of the input a user can achieve with this input method.

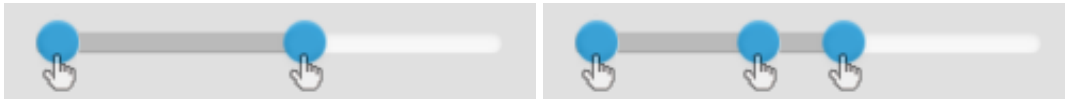
*Table 3.1 – Overview of Classification Techniques for Input Methods dealing with Uncertainty(Randomness)* shows an overview of the proposed classification techniques for input methods dealing with uncertainty, especially in the form of randomness.

#### 3.4.2. Expressiveness

Here we discuss the reasons for the expressiveness classification. A higher expressiveness level has some advantage over lower expressiveness levels, but this also comes with the price of potentially higher complexity, more required knowledge, and lower usability. This is a trade-off one has to consider when integrating different input methods into user interfaces. If the target group consists of specialists with sufficient background knowledge and who receive extensive training, a high expressiveness level may provide both, a high amount of information and good usability and user interface understanding. In contrast to that a user interface for laymen who do not receive any training likely perform better with input methods with lower expressiveness

Input Method Classification Techniques	
Name	Values
Absolute or Comparative	Absolute, Comparative
Input Value Precision	Precise, Close, Imprecise
Representation Type	Graphical, Numerical, Descriptive
Data Value Quantity	Single-Value, Multi-Value
Data Dimensionality	1D, 2D, 3D, ...
Value Set	$\mathbb{N}$ , $\mathbb{R}$ , RGB, ..., self-defined Set
Data Extent	Interval for Location + Interval for Value
Reference Class	Probability, Generic Value, Time, Size, Quality, Colour, ...
Expressiveness	Level 1 - Level 5, Distribution Type

**Table 3.1.:** Overview of Classification Techniques for Input Methods dealing with Uncertainty(Randomness)



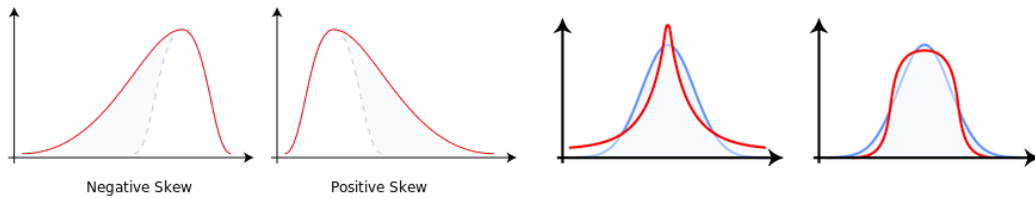
**Figure 3.6.:** Example Input Methods **Left:** Expressiveness Level 3, **Right:** Expressiveness Level 4

and therefore a potentially higher usability and easier understandability. Lower expressiveness levels lead to more uncertainty. An example should clarify this. The input of the two similar input methods (see *Figure 3.6*) with expressiveness level three (left) and four (right), can both be interpreted as a distribution. For the expressiveness level three example, only the lower and upper border is specified. Even if we know that the underlying distribution has one peak only, there are many possibilities. Distributions can differ in skew and kurtosis (see *Figure 3.7*). Since the input method with expressiveness level three does not provide information about the location of the distributions peak we can not tell if the according distribution has a positive or negative skew. Sometimes the lower and upper border is sufficient, but otherwise this leads to uncertainty about the correct distribution. Expressiveness level four allows the specification of the mean and therefore also the location of the peak of the distribution. Although there is no uncertainty due to unawareness of the skew it does not provide information about the distributions kurtosis (see right side of *Figure 3.7*). Input methods with expressiveness level five try to address this problem.

The expressiveness levels provide different amounts of information and requirements:

**Level 1:** *Single Value / Median / Best Estimate / Average*

The most basic approach provides, in terms of distributions, low information. When defining only the mean or mode there is no information about the value range. A positive aspect is that



**Figure 3.7.:** Different Forms of Distributions **Left:** Skew [Rod08], **Right:** High and Low Kurtosis [Chr09]

laymen may be able to use these kinds of input methods, even without training. We believe that it is more intuitive to ask people to enter the best estimate or mean instead of the mode.

**Level 2:** *Range of fixed size*

If the range size is already known, or the input for the user should be simplified and there is at least a good guess for the range size, this can take the work of finding the range size off the users' hands. We already mentioned the problem of the missing information of the distributions skewness and kurtosis, but for distributions with a probability density function that is symmetric around  $x = 0$  it is sufficient to know the lower and upper bound to determine the whole distribution. For a normal distribution it may be sufficient to consider an interval of  $\pm 3\sigma$  from the mean, which covers 99.73% of all measured values.

**Level 3:** *Range of dynamic size (minimum and maximum)*

If the range of the input can not be determined beforehand the user needs to have the ability to specify a range of dynamic size. Like the input methods with expressiveness level two, this one also suffers from the problem of not knowing about the distributions' skewness and kurtosis.

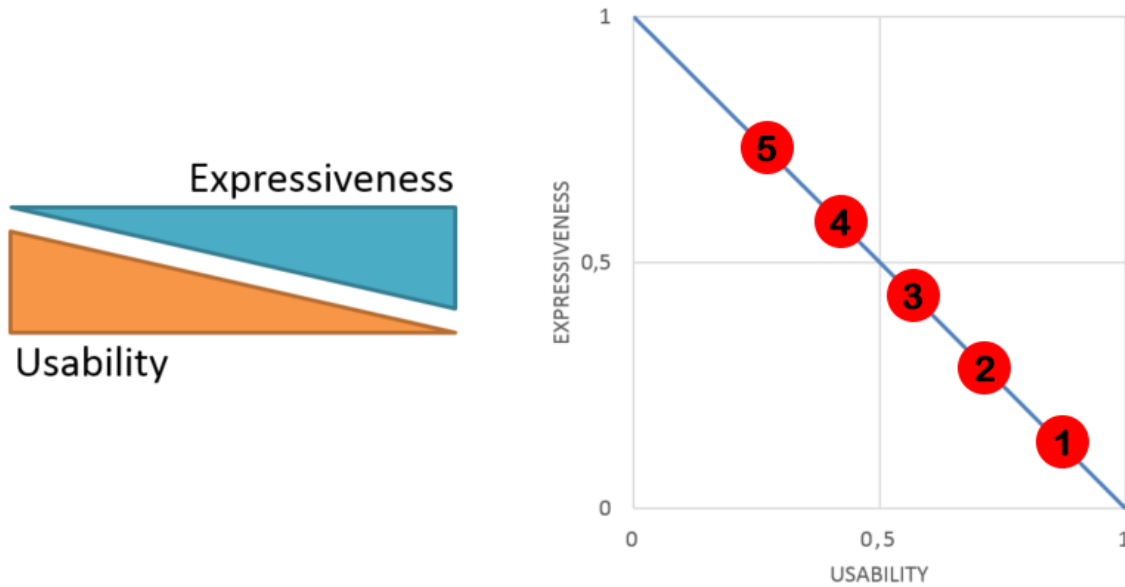
**Level 4:** *Range of dynamic size + peak*

To solve the problem of unknown skewness, we need to know the location of the distributions' peak. Therefore input methods with expressiveness level four require, in addition to the lower and upper bound, the input of the distributions' mode.

**Level 5:** *Range of dynamic size + peak + two nodes*

Since the expressiveness level four still provides no information about the kurtosis, level five input methods add two additional nodes, one on the left and one on the right of the mean. These nodes have the half height of the distributions' peak. This works well to specify the kurtosis of a single-peaked distribution, but for multiple peaks a problem arises if the height between two peaks does not fall under the half height of the higher peak.

The online survey and the user study show how well people can cope with input methods of different expressiveness levels and to what extent they influence the usability. We discuss this issue more precisely in *Section 3.4.2.1 – Expressiveness vs. Usability*.



**Figure 3.8.:** Expressiveness vs. Usability Estimate before Survey and Study

#### 3.4.2.1. Expressiveness vs. Usability

As we have already mentioned, we expect that the usability drops more the higher the expressiveness level gets. This is due to the higher required knowledge about probability distributions in general and about the data, a distribution in this case, that the user wants to enter. In addition to that the input method becomes more overloaded, because there are more input elements. We expect a linear relationship between expressiveness and usability (left hand of *Figure 3.8*). In this case the input methods should be ordered like on the right hand of *Figure 3.8*. Kenneth R. Mylne [Myl02] states that the value of a weather forecast depends on the application as well as on the skill of the forecaster. Applied to input methods this means that its value depends on the usability, which determines how well the user can work with it, and the effectivity, which determines the value of information. If the survey and study show that some of the input methods lie on top-right side of the graphs' diagonal blue line in *Figure 3.8* their usability is overproportional to their expressiveness. These are potential candidates for further extensions and developments. The distance between the blue line and input method, on the top-right side, determines the input methods' overall quality, concerning its usability and expressiveness. Consequently those input methods on the lower-left side of this line do provide a lower usability than their expressiveness suggests. We can use this to learn which aspects caused this for further improvements of promising input methods.

## 3.5. Input Method Metrics

In order to rate our input methods we need to measure qualitative attributes. The classifications in *Section 3.4.1 – Classification Techniques* are neither good nor bad, but provide a way to distinguish between different types of input methods. At first this does not seem to be obvious for some of the classification techniques. One could say that an input method with a broader data value extent is superior to one with a small extent. In this case we have to consider that for different inputs some input methods work better than others and these parts of an input methods that lead to a great data extent may be misleading and inappropriate. A greater data extent increases the flexibility, but this might affect how well it matches certain tasks. Therefore this is more a specialization vs. generalization trade-off than a qualitative comparison in flexibility. The same applies to precision. If an imprecise value is sufficient or one wants to avoid to induce the presence of a degree of precision, an imprecise input method may be superior in some cases. In addition to that imprecise input methods may grant new possibilities in design and representation. One special case is the expressiveness classification which we already discussed in *Section 3.4.2 – Expressiveness* and *Section 3.4.2.1 – Expressiveness vs. Usability*. We want to focus on usability and flexibility for qualitative measurement, while effectivity is covered partly by the expressiveness classification. Since effectivity depends on the context of usage a general effectivity rating beyond expressiveness could be misleading. For the purpose of determining the input methods usability we have a look at the work from Nigel Bevan [Bev08], which we have already inspected in *Section 2.5 – Input Methods*. Bevan introduces a table with factors that contribute to system usability and user experience (see *Figure 3.9*). We incorporate the information that is contained in this table.

**Effectivity and Efficiency** appears twice in this table. The first occurrence *To be effective and efficient* refers to the users' subjective perception of how effective and efficient one can work with the user interface. The other occurrence, *Effectiveness and Productivity in use: effective task completion and efficient use of time*, refers to objective measures for effectivity and efficiency. The question of how to objectively measure effectivity and efficiency arises. Effectivity determines if and to what degree the desired task, e.g. entering data, can be accomplished. This can be measured by correctness of input and amount of input cancellation, which indicates that it was not possible for the user to enter the data at all. Effectivity tells us something about the amount of effort one has to make to accomplish the goal. In this case we can measure the time needed and the amount of corrections and clicks.

**User Experience** is influenced by the design, the way of interaction and the look-and-feel. A users' confidence and satisfaction have a positive effect on the user experience. Therefore we consider *Experience of Interaction, Likability and Comfort* and *Trust* in this context.

**User satisfaction** can be measured by the extent to which users have achieved their pragmatic and hedonic goals. Nigel Bevan refers to *ISO/IEC CD 25010.2*, which suggests four types of measures for user satisfaction: likability, pleasure, comfort, trust. **Likability** and **Pleasure** are hard to measure objectively, since they depend on subjective perception. Therefore we choose

Quality characteristic	UX	Functionality	User interface usability	Learnability	Accessibility	Safety	
Product attributes	Aesthetic attributes	Appropriate functions	Good UI design (easy to use)	Learnability attributes	Technical accessibility	Safe and secure design	
UX pragmatic do goals	To be effective and efficient						
UX hedonic be goals	Stimulation, identification and evocation						
UX: actual experience	Visceral	Experience of interaction					
Usability (= performance in use measures)	Effectiveness and Productivity in use: effective task completion and efficient use of time			Learnability in use: effective and efficient to learn	Accessibility in use: effective and efficient with disabilities	Safety in use: occurrence of unintended consequences	
Measures of UX consequences	Satisfaction in use: satisfaction with achieving pragmatic and hedonic goals						
	Pleasure	Likability and Comfort				Trust	

**Figure 3.9.:** Factors Contributing to System Usability and User Experience [Bev08]

to determine these by questionnaire. This also applies to *Comfort*. Although *Confidence and Trust* appear to be very subjective there are some indicators for the degree of confidence. We have already mentioned the possibility of adding an input element for the users' confidence about the input. However, we can not tell to what extent this confidence applies to the lack of understanding and to what degree this uncertainty arises due to a lack of knowledge (compare *Figure 3.4* (1) and (2)). The users' request for help indicates lower confidence and a help button may give hints. Measuring how often and how long the additional help has been requested provides information about how confident a user is about his input. The request for different levels of help, such as a brief and a detailed description, are possible indicators, too. Extraordinary high numbers of input correction do also point to low confidence.

**Learnability** determines how well a user initially performs with a user interface and how fast and with what amount of effort users can learn to effectively and efficiently use it. This points to an important keyword, intuitiveness, which increasingly draws attention, especially in the context of web-based interaction. Training users in the use of user interfaces is expensive in effort and time and it is not always possible to provide the required training due to resource-based, distance and communication problems. In addition to that, users are usually impatient when working with user interfaces in browsers or computer programs. Interaction with non-intuitive input methods are likely to be aborted fast. Since our input methods are created for a web-based simulation tool we have to keep an eye on that. We hope that users accept some amount of required training if they consider the context of simulations. To measure learnability the users' own assessment can be valuable. For objective measures we compare the values of

effectivity and efficiency of the first use to those of the interaction after multiple uses and after additional training or provision of more detailed descriptions.

**Flexibility** is described as a measure of the extent to which the product is usable in all potential contexts of use, including accesibility [PWL97]. Accesibility is considered as the degree of usability in use, in contexts beyond those initially intended, but we want to extend this to also cover the usability of users with different degrees of background information. In other words: Does the input method work for experts only or even for laymen? We can measure this by comparing usability values of people with different background knowledge. We expect especially learnability to be greatly affected by this. The other component of flexibility is its usability in different contexts. It is desirable that an input method can be used in many different contexts. The classification techniques *Data Extent* and *Value Set* play a role here, too. However, in this connection we have discussed that this is likely to be a generalization vs. specialization trade-off and since the use case of our interest is in the context of a simulation this component provides interesting indications, but we give this less weight for qualitative classification.

**Safety** is also mentioned in *Figure 3.9*. High safety provides an acceptable level of risk of harm to people, business, data, software, property or the environment in the intended context of use [PWL97]. Risk of harm in data and business is, in our context, probably the most important. This refers to the decisions a user might make because of simulation results. Since we have no influence on the simulation correctness from the position of input method design. Input methods that are easy to understand, and therefore usable, help to minimize these risks. Therefore, in the context of input method design, only usability influences safety in general.

## 3.6. Summary and Discussion

We discussed the influence of the users' prior knowledge and education on the best choice for the most appropriate input methods. It seems to be a sound strategy to adapt the input methods complexity to the abilities and education of the user. Statistics about the education obtained in school suggest that we have to go for rather simple approaches quite often, if it has a significant influence on the usability. We considered the argument of Gigerenzer et al. [GHv<sup>+</sup>05] that the bad performance people show in the context of uncertain data refers to the missing or at least unclear specification of the reference class. We have revealed different sources of uncertainty that may occur when dealing with simulations, whereas not all of them refer to the intrinsic uncertainty in the in- and output data. We determined those sources of uncertainty that are relevant for user data input scenarios and discussed ways to deal with them. We reply some of those threats with a high usability and the choice of appropriate input method configuration parameters. This also arised the problem of the input methods' expressiveness in the context of probability distributions. We discussed why we expect the usability to suffer from a high degree of expressiveness. To be able to differ from various degrees of expressiveness



we defined five expressiveness levels. This is one of the nine classification techniques we suggest for the classification of input methods for data with uncertainty. At last we discussed usability measures for input methods on the basis of an article about system usability and user experience [Bev08]. We take up these reflections in the following chapters.



## 4. Input Method Prototypes

In this chapter we discuss the design of the developed input methods for data containing uncertainty. We only present a selection of input methods and classify them according to the classification technique presented in *Section 3.4.1 – Classification Techniques*. Since we want to go for input methods that are also accessible for laymen, we start out with simple input methods and try to approach appropriate concepts from this side. We also pick up common web-based input methods and classify them to found a basis for further developments and for comparison. In some cases we skip the expressiveness level two approaches, because the results did not appear to be satisfying.

Due to the increasing importance of mobile devices, such as smartphones and tablets, which provide new and different forms of user interface interaction, we have to think about the input methods suitability for input techniques such as touch, accelerometer and others. Schwarz et al. [SHMW] reveal that these input techniques often suffer losses in input accuracy and therefore introduce an unnecessary source of uncertainty (see *Figure 3.4 – (2)* and *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*), which we want to avoid for better evaluation.

### 4.1. Assembly and Classification of Input Methods

In this section we present, classify, and briefly describe a selection of input methods. We brainstormed and developed a bunch of options and made the selection so that it includes input methods of different classifications on the one hand and covers those input methods that led to the development of our most promising, preselected input methods on the other hand.

#### Simple Slider

The simple slider is a web-based input method that should be well-known by most people. It appears on many websites and is usually applied when the minimum and maximum possible value is known. Therefore this is often utilized to enter percentages in

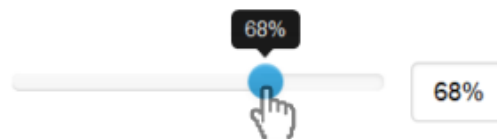


Figure 4.1.: Simple Slider (Percentage) Example

an interactive way. Also for other scenarios, where the lowest and largest possible value is known, sliders can be used, like for example for the specification of body height. Consequently, sliders are suitable for the input of both, generic values and percentages and suits for the input of arbitrary small and large numbers in a clearly defined interval of arbitrary size. The visual representation helps the user to get a feeling for the ratio that lies behind the input. This is just an example for any possible web-based slider and in this case we used a bootstrap slider [Ste].

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Precise	Graphical (Numerical)
Data Value Quantity	Data Dimensionality	Value Set
Single-Value	1D	$\mathbb{R}$
Data Extent	Reference Class	Expressiveness
location: $(-\infty, \infty)$ , extent: $(0, \infty)$	Probability(Percentage) / Generic Value	Level 1

**Table 4.1.:** Classification of *Simple Slider* Input Method

### Ratio Selector

This input method is derived from star rating inputs that are very commonly used in websites nowadays. Actually a star rating is nothing else but the specification of the amount of stars, from a maximum amount of possible stars, one assigns to an artifact. This is, indeed, the specification of a ratio and therefore we pick this up to extend it. In

*Chapter 2 – Related Work* we saw that in some cases the representation of probability as ratios provides significant advantages over percentage representations. But there are disadvantages for this kind of visual input method. The simple size of the glyphs limits the data extent. In addition to that, the denominator of the fraction can not be changed. In some input scenarios this might reduce the accuracy of the input, for example if the denominator is 5 and the probability is  $\frac{2}{6}$ . However, for certain scenarios and with the correct choice of the denominator this input method can be absolutely precise. Like for the *Simple Slider* the visualization helps the user to envision the meaning of the specified ratio. This implementation is based on Krajee’s Bootstrap Star Rating [Kra].



**Figure 4.2.:** *Ratio Selector* Example

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Precise/Close	Graphical (Numerical)
Data Value Quantity	Data Dimensionality	Value Set
Single-Value	1D	$\mathbb{Q}$
Data Extent	Reference Class	Expressiveness
location: $[0, 10]$ , extent: $[1, 10]$	Probability(Ratio)	Level 1

**Table 4.2.:** Classification of *Ratio Selector* Input Method

### Extended Ratio Selector

Addressing the problem of the fixed denominator of the *Ratio Selector*, we extended the input method to enable the user to change both, the numerator and the denominator. We considered different ways of extending the *Ratio Selector*. One possibility is to directly modify the denominator with the help of two buttons („add“and „remove“) or by dragging the right border of the last glyph. Another possibility is to completely drop the visual interaction part of the input method and to provide two own number inputs for the numerator and the denominator. This alternative has the advantage that the problem of size of glyphs for a high amount of denominators does not apply for purely numerical representations and therefore a higher data extend is possible. Here we present a mix of these two alternatives. We keep the visual part but the interaction is restricted to the input boxes (see *Figure 4.3*).



**Figure 4.3.:** *Extended Ratio Selector* Example

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Precise/Close	Numerical (Graphical)
Data Value Quantity	Data Dimensionality	Value Set
Multi-Value	1D	$\mathbb{Q}$
Data Extent	Reference Class	Expressiveness
location: $[0, 10]$ , extent: $[1, 10]$	Probability(Ratio)	Level 1

**Table 4.3.:** Classification of *Extended Ratio Selector* Input Method

### Ratio Range Selector

In order to expand the input possibilities the *Ratio Range Selector* provides the possibility to specify a range of possible values for the numerator. By setting the minimum and maximum of the range to the same value it can be used in the same way as the *Extended Ratio Selector*. *Figure 4.4* shows the visual realization of this input method. Specifying a range



**Figure 4.4.:** *Ratio Range Selector* Example

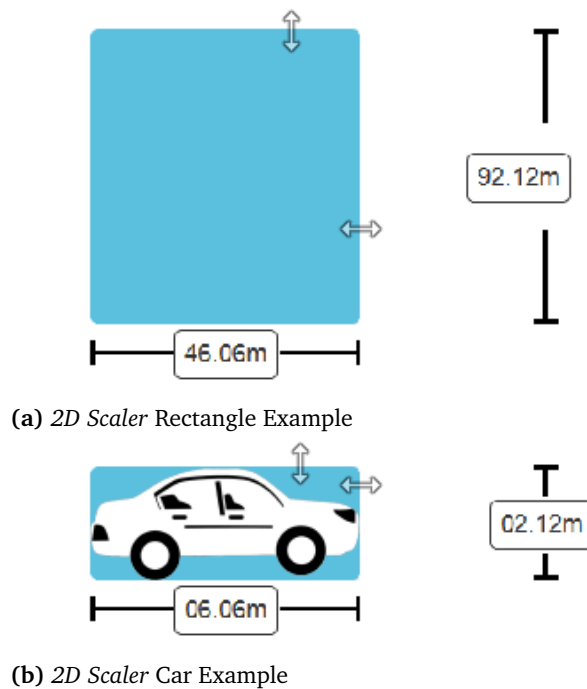
of possible values for the numerator, a problem strikes attention: The input of uncertain information about a probability leads to multiple uncertainties one has to deal with. Cumulative uncertainties can reach a high degree of complexity and therefore might be too complicated for many people to use confidently. That is why we do not go beyond level three in this case.

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Precise/Close	Graphical (Numerical)
Data Value Quantity	Data Dimensionality	Value Set
Multi-Value	1D	$\mathbb{Q}$
Data Extent	Reference Class	Expressiveness
location: $[0, 10]$ , extent: $[1, 10]$	Probability(Ratio)	Level 3

**Table 4.4.:** Classification of *Ratio Range Selector* Input Method

## 2D Scaler

A *Simple Slider* can be used to specify the length of an object or a distance, either by directly modifying its size in a fixed unit, or by specifying a scale factor. To enter the size of a surface one usually needs to enter 2 values, except for special cases, such as the surface of a circle (see *Figure 4.5a*). In some cases the length and width are not independent from each other. If one wants to specify the height and length of a car a fixed ratio between those values can be expected. Enabling the user to specify the size either by modifying the cars length or height can help the user to find the desired input. Remember, that we are working with uncertain data and in this case the user is trying to input a guess or a best estimate. The ability to enter the data in different ways may help to receive more reliable data. This input method has been designed to also provide the functionality of independent values for the two dimensions. The visualization supports the user in envisioning the meaning of the input. The square form can be overlaid with an image to even enforce this envisioning effect (see *Figure 4.5b*).



**Figure 4.5.:** 2D Scaler Examples

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	2D	$\mathbb{R} * \mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty) * [0, \infty)$ , extent: $[0, \infty) * [0, \infty)$	Size	Level 1

**Table 4.5.:** Classification of 2D Scaler Input Method

### 2D Range Scaler

In contrast to the *Ratio Selector* this kind of input method does not refer to probability values and therefore we do not encounter the problem of cumulative uncertainties. The user can specify a lower and an upper bound for both, the x- and the y-axis. A fixed ratio between the axes can be set or omitted independently for the lower and the upper bound. The area below the lower bounds is visualized in a different colour to communicate that this area is valid in any case. The rest of the area is uncertain and therefore receives no new colouring.

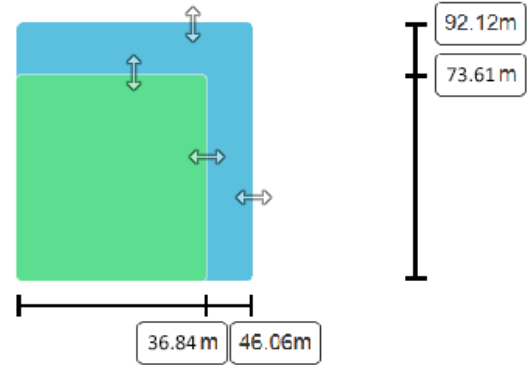


Figure 4.6.: 2D Range Scaler Example

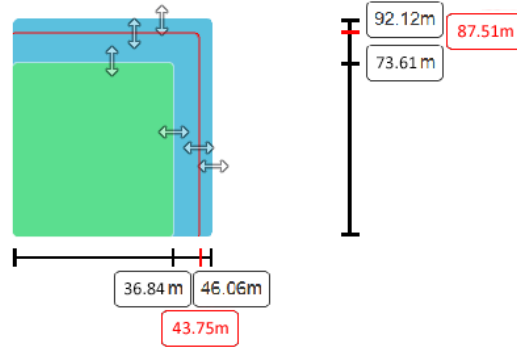
Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	2D	$\mathbb{R} * \mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty) * [0, \infty)$ , extent: $[0, \infty) * [0, \infty)$	Size	Level 3

Table 4.6.: Classification of 2D Range Scaler Input Method



## 2D Range Best Estimate Scaler

With the expressiveness level four we add an input element for a best estimate, median, mean or mode value for both axes. The according type of this input element value has got to be clearly defined to not cause unnecessary confusion. This can be done in a text close to the input method or in the input method itself, e.g. next to the box containing the value (red boxes in *Figure 4.7*). With the help of this input method we can try to interpolate the probability density function of an underlying two dimensional distribution. Two dimensional distributions might be too complicated for laymen to work with and therefore the approach of providing a plot of such functions is dropped here. The inclusion a distribution visualization as colour gradient, as described in *Section 4.4.1 – Distribution as Colour Gradient*, does also not work for this input method due to its two-dimensionality and the unsuitable shape of the uncertain area.



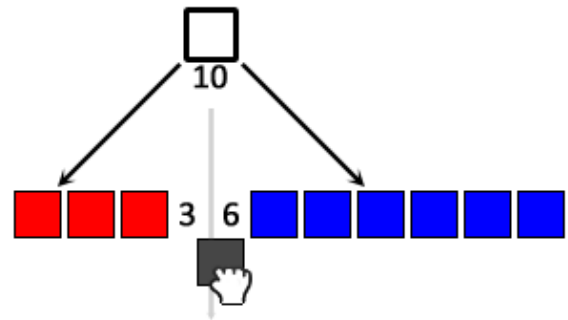
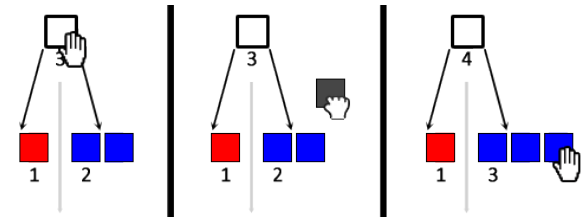
**Figure 4.7.:** *2D Range Best Estimate Scaler Example*

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	2D	$\mathbb{R} * \mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty) * [0, \infty)$ , extent: $[0, \infty) * [0, \infty)$	Size	Level 4

**Table 4.7.:** Classification of *2D Range Best Estimate Scaler* Input Method

## Ratio Drag And Drop

Hoffrage et al. found that for the communication of risk in cancer screening tests people perform better in interpreting natural frequencies than percentages [Hof00]. Consequently we want to provide input methods that support users in entering data with a yes-no or good-bad relationship. *Figure 4.8a* shows the *Ratio Drag And Drop* input method which has been designed to do exactly this. Take the example of a user who is asked to specify how many people out of ten will develop cancer. This input method colours the negative outcome in red and the positive in blue and opposes the proportions directly to help the user to compare them. By dragging and dropping the user has the explicit feeling of removing one unit from the one side to assign it to the other. The total amount glyphs are summarized by the number under the glyph. Since users can not overlook too many elements and they take too much screen size, the glyphs are summarized, as you can see for the total element amount in *Figure 4.8a*.

(a) *Ratio Drag And Drop* Example(b) *Ratio Drag And Drop* Change Total Amount Example**Figure 4.8.:** *Ratio Drag And Drop* Examples

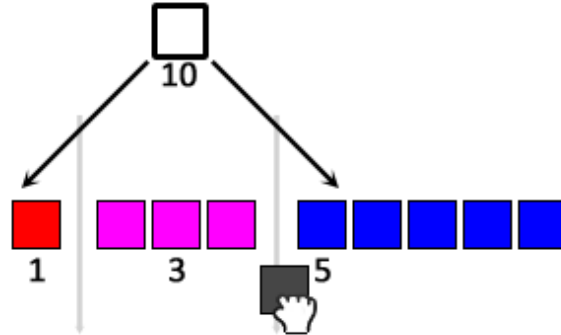
For the modification of the total amount we made up the following approach: The user can just drag another glyph from the total amount area and drop it to one of the two sides (see *Figure 4.8b*).

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{Q}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty]$ , extent: $[2, \infty]$	Probability(Ratio)	Level 1

**Table 4.8.:** Classification of *Ratio Drag And Drop* Input Method

### Ratio Range Drag And Drop

When specifying a range of possible values there are values that we can not certainly assign to one of the two sides (red and blue side for the example in *Figure 4.9*). Assuming that the user drops the dragged element to the blue side the input can mean either a 1 : 9 ratio, if the three elements in the middle belong to the blue, right side, or a 4 : 6 ratio, if they belong to the red, left side. This expressiveness level three input differs from most other approaches, where the user usually directly manipulates the borders of the possible ranges instead of modifying its position and size by transferring elements from a certain area to the uncertain area. Without further explanation this input method might cause confusion because there are three numbers displayed at the bottom while there are only two numbers describing the ratio of interest.



**Figure 4.9.:** *Ratio Range Drag And Drop Example*

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{Q}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty]$ , extent: $[2, \infty]$	Probability(Ratio)	Level 3

**Table 4.9.:** *Classification of Ratio Range Drag And Drop Input Method*

Ratio Range Best Estimate Drag And Drop

Trying to add an input element for the best estimate was a hard task. Highlighting one element in the uncertain area does not clarify if this element is to be assigned to the left or the right side if the best estimate applies. This has to be clarified and we solved this problem by defining the best estimate glyph to be assigned to the left side. Additionally the best estimate ratio is displayed as numbers to make this even clearer. Since the input of this new input element works different than the other interactions, namely dragging instead of drag-and-drop, it appears to be a bit misplaced and confusing. If one has already seen the *Ratio Drag And Drop* input method it is clear how to use this one, but without prior knowledge it is likely to be quite unintuitive.

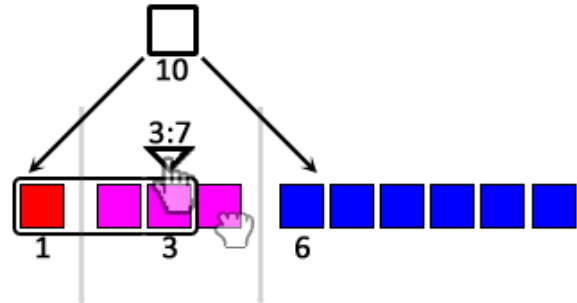


Figure 4.10.: Ratio Range Best Estimate Drag And Drop Example

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{Q}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, \infty]$ , extent: $[2, \infty]$	Probability(Ratio)	Level 4

Table 4.10.: Classification of *Ratio Range Best Estimate Drag And Drop* Input Method

## Term Selector

Wallsten et al. [WBR<sup>+</sup>86] talk about the interpretation of probability terms. Since people are used to express with the help of such terms an according input method makes sense. The user can choose amongst a pre-selection of terms to specify the input. The focus lies more on the choice of provided options than on the input method design. There are already many kinds of web-based selection input methods and they do all provide different advantages and disadvantages. For this example we chose a simple dropdown selector. The selection of terms can be varied in different scenarios and is highly debatable. People have a different understanding of such terms and therefore their usage has to be evaluated carefully. A simple selector with terms can also be used for other reference classes, such as quality or speed. Such terms are, due to their subjective interpretation with multiple possibilities, quite imprecise and it has to be well defined what they refer to.

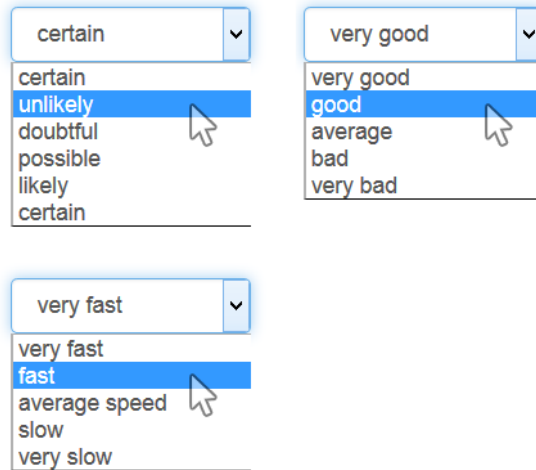


Figure 4.11.: *Term Selector* Example

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Imprecise	Worded
Data Value Quantity	Data Dimensionality	Value Set
Single-Value	1D	Set T of all possible terms describing the reference class
Data Extent	Reference Class	Expressiveness
location: T, extent: T	Probability, Quality, Speed, ...	Level 1

Table 4.11.: Classification of *Term Selector* Input Method

### Term Range Selector

When defining a range of values for probabilities the contained information may become very imprecise. Other reference classes seem to work better, but this is to be evaluated. The advantage that people are used to express things like probability, quality and speed in natural terms comes with the price of precision and when specifying a lower and an upper bound this uncertainty can sum up to the double extent in the worst case. In natural language it is common to define a range of possible values when not sure, but one usually does not add a best estimate. That's probably why approaches for an input method with expressiveness level four appeared to be odd and unnatural.



Figure 4.12.: Term Range Selector Example

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Imprecise	Worded
Data Value Quantity	Data Dimensionality	Value Set
Multi-Value	1D	Set <b>T</b> of all possible terms describing the reference class
Data Extent	Reference Class	Expressiveness
location: <b>T</b> , extent: <b>T</b>	Probability, Quality, Speed, ...	Level 3

Table 4.12.: Classification of Term Range Selector Input Method

## Quality Thumb Rotator

An alternative to the specification of quality with the help of a *Term Selector* is the usage of a *Simple Slider* to specify the quality as a value from 0.0 to 1.0. This approach is commonly used in the web and it has the advantage to allow step-less specification of quality. This works the same way as specifying any other value with the help of a *Simple Slider*. Therefore we use another approach that involves colours and symbols. Green colours are usually associated with the term „good“, while red is usually interpreted as „bad“. A symbol that people use to call something good or to mark something they like, e.g. in social media, is the thumbs-up symbol. Analogous people use a thumbs-down symbol to communicate their dislike. This can be used to create an input method for the specification of quality. *Figure 4.13* shows an input method for step-less specification of quality that is supported by colours and symbols. Since we wanted to avoid the interpretation of quality as a number we avoided displaying the value, in the interval  $[0, 1]$ , that lies behind the input.



**Figure 4.13.:** *Quality Thumb Rotator Example*

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Close	Graphical
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, 1]$ , extent: $[0, 1]$	Quality	Level 1

**Table 4.13.:** Classification of *Quality Thumb Rotator* Input Method

### Quality Thumb Range Rotator

There are several problems with the specification of a quality range for this input method. We need one thumb for both, the minimum and the maximum value, which overlay. The coloured background marks the area of possible values and the colour gradient supports the user in predicting the meaning of the quality range. Since a sufficient usability of this input method is quite questionable we did not add the possibility to specify a best estimate.



Figure 4.14.: Quality Thumb Range Rotator Example

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Close	Graphical
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[0, 1]$ , extent: $[0, 1]$	Quality	Level 3

Table 4.14.: Classification of Quality Thumb Range Rotator Input Method



### Simple Location Range Picker

The goal of picking a discrete location can be accomplished by using existing input methods. Their usage is not different when specifying a best estimate. If users are not sure about a location, they might want to enter an area where the correct location lies in. The red area in *Figure 4.15* can be dragged by the user to find a best fit for the fixed radius so that it covers the desired area. Since the radius has fixed size the covered area might be too big, which introduces unnecessary inaccuracies, or too small, which might cause the area to not cover all possible locations.

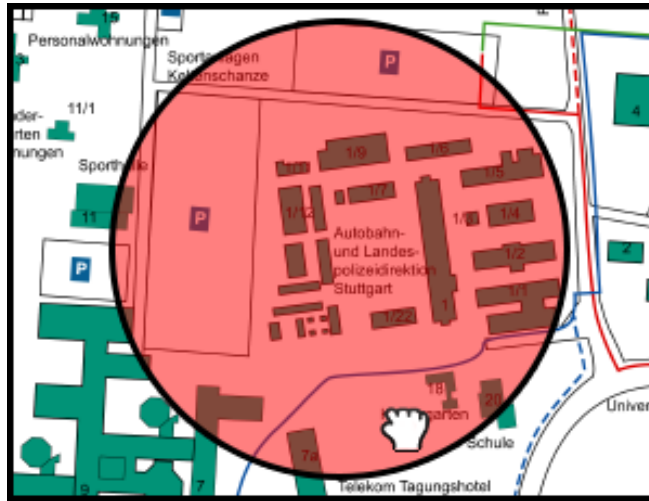


Figure 4.15.: Simple Location Range Picker Example

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	2D	$\mathbb{R} * \mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-90, 90] * [-180, 180]$ , extent: $(0, 180] * (0, 360]$	Location	Level 2

Table 4.15.: Classification of Simple Location Range Picker Input Method

## Location Range Picker

To enable the user to set the size of the area we thought about two approaches. One possibility is to let the user freely paint the borders of the area. If people are asked to mark such an area, where a desired location lies in, they usually draw a circle. Therefore, and to simplify the interaction, we chose the other approach, namely to specify the center and the radius. The radius can be changed by scaling the circle borders and the center can be dragged. Here we want to mention that this input method can also be used for a colour picker when a colour on a colour circle is to be chosen.

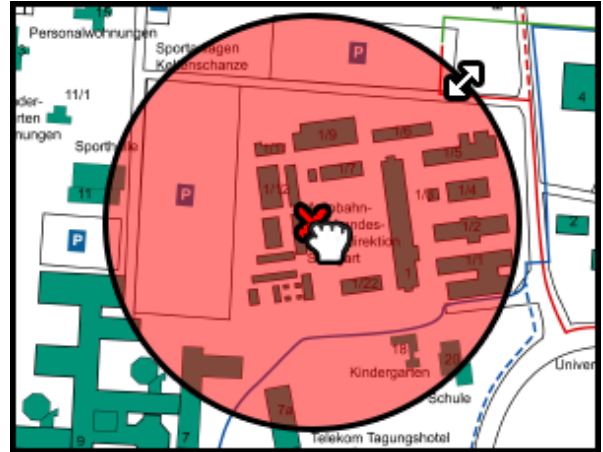


Figure 4.16.: Location Range Picker Example

Classification		
Absolute/Comparative	Input Value Precision	Representation Type
Absolute	Precise	Graphical
Data Value Quantity	Data Dimensionality	Value Set
Multi-Value	2D	$\mathbb{R} * \mathbb{R}$
Data Extent	Reference Class	Expressiveness
location: $[-90, 90] * [-180, 180]$ , extent: $(0, 180] * (0, 360]$	Location	Level 3

Table 4.16.: Classification of Location Range Picker Input Method

### Location Likely Range Picker

The expressiveness level four usually adds another single value as a best estimate. In this case we considered adding another point to specify within the area or to add another circle where the best estimate lies on. We also came up with a third option, namely to define a smaller radius with a higher probability. This way we can interpolate a 2 dimensional distribution where the peak is in the middle and the kurtosis depends on the radius of the inner circle. The other approach, where the user sets a single best estimate location, allows the generation of a distribution with a variable peak, but we do not have information about its kurtosis. It is up to the input interface designer to decide which approach to choose.

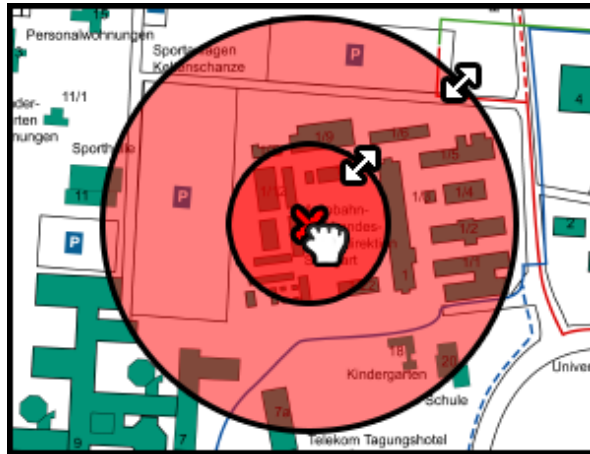


Figure 4.17.: Location Likely Range Picker Example

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	2D	$\mathbb{R} * \mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-90, 90] * [-180, 180]$ , extent: $(0, 180] * (0, 360]$	Location	Level 4

Table 4.17.: Classification of Location Likely Range Picker Input Method

### Bar Slider

Similar in usage, compared to the *Simple Slider*, the *Bar Slider* provides a different look and feel. The design is inspired by bar charts. For choosing percentages it gives a better visual support for determining the size of the chosen proportion. This input method also

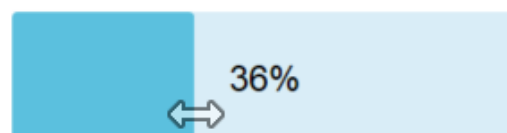


Figure 4.18.: Bar Slider Example

#### 4. Input Method Prototypes

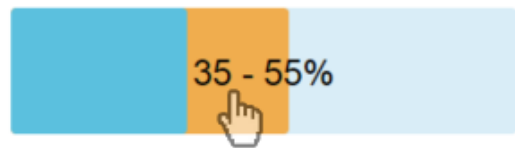
works well for the input of generic values and visualizes the value location in the set of possible values.

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-\infty, \infty]$ , extent: $(0, \infty]$	Probability(Percentage) / Generic Value	Level 1

**Table 4.18.:** Classification of *Bar Slider* Input Method

#### Fixed Range Bar Slider

One advantage of defining a range over specifying one best estimate value is, that it is obvious that the input contains uncertainty. For input methods with expressiveness level one, the uncertainty of the input value has to be pointed out, since it is an uncertain guess or the most probable outcome. If the range size is chosen wisely, or it is already known, the input can be valuable and precise.



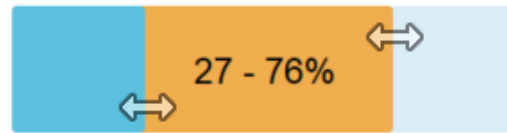
**Figure 4.19.:** *Fixed Range Bar Slider* Example

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Single-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-\infty, \infty]$ , extent: $(0, \infty]$	Probability(Percentage) / Generic Value	Level 2

**Table 4.19.:** Classification of *Fixed Range Bar Slider* Input Method

### Dynamic Range Bar Slider

The problem of the *Fixed Range Bar Slider* is, that the predefined range does not match the needed size in most cases. *Figure 4.20* shows the *Dynamic Range Bar Slider* which allows the free specification of the lower and the upper bound. For distributions with symmetric probability distribution functions, such as a normal distribution, this input method is sufficient to completely define the distribution.



**Figure 4.20.:** *Dynamic Range Bar Slider Example*

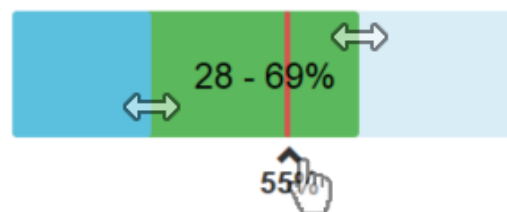
In statistics, normal distributions play an important role and by predefining the interval to  $[-3\sigma, 3\sigma]$ , for example, the user can easily specify the distribution.

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-\infty, \infty]$ , extent: $(0, \infty]$	Probability(Percentage) / Generic Value	Level 3

**Table 4.20.:** Classification of *Dynamic Range Bar Slider* Input Method

### Dynamic Range Bar Best Estimate Slider

For non-symmetric distributions we need more information to determine the peak. By dragging the best estimate element the user can set the distributions peak. Which colours to use for the different graphical input elements depends on the purpose of the input and on the context. The colours in *Figure 4.21* could lead to problems for people with red-green blindness and other variations might not provide sufficient contrast or lower the visual likability. Choosing the correct colours is up to the input interface designer and depends on the context and purpose.



**Figure 4.21.:** *Dynamic Range Bar Best Estimate Slider Example*

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-\infty, \infty]$ , extent: $(0, \infty]$	Probability(Percentage) / Generic Value	Level 4

**Table 4.21.:** Classification of *Dynamic Range Bar Best Estimate Slider* Input Method

We developed a bunch of other input methods, which we do not discuss here in detail, such as a percentage chooser with a pie chart visualization, visual time and date pickers, inputs for generic values with different visual appearance and precision, size input methods for different amounts of dimensions, probability and likability input methods using different colour and glyph representations. The presented input methods above are a selection of the most promising candidates for reliable inputs.

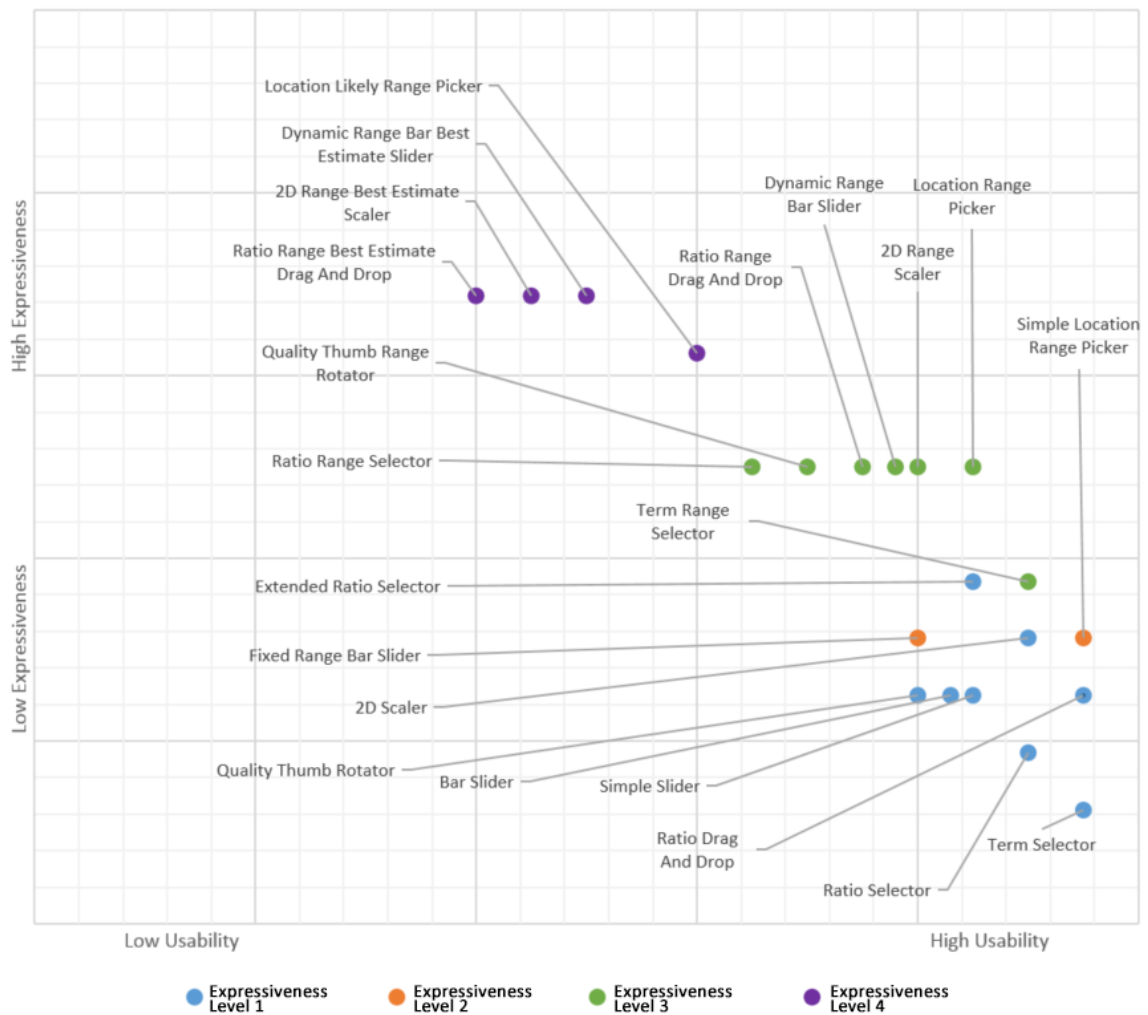
## 4.2. Expressiveness vs. Usability Estimation

In *Section 3.4.2.1 – Expressiveness vs. Usability* we have already pointed out that we expect the usability to drop lower the higher the expressiveness level gets. After briefly testing the input methods, with the help of some volunteers from our personal environment, we created an overview of our usability vs. expressiveness estimations. These are no representative or reliably collected data, but they still give hints about the value of the input methods. *Table 4.22* contains some comments about the reasons for their placement in *Figure 4.22*. Input methods with the same expressiveness level are displayed with the same colour.

Input Methods Usability and Expressiveness Comments	
<b>Simple Slider</b>	High usability due to familiarity with this input method
<b>Ratio Selector</b>	High usability due to familiarity of star rankings and supportive visualization, lower expressiveness because only numerator can be modified
<b>Extended Ratio Selector</b>	Higher expressiveness due to the possibility of modifying the denominator
<b>Ratio Range Selector</b>	The amount of input elements and the specification of a range seem confusing, supportive visualization
<b>2D Scaler</b>	Little abstraction required, intuitive and easy interaction
<b>2D Range Scaler</b>	A bit complicated at first, but easy to use after short explanation
<b>2D Range Best Estimate Scaler</b>	Appears overloaded, problems with interaction when minimum, maximum and best estimate are close
<b>Ratio Drag And Drop</b>	Supportive visualization and colours, intuitive drag-and-drop interaction, uses ratios instead of percentage which increases usability
<b>Ratio Range Drag And Drop</b>	Looses some intuitivity, but works quite well after short explanation
<b>Ratio Range Best Estimate Drag And Drop</b>	Low intuitivity, inconsistent interaction
<b>Term Selector</b>	Quite imprecise, but easy to use due to usage of natural terms
<b>Term Range Selector</b>	The same goods and bads as the Term Selector
<b>Quality Thumb Rotator</b>	Supporting user with the help of colours and glyphs, possibly precise input, unfamiliar at first
<b>Quality Thumb Range Rotator</b>	Seems a bit overloaded, because thumbs overlay
<b>Simple Location Range Picker</b>	Easy to use, intuitive, familiar
<b>Location Range Picker</b>	Quite intuitive and easy
<b>Location Likely Range Picker</b>	After short explanation quite easy to use, provides a bit less information that other input methods with expressiveness level 4, because the position of the inner circle is bound to the outer circle center
<b>Bar Slider</b>	Very similar to the Simple Slider, slightly better visual support, a little less familiar
<b>Fixed Range Bar Slider</b>	Easy to use, drawbacks of fixed range size
<b>Dynamic Range Bar Slider</b>	Quite easy to use and seems familiar because it is comparable to existing slider variations
<b>Dynamic Range Bar Best Estimate Slider</b>	More complicated and less intuitive to use

Table 4.22.: Input Methods Expressiveness vs. Usability Estimation

## 4. Input Method Prototypes



**Figure 4.22.:** Expressiveness vs. Usability Estimation for presented Input Methods after brief Testing

### 4.3. Discussion and Preselection

If we inspect the positioning of the input methods in *Figure 4.22* it is striking that the following input methods showed a higher usability than other input methods of the same expressiveness level: *Simple Location Range Picker*, *Location Range Picker* and *Location Likely Range Picker*. This makes them good candidates for further inspection and testing. However, they work for the specification of locations only and therefore it is a more specialized input method. Lipkus et al. [LSR01] found that people deal better with frequencies than with probabilities. Consequently the *Ratio Drag And Drop* input method, the *Ratio Selector* and their advancements are of interest. The increase in usability, due to the usage of ratios instead of percentages, is

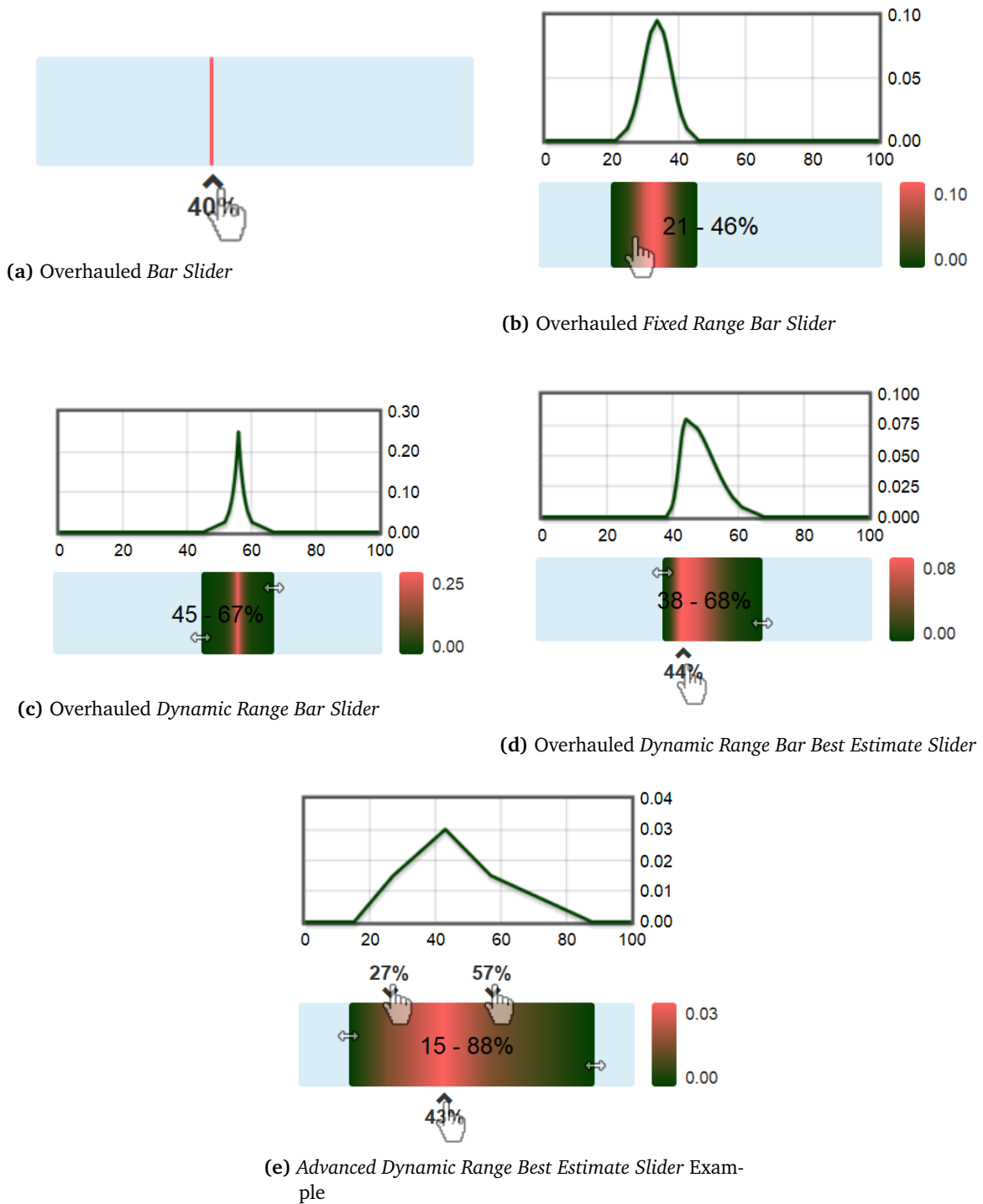


not as high as expected and in addition to that these input methods can only be used for the input of probabilities. We want to focus on the *Bar Slider* and its followers because they are similar to the well known *Simple Slider* and they are suitable for the input of both, probabilities generic values. In addition to that the *Dynamic Range Bar Best Estimate Slider* shows a higher usability than the other input methods with the same expressiveness level, except the *Location Likely Range Picker*.

#### 4.4. Upgrading most promising Input Methods

With the *Bar Slider* and its followers chosen, we inspected these input methods concerning their potential of improvement and their negative aspects. What we did not like about the input methods, except for the *Bar Slider*, was the coloured area on the left of the marked range. It does not provide any additional information and distracts from the actual important area, the value range. It may also lead to the interpretation of two proportions which have to be chosen. Therefore we removed the leading bar, which led to a different look and feel for the *Bar Slider* (see *Figure 4.23a*). In this case the unchanged input method might be preferred over the overhauled one, but for consistency aspects this fits better to the other input methods. In addition to that it looks more different to the *Simple Slider* and therefore it is more unlikely to interpret the interaction as the specification of a concrete value instead of the input of a best estimate or guess. The classification for these input methods is the same as for their unoverhauled counterpart. *Figure 4.24* shows how they look like after the enhancements.

#### 4. Input Method Prototypes



**Figure 4.23.:** Overhauled Input Methods

We made some enhancements to visualize the underlying probability distribution function. We discuss them in the following section.

### Advanced Dynamic Range Best Estimate Slider

In addition to the graphical elements we went one step further and created an input method with expressiveness level five. In order to enable the user to modify the kurtosis we added two more nodes, each between the peak and the minimum or maximum. The colour gradient and the probability density function plot help the user to see what kind of distribution the input results in (see *Figure 4.23d*).

Classification		
<b>Absolute/Comparative</b>	<b>Input Value Precision</b>	<b>Representation Type</b>
Absolute	Precise	Graphical (Numerical)
<b>Data Value Quantity</b>	<b>Data Dimensionality</b>	<b>Value Set</b>
Multi-Value	1D	$\mathbb{R}$
<b>Data Extent</b>	<b>Reference Class</b>	<b>Expressiveness</b>
location: $[-\infty, \infty]$ , extent: $(0, \infty]$	Probability(Percentage) / Generic Value	Level 5

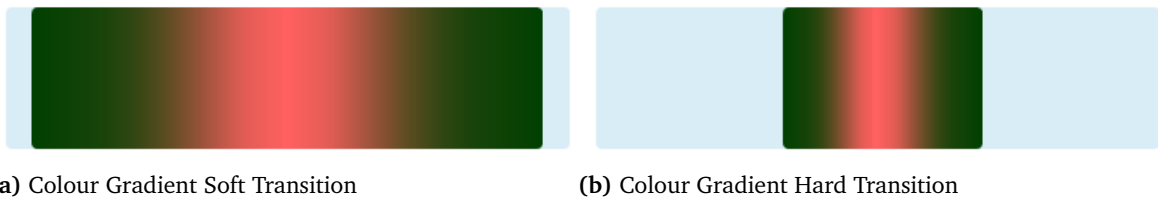
**Table 4.23.:** Classification of *Advanced Dynamic Range Best Estimate Slider* Input Method

#### 4.4.1. Distribution as Colour Gradient

To visualize the underlying distribution we put a colour gradient, that provides information about the height of the probability density function, over the range bar. When the function is flat and there is low change in probability, the colour transition is soft (see *Figure 4.24a*), while a steep curve leads to a harder colour transition (see *Figure 4.24b*). The colour gradient should give the user an intuitive feeling for the density of the underlying distribution, even when the user does not associate the colour transition directly with the probability density functions' change of height. The choice of colour is up to the user interface designer. We suggest to use opposing colours, but the best colours depend on the context and purpose of the input method and one should also consider problems, such as red-green blindness.

#### 4.4.2. Gradient Height Legend

The colour gradient gives information about the relative change of the probability density functions' height, but one can not determine the absolute height of the peak or another functional value. Each colour on the range bar colour gradient can be assigned to a height



**Figure 4.24.:** Colour Gradient Transition Examples

value. To help the user to determine the height of different points of the curve we added a legend on the right of the input bar (see *Figure 4.24*). Users can directly read the height of the peak from the top of the gradient height legend. The rest of the values fall off linearly to zero at the very bottom of the legend.

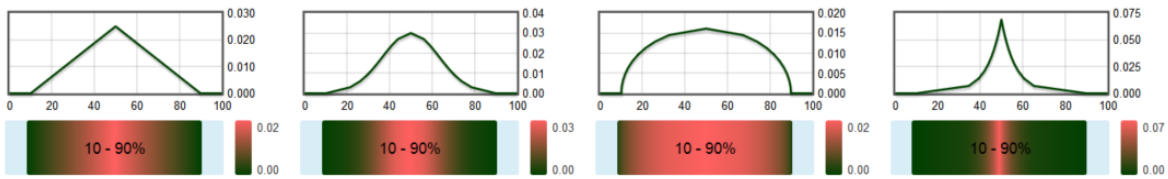
### 4.4.3. Additional Probability Density Function Plot

For those users who are familiar with the representation of the probability density as a function, we added a plot over the input bar (see *Figure 4.24*). With the help of the plot the height of a certain point on the function can be determined easily by simply checking the y-axis. We used Flot Charts [Ole14] to implement the plot.

### 4.4.4. Predefined Probability Distributions

Since the inputs can now be associated with the underlying distribution, the *Additional Probability Density Function Plot* and *Colour Gradient* visualize a certain type of distribution which influences the users' input. Therefore one has to consider the effect of displaying, for example, a normal distribution above the input bar when asking the user for the input. The user probably enters different data when provided with a normal distribution instead of a triangular or even a Wigner semicircle distribution. This problem does not arise for *Advanced Dynamic Range Best Estimate Slider*, because in this case the kurtosis can be chosen freely. But for *Fixed Range Bar Slider*, *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider* the underlying distribution is predefined by the interface developer. An additional possibility, worth considering, is to provide a drop-down selection for the user to choose the distribution type. However this requires the user to be familiar with different kinds of distributions and distributions in general. That is why we dropped this possibility in the first instance. We started out with four simple predefined distributions:

- Triangular Distribution
- Normal Distribution ( $\pm 3\sigma$ )
- Wigner Semicircle Distribution ( $r = 1$ )



**Figure 4.25.:** *Dynamic Range Bar Slider* with predefined Distribution Types (from Left to Right: Triangular Distribution, Normal Distribution, Wigner Semicircle Distribution, Laplace Distribution)

- Laplace Distribution ( $b = 1$ )

The triangular distribution is one of the simplest forms of a distribution and the kurtosis is neither positive, nor negative. The Wigner distribution is the deputy for negative curvatures. The normal distribution approaches very often and therefore this is a must-have for initial testings. Although the normal distribution has a positive kurtosis we also added the Laplace distribution as a direct counterpart for the Wigner semicircle distribution. *Figure 4.25* shows the four distributions applied to the *Dynamic Range Bar Slider*.

## 4.5. Input Method Implementation

The implementation is split up into three parts: HTML, CSS and JavaScript (jQuery). In a HTML template file, a div-element with a certain class and some set of attributes has to be defined. In addition to that the correct CSS and JavaScript file has to be included. When the document has loaded, div-elements with the according classes are recognized by a JavaScript snippet and a listener checks for user interactions with the input methods (see *Listing A.11*). The current input is stored in one or multiple attributes of the div-element and therefore can easily be retrieved by JavaScript.

## 4.6. Summary and Discussion

In this chapter, we presented a selection of input method prototypes, which we developed. We also classified them according to the classification techniques from *Section 3.4 – Classification of Uncertainty Representations*. The selection covers most reference classes and representation types with different levels of precision. That includes input methods, based on sliders, star ratings, and location pickers, but we also introduced new approaches, such as the *Ratio Drag And Drop* input method. For each of the presented input methods we provided estimated usability and expressiveness values. Based on a short discussion about which of those prototypes are the most promising, we presented upgraded versions of them and one even more advanced version that covers expressiveness level five. For the upgraded versions we described the

#### 4. Input Method Prototypes

---

three additional supportive graphical elements: *Distribution as Colour Gradient*, *Gradient Height Legend* and *Additional Probability Density Function Plot*. We mentioned that when displaying a probability density function plot, the presented kind of distribution may influence the users' input. At last, we briefly outlined the implementation approach for the input methods. These prototypes are analysed in the following chapters.

## 5. Web-Based Simulation Tool

The scope of this work includes the creation of a web-based simulation tool, which integrates the developed input methods. Therefore these prototypes had to be implemented in a way that allows the manual integration of the input methods as well as the automated integration by the simulation tool. Both, a good usability for users and the easy creation of new simulations was demanded. For the implementation of the simulation tool, we used the Django and Bootstrap framework, which we introduce in *Section 5.1*. In addition to that we made extended use of CSS, JavaScript, jQuery, Ajax, json, Django Apps and Django Template Tags.

In the last years portable devices, such as tablets and smartphones became more important. Their computing power increases and nowadays most of those devices are equipped with dual- or even quad-core processors. The simulation tool should be usable, even by laymen, and this arises the question if we want to run the simulation calculations client- or server-sided. This makes even more sense if we consider that personal computers and laptops usually have even more computing power. After well consideration we decided to go for server-sided calculations to support older, weaker devices and to cover very resource consuming simulations.

### 5.1. Django and Bootstrap

As mentioned above we used Django [Dja15] and Bootstrap [Boo15] to create the simulation tool. On the Django website it is described as a high-level Python Web framework that encourages rapid development and clean, pragmatic design [Dja15]. As this free, open-source framework already integrates Python, this makes it easy for us to use Python to implement the simulation environment. Django comes with more advantages that help to improve our web-based simulation tool: Django applications, Django Template Tags, security features, scalability and its operation with an underlying database. In the following *Section 5.1.1 – Django Template Tags* and *Section 5.1.2 – Django Apps* we briefly look into Django Template Tags and Django Apps, because these are two very Django specific features.

#### 5.1.1. Django Template Tags

Django provides a useful feature: Django Template Tags. Before sending a requested HTML Template, the Django framework dynamically processes the Template Tags in this document,

---

### Listing 5.1 Django Template Tag Example (for-loop, dictionary and filters)

---

```
<!-- Input methods -->
{% for dict_item in inputs %}
  <h4>{{ dict_item.title }}</h4>
  {% include "input_templates/input_method_"|add:dict_item.type_id|add:".html" with
    params=dict_item.params %}
  <br><br>
{% endfor %}
```

---

depending on the state and context. These tags can be used to integrate constants, calculate dynamic content with the help of python methods, or to make use of simple programming constructs, such as conditionals and for-loops (see *Listing 5.1*). We use these template tags to dynamically create the required html-elements for the input methods and to populate them with the required attributes to configure them as specified in the simulation specification file.

#### 5.1.2. Django Apps

The term *Django Apps* stands for *Django Applications* and describes a Python package that provides some set of features. It is a set of code that interacts with various parts of the framework. After registering a Django App to the website it provides new constants, Template Tags, models, Python snippets and other resources to the website. We provide an example for a Django Application in *Section 5.2 – Concept and Development*. The simulation functionality is realized as a Django App.

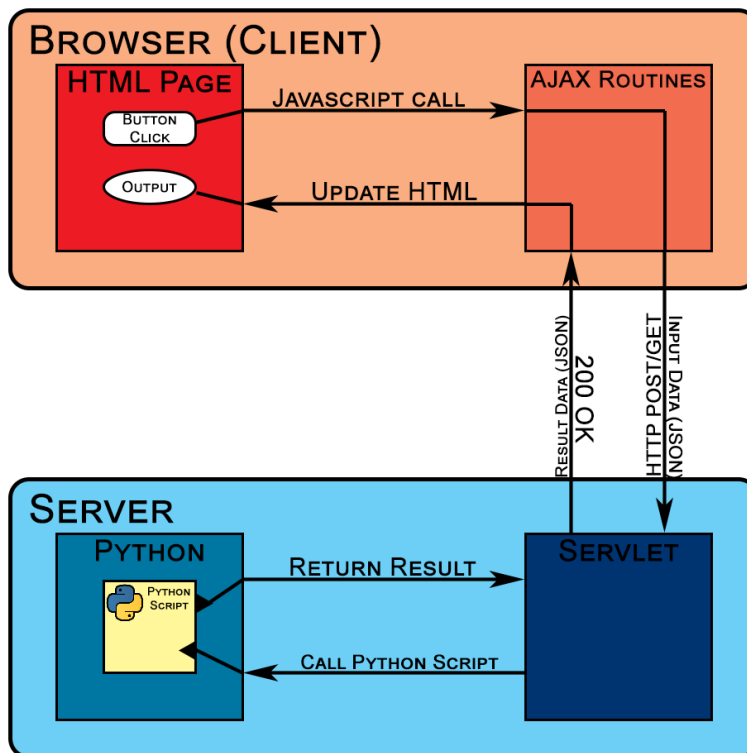
#### 5.1.3. Bootstrap

The Bootstrap website advertises Bootstrap as the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web [Boo15]. The provided features scale with a single code base from phones to tablets to desktops with CSS media queries. The already integrated CSS and JavaScript functionality helped a lot to build up on this to quickly develop and integrate new input methods and other required features.

## 5.2. Concept and Development

Since we decided to calculate the simulation result on the server side, a first step is to run a Python code after triggering from a website on a client device and send the results back to the requestor and display it on the client interface. Therefore we created a simple test application that calculates the square of a whole number server-sided. After clicking a button on the website (client side) a JavaScript function reads the input number from an input field





**Figure 5.1.:** Simulation Tool Client Server Communication Overview

and sends it with the help of a HTTP Post to a specific URL of the website (see *Listing A.8*). A Django view function<sup>1</sup>, specified in the test application, receives this request and processes it (see *Listing A.1*). A python function which calculates and returns the square of the input, is called (see *Listing A.2*). The response to the HTTP Post contains the result as payload, wrapped in a json dictionary. On the client side the JavaScript function waits for the response and populates a prepared textfield with the result number. *Figure 5.1* illustrates that.

This lays a good foundation for creating a complete simulation tool. In order to make it easy for developers to create new simulations we created a simulation template that requires the specification of a predefined set of parameters. Besides a unique identifier, a name, and a description text, there are 3 main parameters:

- Input
- Output

<sup>1</sup>A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response

- `run_simulation()` function

To create a new simulation a derivation of the abstract `UIST_Simulation` class has to be created and the new simulation class has to be registered in the `SIMULATION_LIST` in the registration file (see *Listing A.7*). By redefining the constants, inherited from the abstract simulation base class, the input parameters can be specified (see *Listing A.3*). The `INPUT` constant is an array of dictionaries, while each dictionary contains parameters to specify and configure the required input methods (see *Listing A.9*). Analogously the `OUTPUT` constant specifies the output methods. For each in and output method one has to assign a unique identifier. The simulation application uses this identifier to catch the input from the input elements, when triggering the simulation and to find the output elements to display the results after the calculations. The simulation implementation is to be put in the `run_simulation()` function. After fetching the input values the simulation application calls this method and takes the returned dictionary to populate the prepared output methods (see *Listing A.10*). To provide high responsiveness we added a progress bar to the front-end. In the simulation implementation the progress value has to be updated by calling the `sim_base.UIST_Simulation.update_progress()` function, otherwise the progress bar shows the 0% and 100% steps only. The simulation tool builds the simulation front-end automatically and the simulation developer can concentrate on the implementation instead of the creation of an interface for the in and output.

### 5.3. Example Simulation

We created a very simple test simulation which can be used by developers as a template and to be able to test the whole system, rather than single components only. It is called *Hitting the Bull's-eye*, because it is about an archer who tries to hit a target. Depending on the input, namely the change to hit the bullseye and the amount of arrows in his quiver, the amount of arrows that hit the bullseye are calculated. The input method `type_id 1` refers to the *Simple Slider* and the `type_id 2` refers to the *Ratio Selector*. The first item of the array, which is the value of the `params` key, is the unique identifier. The according input value is put into the input dictionary with this identifier as key. In the `run_simulation()` function the value can be easily fetched by calling `input_dict.get('< key >')`. According to the identifier names for the output methods, the output values have to be stored with the output method identifier as key. *Listing 5.2* and *Listing 5.3* show the implementation of the test simulation and *Figure 5.2* shows a screenshot of the front-end after simulation execution.

### 5.4. Summary and Discussion

In this chapter we presented the web-based simulation tool that we created. It makes use of the frameworks Django and Bootstrap, which we briefly described. An important demand was that this simulation tool enabled developers to easily create new simulations and integrates

**Listing 5.2** *Hitting the Bull's-eye* Test Simulation Python Code – Part 1

```

"""
-----
NAME:      apps/simulation/sim/simulation_bullseye.py
AUTHOR:    Marius Kleiner
DATE:      15.01.2015
MODIFICATION LOG:
    18.01.2015 [Marius Kleiner]   Added documentation
-----
A very simple simulation of an archer trying to hit the bullseye of a target.
----- """

# IMPORT abstract simulation base class
from apps.simulation.sim import simulation_base as sim_base

# IMPORT misc for simulation
from time import sleep
from random import randint

#Simulation class
class Simulation(sim_base.UIST_Simulation):

    #: Simulation identifier
    IDENT = 'HittingTheBullseye'
    #: Simulation name
    NAME = 'Hitting The Bulls-Eye'
    #: Simulation description (short)
    DESC = 'An archer uses his bow and arrow to hit a target. How many of the archers arrows
        will hit the bullseye?'
    #: List of simulation input methods
    INPUTS = [
        # INPUT 1 (arrow hit chance)
        {
            'type_id' : '1',
            'title' : 'Chance to hit the bullseye',
            'desc' : 'A arrows chance to hit the bullseye.',
            'params' : ['chance_bullseye', 'Chance: ', '%', '10', '100', '1', '50']
        },
        # INPUT 2 (amount of arrows)
        {
            'type_id' : '2',
            'title' : 'Arrows in quiver',
            'desc' : 'The amount of arrows in quiver.',
            'params' : ['arrow_amount', '4', '10', '5', "093", "12"]
        }
    ]
    #: List of simulation outputs
    OUTPUTS = [
        # OUTPUT 1 (amount of successful arrows)
        {
            'type_id' : 'number_simple',
            'title' : 'Arrows in bullseye',

```

---

### Listing 5.3 *Hitting the Bull's-eye* Test Simulation Python Code – Part 2

---

```
        'desc' : '# of successfully arrows',
        'params' : ['arrow_success_num']
    }
]

@classmethod
def run_simulation(cls, input_dict, csrf_token, uniqid):
    """ Run 'Hit The Bulls-Eye' simulation

    PARAMS:
        input_dict      Dictionary containing input parameters
        csrf_token      Django CSRF token
        uniqid          A unique id for this simulation instance
    RETURN:    Amount of arrows that hit the target bullseye
    """
    # Check if all expected input keys exist
    if Simulation.input_keys_exist(input_dict) == False:
        return sim_base.UIST_Simulation.ERROR_KEY_MISSING
    # Get input values, convert to desired datatype and put them in
    # local variables.
    # Note, that there's no need to define an alternative value for
    # the dict.get() function because existence has been checked in
    # Simulation.input_keys_exist() already.
    chance_bullseye = int(input_dict.get('chance_bullseye'))
    arrow_amount = int(input_dict.get('arrow_amount'))
    # Simulation logic...
    successful_shots = 0
    for arrow_num in range(0, arrow_amount):
        attempt = randint(1,100)
        if attempt <= chance_bullseye:
            successful_shots += 1
    # For testing purposes the simulation progress raises by 20%
    # every second. (Visualize simulation calculation progress)
    sleep(1)
    sim_base.UIST_Simulation.update_progress(csrf_token, uniqid, 20)
    sleep(1)
    sim_base.UIST_Simulation.update_progress(csrf_token, uniqid, 40)
    sleep(1)
    sim_base.UIST_Simulation.update_progress(csrf_token, uniqid, 60)
    sleep(1)
    sim_base.UIST_Simulation.update_progress(csrf_token, uniqid, 80)
    sleep(1)
    sim_base.UIST_Simulation.update_progress(csrf_token, uniqid, 100)
    # Once all calculations have been finished prepare and return an
    # output dictionary containing a value for all defined outputs.
    output_dict = {
        'arrow_success_num' : successful_shots
    }
    return output_dict
```

---

**Description**

This is a very simple simulation for testing purposes.

An archer is training for a tournament. He does not yet know about the amount of arrows and target distance. The archer exactly knows the chance to hit the bullseye depending on the target distance.

How many arrows will hit the bullseye?

**Simulation Inputs**

Chance to hit the bullseye

Chance: 50%

Arrows in quiver

5 of 12

min max

**Run simulation** 100%

**Simulation Results**

Arrows in bullseye

2

**Restart Simulation**

**Figure 5.2.:** *Hitting the Bull's-eye Test Simulation Front-End Screenshot*

the created input methods. We believe that we achieved that and we described the creation of a simple sample simulation to illustrate that.



## 6. Online Survey

The preselected input methods that we enhanced with supportive graphical elements in *Chapter 4 – Input Method Prototypes* have to be evaluated for their usability and applicability. Therefore we used an online survey for a first step of evaluation. It requires relatively low effort for its provided benefit and it is easy to reach a large number of participants. People are probably more willing to conduct a 10 minute online survey at home in comparison to moving to the laboratory to participate in a local study. The disadvantage is that we can not use special measurement techniques, such as an eye-tracker. However we deal with web-based input methods and therefore an online survey where users participate using their home computer device provides the environment where these input methods are planned to be used in. The usability measures, which we discussed in *Section 3.5 – Input Method Metrics*, can be measured in a subjective way, by questionnaire, or in an objective way. We decided to stick with subjective measures for the online survey to receive some first insights and to use the gained information to improve the user study, which additionally uses objective measurement techniques.

### 6.1. Survey Composition

In *Section 4.4 – Upgrading most promising Input Methods* we made a preselection of prototypes. These prototypes were the subject of interest in this online survey:

- *Bar Slider*
- *Fixed Range Bar Slider*
- *Dynamic Range Bar Slider*
- *Dynamic Range Bar Best Estimate Slider*
- *Advanced Dynamic Range Best Estimate Slider*

We inspected system usability and user experience measures in *Section 3.5 – Input Method Metrics*. In this study we determine the following measures:

- Effectivity
- Efficiency
- Confidence

- Satisfaction
- Learnability

For each of the input methods we asked questions to gather information about those measures in a subjective way.

### 6.2. Questionnaire

Each participant was asked to answer some general questions for demographic classification.

An example question and a working instance of the according input methods was provided. For each input method we created an analogue question block, which we presented in random order for each participant. First the participants were shown an example task with the possibility to test the input method, as it could appear in a simulation data input situation. The task was to specify how often Sam Sample uses his car to go to work per month. To make sure the participants do not expect this to be a single value that is always the same, instead of a distribution, we added a table that showed how often Sam took his car in the last 36 months. The values in the table were normal distributed, but without detailed inspection this was hard to see. Anyway the table should mainly make sure the participants try out the input methods to enter a distribution. The range of possible values the users could enter was, from 0 to 31. For input method *Fixed Range Bar Slider*, *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider* the predefined distribution type, as described in *Section 4.4.4 – Predefined Probability Distributions*, was already set to normal distribution. The range for the *Fixed Range Bar Slider* input method was set to the correct range, so that the user was only required to drag it to the correct position. For the *Bar Slider* we expected the participants to specify the most likely value, or the mode of the probability density function. The hardest task was for the *Advanced Dynamic Range Best Estimate Slider* input method, where the user was asked to specify the mean, the range and two nodes with the half height of the mode. In addition to each task, we provided a brief description for the input method that explained the interaction with the different input elements. We did not track the correctness and duration of the input since we were, so far, only interested in the questionnaire results.

#### 6.2.1. Demographic Information

In order to check if the set of participants provides representative informative value, or if a certain demographic group gives different results, we requested some demographic information, such as:

- Age
- Gender



- Highest Degree of completed School
- Employment Status

The input methods are more complicated to use than their standard web-based relatives. In addition to that the underlying math is more complex and deals with probability theory, stochastics and statistics. If the user is not familiar with such things, this might impede the interaction with the input method. Therefore we also asked questions about the participants familiarity with common web-based input methods and about their knowledge about the following topics:

- Probability Theory
- Stochastics
- Statistics
- Probability Distributions

### 6.2.2. Usability Questions

We asked the following questions for each of the five input method question blocks to measure usability:

- **Effectivity:** „I am confident that I am able to correctly enter data with this input method“
- **Efficiency:** „I was able to quickly enter data using this input method“
- **Confidence:** „It was simple to use this input method“
- **Satisfaction:** „I liked using this input method“
- **Learnability:** Selection of one from five predefined formulations, ranging from „I could use this input method intuitively (without reading description)“ to „I doubt I will ever be able to confidentially use this input method (even after training)“.

A text field allowed the participants to add free-text comments for each input method. These comments were used for conclusions and to improve the input methods.

### 6.2.3. Final Question Block

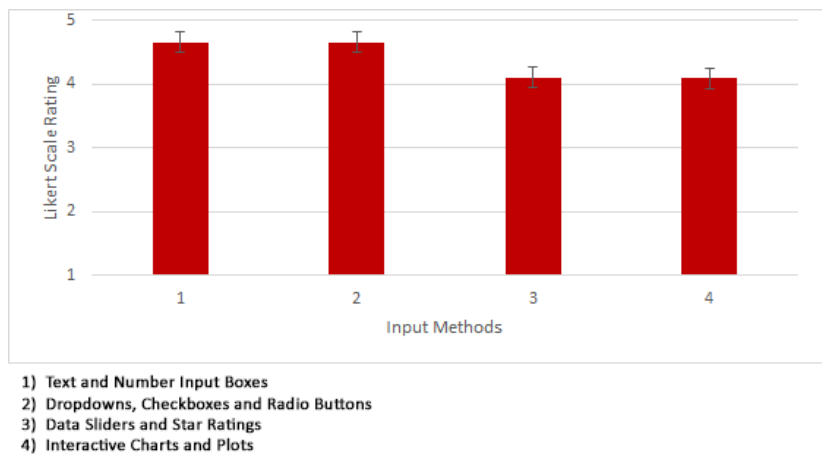
After asking questions for each of the five input methods independently, the users were asked to rank them for likability and usefulness. The supportive graphical elements *Distribution as Colour Gradient*, *Gradient Height Legend* and *Additional Probability Density Function Plot*, were created to help the users to work with the input methods. Therefore we asked the participants to rate and comment on their understandability, likability, and usefulness. Three text fields provided the possibility to explain their rankings and judgement of the graphical elements to the participants, to tell for which tasks and which applications they would like to use this input methods, and to comment on the whole survey.

### 6.2.4. Deployment and Distribution

For the survey conduction we used the Lime Survey tool and deployed the survey on the Lime Survey server of the HCI group [Car15]. It was an anonymous and open survey. Consequently everyone with the link to the questionnaire was able to participate. The survey was distributed over e-Mail, Facebook, WhatsApp and other communication channels. The survey was active for about three weeks.

## 6.3. Participants

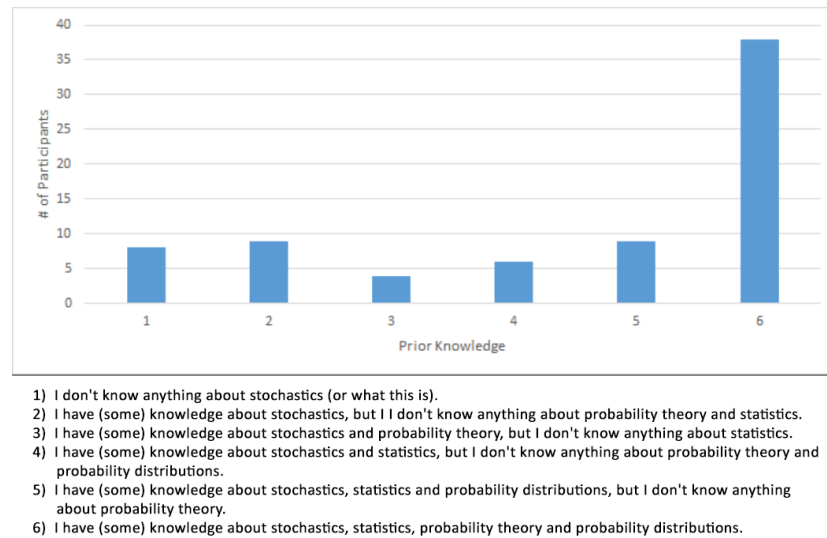
A total of 72 participants completely finished the survey, while there were about another 155 data sets with incomplete answers. We suspect that some of these people felt overstrained with some input methods and therefore skipped the survey instead of communicating this by answering the questionnaire. Some people communicated problems when using the chrome browser and some of the incomplete answer sets were completely empty, which means these people did not even try to do the survey and therefore they can be ignored. We checked if most of these participants stopped at a certain point in the survey, for example at the more complicated question blocks, and if the dropouts specified a low familiarity with web-based input methods or little knowledge about probability theory, statistics and stochastics. There was no regularity to be found in the demographic data of the dropouts. Consequently they probably skipped the survey due to the duration of the survey or the problems with chrome browsers. The participants were, in average, 25.9 years old, with a standard deviation of 5.9. 39 of the 72 subjects were male, 32 female and one participant preferred not to tell. About 20% of the participants stated to be employed for wages and about 78% were students. The highest degree of education for 30 of the 72 participants was a high school graduate, diploma or equivalent (Abitur), whereas about 32% had a Bachelor's degree and 15% even had a Master's degree. *Figure 6.1* shows the average results for the question about the familiarity with common web-based input methods. Most participants were quite familiar with text and number input boxes and with dropdowns, checkboxes and radio buttons. They were slightly less familiar



**Figure 6.1.:** Participants' average Familiarity with different Web-Based Input Methods as Likert Scale Rating with Standard Error Bars

with data sliders and star ratings and with interactive charts and plots, but this should be enough to make sure the results do not significantly deteriorate due to solely unfamiliarity with such input techniques.

*Figure 6.2* provides valuable insight in how much knowledge the participants had about probability theory, statistics and stochastics in general. We expected people with such knowledge, especially about probability distributions, to perform better on tasks as we presented it in the survey. The chart shows that most of them knew much about probability theory, statistics, stochastics in general and even about probability distributions.



**Figure 6.2.:** Amount of Participants' with average prior Knowledge about Statistics, Probability Theory and Stochastics in General

## 6.4. Results

In this section we present the results of the online survey about the subjective measurement of the usability of the five preselected input methods. Therefore we make use of diagrams and statistical evaluations, such as Friedman test and Wilcoxon signed-rank test. In this chapter we often refer back to our expected relationship between the input methods' usability and expressiveness (see *Section 3.4.2.1 – Expressiveness vs. Usability*).

The Friedman tests for each usability measure showed a significant difference between the different input methods (see *Table 6.1*). Post hoc analysis with Wilcoxon signed-rank tests

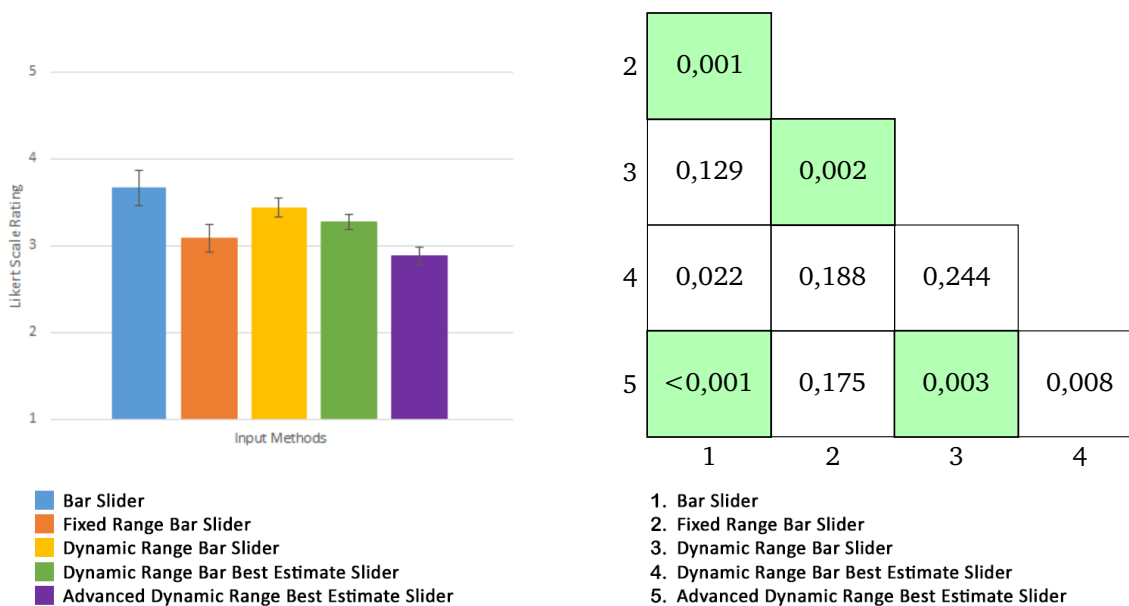
Friedman Test					
	Effectivity	Efficiency	Confidence	Satisfaction	Learnability
N	74	74	74	74	74
$\chi^2(4)$	25,32	61,56	63,85	16,72	75,73
p	< 0,001	< 0,001	< 0,001	0,002	< 0,001

**Table 6.1.:** Friedman Test Results for the five Usability Measures in the Online Survey

was conducted for each usability measure with a Bonferroni correction applied on  $p < 0.05$ , resulting in a significance level set at  $p < 0.005$ . See *Figure 6.3b*, *Figure 6.4b*, *Figure 6.5b*, *Figure 6.6b* and *Figure 6.7b* for exact results.

## 6.4.1. Effectivity

Looking at the answers for effectivity, asking if the users think that their entered data was correct, the assumption from *Section 3.4.2.1 – Expressiveness vs. Usability* seems to be confirmed, with one exception only. Figure 6.3a shows the average values for the Likert Scale effectivity questions. We would expect the *Fixed Range Bar Slider* input method to have an average effectivity value between the values of the *Bar Slider* and the *Dynamic Range Bar Slider*. The difference between the simple input methods and the more complex input methods was slightly lower for effectivity, compared to the other measures.



(a) Avg. Likert Scale Values with Standard Error Bars      (b) Wilcoxon Signed-Rank Test

**Figure 6.3.:** Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Effectivity for the five preselected Input Methods and an applied Bonferroni Correction, resulting in a Significance Level of  $p < 0,005$ . Statistically significant Values are highlighted with green Colour.

The Wilcoxon signed-rank test (see *Figure 6.3b*) confirms that the effectivity rating for the *Fixed Range Bar Slider* was, indeed, significantly lower than for the *Bar Slider* and the *Dynamic Range Bar Slider*. The same applies to the *Advanced Dynamic Range Best Estimate Slider*.

6.4.2. Efficiency

The values for efficiency confirm the assumption that with rising expressiveness the usability drops. Again the *Fixed Range Bar Slider* falls out the series. It shows an average efficiency level of about the same height as the *Dynamic Range Bar Slider*. *Figure 6.4a* suggests that this relation between usability and expressiveness is not linear, but the usability drops exponentially for rising expressiveness levels.

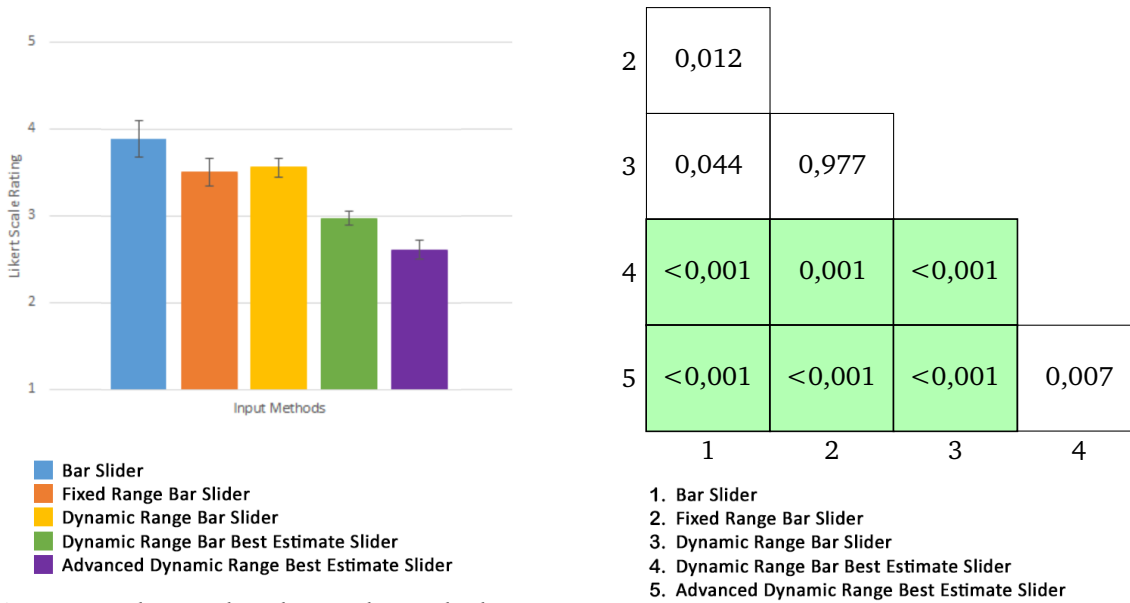


**Figure 6.4.:** Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Efficiency for the five preselected Input Methods and an applied Bonferroni Correction, resulting in a Significance Level of  $p < 0,005$ . Statistically significant Values are highlighted with green Colour.

All differences, but those between the *Bar Slider*, the *Fixed Range Bar Slider*, and the *Dynamic Range Bar Slider*, were significant. This highlights that the *Fixed Range Bar Slider* was not rated significantly better than the *Dynamic Range Bar Slider*, but it also shows that the *Bar Slider* was not rated significantly better than the *Dynamic Range Bar Slider*, which we would expect if the assumption from *Section 3.4.2.1 – Expressiveness vs. Usability*) applies.

6.4.3. Confidence

Again, the low average values for the *Fixed Range Bar Slider* and the disproportionate high values for the *Dynamic Range Bar Slider* are striking (see *Figure 6.5a*). Eliminating the input method with expressiveness level two from this chart, the relationship between usability and expressiveness shows a linear tendency.



(a) Average Likert Scale Values with Standard Error Bars

(b) Confidence

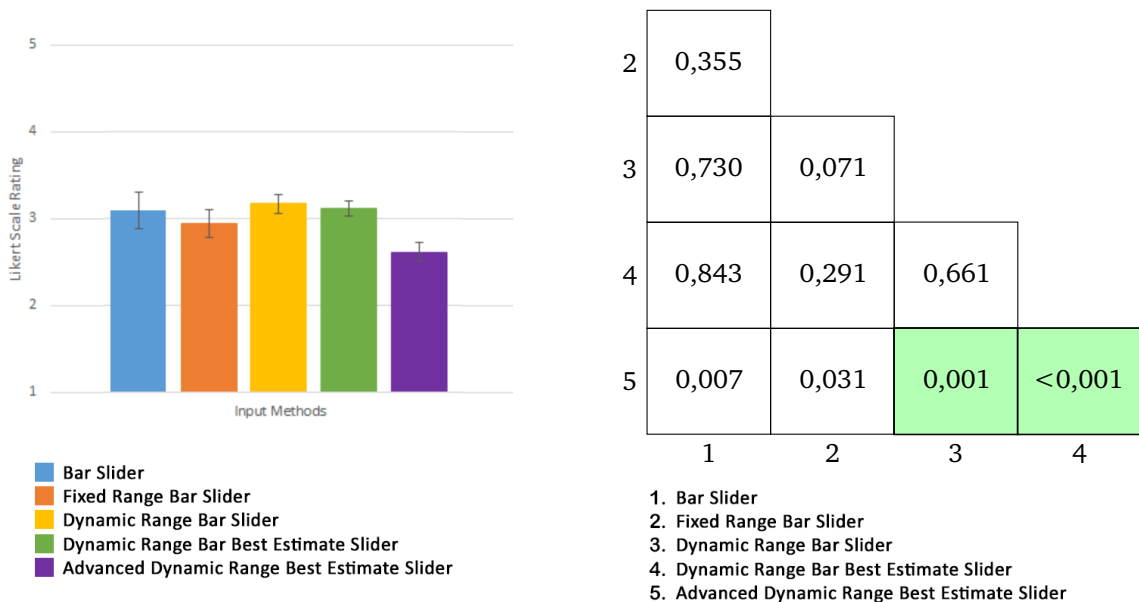
**Figure 6.5.:** Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Confidence for the five preselected Input Methods and an applied Bonferroni Correction, resulting in a Significance Level of  $p < 0,005$ . Statistically significant Values are highlighted with green Colour.

For confidence, the *Dynamic Range Bar Best Estimate Slider* and the *Advanced Dynamic Range Best Estimate Slider* input methods showed significantly lower average values than the other three input methods. At the point where the expressiveness reaches a level that enables an at least rough specification of a probability distributions the participants performed significantly worse, enforcing the exponential relation

6.4.4. Satication

The values for satisfaction are completely out of the ordinary. Like for the other measures the *Fixed Range Bar Slider* has lower average values as expected. The relatively high value of the

*Dynamic Range Bar Best Estimate Slider* is about as high as the values of the *Bar Slider* and *Dynamic Range Bar Slider*. There are not as much variations in the values of the different input methods than for other measures, expect for the *Advanced Dynamic Range Best Estimate Slider*, which has the lowest value for all the measures (see *Figure 6.6a*).



(a) Average Likert Scale Values with Standard Error Bars

(b) Satisfaction

**Figure 6.6.:** Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Satisfaction for the five preselected Input Methods and an applied Bonferroni Correction, resulting in a Significance Level of  $p < 0,005$ . Statistically significant Values are highlighted with green Colour.

In contrast to the values for confidence, the chart in *Figure 6.6a* indicate an exponential tendency for the relationship between usability and expressiveness. The significant lower value for the *Advanced Dynamic Range Best Estimate Slider*, compared to those of the *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider*, and the lower, not significant differences between the other four input methods, enforce this assumption, with the exception of the *Fixed Range Bar Slider*, which did not match this pattern.

### 6.4.5. Learnability

The chart for **learnability** (see *Figure 6.7a*) shows similar trends like those for efficiency and confidence, but in contrast to them, the values here are relatively high. Even the *Advanced*



*Dynamic Range Best Estimate Slider* input method, which showed disproportionate low values for the other measures, is closer to the other values.



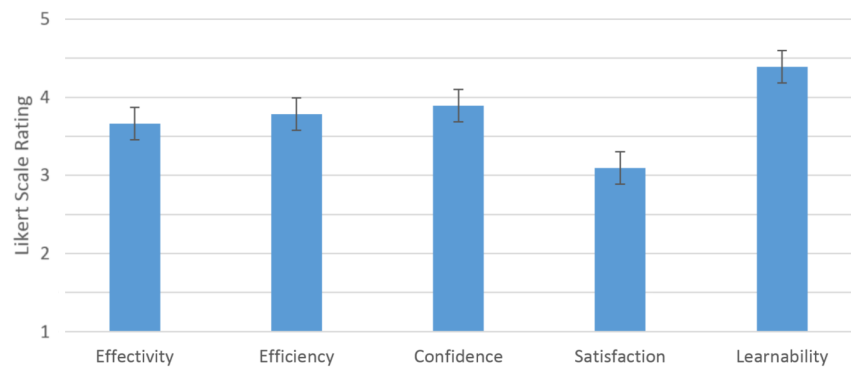
**Figure 6.7.:** Average Likert Scale Values with Standard Error Bars and Results of the post-hoc Analysis with a Wilcoxon Signed-Rank Test for Usability Measure Learnability for the five preselected Input Methods and an applied Bonferroni Correction, resulting in a Significance Level of  $p < 0,005$ . Statistically significant Values are highlighted with green Colour.

All values, but those comparing the *Fixed Range Bar Slider* and the *Dynamic Range Bar Slider*, and those comparing the the *Dynamic Range Bar Best Estimate Slider* and *Advanced Dynamic Range Best Estimate Slider*, showed significant differences.

#### 6.4.6. Results for Input Method Results

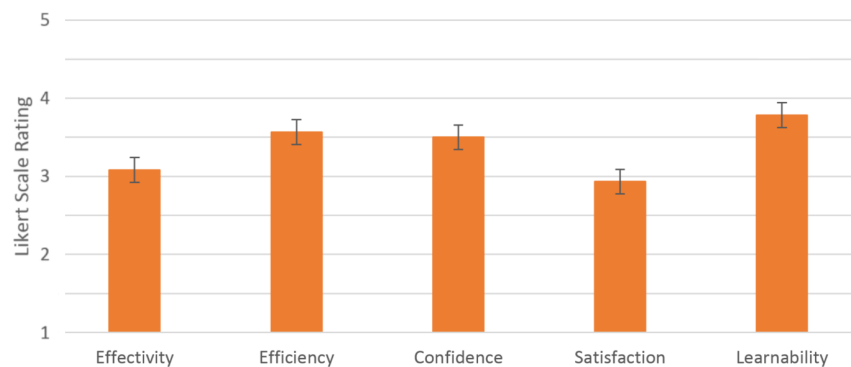
Although users should be quite familiar with the usage of simple sliders, the values for the *Bar Slider* are lower as expected. We expected those values to be at least close the the average familiarity with web-based input methods (see *Figure 6.1* and *Figure 6.8*).

The most striking input method in this series is the *Fixed Range Bar Slider*. It has lower average values as expected for all the five usability measures. The assumption from *Section 4.2 – Expressiveness vs. Usability Estimation* would suggest average values between those of the *Bar Slider* and the *Dynamic Range Bar Slider*, for the *Fixed Range Bar Slider*. In some cases it has



**Figure 6.8.:** Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the *Bar Slider* Input Method

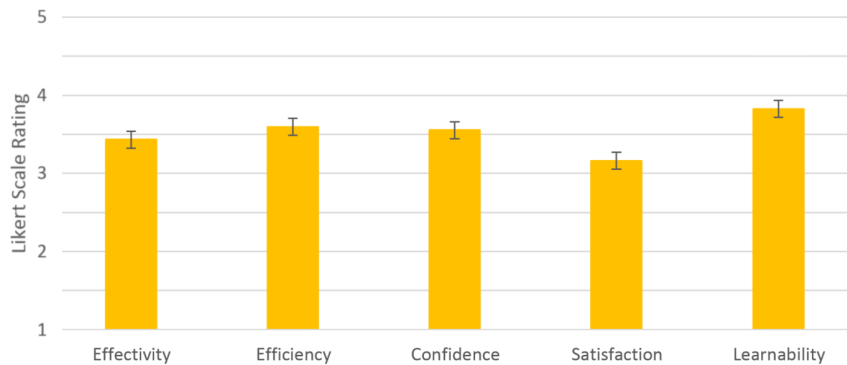
even lower values than the *Dynamic Range Bar Best Estimate Slider*. Especially its effectivity was assessed very low (see *Figure 6.9*).



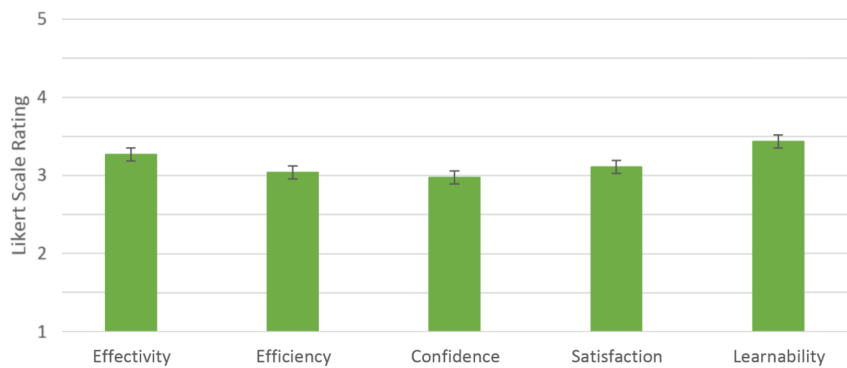
**Figure 6.9.:** Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the *Fixed Range Bar Slider* Input Method

We expected the *Dynamic Range Bar Slider* to perform quite well since there are already slider for range specification in the web. It showed relatively high values for the most measures, compared to the other input methods. The *Dynamic Range Bar Slider* provided quite good results, which were for all usability measures, but likability, not significantly lower than those for the *Bar Slider*.

The *Dynamic Range Bar Best Estimate Slider* clearly falls behind the first three input methods if we consider its efficiency, confidence and learnability. However it shows good values for effectivity and satisfaction. This input method provided results as we mostly expected. *Figure 6.11* sums up its average values for the five usability measurements.



**Figure 6.10.:** Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the *Dynamic Range Bar Slider* Input Method

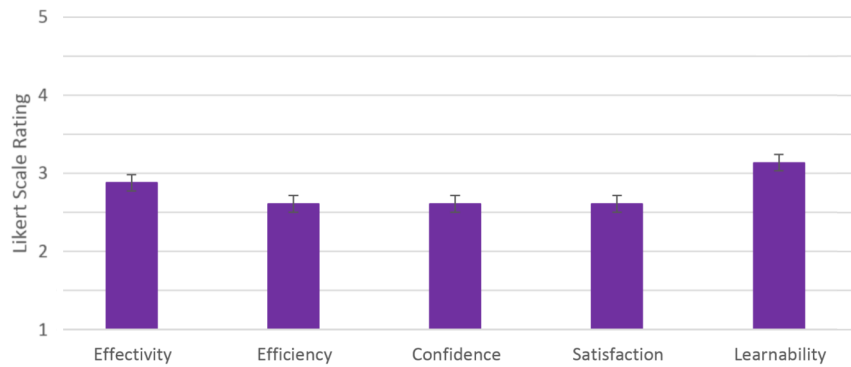


**Figure 6.11.:** Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the *Dynamic Range Bar Best Estimate Slider* Input Method

It is not surprising that the *Advanced Dynamic Range Best Estimate Slider* showed the worst results in all categories. Like the *Dynamic Range Bar Best Estimate Slider*, it suffers from the average values for efficiency, confidence and satisfaction (see *Figure 6.12*, *Figure 6.4a*, *Figure 6.5a* and *Figure 6.7a*).

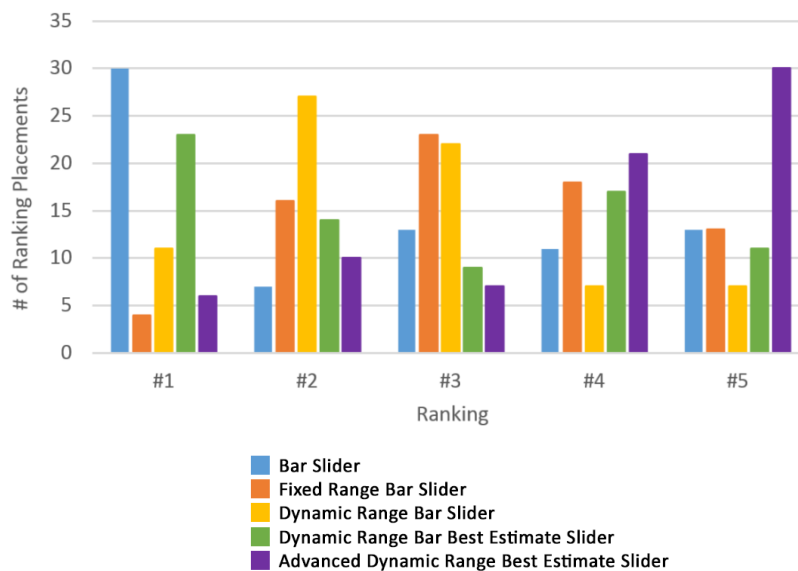
#### 6.4.7. Likability and Usefulness Rankings for Input Methods

We asked the participants to rank the presented input methods for their likability and usefulness. *Figure 6.13* and *Figure 6.14* show the results of the rankings. Besides the *Bar Slider* which we expected to be the most liked prototype, the *Dynamic Range Bar Best Estimate Slider* was specified as the most liked input method by about 31% of the participants and about 19% liked it second-most. This is quite interesting, because this input method has a relatively high

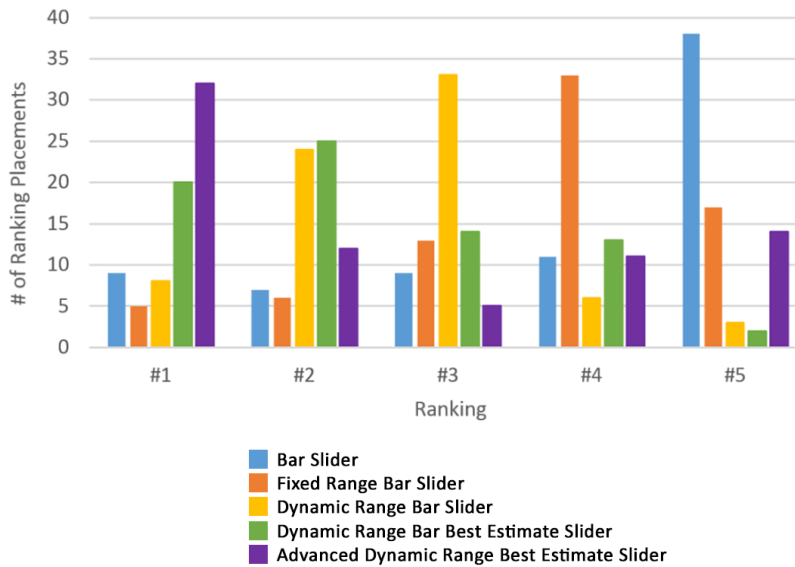


**Figure 6.12.:** Average Likert Scale Values with Standard Error Bars for the five Usability Measures for the *Advanced Dynamic Range Best Estimate Slider* Input Method

expressiveness. The *Fixed Range Bar Slider* was mostly rated as the third-most liked input method. The *Dynamic Range Bar Slider* takes its place as the second-most liked input method. The *Advanced Dynamic Range Best Estimate Slider* is in most cases rated as the fourth-most or least liked input method, but they recognized its potential for meaningful inputs. The rankings for usefulness are mostly as expected. *Advanced Dynamic Range Best Estimate Slider* is considered as the most useful, while *Bar Slider* is rated as the least useful. The input methods *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider* did both receive some more ratings as the most, respectively second-most, useful input methods.



**Figure 6.13.:** Input Methods Likability Ranking

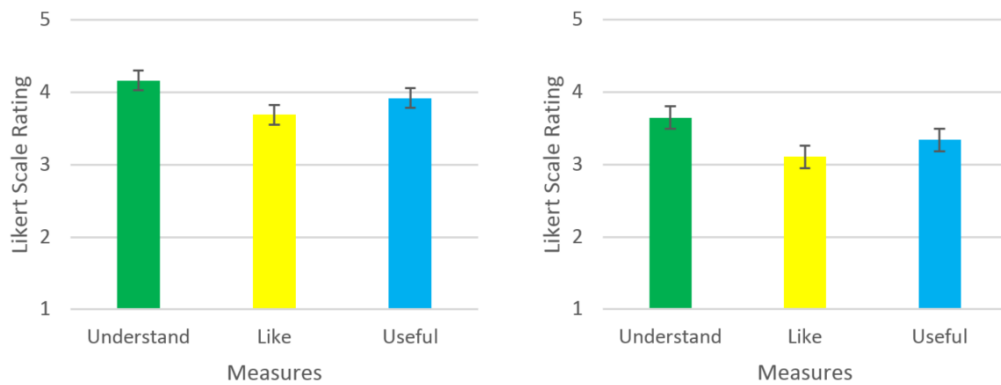


**Figure 6.14.:** Input Methods Usefulness Ranking

#### 6.4.8. Rating of the supportive graphical Elements

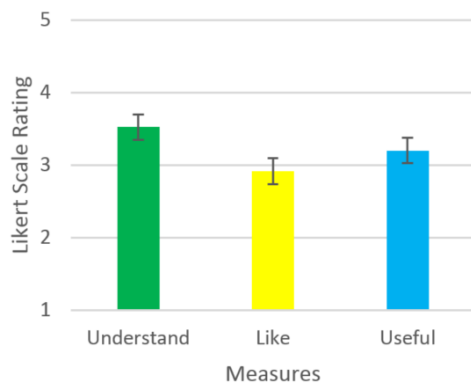
For all of the three supportive graphical elements *Additional Probability Density Function Plot*, *Distribution as Colour Gradient* and *Gradient Height Legend* we asked the user to rate three Likert Scale statements, namely about their understandability, likability and usefulness (see *Figure 6.15a*, *Figure 6.15b* and *Figure 6.15c*). The average values for the *Gradient Height Legend* and the *Distribution as Colour Gradient* are nearly the same, which is not surprising, since these elements are supposed to be used as a tandem and they make use of the same visualization technique. The *Gradient Height Legend* was rated even slightly worse in likability compared to the *Distribution as Colour Gradient*. The *Additional Probability Density Function Plot* shows higher average values for all the three categories. Especially likability and usefulness received significantly higher ratings. However, we have to consider that over 50% of the participants specified that they had good knowledge about statistics, stochastics and even probability distributions. People like to work with familiar things, a probability density function plot in this case. For people with no knowledge about these topics, the plot might give no informative value and the colour gradient might be more intuitive to use.

## 6. Online Survey



(a) *Additional Probability Density Function Plot*

(b) *Distribution as Colour Gradient*



(c) *Gradient Height Legend*

**Figure 6.15.:** Average Likert Scale Values with Standard Error Bars for the Measures Understandability, Likability and Usefulness for the three supportive graphical Elements

### 6.4.9. Comments from Participants

Some overzealous participants calculated the mean of the table of sample values and complained about the inability to specify 15,6666 with the help of the *Bar Slider*, *Dynamic Range Bar Best Estimate Slider* and *Advanced Dynamic Range Best Estimate Slider*. This refers to the problem of uncertainties due to input method limitations (e.g. in precision) mentioned in *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*. However, the users were not supposed to calculate the mean value and in addition to that the mean of the underlying standard distribution was 15, but the amount of 36 values was too low to lead to the correct value to be calculated.

For the Bar Slider some users requested a look and feel that is closer to standard web sliders and asked for the possibility to specify the value by directly clicking on the desired position on the bar instead of dragging the node all the way along to the correct position. Also marks for value steps (e.g. marking each multiple of 10) were requested. An additional display for the input value was suggested, because the value is sometimes hidden by the mouse when dragging the node. Besides these things the comments described this input method as quite simple and easy to use.

The predefined size of the Fixed Range Bar Slider was supposed to help the users and ease the interaction. However the overall tone was that this input method was „super easy“ to use, but the impossibility to change the range size was considered as more a rather negative than positive fact. Therefore was, amongst others, described „horrible“ and „annoying“.

Like for the Bar Slider participants requested the possibility to directly set the values by clicking in the bar when working with the Dynamic Range Bar Slider. The comments ranged from „has not enough freedom to enter data“ up to „awesome and quite self-explaining“ and „easy and visually appealing“.

After working with some of the other input methods some participant felt that the Dynamic Range Bar Best Estimate Slider was „easy“ and „intuitive“ to use. But they recognized the lower usability that comes with the higher expressiveness and some participants mentioned this in their comments. Generally people stated that this is a good trade-off between usability and expressiveness, though they did not use those terms. They used descriptions like „The more power the method holds, the more complex and annoying the interaction got. This one was a good compromise.“ and „I think this one is where you provide the most information, while not being very cumbersome at the same time.“.

Comments for the Advanced Dynamic Range Best Estimate Slider included both, positive and negative feedback, while the negative feedback was predominant. Although some participants did not understand it at all and many of them called it „too complicated“, „too complex“ and „overloaded“, others rated it as a „very good input method“ and they appreciated the higher expressiveness while stating that they could learn to use it eventually. For the negative comments the inconsistent interaction types and the high amount of input elements were the main issues.

The supportive graphical elements were taken with mixed feelings. Some participants explained that they did not get the functionality of the Gradient Height Legend, describing it as „not necessary, because the colours are intuitive, even without the legend“ and „not sufficiently practicable“, but they gave more positive feedback for the transition gradient, which some participants called „useful to see the density“. Most participants summed up the Additional Probability Density Function Plot and the Distribution as Colour Gradient as „simple and straightforward“. Confirming the results of the elements ranking for usability, likability and usefulness, the Additional Probability Density Function Plot received the most positive feedback.

Few feedback commented negative about it, but the numerous positive comments ranged from „useful“ up to „fantastic“.

Generally the comments confirmed the interpretation of the answers of the usability questions. Participants confirmed the theory that with rising expressiveness the usability drops, but they made clear that the *Fixed Range Bar Slider* showed very low usability, compared to its expressiveness.

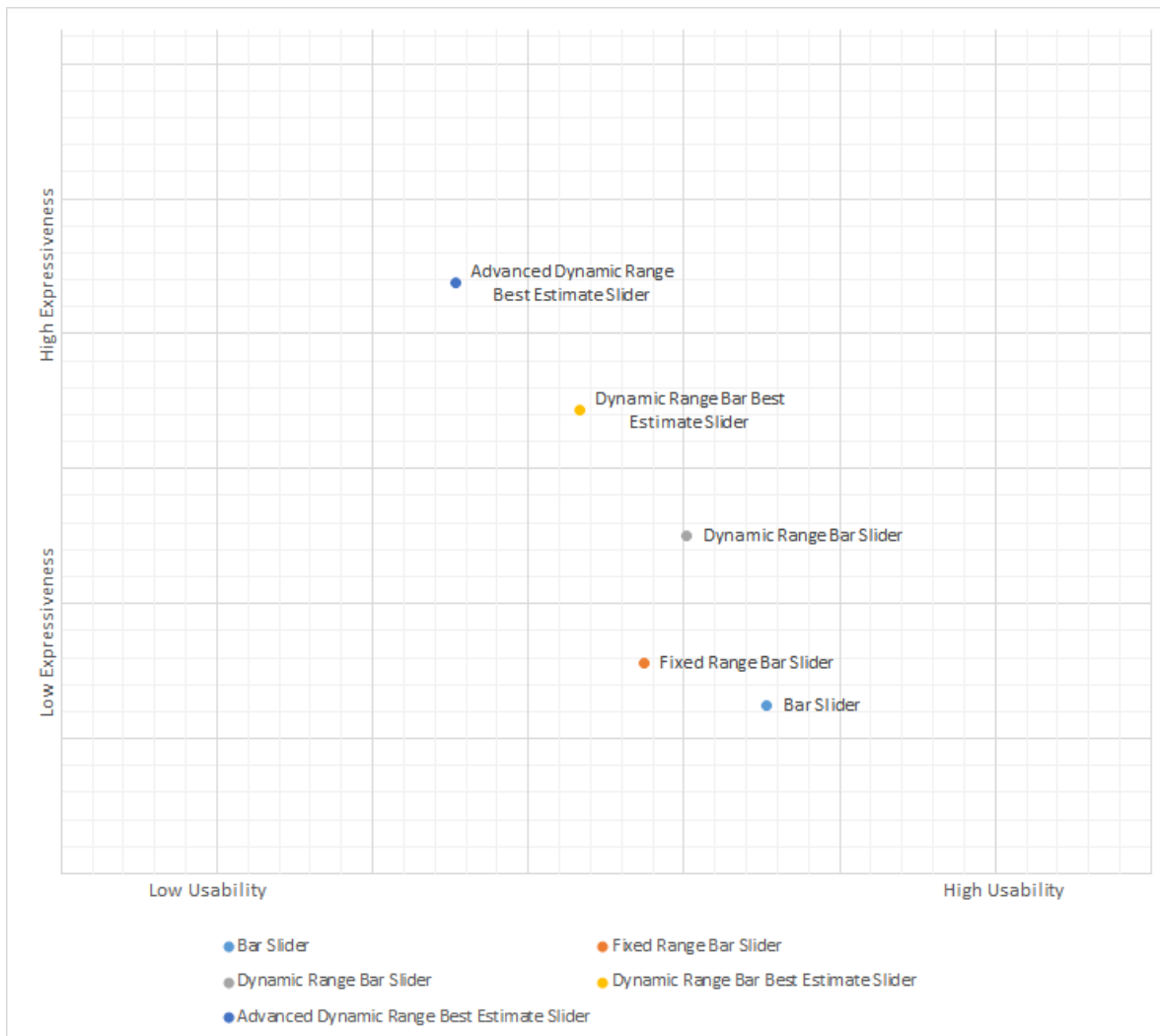
### 6.4.10. Expressiveness vs. Usability Estimation after Online Survey

In *Section 4.2 – Expressiveness vs. Usability Estimation* we presented an estimation of the usability and expressiveness of the various input methods. With the help of the new information that we received from the online survey we can make new assumptions. We calculated a usability value for the input methods, which is the average of the average values for the five usability measures. This is no standardized calculation and it should be considered as an estimated value only. *Figure 6.16* shows these estimations in a diagram. Considering this diagram the slightly higher usability of the *Bar Slider* does not make up for its low expressiveness. The *Fixed Range Bar Slider* is out of discussion. The other input methods seem to be valid alternatives and therefore their application depends on the focus we put, high usability or high expressiveness.

## 6.5. Discussion

The differences between the input methods for effectivity were lower, which might originate from the participants' high average knowledge about stochastics and the participants' resulting high assesment of the input methods' effectivity. The values for effectivity are, compared to the average values for confidence and learnability, slightly lower. This indicates that people were, at least in some cases, not able to enter the desired data correctly. The lower average values for effectivity indicate that some participants had problems entering the desired data. There are several possible reasons for this. Some people mentioned that they wanted to enter comma values while the input methods allowed the specification of whole numbers only. Some problems arised due to a lack of knowledge about distributions, for example participants did not know what the mode is or they did not get along with distributions at all. In addition to that the specification of a node with the half height of the mode is quite uncommon. The unfamiliar look and feel of the input methods and their relatively high complexity may have caused lower effectivity, too. The average values for efficiency suggest an exponential relation between usability and expressiveness. That would mean that input methods with even higher expressiveness led to very negative results. The results of the according Wilcoxon signed-rank test underpin this assumption, since the *Bar Slider*, the *Fixed Range Bar Slider* and the *Dynamic Range Bar Slider* input methods do not show significant differences (meaning a flat start of





**Figure 6.16.:** Expressiveness vs. Usability Estimation for preselected Input Methods after Online Survey, incorporating initial Testings and Survey Results.

the curve), while the *Dynamic Range Bar Best Estimate Slider* and the *Advanced Dynamic Range Best Estimate Slider* show significantly different values and the difference between those two input methods are also significant (resulting in a precipitous curve). Participants rated their confidence for the *Dynamic Range Bar Best Estimate Slider* and the *Advanced Dynamic Range Best Estimate Slider* significantly lower than for the other input methods. This might originate from the mode specification element, which both input methods share, from the too high amount of combined input elements, or from a lack of understanding of probability distributions. The very close outcomes of the average values for satisfaction possibly result from the similar visual appearance and interaction mechanisms that these input methods share. An exception is the *Advanced Dynamic Range Best Estimate Slider*, which was rated significantly worse than the

*Dynamic Range Bar Slider* and the *Dynamic Range Bar Best Estimate Slider*. Again we expect the high complexity and the inconsistency in the required interaction techniques to be the reason for this. The generally high average values for the input methods' learnability suggest that people are confident that they could learn to use these input methods, although the initial effectivity was relatively low. The data show that the *Fixed Range Bar Slider* was rated as the worst input method. The fixed range was supposed to help the users, so that they have to find the correct location only, instead of specifying both, the location and the range. Especially for normal distributions, where the standard deviation is known beforehand, this simplifies the task tremendously. However, users disliked it and we believe that the inability to change the range makes this input method feel inflexible and if the user does not agree with the predefined range, it is impossible for the user to enter the desired data. The feeling of being limited to a small subset of the possible values seems to heavily decrease the input methods usability.

The results substantiates our assumption about the relationship between usability and expressiveness with significant values (see *Figure 6.7b*), but also showed some exponential tendencies. Higher expressiveness resulted in a lower usability for all the input methods, except for the *Fixed Range Bar Slider*. We incorporated the input methods' usability results in *Section 6.5 – Discussion*.

### 6.6. Summary

In this chapter we described the setup and conduction of an online survey and we presented the composition of participants, the results and the conclusion, based on these results. They confirmed the assumption that we made in *Section 4.2 – Expressiveness vs. Usability Estimation*, assuming that with rising expressiveness the usability drops. The *Fixed Range Bar Slider* fell out of the series and performed, in contrary to our expectations, very bad. Assuming that *Figure 6.16* provides sufficient validity the *Dynamic Range Bar Slider*, *Dynamic Range Bar Best Estimate Slider* and *Advanced Dynamic Range Best Estimate Slider* are good candidates.

## 7. User Study

After the measurement of subjective indicators for usability measures with the help of an online survey (see chapter 6 – Online Survey), we considered objective measurement techniques in the user study, which we outline in this chapter.

### 7.1. Study Composition

For the user study we considered the same input methods as we already did in the online survey, but we discussed to drop the Fixed Range Bar Slider due to its bad results in the online survey, but we decided to not yet abandon it and to compare with the results of the user study first. Consequently, as for the online survey, the following prototypes were the subject of evaluation:

- Bar Slider
- Fixed Range Bar Slider
- Dynamic Range Bar Slider
- Dynamic Range Bar Best Estimate Slider
- Advanced Dynamic Range Best Estimate Slider

To be able to compare the results with the online survey we also used the same usability measures. These are determined with the help of a ten-item attitude Likert Scale, namely the System Usability Scale [Bro96], and by tracking various data of user interaction with the test tool.

#### 7.1.1. Objective Usability Measurement

Our user study tool tracks different indicators for the five usability measures.

**Effectivity:** The effectivity can be determined by comparing the user input to the correct values. The more the entered data differ from the correct values the lower we rate the effectivity. Some of the input methods have multiple input elements which each allow the specification of one

value. Therefore the correct input may depend on multiple values and we have to think about how to evaluate this.

**Efficiency:** For efficiency there are basically two things to measure: the amount of clicks the user makes on the different input elements and the time the user spends to achieve the required input. A high amount of clicks suggests that the interaction with the input method is cumbersome and the user had to make many corrections. We measured the input time with the help of javascript and with the help of the stationary SMI eye-tracker [?], which generated heat maps for each question block.

**Confidence:** To measure the users' confidence is harder than the measurement of the effectivity and efficiency, which is quite straightforward. Since this is subjective by nature it seems valid to determine the users' confidence about his input with the help of a Likert Scale questionnaire. But there are also objective indicators for the users' confidence. We provided a help button to the participants and counted how often they used it to requests additional information and how much time they spent reading the help text. The eye-tracker gave hints about how much time the participants spent on reading the description or working with the input method.

**Satisfaction:** The users' satisfaction with the input method interaction is naturally subjective and there are few objective indicators. One could consider techniques, such as the measurement of the endorphin level in blood, brain activity and the interpretation of unconscious behaviour, but yet there are no practical ways to integrate these in a user study and these methods are not yet researched enough. Therefore we have to count on the measurement by questionnaire.

**Learnability:** This refers to the change of the other values over time, respectively the change after using the input method more frequently. We can compare the study tasks which are processed first with those processed at last to draw conclusions about learnability.

### 7.1.2. Study Environment

The participants answered the tasks and questions on a common desk with a 21" flat screen monitor and an ordinary computer mouse. The simulation tool, which guided the participants through the study, ran in a Firefox browser on a Windows system. This has the advantage that the participants experience the input methods in an environment they were designed for. The study tool has been implemented in HTML and JavaScript and used the Bootstrap framework. JavaScript is restricted to act within the browser environment and can not access to the host computers file system. To store the data, we created a Java server which runs on the same machine, communicates with the JavaScript part of the implementation over HTTP requests and saves the results in a .csv-file. In addition to that, we used an eye-tracker to see on which parts of the input methods the participants focused.

## 7.2. Tasks and Questionnaire

We prepared three different tasks for the participants which the users had to solve with each of the five input methods. The order of the tested input methods was random to reduce the influence of learning effects during the study execution. Figure A.1 shows a screenshot of Task 2 for the Dynamic Range Bar Best Estimate Slider input method.

### Task 1: "Using the Public Transport"

We used the example task from the online survey as the first of three tasks for the user study to be able to compare the results. We slightly modified the description of this task so that it is formulated similar to the other two tasks. The first task was to specify how often a sample person uses public transportations to go to work per month. A table showed how often the public transportation was used in the last 36 months. Based on these information the participants were asked to use the input methods to answer this question. We prepared five different tables which were presented randomly for the different input methods.

### Task 2: "Buy, Buy, Buy"

The second task was to specify how much money the participant spends in the supermarket. We believe that some people have problems with processing the information in the table of task 1 and therefore we created a task where the participant are asked to enter data which they already know and which they do not have to process and interpret first. The disadvantage is that we can not measure the effectivity for this task, since we can not proof if the entered values are correct.

### Task 3: "Alea Iacta Est"

The third task had the goal to ask a question about a topic the participant probably know and where we can check the answer for its correctness. We chose to ask a question about the possible outcomes of dice rolls. Multiple rolls produce a probability distribution which can be well specified with the help of our input methods.

### System Usability Scale

After the three tasks for one of the input methods we asked participants to answer the ten-item Likert Scale questions of the System Usability Scale [?]. John Brooke describes it as a „quick and dirty, reliable tool for measuring usability“. This way we measure the system usability in a subjective way that allows us to compare the result to other studies which made use of System

## 7. User Study

---

Usability Scale. In order to modify the questions to fit for the evaluation of input methods, instead of whole user interfaces, we adapted the items as follows:

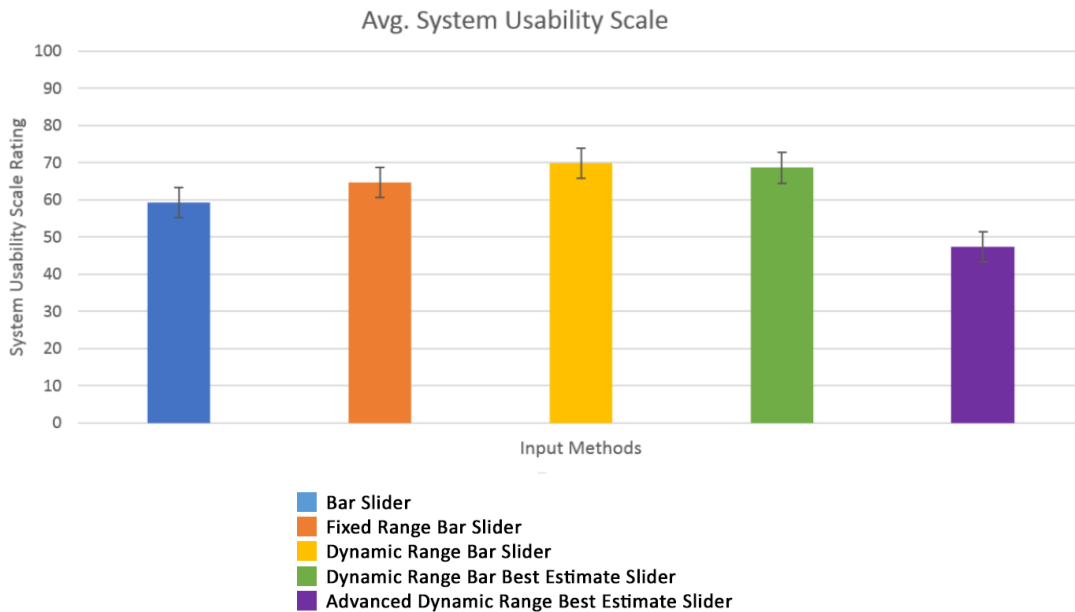
1. I think that I would like to use this input method frequently.
2. I found the input method unnecessarily complex.
3. I thought the input method was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this input method were well integrated.
6. I thought there was too much inconsistency in this input method.
7. I would imagine that most people would learn to use this input method very quickly.
8. I found the input method very cumbersome to use.
9. I felt very confident using the input method.
10. I needed to learn a lot of things before I could get going with this input method.

### 7.3. Participants

We had 14 people participating in this study. 5 of the 14 participants were female and they were from 21 to 45 years old. Amongst the participants were nine students, one person seeking work, three employees and one person preferred not to tell. One participant specified to be very unfamiliar with web-based input methods while a majority of 12 participants claimed to have average or good knowledge about the usage of such input methods. Only three participants stated to be very familiar, even with extended web-based input methods. All the participants specified at least the Abitur as their highest educational degree, three already finished their Bachelor Thesis, one participant had a Master degree and one even had a doctorate. Two participants stated to know nothing about stochastics and another to know at least nothing about statistics. Four other participants knew nothing about probability theory, but they had at least basic knowledge about statistics. The other eight participants specified to know about both, statistics and probability theory and even about probability distributions.

### 7.4. Results

The System Usability Scale allows the calculation of a value between 0 and 100 which denotes the usability. Based on the answers from this user study we created a SUS scale for the five subjected input methods. *Figure 7.1* shows these scales. The *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider* performed best and *Advanced Dynamic Range Best Estimate Slider* gave the worst result. The *Bar Slider* and *Fixed Range Bar Slider* performed



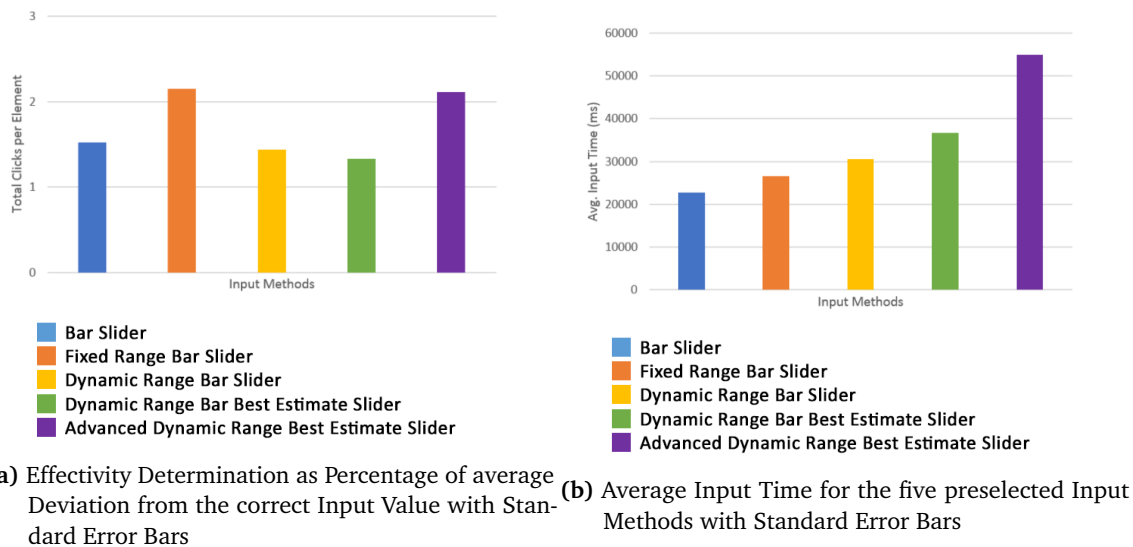
**Figure 7.1.:** Average System Usability Scale Results with Standard Error Bars for the the five preselected Input Methods

worse as expected. This does not confirm the assumption from *Section 3.4.2.1 – Expressiveness vs. Usability*.

We measured effectivity by determining how far participants' inputs differed from the correct input values. For *Task 2: "Buy, Buy, Buy"* we could not do that, because the users were asked to enter facts about their personal behaviour which we can not verify. *Figure 7.2a* shows the average deviation from the correct value in percent. As expected the inputs for the *Advanced Dynamic Range Best Estimate Slider* were tremendously worse than for the other input methods. The *Dynamic Range Bar Slider* and the *Dynamic Range Bar Best Estimate Slider* showed similar results, speaking for the *Dynamic Range Bar Best Estimate Slider*, since it provides higher expressiveness. It is surprising that the lowest deviations were measured for the *Fixed Range Bar Slider*, which performed badly for usability measures. The predefined range seems to have helped the participants to a great extent, especially for *Task 1: "Using the Public Transport"*.

The *Fixed Range Bar Slider* scores well for the measures help time and input time, too. People made use of the help description for the shortest average time amongst the five subjected input methods. This suggests that it was quite intuitive and easy to use. Likewise, the *Dynamic Range Bar Best Estimate Slider* performed quite well and required even less help than the *Dynamic Range Bar Slider*. While the average input times show an expected pattern (see *Figure 7.2b*), namely higher input times for higher expressiveness levels, the average total clicks per input element chart contains more striking information (see *Figure 7.5*). The *Dynamic Range Bar Best Estimate Slider* required the least clicks per input element and the higher amount of modifiable

## 7. User Study



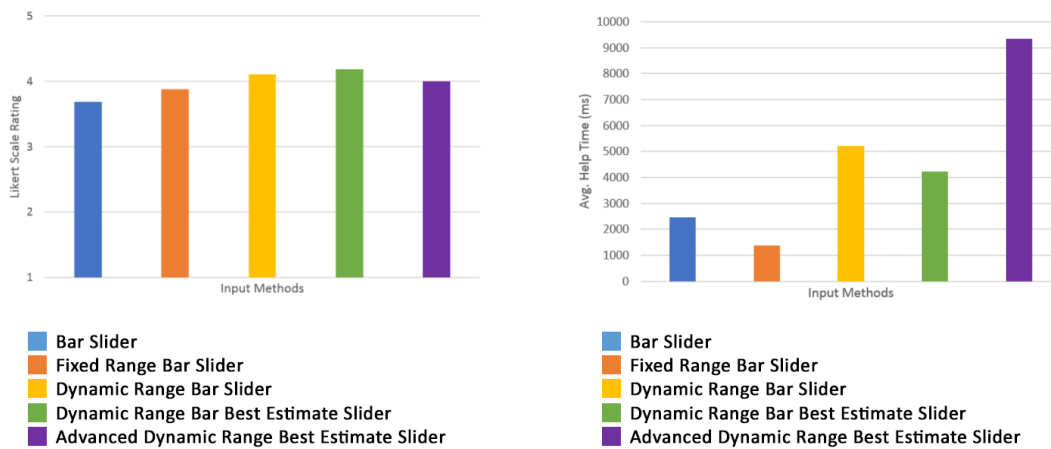
**Figure 7.2.:** Average Input Time and Effectivity Results for the five preselected Input Methods

elements relativizes the slightly higher input time compared to *Dynamic Range Bar Slider* and *Fixed Range Bar Slider*. Moreover the average total clicks per input element for the *Fixed Range Bar Slider* are even higher than for the *Advanced Dynamic Range Best Estimate Slider*. Maybe the modification of one input element to specify two values, namely the minimum and the maximum, leads to several rapid modifications until a best-fit has been found. Still the required input time is the second lowest. *Figure 7.4b* shows the average time the participants made use of the help function for the three different tasks in this user study. The tasks were always presented in order and therefore we can see that the more frequently the input method has been used the lower the required help dropped.

Participants stated to be most confident for inputs with the *Dynamic Range Bar Slider* and the *Dynamic Range Bar Best Estimate Slider*, whereby the confidence was actually highest for the *Dynamic Range Bar Best Estimate Slider* (see *Figure 7.3a*). It is interesting that users were less confident when working with the *Bar Slider* and the *Fixed Range Bar Slider*, although they provided more decent values for effectivity (see *Figure 7.2a*). Also the participants were quite confident for their inputs with the *Advanced Dynamic Range Best Estimate Slider*, although the results were worst for this input method. *Figure 7.4a* shows the average confidence for the three task blocks. The low confidence for *Task 1: "Using the Public Transport"* results from the relatively high difficulty of the assignment. *Task 2: "Buy, Buy, Buy"* was the only remit where the user was asked to enter data which they did not have to determine with the help of a table or by calculation. Therefore confidence was significantly higher.

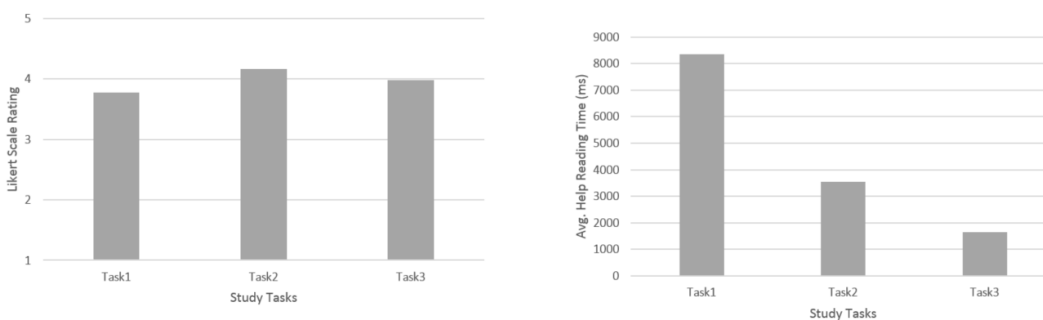
Values and comments about the *Bar Slider* suggest that people felt the input of a single best estimate value with the help of a slider too cumbersome and fiddly for high values and they





(a) Average Confidence Likert Scale Rating for the five preselected Input Methods with Standard Error Bars (b) Average Help Time for five preselected Input Methods with Standard Error Bars

**Figure 7.3.:** Average Help Reading Time and Confidence for five preselected Input Methods



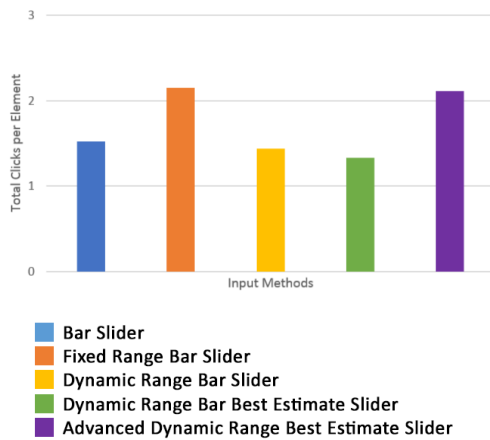
(a) Average Confidence Likert Scale Rating for the three Task Blocks of the User Study with Standard Error Bars (b) Average Help Reading Time for the three Task Blocks of the User Study with Standard Error Bars

**Figure 7.4.:** Average Help Reading Time and Confidence for the three Task Blocks of the User Study

stated that they would like to enter more data than this input method allows. Also the comments about the *Fixed Range Bar Slider* were mostly negative and described this input method as inflexible and unsatisfying. The *Dynamic Range Bar Slider*, which received the highest System Usability Scale, received mostly positive results and people explicitly stated that specifying a range is quite simple and satisfying. However the design of this input method often required the usage of the help button to determine the scaling functionality of the slider bar. The low amount of clicks per input element, the usability, the input time and the help time results plead for the *Dynamic Range Bar Best Estimate Slider* as a very good candidate. Most participants named this input method when asked for the one they liked most and which

## 7. User Study

---



**Figure 7.5.:** Average total Clicks per Input Element for the five preselected Input Methods with Standard Error Bars

one they consider as the best one. In contrast to that the participant liked the *Advanced Dynamic Range Best Estimate Slider* least. Many recognized the trade-off between usability and expressiveness and stated that the complexity of the *Advanced Dynamic Range Best Estimate Slider* does not make up for its higher expressiveness.

General comments suggest that some people prefer playing around and trying out the input methods instead of using a description text or descriptive popup toolboxes. Those participants who preferred using the help function said that they learned to use the input methods fast and barely required any help for the last task, after using the according input methods twice.

### 7.5. Discussion

We have seen in *Section 3.1 – Humans Dealing with Uncertainty* that less than 30% of the population in Germany acquires university entrance qualification. Since all the participants in this user study specified to have at least obtained the Abitur degree, this is a limitation of this study, because we only cover a small, well-educated proportion of the population in Germany. Another limitation is the low amount of participants, which did not allow a statistical analysis with significant results.

The participants showed mainly two different behaviours when it comes to the determination of the input methods functionality. While some users preferred to play around and try it out until they understood how it worked, others made use of the provided help description. There was no significant demographic tendencies for one of the two behaviours. In order to make the input methods accessible for all people the intuitivity has to be increased to also support

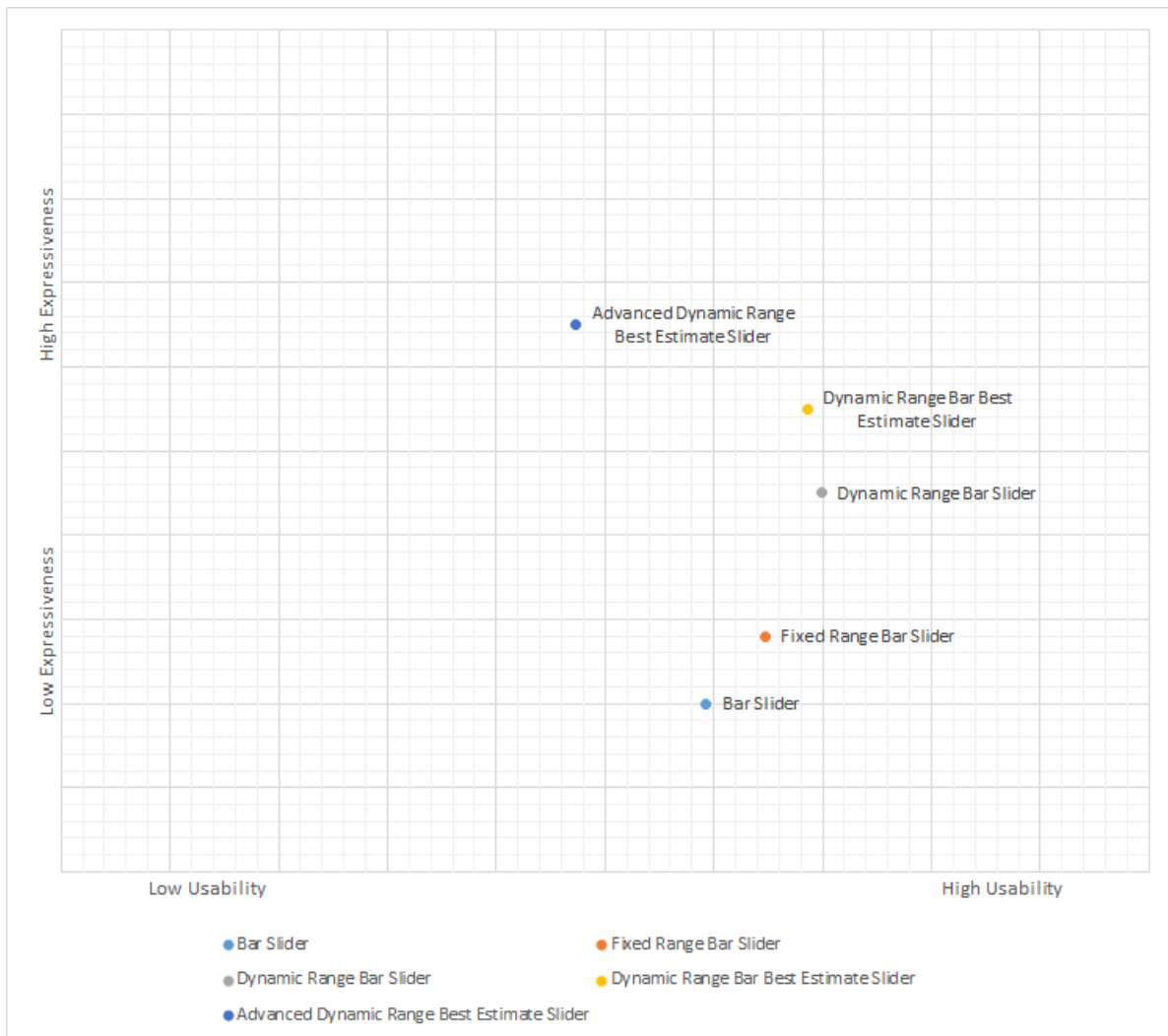
such users who prefer trying out over reading a description. However, for those who read the descriptions, the input methods were quickly understood in most cases.

The low confidence for the *Bar Slider* and the *Fixed Range Bar Slider* might be due to the demand of specifying more values than these input methods allowed. They communicated this with the help of their confidence specification. This also explains the relatively high confidence for the *Advanced Dynamic Range Best Estimate Slider*, although the effectivity results would suggest a much lower confidence. The possibility of specifying many values raises the confidence and the feeling that at least a part of the input is correct. The higher confidence for *Task 2: "Buy, Buy, Buy"* suggest a higher confidence for the input methods than *Figure 7.3a* shows, because such input methods are used for the input of data the users know instead for the input of data which the user has to determine with the help of a table or by calculation.

While the *Bar Slider* performed quite badly in this study the *Fixed Range Bar Slider* showed surprisingly good results, especially for effectivity. Therefore, in contrary to the conclusions after the online survey, there seems to be a place for this input method when the value range is known beforehand. Still, users showed little satisfaction and confidence when using this input method, which is enforced by the high amount of correction clicks the users did, and therefore one have to weigh up its application. The *Dynamic Range Bar Slider* seems to be a good flexible choice and showed the highest usability amongst the subjected input methods, but the relatively high help time and the participants comments advice to do some revision. Comparable results for usability and effectivity received the *Dynamic Range Bar Best Estimate Slider*, while providing a higher expressiveness level, making this also a good candidate. The *Advanced Dynamic Range Best Estimate Slider* performed badly for the most measures.

### 7.5.1. Expressiveness vs. Usability Estimation after User Study

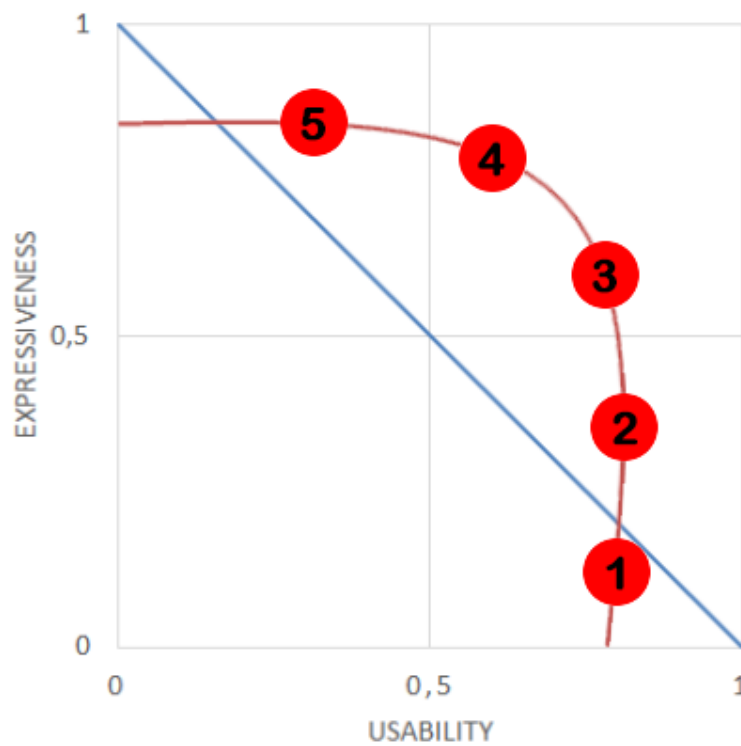
The estimation of the usability and expressiveness of the various input methods, presented in *Section 4.2 – Expressiveness vs. Usability Estimation* and the estimation after the online survey in *Section 6.5 – Discussion* generally agreed. After considering the user System Usability Scale and the new information we received from the user study we can provide a new estimation for the relation of usability and expressiveness for the five subjected input methods (see *Figure 7.6*). Compared to the initial assumption from *Section 3.4.2.1 – Expressiveness vs. Usability* the chart shows a different pattern. It seems that once the users are aware of the amount of information that they can provide about uncertain data, they felt unsatisfied with the determination of a single best estimate or a fixed range and while trying to find an input value that fits the underlying distribution best, the input methods appear to be cumbersome and unsuitable, resulting in a low usability rating. This explains the low rating for the *Bar Slider* and the *Fixed Range Bar Slider*. Still the assumption that a too high complexity due to high expressiveness leads to lower usability. *Figure 7.7* shows an overhauled estimation for the general relation between usability and expressiveness.



**Figure 7.6.:** Expressiveness vs. Usability Estimation for preselected Input Methods after Online Survey and User Study. Usability Values are a Combination of the SUS Results, the Online Survey Results and the initial Testings.

## 7.6. Summary

We described the setup and the conduction of a user study in this chapter. After the outline of the study environment and tasks we subjected the participants of this study. After presenting and drawing conclusions from the results we compared the estimation of the subjected input methods usability and expressiveness relation after this study to the estimations after and before the online survey. Based on that we updated the general assumed pattern for the general relation of expressiveness and usability for input methods.



**Figure 7.7.:** Updated Expressiveness vs. Usability Pattern Estimate, based on the Results from the Online Survey and the User Study.

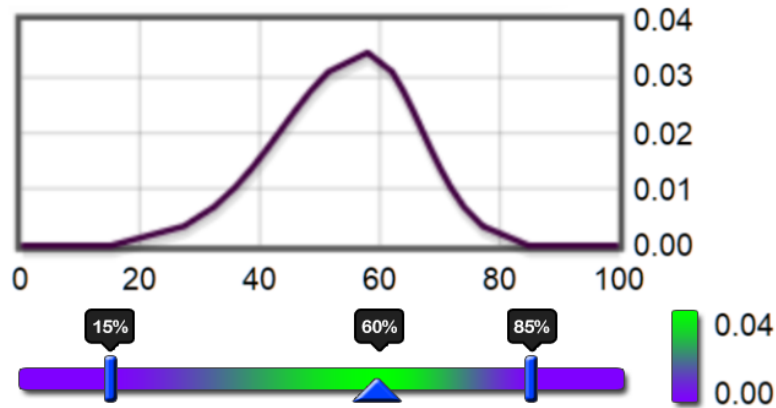


## 8. Further Steps on the Basis of Survey and Study Results

Based on the results of the online survey (see *Chapter 6 – Online Survey*) and user study (see *Chapter 7 – User Study*) we discuss further steps to improve or alter the tested input methods in this chapter.

### 8.1. Further Enhancements and Alternatives

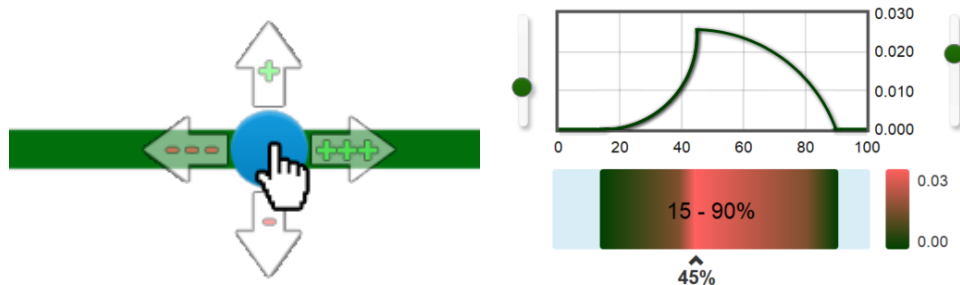
The user study revealed that, while some people make use of helping descriptions and describing toolboxes, others completely ignore those and like more to play around and try out until they figured out how the input method works. We experienced that the more scientific their educational and employment background is, the more likely they use descriptions and help texts in order to determine how to use an input method. Therefore we have to improve the intuitivity and reconsider if the *Advanced Dynamic Range Best Estimate Slider* is a viable alternative, since this one could barely be used without additional help. There are several starting points for better intuitivity, which we get from the comments of the participants of the online survey and the user study. People often demanded the input methods to look more like ordinary web sliders. This leads to some problems. We originally changed the design for two reasons: firstly, the size of the slider element leads to coverage of the slider background gradient and, secondly, it is hard to see the colour gradient on a thin slider background bar. Some participants suggested that the colour gradient should also be applied to the area over the maximum and below the minimum, since these areas have a height of 0 on the probability density function. We cut the gradient at the borders in order to be able to allow the determination of the borders. Another problem which has been mentioned frequently was the inconsistency of interactions. While the borders are changed by scaling, the node elements have to be dragged. Technically this is the same action but for the human brain these are different interactions. *Figure 8.1* shows a modified version of the *Dynamic Range Bar Best Estimate Slider*, which contains the changes we suggest to overcome the mentioned problems. These changes are not yet implemented, because this should not be done without consecutively testing and evaluating. We reduced the height of the slider background bar to be closer to the size of an ordinary web slider, but kept it big enough to enable the determination of the probability density with the help of the colour gradient. The slider elements at the upper and lower bound look more like those of an ordinary web-slider, but we kept them smaller



**Figure 8.1.:** *Dynamic Range Bar Best Estimate Slider* with further Enhancements based on the Results of the Online Survey and User Study

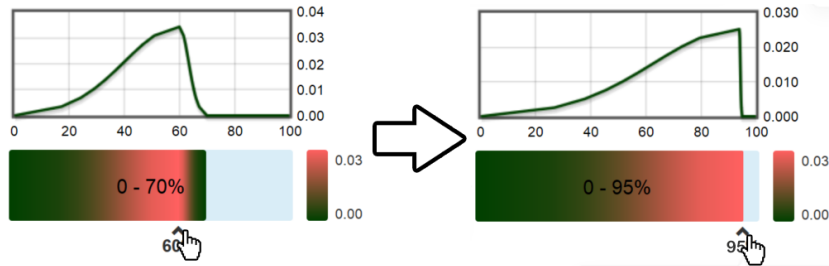
in width to cover less of the slider background. With this change, we can expand the colour gradient to cover the whole background, since the border positions can now be determined by the slider elements. By the placement of slider elements for the borders, now all input elements can be changed by dragging and therefore the interaction is more consistent. Some users also complained that the numbers in the center of the slider background are hard to read, depending on the chosen gradient colours, and that sometimes the mouse overlaid the numbers close to the drag elements. In addition to that in older browsers, where the input number text could not be set to be unselectable by CSS, this leads to problems in working with the input method when sliding or scaling, while the text was selected. Therefore we put all the input number representation into separate boxes with enough space between them and the drag elements to prevent overlay of relevant numbers and to support older browsers. Additionally, the participants of the online survey and user study mentioned that it is not always possible to enter the exact value of desire. This refers to input method limitations in accuracy as we introduced in *Section 3.2 – Uncertainty Occurrences when Dealing with Simulations*. The precision of web sliders are usually bound to their size in pixels. For extended data ranges this leads to the problem that the value steps become quite big. In the user study the input methods for *Task 2* (see *Section 7.2 – Task 2: "Buy, Buy, Buy"*) had a step size of five only. In this case, it still allowed the specification of the correct values, but in other cases this might not be sufficient. In addition to that, participants experienced this task to be too fiddly. In order to enable the user to specify values more precisely, we modified the basic functionality of the slider elements. The values can be slid horizontally, as usual, but in addition that the value can be changed in lower steps with a vertical movement (see *Figure 8.2a*). This way we can overcome pixel precision and help the user to easily choose the desired input value. For the *Dynamic Range Bar Best Estimate Slider* and the *Advanced Dynamic Range Best Estimate Slider* people commented that it was cumbersome that the slider node positions were bounded by





(a) Precise Slider Improvement by Vertical Dragging (b) Alternative Approach for *Advanced Dynamic Range Best Estimate Slider* with vertical Sliders to modify the Kurtosis

**Figure 8.2.:** Precise Slider Improvement by Vertical Dragging and alternative Approach for *Advanced Dynamic Range Best Estimate Slider* to modify Kurtosis



**Figure 8.3.:** Slider Node moving its Neighbour Node when Dragging over its Borders

their neighbour slider elements and therefore were blocked when they got too close to the next node on the right or left. The neighbour node has to be moved first, in order to move a node to a value which lays behind this node. As an enhancement, we suggest that a node also moves its neighbour when moving over its borders (see *Figure 8.3*). The *Advanced Dynamic Range Best Estimate Slider* performed quite badly, especially for laymen. The determination of two nodes with the half height of probability density functions peak was inconvenient to those who knew about probability distributions and incomprehensible for most of the others. Consequently this input method is more suitable for experts. However, such users commented in the online survey and user study that they preferred to draw the probability density function freely or to determine the kurtosis in another way. *Figure 8.2b* shows an alternative approach for the *Advanced Dynamic Range Best Estimate Slider* with vertical sliders to change the kurtosis.

### 8.2. Summary and Discussion

We presented some possible modifications for the preselected input methods, based on the results of the online survey and the user study. We believe that these enhancements may improve the usage significantly and solve many problems people often encountered. For the different input methods we gave a suggestion in which contexts they are worth considering for application. *Dynamic Range Bar Slider* and *Dynamic Range Bar Best Estimate Slider* seem to be suitable for both, laymen and experts and provide relatively high expressiveness for their usability and therefore they are the input methods which are the most flexible for both, different kinds of application case and different kind of users. *Advanced Dynamic Range Best Estimate Slider* was too complicated for laymen and for experts other input methods likely provide better results. *Fixed Range Bar Slider* performed quite badly and might only be worth considering for very special cases, but we suggest to always prefer *Dynamic Range Bar Slider* over this one.

## 9. Summary and Conclusions

Before presenting our conclusions we provide a brief summary of the aims of this thesis and what we have done to achieve the milestones. At last we point out an outlook for future work and possible applications for the gathered knowledge and the developed input methods. For the preselected input methods, we give a suggestion in which scenarios it is advised to apply them and discuss their general suitability.

### 9.1. Summary

After providing some background information and inspecting related work dealing with input methods for data with uncertainty, the topic of this thesis, and such articles dealing with one of the subtopics *uncertainty*, *visualization* and *input methods*. To satisfy the demands of the main task, which was to design and develop input methods, we created various input methods with different main focuses and properties. These input methods were classified with the help of a classification technique we developed and presented in *Chapter 3 – Inspecting Uncertainty and Input Methods*, where we also analysed uncertainty occurrences in simulations and explained a central assumption of this thesis: the higher an input methods expressiveness gets, the lower its usability drops. We chose a preselection of the most promising input methods for further inspection and evaluation. To evaluate the input methods and to review our assumption about the relationship between expressiveness and usability we conducted an online survey and a user study for augmented versions of the preselected input methods. Based on the information gathered in the survey and study we determined the best candidates for various scenarios and suggested some possible enhancements in *Section 8.1 – Further Enhancements and Alternatives*. Besides the development of input methods the creation of a web-based simulation tool, which we presented in *Chapter 5 – Web-Based Simulation Tool* and which we implemented by mainly using Django, Bootstrap, Python, HTML, JavaScript, jQuery, Ajax and CSS, was demanded.

### 9.2. Conclusions

There is not much related work in this area, although this is a broad and interesting subject area which is quite important when it comes to providing appropriate input methods for data with uncertainty, for example in the context of simulations. Input methods for uncertain data play a

special role in simulations. We have discussed the various sources of uncertainty in simulations and it is highly recommended to consider these sources of uncertainty when processing simulation input data. We revealed that finding the most suitable input method depends on many things, mainly on the users' expected knowledge about stochastics, the required expressiveness and usability and the input data's reference class. It is generally important to find the best trade-off between expressiveness and usability and between generalization and specialization. For the five preselected input methods, which we subjected in the online survey and user study, the *Dynamic Range Bar Slider* and the *Dynamic Range Bar Best Estimate Slider* turned out to be the best prototypes in overall rating, whereas the *Fixed Range Bar Slider* and the *Advanced Dynamic Range Best Estimate Slider* can still be valid choices for some special scenarios. We discussed the input methods' suitability for different scenarios more detailed in *Section 9.3 – Suitability for different Scenarios*. We believe that with the upgrades, proposed in *Section 8.1 – Further Enhancements and Alternatives*, these input methods show good results, considering the few design and development iterations. The selection of presented input methods, the classification techniques, and the tested input methods form a good basis for further research in this subject area. The assumption from *Section 3.4.2.1 – Expressiveness vs. Usability* has been mostly confirmed by the online survey, but the results from the user study demanded an adjustment of the expected relations between usability and expressiveness for input methods for data with uncertainty. Generally it can be said that the higher the expressiveness gets, the lower the usability drops. However, when the expressiveness is too low the usability suffers from the low effectivity and satisfaction, since the users want to input more information the input method allows. This is likely also the reason for the bad results for the *Fixed Range Bar Slider*, since users felt restricted by the fixed range and therefore they could not input the desired data in a sufficient expressive way.

### 9.3. Suitability for different Scenarios

Here we give a suggestion about in what situations to use the input methods which we subjected in the online survey and in the user study, based on our experiences we made in our survey, study, tests and discussions.

**Bar Slider:** This input method was a deputy for ordinary web sliders. Though, the information provided is quite low nearly everyone has already used such sliders and therefore it can be used as a backup whenever the other input methods do not come into question. However, based on the data an ordinary number input box might provide similar or even better results, but this is very situational.

**Fixed Range Bar Slider:** The *Fixed Range Bar Slider* was not very popular, neither amongst the participants of the survey and study, nor amongst developers. People did not feel satisfied with the interaction and they were inconfident of when the range size did not fit their expectations. We suggest to always prefer *Dynamic Range Bar Slider* over the *Fixed Range Bar Slider*, except

for some very special situations where the range is already known beforehand, e.g. for normal distribution with known standard deviation but with unknown mean.

**Dynamic Range Bar Slider:** This input method is appropriate for the most cases. Most people, even those without much knowledge about stochastics and with little familiarity with web-based input methods, were quite capable of using and correctly entering data with the help of this input method. Specifying a minimum and maximum value seems to be a simple and intuitive task most people could easily solve. In addition to that usability and confidence were relative high for the *Dynamic Range Bar Slider*.

**Dynamic Range Bar Best Estimate Slider:** The *Dynamic Range Bar Best Estimate Slider* had good results in usability tests. Most of our survey and study participants were at least slightly familiar with probability distributions. Laymen might perform worse since this kind of input method is not common. However, we experienced a very high learnability for this input method and in most cases people could cope with it after short testing. We believe that this input method is, after the changes suggested in *Section 8.1 – Further Enhancements and Alternatives*, in most situations, superior compared to *Dynamic Range Bar Slider*, except for the case of symmetric distributions where the mode can be determined with the help of the minimum and maximum.

**Advanced Dynamic Range Best Estimate Slider:** In most situations this input method provided bad results for both, usability and effectivity. For people who often work with probability distributions the *Advanced Dynamic Range Best Estimate Slider* might work out, but such people would likely prefer drawing the probability density function freely by themselves or at least use the proposed alternative in *Figure 8.2b*. Therefore we suggest to use this input method rarely or never and better stick with *Dynamic Range Bar Best Estimate Slider* or input methods for experts.

## 9.4. Outlook

First of all there are many other possible input method designs and concepts which can be included and developed, especially for the input of data that refer to the reference classes besides uncertainty in the form of probability. But there are also more possibilities for input methods for probabilities. In addition to that we had some more input method concepts which we did not present in this thesis. These and more prototypes can be classified with the help of our classification techniques and implemented. More surveys and studies could help to compare the different input approaches and to determine the best candidates for different scenarios, which can be described with the help of the classification techniques and the input methods usability and expressiveness. The classification techniques might be complemented by more categories or be altered to be more fine grained, after gathering more experience with such input methods. The five input methods, subjected in the online survey and in the user study, can be toolled up with the suggested enhancements and further test to reveal their impact

## 9. Summary and Conclusions

---

and to determine more required changes should be conducted. The simulation tool is still in an early stage and does not yet provide tools to ease the creation of input method setups visually and more templates for the creation of different kinds of simulations or even a tool that helps creating simulation implementations. New developed input methods and more output methods are to be implemented to improve this simulation tool. Another beneficial functionality would be the possibility to do multiple runs and to visualize intermediate results.

# A. Appendix

User Study - Input Methods

STEP 2/20

---

### Kaufen, kaufen, kaufen!

Ein Marktforschungsinstitut interessiert sich für das Einkaufsverhalten von Studenten. Es wurde bereits ermittelt wie häufig und wo gerne eingekauft wird. Die Produktauswahl und der Service sollen auf Studenten abgestimmt werden. Um das Einkaufsverhalten eines Studenten zu simulieren werden noch weitere Informationen benötigt. Insbesondere wie viel ein Student pro Einkauf ausgeben ist von Interesse.

### Aufgabe

Gebe, falls es die Eingabemethode zulässt, folgendes an:

- **Wie hoch ist der wahrscheinlichste Betrag den Sie pro Einkauf ausgeben?**
- **Wie viel geben Sie mindestens/höchstens pro Einkauf aus?**
- **Für welchen Betrag ist die Wahrscheinlichkeit halb so hoch als für den wahrscheinlichsten Betrag?**

Mache deine Eingabe...

Hilfe ausblenden

Information

Probability Density Function Plot

Probability Density

Scaling Bar representing min-max range

Color - Probability Density mapping

Best Estimate (Mode - not Median or Mean)

Fertig?

Meine Eingabe ist bestimmt richtig!

Ich denke meine Eingabe ist zumindest fast richtig!

Ich bin mir nicht sicher, ob meine Eingabe korrekt ist?!

Ich bezweifle, dass meine Eingabe richtig ist!

Ich verstehe das nicht!

• Skalriere den Balken so, dass er den **Bereich der möglichen Werte** so gut wie möglich abdeckt.  
Durch die Skalierung des Balkens kann **untere- bzw. obere Grenze** des Wertebereichs angegeben werden.

• Benutze außerdem den Schiebegriff um den **wahrscheinlichsten Schätzwert**, nicht den Durchschnitt, anzugeben.  
Mathematisch ausgedrückt: Gib' den Modus an, nicht den Mittel- oder den Zentralwert(Median).

Figure A.1.: Screenshot from User Study, Task 2, Dynamic Range Bar Best Estimate Slider

---

**Listing A.1** Test Script Django View Function

---

```
# imports
...
# IMPORT simulation base
from apps.simulation.sim import simulation_base as sim_base
# IMPORT simulation register
from apps.simulation.sim import reg_sim as reg

def simulation_ajax(request):
    """ Send http response in json by ajax to respond the request sent by javascript

    PARAMS:
        request      Http request

    RETURN:
        Response to request depending on input processing
    """
    # Check for csrf token and required dictionary keys first
    if 'csrfmiddlewaretoken' in request.POST and 'uniqid' in request.POST and 'ident' in
    request.POST:
        # Create input dictionary
        input_dict = {}
        for key in request.POST:
            input_dict[key] = request.POST[key]
        # Try to load simulation - error response if not found
        sim = reg.get_sim_by_ident(request.POST['ident'])
        if sim == None:
            # If this simulation does not exist respond with html template
            return render_to_response('simulation_tool/uist_simulation.html',
                context_instance=RequestContext(request))
        else:
            # Start calculation
            outputdict = sim.run_simulation(input_dict,
                request.POST['csrfmiddlewaretoken'], request.POST['uniqid'])
            # Calculation finished - delete progress in list
            sim_base.UIST_Simulation.del_progress(request.POST['csrfmiddlewaretoken'],
                request.POST['uniqid'])
            # Return calculation result output dictionary
            return HttpResponse(json.dumps(outputdict))
    else:
        # On error respond with html template
        return render_to_response('simulation_tool/uist_simulation.html',
            context_instance=RequestContext(request))
```

---



---

## Listing A.2 Python Function to calculate Square

---

```
""" .....
NAME:      apps/testpyscript/scripts/runscript_test.py
AUTHOR:    Marius Kleiner
DATE:      15.01.2015
MODIFICATION LOG:
    19.01.2015 [Marius Kleiner]   Added documentation
.....
Simple test script that runs server-side for testing.
..... """

def square(x):
    """ Calculate square of parameter x

    PARAMS:
        x          Input parameter to calculate square from
    RETURN:      Square of given parameter
    """
    return int(x) * int(x)
```

---

**Listing A.3** Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 1

---

```
""" -----
NAME:      apps/simulation/sim/simulation_base.py
AUTHOR:    Marius Kleiner
DATE:      15.01.2015
MODIFICATION LOG:
            18.01.2015 [Marius Kleiner]   Added documentation
-----

This file contains the simulation base class and some commented dummy constants that
are required for any simulation.
----- """

# IMPORTS abstract class
import abc

class UIST_Simulation(abc.ABCMeta):
    """ Abstract simulation base class that every simulation should be derived from.

    Contains various class methods that provide simulatio information access for
    other functions (like the django view.py and template-tags).
    """

    #: Simulation identifier
    IDENT = '<simulation identifier>'
    #: Simulation name
    NAME = '<simulation name>'
    #: Simulation description (short)
    DESC = '<A brief description of the simulation>'
    #: List of simulation input methods
    INPUTS = [
        {
            'type_id' : '<input method type id>',
            'title' : '<display title for this input>',
            'desc' : '<brief description of this input>',
            'params' : ['<unique identifier>', '...']
        },
        # ...
    ]
    #: List of simulation outputs
    OUTPUTS = [
        {
            'type_id' : '<output type identifier>',
            'title' : '<display title for this output>',
            'desc' : '<brief description of this output>',
            'params' : ['<unique identifier>', '...']
        },
        # ...
    ]
]
```

---

---

**Listing A.4** Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 2

---

```
#: Simulation error identifier(s)
ERROR_ON_RUN = 'SIMULATION_ERROR'
#: Simulation error dictionaries
ERROR_KEY_MISSING = { ERROR_ON_RUN : 'At least on input key, that is required for this
    simulation, is missing!'}

#: Progress dictionary stores the progress(%) of all running simulations
prog_dict = {}

@classmethod
def get_ident(cls,*args,**kwargs):
    """ Get identifier for simulation class

    RETURN:    Simulation identifier
    """
    return cls.IDENT

@classmethod
def get_name(cls,*args,**kwargs):
    """ Get name for simulation class

    RETURN:    Simulation name
    """
    return cls.NAME

@classmethod
def get_desc(cls,*args,**kwargs):
    """ Get short description for simulation class

    RETURN:    Simulation description (short)
    """
    return cls.DESC

@classmethod
def get_inputs(cls,*args,**kwargs):
    """ Get input method information for simulation class

    RETURN:    Simulation input method info
    """
    return cls.INPUTS
```

---

---

**Listing A.5** Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 3

---

```
@classmethod
def update_progress(cls, csrf_token, uniqid, value):
    """ Update progress of simulation instance

    PARAMS:
        csrf_token    Django CSRF token
        uniqid        A unique id for this simulation instance
        value         New progress value for simulation instance
    """
    old_value = UIST_Simulation.prog_dict.get(csrf_token + uniqid, 0)
    if value > old_value and value <= 100:
        UIST_Simulation.prog_dict.update({ csrf_token + uniqid : value })

@classmethod
def get_progress(cls, csrf_token, uniqid):
    """ Get progress of simulation instance

    PARAMS:
        csrf_token    Django CSRF token
        uniqid        A unique id for this simulation instance
    RETURN:    Simulation instance progress value
    """
    return UIST_Simulation.prog_dict.get(csrf_token + uniqid, 0)

@classmethod
def del_progress(cls, csrf_token, uniqid):
    """ Delete progress of simulation instance

    PARAMS:
        csrf_token    Django CSRF token
        uniqid        A unique id for this simulation instance
    """
    UIST_Simulation.prog_dict.pop(csrf_token + uniqid, None)

@abc.abstractmethod
def run_simulation(cls, input_dict, csrf_token, uniqid):
    """ Run simulation of derived simulation class

    PARAMS:
        input_dict    Dictionary containing input parameters
        csrf_token    Django CSRF token
        uniqid        A unique id for this simulation instance
    RETURN:    Simulation result output dictionary
    """
    pass
```

---

---

**Listing A.6** Python Simulation Base File (All new Simulations have to be a Derivation of this Class) Part 4

---

```
@classmethod
def input_keys_exist(cls, input_dict):
    """ Check if all input keys exist in the input dictionary

    PARAMS:
        input_dict      Dictionary containing input parameters
    RETURN:   If all input keys exist
    """
    for im in cls.get_inputs():
        params_ary = im.get('params', None)
        if params_ary == None:
            return False
        else:
            if input_dict.get(params_ary[0], None) == None:
                return False
    return True

@classmethod
def get_outputs(cls,*args,**kwargs):
    """ Get output method information for simulation class

    RETURN:   Simulation output method info
    """
    return cls.OUTPUTS
```

---

---

**Listing A.7** Python Simulation Register File

---

```
""" .....
NAME:      reg_sim.py
AUTHOR:    Marius Kleiner
DATE:      15.01.2015
MODIFICATION LOG:
            18.01.2015 [Marius Kleiner]   Added documentation
.....

The created simulations have to be registered here in order to be recognized by the
website. In order to do so import the simulation file (in the sim folder) and add
the simulation to the SIMULATION_LIST array constant.
..... """

# IMPORTS simulations
from apps.simulation.sim import simulation_bullseye
# ...

#: Contains all the registered simulations
SIMULATION_LIST = [
                                # Hit the bulls-eye (test) simulation
                                simulation_bullseye.Simulation,
                                # ...
                                ]

def get_sim_by_ident(ident):
    """ Get the simulation class matching the given identifier.

    PARAMS:
            ident      Simulation identifier
    RETURN:
            Matching simulation class or None if no match
    """
    for sim in SIMULATION_LIST:
        if ident == sim.get_ident():
            return sim
    return None
```

---

---

### Listing A.8 Client-Side of Test Script (JavaScript)

---

```
$(document).ready(function() {
    // On button click call function...
    /**
     * Start script (communication) on run-button click
     * @return {Boolean} False in any case
     */
    $("#button").click(function() {
        // Read input for x value from input-form
        var input_string = $("#x_input_form").val();
        // Send ajax request to server including the input data in json
        $.ajax({
            url : "/runscript_json",
            type : "POST",
            dataType: "json",
            data : {
                client_x_value : input_string,
                // IMPORTANT! Add the csrf_token that has also been loaded in the .html
                // template
                csrfmiddlewaretoken: '{{ csrf_token }}'
            },
            /**
             * On success write the result into desired html element
             * @param json {json dictionary} Result/Output dictionary
             */
            success : function(json) {
                document.getElementById("result_p").innerHTML = json.x_result;
            },
            /**
             * Throw error on fail
             */
            error : function(xhr,errmsg,err) {
                // error handling
            }
        });
        return false;
    });
});
```

---

### Listing A.9 HTML Snippet for the Inclusion of Input Method Templates with the Help of Django Template Tags

---

```
<!-- Input methods -->
{% for dict_item in inputs %}
    <h4>{{ dict_item.title }}</h4>
    {% include "input_templates/input_method_"|add:dict_item.type_id|add:".html" with
        params=dict_item.params %}
    <br><br>
{% endfor %}
```

---

**Listing A.10** Javascript Simulation Run Snippet

---

```
/**
 * Start simulation (communication) on run-button click
 * @return {Boolean} True in any case
 */
$("#button_run_simulation").click(function() {
    // Disable run simulation button and input methods
    this.disabled = true;
    disable_all_inputs();
    // Prepare input dictionary containing all input method values
    var input_dict = { 'dummy' : 'dummy' };
    {% for dict_item in ident|inputs %}
        input_dict['{{ dict_item.params.0 }}'] = call_by_name('get_input_value_{{
            dict_item.params.0 }}');
    {% endfor %}
    // Data dictionary (contains simulation identifier, csrf token and unique session
    identifier)
    var data_dict = { 'ident' : '{{ ident }}', 'uniqid' : uniqid, 'csrfmiddlewaretoken' : '{{
        csrf_token }}';
    for (var attrname in input_dict) { data_dict[attrname] = input_dict[attrname]; }
    // Send ajax request to server including the data dictionary (json)
    $.ajax({
        url : "/simulation_json",
        type : "POST",
        dataType: "json",
        data : data_dict,
        /**
         * On success display results
         * @param json {json dictionary} Output dictionary
         */
        success : function(json) {
            received_sim_result = true;
            update_progress(100);
            change_color_progress_bar_green();
            show_result_container();
            // Display outputs
            {% for output_item in ident|outputs%}
                call_by_name_with_param('set_output_value_{{ output_item.params.0 }}',
                    json.{{ output_item.params.0 }});
            {% endfor %}
        },
        ...
    });
    received_sim_result = false;
    // Spawn progress bar
    show_progress_bar_container();
    // Start loop for fetching progress
    progress_loop();
    // ...
    return true;
});
```

---



---

**Listing A.11** Example for the Javascript Part of the Input Method Implementation (Bar Slider)  
Part 1

---

```
/**
 * ID:                017
 * NAME:              Bar Slider
 * AUTHOR:            Marius Kleiner
 * DESCRIPTION:       Javascript for input method 017
 * NOTES:             -
 */
// Constants
var UIST_IM_FCA_ORIENTATION_HORIZONTAL = 'horizontal';
var UIST_IM_FCA_ORIENTATION_VERTICAL = 'vertical';
var UIST_IM_FCA_BE_HEIGHT = 24;
var UIST_IM_FCA_BE_MARGIN = 15;
// Global Variables
var uist_im_fca_element = null;
var uist_im_fca_mouse_down = false;
var uist_im_fca_pointer_pos = [0,0];
/**
 * Initialize fca elements
 */
$( document ).ready(
    function() {
        $( '.uist-im-fca-container' ).each(
            function() {
                var cont = $(this);
                uist_im_fca_setValue(cont, parseFloat(cont.attr('value')));
                // Reposition tooltips
                var target_ary = cont.find('div[' + TOOLTIP_UIST_ATTR_NAME + ']');
                for (i = 0; i < target_ary.length; i++) {
                    $(target_ary[i]).qtip('reposition');
                }
            }
        );
    }
);
/**
 * Set global mousedown variable to true only if mouse is over a fca border
 */
$(document).mousedown(
    function() {
        if (uist_im_fca_element != null ) {
            uist_im_fca_mouse_down = true;
        }
    }
);
/**
 * Reset mousedown variable on mouse up
 */
).mouseup(
    function() {
        uist_im_fca_mouse_down = false;
    }
);
```

---

**Listing A.12** Example for the Javascript Part of the Input Method Implementation (Bar Slider)  
Part 2

---

```
/**
 * On mousemove check if mouse is over fca border (and mouse is pressed)
 * If yes resize the fca element
 */
$( '.uist-im-fca-container' ).mousemove(
    function( event ) {
        if ($(this).attr( 'disabled' ) == 'true') {
            return;
        }
        // Relevant fca div elements
        var cont = $(this);
        var background_element = cont.children( '.uist-im-fc-resize-element-back' );
        var target_be = background_element.children( '.uist-im-fberc-best-estimate' );
        var p_be_element = background_element.children( '.uist-im-fberc-best-estimate' ).
            children( '.uist-im-fberc-freq-mark-xs' ).children('p');
        var txt_element = background_element.children( '.uist-im-fberc-value' ).children('p');
        // Attributes and Parameters
        var value_desc_pre = cont.attr('value_desc_pre');
        var value_desc_post = cont.attr('value_desc_post');
        var value_min = cont.attr('value_min');
        var value_max = cont.attr('value_max');
        var value_min_sel = cont.attr('value_min_sel');
        var value_max_sel = cont.attr('value_max_sel');
        var deltaX; var deltaY; var cursor;
        var be_value; var left_new; var bottom_new;
        var value_size = parseFloat(value_max) - parseFloat(value_min);
        var max_size_x = parseFloat(background_element.width());
        var max_size_y = parseFloat(background_element.height());
        var top_border_be = target_be.offset().top;
        var bottom_border_be = target_be.offset().top + target_be.height();
        var left_border_be = target_be.offset().left;
        var right_border_be = target_be.offset().left + target_be.width();
        var orientation = cont.attr('orientation');
        // Check if mouse is over BE line
        var be_hover_horizontal = (orientation == UIST_IM_FCA_ORIENTATION_HORIZONTAL) &&
            (event.pageY > bottom_border_be) &&
            (event.pageY < bottom_border_be +
                UIST_IM_FCA_BE_HEIGHT) &&
            (event.pageX > left_border_be -
                UIST_IM_FCA_BE_MARGIN) &&
            (event.pageX < right_border_be +
                UIST_IM_FCA_BE_MARGIN);
        var be_hover_vertical = (orientation == UIST_IM_FCA_ORIENTATION_VERTICAL) &&
            (event.pageX > right_border_be) &&
            (event.pageX < right_border_be +
                UIST_IM_FCA_BE_HEIGHT) &&
            (event.pageY > top_border_be -
                UIST_IM_FCA_BE_MARGIN) &&
            (event.pageY < bottom_border_be +
                UIST_IM_FCA_BE_MARGIN);
```

---

---

**Listing A.13** Example for the Javascript Part of the Input Method Implementation (Bar Slider)  
Part 3

---

```
// If mouse is over resize element border store this in the global es variable
if (be_hover_horizontal){
    cont.css('cursor', 'pointer');
    uist_im_fca_element = target_be;
}else if (be_hover_vertical){
    cont.css('cursor', 'pointer');
    uist_im_fca_element = target_be;
}else {
    uist_im_fca_element = null;
    cont.css('cursor', 'default');
}
// If mouse has been pressed and this target is the one with mouse over border...
if (uist_im_fca_mouse_down) {
    if ((uist_im_fca_element != null) && (target_be.get(0) ==
    uist_im_fca_element.get(0))) {
        // Calculate mouse position delta
        deltaX = uist_im_fca_pointer_pos[0] - event.pageX;
        deltaY = uist_im_fca_pointer_pos[1] - event.pageY;
        // Calc new pos & value
        left_new = parseFloat(target_be.css('left'));
        bottom_new = parseFloat(target_be.css('bottom'));
        if (orientation == UIST_IM_FCA_ORIENTATION_VERTICAL) {
            bottom_new = Math.min(Math.max(bottom_new + deltaY,
            target_lower.height()), target_upper.height());
            be_value = bottom_new * value_size / max_size_y;
        } else {
            left_new = Math.min(Math.max(left_new - deltaX, 0),
            parseFloat(max_size_x));
            be_value = parseFloat(parseFloat(value_size) * parseFloat(left_new)
            / parseFloat(max_size_x));
        }
        // Check if exceed min-/max-value
        if ( (be_value < parseFloat(value_min)) || (be_value >
        parseFloat(value_max)) ) {
            be_value = Math.min(Math.max(be_value, parseFloat(value_min_sel)),
            parseFloat(value_max_sel));
            if (orientation == UIST_IM_FCA_ORIENTATION_VERTICAL) {
                bottom_new = be_value * parseFloat(max_size_y) /
                parseFloat(value_size);
            } else {
                left_new = be_value * parseFloat(max_size_x) /
                parseFloat(value_size);
            }
        }
        // Update position and value
        be_value = be_value.toFixed(0);
        target_be.css('left', left_new);
        target_be.css('bottom', bottom_new);
        cont.attr('value', be_value);
        p_be_element.text(value_desc_pre + be_value + value_desc_post);
    }
}
```

---

**Listing A.14** Example for the Javascript Part of the Input Method Implementation (Bar Slider)  
Part 4

---

```
        // After each mouse move store position in global pointer pos value
        uist_im_fca_pointer_pos = [event.pageX , event.pageY];
    }
);
/**
 * Set value for this fca input method
 * @param: cont {jQuery object} FCA input method container element
 * @param: be_value {Number} Best estimate value
 */
function uist_im_fca_setValue(cont, be_value) {
    // Relevant fca div elements
    var background_element = cont.children( '.uist-im-fc-resize-element-back' );
    var target_be = background_element.children( '.uist-im-fberc-best-estimate' );
    var p_be_element = background_element.children( '.uist-im-fberc-best-estimate' ).
        children( '.uist-im-fberc-freq-mark-xs' ).children('p');
    // Attributes and Parameters
    var value_desc_pre = cont.attr('value_desc_pre');
    var value_desc_post = cont.attr('value_desc_post');
    var orientation = cont.attr('orientation');
    var value_min = cont.attr('value_min');
    var value_max = cont.attr('value_max');
    var value_min_sel = cont.attr('value_min_sel');
    var value_max_sel = cont.attr('value_max_sel');
    var left_new; var bottom_new;
    var value_size = parseFloat(value_max) - parseFloat(value_min);
    var max_size_x = parseFloat(background_element.width());
    var max_size_y = parseFloat(background_element.height());
    // Best estimate value
    be_value = Math.min(Math.max(be_value, parseFloat(value_min)), parseFloat(value_max));
    left_new = parseFloat(target_be.css('left'));
    bottom_new = parseFloat(target_be.css('bottom'));
    if (orientation == UIST_IM_FCA_ORIENTATION_VERTICAL) {
        bottom_new = be_value * max_size_y / value_size;
        bottom_new = Math.min(Math.max(bottom_new, 0), max_size_y);
        be_value = bottom_new * value_size / max_size_y;
    } else {
        left_new = be_value * max_size_x / value_size;
        left_new = Math.min(Math.max(left_new, 0), max_size_x);
        be_value = left_new * value_size / max_size_x;
    }
    be_value = be_value.toFixed(0);
    // Update position and value
    target_be.css('left', left_new);
    target_be.css('bottom', bottom_new);
    target_be.css('background-color', UIST_IM_CONST_COLOR_2_STEP_GRADIENT[0]);
    cont.attr('value', be_value);
    p_be_element.text(value_desc_pre + be_value + value_desc_post);
}
```

---

# Bibliography

- [acc91] accuracy. (n.d.) Collins English Dictionary – Complete and Unabridged. (1991, 1994, 1998, 2000, 2003). Retrieved May 6 2015 from <http://www.thefreedictionary.com/accuracy>. 1991. URL <http://www.thefreedictionary.com/accuracy>. (Cited on page 32)
- [BCE<sup>+</sup>92] K. W. Brodlie, L. Carpenter, R. A. Earnshaw, Gallop, R. J. Hubbard, Am Mumford, C. D. Osland, P. Quarendon. *Scientific visualization: techniques and applications*. Springer-Verlag New York, Inc, 1992. (Cited on pages 17, 22, 30 and 31)
- [Bev08] N. Bevan. Classifying and selecting UX and usability measures. In *International Workshop on Meaningful Measures: Valid Useful User Experience Measurement*, pp. 13–18. 2008. (Cited on pages IX, 19, 20, 22, 40, 41 and 43)
- [BHG89] R. D. Bergeron, K. Hall, G. G. Grinstein. A Reference Model for the Visualization of Multi-dimensional Data. 1989. (Cited on pages 17 and 22)
- [Boo15] Bootstrap founding team, core contributors and community. Bootstrap: Responsive front-end framework, 2015. URL <http://getbootstrap.com/>. (Cited on pages 73 and 74)
- [Bro96] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996. (Cited on page 101)
- [But99] B. Butterworth. *What counts: How every brain is hardwired for math*. Free Press, New York, 1999. (Cited on page 14)
- [Car15] Carsten Schmitz, Lime Survey Team and Contributors. Lime Survey: Free Open Source Software survey tool on the web, 2015. URL <https://www.limesurvey.org>. (Cited on page 84)
- [CD13] K. Chowdhary, P. Dupuis. Distinguishing and integrating aleatoric and epistemic variation in uncertainty quantification. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47(03):635–662, 2013. (Cited on page 8)
- [Chr09] Chrischi. [wikimedia.org](http://wikimedia.org): Steilgipflig.svg & Flachgipflig.svg, 2009. (Cited on pages IX and 38)
- [CR00] A. Cedilnik, P. Rheingans. Procedural annotation of uncertain information. In *Visualization 2000. Proceedings*, pp. 77–84. 2000. (Cited on page 18)

- [DCG99] X. Du, W. Chen, R. Garimella. Propagation and management of uncertainties in simulation-based collaborative systems design. In *The 3rd World Congress of structural and Multidisciplinary Optimization*, pp. 17–21. 1999. (Cited on page 26)
- [Dic15a] The American Heritage® Science Dictionary. 2015. URL <http://dictionary.reference.com/browse/randomness>. (Cited on page 8)
- [Dic15b] Collins English Dictionary - Complete & Unabridged 10th Edition. 2015. URL <http://dictionary.reference.com/browse/ambiguity>. (Cited on page 8)
- [Dix09] A. Dix. *Human-computer interaction*. Springer, 2009. (Cited on page 11)
- [Dja15] Django Software Foundation and individual contributors. Django: The web framework for perfectionists with deadlines, 2015. URL <https://www.djangoproject.com/>. (Cited on page 73)
- [DK97] T. J. Davis, C. P. Keller. Modelling and visualizing multiple spatial uncertainties. *Computers & Geosciences*, 23(4):397–408, 1997. (Cited on page 18)
- [Dr.14] Dr. Ken Kahn at the University of Oxford. Behaviour Composer, 2014. URL <http://m.modelling4all.org/>. (Cited on page 21)
- [Eic] S. G. Eick. Data visualization sliders. In P. Szekely, editor, *the 7th annual ACM symposium*, pp. 119–120. doi:10.1145/192426.192472. (Cited on pages 20 and 22)
- [Fed13] Federal Statistical Office of Germany. Educational attainment: Educational attainment of the population in Germany, 2013. URL <https://www.destatis.de/EN/FactsFigures/SocietyState/EducationResearchCulture/EducationalLevel/Tables/EducationalAttainmentPopulationGermany.html>. (Cited on pages IX, 23 and 24)
- [For15] Forio Corporation. Forio Epicenter, 2015. URL <http://forio.com/products/epicenter/>. (Cited on pages IX and 21)
- [GHv<sup>+</sup>05] G. Gigerenzer, R. Hertwig, van den Broek, Eva, B. Fasolo, K. V. Katsikopoulos. A 30% chance of rain tomorrow: how does the public understand probabilistic weather forecasts? *Risk analysis : an official publication of the Society for Risk Analysis*, 25(3):623–629, 2005. doi:10.1111/j.1539-6924.2005.00608.x. (Cited on pages 15, 25, 35 and 42)
- [Hof00] U. Hoffrage. MEDICINE: Communicating Statistical Information. *Science*, 290(5500):2261–2262, 2000. doi:10.1126/science.290.5500.2261. (Cited on pages 14, 16, 22 and 52)
- [HPv94] L. Hesselink, F. H. Post, van Wijk, Jarke J. Research issues in vector and tensor field visualization. *Computer Graphics and Applications, IEEE*, 14(2):76–79, 1994. (Cited on pages 17, 22, 31, 33 and 34)

- [Hub10] D. W. Hubbard. *How to measure anything: Finding the value of intangibles in business*. Wiley, Hoboken, N.J, 2nd ed. edition, 2010. (Cited on page 6)
- [IM87] H. Ibrek, M. G. Morgan. Graphical Communication of Uncertain Quantities to Nontechnical People. *Risk Analysis*, 7(4):519–529, 1987. doi:10.1111/j.1539-6924.1987.tb00488.x. (Cited on pages 16 and 22)
- [Ind08] Inductiveload. [wikimedia.org: Normal Distribution PDF.svg](http://commons.wikimedia.org/wiki/File:Normal_Distribution_PDF.svg), 02/04/2008. URL [http://commons.wikimedia.org/wiki/File:Normal\\_Distribution\\_PDF.svg](http://commons.wikimedia.org/wiki/File:Normal_Distribution_PDF.svg). (Cited on pages IX and 10)
- [KK93] P. R. Keller, M. M. Keller. *Visual cues: practical data visualization*, volume 2. IEEE Computer Society Press Los Alamitos, CA, 1993. (Cited on pages 17 and 30)
- [Kra] Kraje. Bootstrap Star Rating. URL <http://plugins.kraje.com/star-rating>. (Cited on page 46)
- [LL08] K. J. Laskey, K. B. Laskey. Uncertainty Reasoning for the World Wide Web: Report on the URW3-XG Incubator Group. In *URSW*. 2008. (Cited on pages IX, 6 and 7)
- [LSR01] I. M. Lipkus, G. Samsa, B. K. Rimer. General Performance on a Numeracy Scale among Highly Educated Samples. *Medical Decision Making*, 21(1):37–44, 2001. doi:10.1177/0272989X0102100105. (Cited on pages 14, 23, 25 and 66)
- [Mac92] A. M. MacEachren. Visualizing uncertain information. *Cartographic Perspectives*, (13):10–19, 1992. (Cited on page 18)
- [MDL08] R. E. Morss, J. L. Demuth, J. K. Lazo. Communicating Uncertainty in Weather Forecasts: A Survey of the U.S. Public. *Weather and Forecasting*, 23(5):974–991, 2008. doi:10.1175/2008WAF2007088.1. (Cited on pages 14 and 22)
- [MLD10] R. E. Morss, J. K. Lazo, J. L. Demuth. Examining the use of weather forecasts in decision scenarios: results from a US survey with implications for uncertainty communication. *Meteorological Applications*, 17(2):149–162, 2010. doi:10.1002/met.196. (Cited on pages 13 and 23)
- [MLFW80] A. H. Murphy, S. Lichtenstein, B. Fischhoff, R. L. Winkler. Misinterpretations of Precipitation Probability Forecasts. *Bulletin of the American Meteorological Society*, (61):695–701, 1980. (Cited on page 15)
- [Myl02] K. R. Mylne. Decision-making from probability forecasts based on forecast value. *Meteorological Applications*, 9(3):307–315, 2002. doi:10.1017/S1350482702003043. (Cited on pages 13, 14, 22, 23 and 39)
- [Nie03] J. Nielsen. *Usability 101: Introduction to usability*, 2003. (Cited on page 11)

- [Ole07] Oleg Alexandrov. Wikimedia: Fair dice probability distribution.svg, 19.07.2007. URL [http://commons.wikimedia.org/wiki/File:Fair\\_dice\\_probability\\_distribution.svg](http://commons.wikimedia.org/wiki/File:Fair_dice_probability_distribution.svg). (Cited on pages IX and 10)
- [Ole14] Ole Laursen, David Schnur, IOLA. Flot Charts: Attractive JavaScript plotting for jQuery, 2014. URL <http://www.flotcharts.org/>. (Cited on page 70)
- [OM02] C. Olston, J. D. Mackinlay. Visualizing data with bounded uncertainty. In *InfoVis 2002. IEEE Symposium on Information Visualization*, pp. 37–40. 28-29 Oct. 2002. doi:10.1109/INFVIS.2002.1173145. (Cited on page 16)
- [Pek07] Pekaje. wikimedia.org: Accuracy and precision.svg, 2007. URL [http://en.wikipedia.org/wiki/File:Accuracy\\_and\\_precision.svg](http://en.wikipedia.org/wiki/File:Accuracy_and_precision.svg). (Cited on pages IX and 33)
- [pre10] precision. (n.d.) Random House Kernerman Webster's College Dictionary. (2010). Retrieved May 6 2015 from <http://www.thefreedictionary.com/precision>. 2010. URL <http://www.thefreedictionary.com/precision>. (Cited on page 32)
- [PWL97] A. T. Pang, C. M. Wittenbrink, S. K. Lodha. Approaches to Uncertainty Visualization. *The Visual Computer*, 13(8):370–390, 1997. doi:10.1007/s003710050111. (Cited on pages 5, 17, 26, 27, 31, 34 and 42)
- [RBKSC06] M. S. Roulston, G. E. Bolton, A. N. Kleit, A. L. Sears-Collins. A Laboratory Study of the Benefits of Including Uncertainty Information in Weather Forecasts. *Weather and Forecasting*, 21(1):116–122, 2006. doi:10.1175/WAF887.1. (Cited on pages 13 and 14)
- [Rod08] Rodolfo Hermans. wikimedia.org: Negative and positive skew diagrams (English).svg, 2008. (Cited on pages IX and 38)
- [Sha91] B. Shackel. Usability-context, framework, definition, design and evaluation. *Human factors for informatics usability*, pp. 21–37, 1991. (Cited on page 10)
- [SHMW] J. Schwarz, S. Hudson, J. Mankoff, A. D. Wilson. A Framework for Robust and Flexible Handling of Inputs with Uncertainty. In K. Perlin, M. Czerwinski, R. Miller, editors, *the 23rd annual ACM symposium*, p. 47. doi:10.1145/1866029.1866039. (Cited on pages 18, 22 and 45)
- [Shn92] B. Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*, volume 2. Addison-Wesley Reading, MA, 1992. (Cited on page 12)
- [Ste] Stefan Petre. Bootstrap Slider. URL <https://github.com/seiyria/bootstrap-slider>. (Cited on page 46)
- [Tay09] B. N. Taylor. *Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results (rev. DIANE Publishing, 2009)*. (Cited on page 16)



- [Tuf83] E. R. Tufle. The visual display of quantitative information. *CT Graphics, Cheshire*, 1983. (Cited on page 17)
- [TW04] S. Tisue, U. Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, pp. 16–21. 2004. (Cited on pages IX and 21)
- [VVG98] Van der Wel, Frans JM, Van der Gaag, Linda C, B. G. H. Gorte. Visual exploration of uncertainty in remote-sensing classification. *Computers & Geosciences*, 24(4):335–343, 1998. (Cited on page 18)
- [WBR<sup>+</sup>86] T. S. Wallsten, D. V. Budescu, A. Rapoport, R. Zwick, B. Forsyth. Measuring the vague meanings of probability terms. *Journal of Experimental Psychology: General*, 115(4):348–365, 1986. doi:10.1037/0096-3445.115.4.348. (Cited on pages 15 and 55)
- [WPL96] C. M. Wittenbrink, A. T. Pang, S. K. Lodha. Glyphs for visualizing uncertainty in vector fields. *Visualization and Computer Graphics, IEEE Transactions on*, 2(3):266–279, 1996. (Cited on page 18)
- [ZFFU08] B. J. Zikmund-Fisher, A. Fagerlin, P. A. Ubel. Improving understanding of adjuvant therapy options by using simpler risk graphics. *Cancer*, 113(12):3382–3390, 2008. (Cited on page 18)

All links were last followed on July 15, 2015.



### **Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature