Quantitative Methods for Uncertainty Visualization

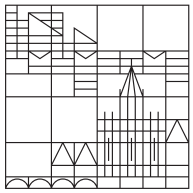
Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von

Jochen Görtler

an der

Universität Konstanz



Mathematisch-Naturwissenschaftliche Sektion
Informatik und Informationswissenschaft

Konstanz, 2021

Advisor
Prof. Dr. Oliver Deussen, University of Konstanz, Germany

Reviewers
Prof. Dr. Oliver Deussen, University of Konstanz, Germany
Prof. Dr. Daniel Weiskopf, University of Stuttgart, Germany

Date of Submission
01 February 2021

Date of Defense
12 July 2021

Abstract

Uncertainty is ubiquitous in the data that we collect. Nevertheless, when users create visualizations of this data, it is frequently neglected. The reason for this is twofold: For one, many common algorithms cannot handle uncertain data. If this is the case, the only option is to omit information and solely consider the most likely realization of the data. The second reason is that uncertainty is difficult to communicate to the user, either due to the lack of suitable visual variables or because users lack literacy in understanding uncertainty and its underlying mathematical model: probability distributions.

The following thesis proposes methods to alleviate some of these problems by tackling two research questions: *"How can we communicate uncertainty with its statistical properties?"* and *"How to adapt visualization methods to uncertainty?"*

First, we discuss sources of uncertainty, how to model it by using probability distributions, and different approaches for propagating uncertainty. Then, we propose a novel treemap technique designed to show uncertainty information. Our method relaxes the requirement of covering the entire designated space that traditional techniques adhere to. We propose modulated sine waves as a quantitative encoding of uncertainty, yet our resulting method is flexible to work with various visual variables.

Next, we investigate how to perform dimensionality reduction on uncertainty data. We identify two general approaches: Monte Carlo sampling and analytical methods. We apply the former to adapt stress-majorization for creating layouts of probabilistic graphs. While Monte Carlo methods can be applied to a wide range of problems, the resulting visualizations can be difficult to interpret. On the other hand, analytical approaches do not share this drawback but are only viable if the uncertainty information can be propagated analytically through the projection. We show how this can be done to arrive at an uncertainty-aware version of principal component analysis. Besides, the analytical approach allows us to understand the projection's sensitivity to uncertainty in the data.

Together with a summary of the developed methods, this thesis concludes with potential directions for future research. For this, we discuss Bayesian methods and their potential applications for handling uncertainty in visualization. Furthermore, we propose stippling, a form of visual abstraction, as a new way to visualize uncertainty in scalar fields.

Zusammenfassung

Beim Sammeln von Daten sind Unsicherheiten allgegenwärtig. Dennoch werden diese beim Erstellen von Visualisierungen häufig vernachlässigt. Hierfür gibt es zwei Gründe. Zum einen können viele gängige Algorithmen nicht mit unsicheren Daten umgehen. In diesen Fällen besteht die einzige Möglichkeit darin, Informationen wegzulassen. Der zweite Grund ist, dass Unsicherheiten dem Benutzer nur schwer zu vermitteln sind, entweder aufgrund des Fehlens geeigneter visueller Variablen oder weil es den Benutzern an allgemeinen Kenntnissen über Unsicherheiten mangelt. In der folgenden Arbeit werden Methoden vorgeschlagen, um diese Probleme zu verbessern. Dabei betrachten wir die zwei folgenden Forschungsfragen: *"Wie können wir Unsicherheiten und deren statistische Eigenschaften dem Benutzer vermitteln?"* und *"Wie können wir Visualisierungsmethoden anpassen, um mit Unsicherheiten umzugehen?"*

Hierfür werden zunächst Quellen von Unsicherheiten identifiziert und es wird erläutert, wie man diese durch Wahrscheinlichkeitsverteilungen modellieren kann. Des Weiteren schlagen wir grundlegende Ansätze zum Propagieren von Unsicherheiten vor.

Als Nächstes präsentieren wir eine neue Treemap-Methode zur Darstellung von Unsicherheiten in Hierarchien. Unser Ansatz lockert die Anforderung von traditionellen Verfahren, den gesamten Zeichenbereich abzudecken. Zusätzlich schlagen wir Sinuswellen als quantitative Codierung von Unsicherheiten vor. Die resultierende Methode ist jedoch flexibel genug, um sie auch mit anderen visuellen Variablen zu verwenden.

Der folgende Teil der Dissertation untersucht Dimensionsreduktion von unsicheren Daten. Wir identifizieren zwei allgemeine Ansätze: Monte-Carlo-Sampling und analytische Methoden. Erstere wenden wir an, um Layouts von probabilistischen Graphen mittels Stress-Majorisierung zu erstellen. Während Monte-Carlo-Methoden vielseitig eingesetzt werden können, sind die resultierenden Visualisierungen oft nur schwer zu interpretieren. Analytische Ansätze haben diesen Nachteil nicht, können allerdings nur verwendet werden, wenn sich die Unsicherheiten direkt durch die Projektion propagieren lassen. Um diese Vorgehensweise zu veranschaulichen, zeigen wir, wie man eine solche Formulierung für die Hauptkomponentenanalyse ableiten kann. Darüber hinaus erlaubt uns der analytische Ansatz, die Sensitivität der Projektion in Bezug auf die Unsicherheit in den Daten zu verstehen.

Zum Abschluss geben wir einen Einblick in mögliche Richtungen für zukünftige Forschung. Hierfür diskutieren wir Bayes'sche Methoden und ihr Potential für den Umgang mit Unsicherheiten in der Visualisierung. Darüber hinaus schlagen wir visuelle Abstraktionen als neue Möglichkeit zur Visualisierung von Unsicherheiten in Skalarfeldern vor.

Acknowledgements

This thesis would not have been possible had it not been for a series of fortunate events and the numerous people that helped me along the way.

First and foremost, I want to thank my advisor Oliver Deussen for accepting me into his group and supervising me. He gave me the freedom to pursue my areas of interest and championed all my endeavors. His ideas and feedback were invaluable. I am also immensely grateful to my second advisor Daniel Weiskopf. He helped me figure out many of the technical details and significantly improved my drafts. Thank both of you — I could not have arrived at this point without you and your support.

Furthermore, I am grateful to all my coauthors. Special thanks go out to Christoph Schulz, who contributed immensely to all of my work. He always came up with new approaches and perspectives when I was stuck. It was exciting and fun to collaborate with Marc Spicker and Rebecca Kehlbeck on our joint projects — thank you for your help and the great time. Although Till Niese and I never had a paper together, I want to thank him here for his endless help in figuring out all the problems with my code.

I also want to thank the rest of the crew at the University of Konstanz: Marvin Gülzow, Julian Kratt, Thomas Lindemeier, Mariam Mahmoud, Jens Metzner, Patrick Paetzold, Thilo Spinner, and all undergraduates. The whole group made work a place where I loved to go, sometimes just to hang out, and often longer than I originally planned. The same goes for Ingrid Baiker, who helped me get settled in at Konstanz. She always had an open ear for what was going on in my life and helped me with everything not directly related to research. She and Claudia Widmann also helped me navigate all the administrative tasks — those are sometimes harder than the research itself. Thank you.

A big thank you to Hendrik Strobelt, who gave me great general advice and steered me towards some of my most successful works. Additionally, I want to thank everyone in the Visualization Design Lab at the University of Utah for my great time there. Alex Lex was not only a great mentor, but he also showed me many beautiful places in Utah. I am also grateful to Dominik Moritz, my internship mentor during my time at Apple, who helped me get a new perspective on many topics in visualization.

I want to thank my family, who taught me the value of education, provided me with every opportunity in life, and showed me to go through the world with open eyes.

My most generous thank you goes to my wife Ania, who has supported me wholeheartedly ever since we met. She always had my back, and I could rely on her to pick me up with a good laugh when I was down or stressed by my work. My son Leon always managed to put a smile on my face after a long day and helped me recognize what's important. I love you both.

Contents

Figures

Tables

Overview 1

Uncertainty is ubiquitous in our lives. We encounter it when we make decisions or worry about the future. A large amount of uncertainty comes from not knowing the complete state of our surroundings, as we can only perceive a tiny fraction of it. As scientists, we try to better understand the world around us and the laws that govern it. We do this through observation and experimentation, supported by measurements. However, the measurements that we perform are inherently inaccurate. The reason for this is that our measurement devices are imperfect and introduce systematic errors or random noise.

Fully capturing a system would require us to observe everything down to single particles. Of course, this is impossible. Not only because of the complexity of such an endeavor, but also because the Heisenberg uncertainty principle states that we cannot measure the exact position and velocity of a particle at the same time[1]. Therefore, when scientists conduct experiments, they need to carefully balance trade-offs to get accurate results for the variables that they want to measure. In the natural sciences, much effort has been put into understanding and quantifying errors. In Chapter 2, we will describe the different types of uncertainty introduced by these errors. Uncertainty also plays a fundamental role in the social sciences because humans are not rational, which makes their actions and reactions hard to predict. While we can make reasonable estimates based on the behavior of larger groups, drilling down and understanding the actions of an individual is often impossible.

1: This is in part due to the observer effect, which describes that some measurements cannot be made without affecting the system itself.

Graphical representations are essential tools for communicating facts and ideas, independent of the domain. Looking at uncertainty from the context of information visualization, we can see its presence at every stage of the visualization pipeline. The first step is understanding the process that has generated the input data and then casting it into a mathematical model. Further preprocessing steps, such as filtering, also need to consider the variability of the data resulting from this process. The mapping stage transforms data into geometric primitives. To convey the uncertainty, we need to rethink and adapt the algorithms we use to create representations because, in their current form, they cannot readily handle this additional information.

The critical challenge is that only a few algorithms are robust against small variations in the inputs. Layout generation is particularly prone to this because of branching: Taking a different execution path can lead to large changes in the resulting arrangement. These discontinuities make it difficult to reason about uncertain data and can ultimately reduce trust in the visualization.

The purpose of visualization is to transport a message, so we need to make sure that it reaches the receiver intact. In concrete terms, we need to convey the information to the user understandably and intuitively — a challenging task even without considering uncertainty. Because of this, few visualizations contain uncertainty estimates. Visualization authors

face the following dilemma: They can honestly communicate the uncertainty in the data, but this risks reducing the viewer's trust in the visualization [Hul20]. By investigating new visual variables, rendering techniques, and algorithms, we aim to mitigate some of these challenges.

In empirical research, there is a distinction between qualitative and quantitative methods. In qualitative research, we are interested in understanding how things work or why people behave in a certain way. Generally, qualitative results are prose or short comments in natural language that describe concepts. Quantitative research, on the other hand, tries to measure variables and relate entities to each other. The results of quantitative research are numerical and come in the form of numbers, magnitudes, or vectors[2]. A similar distinction can be drawn when conveying information: We can communicate aspects of the data qualitatively or quantitatively. The following thesis presents methods that treat uncertainty quantitatively. Starting from input data afflicted with uncertainty, we want to adapt algorithms to propagate this uncertainty and finally communicate the results to the user quantitatively, which allows them to draw well-founded conclusions.

2: One example of a qualitative representation are clothing sizes (S, M, L,...). The quantitative counterpart would be the sizing table, which specifies concrete measurements.

1.1 Research Questions

This thesis aims to develop novel visualization techniques that respect the uncertainty in the data and communicate it faithfully. We see this problem as an interplay between the following two research questions:

RQ1 *How can we communicate uncertainty with its statistical properties?* There are already several established ways to depict uncertainty in low-dimensional data. Examples of this are box-plots or violin-plots. We are interested in finding analogous representations for high-dimensional data.

RQ2 *How to adapt visualization methods to uncertainty?* Modeling uncertainty in the data forces us to rethink the methods we use for visualization because many existing techniques cannot be applied directly to uncertain data.

The following paragraphs give an overview of the challenges that govern these research questions and approaches for tackling them.

Communicating Uncertainty Information Uncertain data are inherently more complex than their certain counterparts: Instead of a single value, a data point can have many possible realizations. Therefore, we need novel visualization techniques that allow us to communicate the variability in the data. There are two general approaches to communicating uncertainty. We can either use summary statistics that describe each data point's distribution and show them in addition to the expected value of the distribution. Alternatively, we can show the variability directly, for example, by plotting all possible realizations. To encode uncertainty information using summary statistics, we can either use a geometric encoding or multiplex the additional information into a secondary channel. A well-known example of the former are error bars. Contrary, value-suppressing color palettes [CMH18] or techniques that encode the infor-

mation into entirely different visual channels, such as glyphs [WPL96], are instances of multiplexing. Furthermore, we also need new visual variables for encoding information about uncertainty. Several methods, such as *sketchiness* and *blur*, have been investigated under the premise of intuitiveness, in many cases with mixed results [Bou+12; Woo+12].

Algorithms are predominantly not continuous over their inputs nor their specific parameters. Therefore, slight changes in either one of them can lead to large changes in the output. Sensitivity analysis is a method to judge the robustness of models and algorithms against these changes. The general idea is to slightly vary the input parameters to see how much changes they inflict on the output. The technique is commonly used when creating mathematical models and simulations, such as weather forecasts, where the resulting set of outcomes is also called an *ensemble*. Sensitivity analysis and quantification of uncertainty are related concepts, as both domains try to model variations in the input to algorithms. Traditionally, sensitivity analysis investigates variations of the input parameters to an algorithm, while uncertainty methods try to quantify the impact of small changes in the actual input data. An interesting line of research would be to unify both under a single framework, allowing the user to answer "What if?"-questions about the model[3]. Understanding the interplay and relationships between variations in the input and the influence of parameters could ultimately strengthen the trust in the method's results.

3: In this thesis, we will use the term sensitivity analysis for both concepts.

Adapting Algorithms Many of the commonly used methods for conveying information cannot readily handle uncertain data. There are several reasons for this. As we discussed earlier, the lack of appropriate visual variables is one of them. Another big challenge is that the algorithms that we use to process data for visualization were designed with absolute, non-variable, data in mind[4].

4: Consider sorting, a well-studied problem in computer science. How would we change the algorithm if we want to sort noisy measurements?

When we want to encode information about the uncertainty geometrically, we need to adapt the algorithms to account for the additional required space. In many cases, this is not readily possible for existing techniques. Consider layout algorithms, for example. For most of them, it is impossible to adapt the layout locally when the data changes. The reason for this is that many layout algorithms optimize for space efficiency globally. In Chapter 3, we will show how changing this paradigm can lead to new designs that are better suited to visualize uncertainty information.

In dimensionality reduction, the goal is to find a projection of the data — typically onto 2D — that retains most of its characteristics in the high-dimensional space, according to various metrics. In Chapters 4 and 5, we introduce two fundamental approaches that we can take to adapt existing methods to uncertain data. First, we consider the problem of finding a layout for a probabilistic graph. The critical insight is that we can use Monte Carlo techniques to go from probability distributions back to single instances that can be handled by the algorithm. However, this merely shifts the problem as we now need to register multiple resulting layouts back to each other. We achieve this by introducing an additional anchoring term that is jointly optimized when creating the layout. In Chapter 5, we show how to perform a principal component analysis for uncertain

input data by taking a different approach: We derive a closed-form solution to finding the covariance matrix, which ultimately yields the principal components, under uncertainty. A significant advantage of analytical methods is that they allow us to investigate the sensitivity of the result to the uncertainty in the data, as described in the previous paragraph.

1.2 Author's Contributions

This thesis presents approaches aimed at answering the research question defined in the previous section. For this, text and images from the author's previous publications have been reused without explicit citation. The following is a list of these publications:

- C. Schulz, A. Nocaj, J. Görtler, O. Deussen, U. Brandes, and D. Weiskopf. „Probabilistic graph layout for uncertain network visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 23 (1 2017) [Sch+17]

  Joint work with Christoph Schulz, Arlind Nocaj, Oliver Deussen, Ulrik Brandes, and Daniel Weiskopf. I contributed to the visualization of clusters in the layout by computing α-shapes of possible realizations of node positions. We then use these α-shapes to draw contours around the realizations to help discern single- and multimodal nodes. Additionally, I helped perform the experiments to visualize travel times by car in southern Germany by collecting and preparing data.

- J. Görtler, C. Schulz, D. Weiskopf, and O. Deussen. „Bubble treemaps for uncertainty visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 24 (1 2018) [Gör+18]

  Collaboration together with Christoph Schulz, Daniel Weiskopf, and Oliver Deussen. I came up with the design for bubble treemaps, the method for drawing contours using biarc curves, and the encodings of uncertainty. I implemented the prototype we used to create the examples in the paper. Daniel Weiskopf helped me refine my ideas together with Christoph Schulz, who contributed the algorithm for modulating sine waves onto the contours used to depict the uncertainty.

- J. Görtler, M. Spicker, C. Schulz, D. Weiskopf, and O. Deussen. „Stippling of 2D scalar fields". In: *IEEE Transactions on Visualization and Computer Graphics* 25 (6 2019) [Gör+19]

  Joint work together with Marc Spicker, Christoph Schulz, Daniel Weiskopf, and Oliver Deussen. We saw the potential of using the Linde–Buzo–Gray stippling algorithm[5] for uncertainty visualization and came up with the idea of using it for visualizing abstracted scalar fields. I contributed the generalization from discretized images to continuous scalar fields and refactored Marc Spicker's initial implementation of the algorithm to adapt to these changes. Marc Spicker and I collaborated on the two methods of embedding contours into stipple representation, and with his help, we created

5: Oliver Deussen and Marc Spicker developed the original version of the LBG stippling algorithm for artistic rendering and other applications in computer graphics [DSZ17].

the examples in the paper. Christoph Schulz devised and conducted the user study and analyzed its results.

- J. Görtler, R. Kehlbeck, and O. Deussen. „A visual exploration of Gaussian processes". In: *Distill* (2019) [GKD19]

  Joint work with Rebecca Kehlbeck and Oliver Deussen. Hendrik Strobelt nudged us to write an article that uses visualizations to explain how machine learning methods work. I implemented Gaussian processes, devised the text's overall structure, and created the concepts for most interactive figures. Rebecca Kehlbeck helped to refine the visualizations and contributed to the section on kernels. Together we designed the visualization of the conditioned distribution with the respective covariance matrix.

- J. Görtler, T. Spinner, D. Streeb, D. Weiskopf, and O. Deussen. „Uncertainty-aware principal component analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 26 (1 2020) [Gör+20]

  Joint work with Thilo Spinner, Dirk Streeb, Daniel Weiskopf, and Oliver Deussen. I came up with the idea of using the mathematical properties of Gaussian distributions for dimensionality reduction and investigating the influence of uncertainty on the final projection. Dirk Streeb suggested the use of principal component analysis and surveyed much of the related work. Thilo Spinner helped me derive the required equations for the projection. I had the idea for sensitivity analysis and framed it mathematically. Thilo Spinner provided the initial implementation of factor traces, which I subsequently refined.

- K. Gadhave, J. Görtler, Z. Cutler, C. Nobre, O. Deussen, M. Meyer, J. Phillips, and A. Lex. „Capturing user intent when brushing in scatterplots". In: *OSF Preprints (Submitted to SIGCHI)* (2021) [Gad+21]

  Collaboration with Kiran Gadhave, Zach Cutler, Carolina Nobre, Oliver Deussen, Miriah Meyer, Jeff Phillips, and Alexander Lex[6]. We saw the need to model user intent in visualization systems. Together, Alexander Lex, Kiran Gadhave, Jeff Phillips, and I developed the initial user intents. Kiran Gadhave implemented the visualization system on the frontend while I developed the machine learning backend of our prototype. Together we came up with the Jaccard index for ranking intents. It was my idea to use Bayesian ranking, as well as using decision trees to simplify range queries. Zach Cutler incorporated the provenance tracking. Carolina Nobre and Miriah Meyer helped to conduct and evaluate the user studies. Alexander Lex coordinated the project and guided our research.

6: This project originated from my research stay with the Visualization Design Lab at the University of Utah, for which I received financial support from the German Research Foundation.

In addition to these, there are other publications to which the author has contributed. We list them below. The first two of them touch upon different topics, so they are not part of this thesis.

[Kat+13] and [Bod+16] originated during my undergraduate and graduate studies at the Karlsruhe Institute of Technology.

- D. Katić, A.-L. Wekerle, J. Görtler, P. Spengler, S. Bodenstedt, S. Röhl, S. Suwelack, H. G. Kenngott, M. Wagner, B. P. Müller-Stich, R. Dillmann, and S. Speidel. „Context-aware augmented reality in laparoscopic surgery". In: *Computerized Medical Imaging and Graphics* 37 (2 2013) [Kat+13]
- S. Bodenstedt, J. Görtler, M. Wagner, H. G. Kenngott, B. P. Müller-Stich, R. Dillmann, and S. Speidel. „Superpixel-based structure classification for laparoscopic surgery". In: *Medical Imaging, SPIE*. 17. 2016. ISBN: 978-1-5106-0021-8 [Bod+16]
- T. Spinner, J. Körner, J. Görtler, and O. Deussen. „Towards an interpretable latent space". In: *Proceedings of the Workshop on Visualization for AI Explainability (VISxAI)*. `https://thilospinner.com/towards-an-interpretable-latent-space/`. Last Accessed: 2021-07-15. IEEE VIS, 2018 [Spi+18]

1.3 Outline

This thesis is structured around answering the two research questions posed in Section 1.1. In Chapter 2, we introduce the various types of uncertainty that have been identified in different domains. We then show how to model uncertainty mathematically by using probability distributions. For this, we first introduce general concepts of statistics that we will use throughout this thesis. Then, we identify two fundamental approaches to propagating uncertainty through the visualization pipeline: Monte Carlo-based sampling and analytical solutions (**RQ2**). We present concrete instances of these two methods in the following chapters.

In Chapter 3 we consider uncertainty in hierarchical data. Starting from a hierarchy where uncertainty is present solely in the leaves, we show how to analytically propagate the uncertainty towards the root (**RQ2**). Once every node in the hierarchy has uncertainty information attached to it, we present a novel treemap algorithm designed to show this information (**RQ1**). What differentiates this algorithm from traditional techniques is that it deliberately leaves open space in the layout, which we reclaim to show information about each node's uncertainty (**RQ2**).

Chapters 4 and 5 focus on dimensionality reduction under uncertainty. First, we will propose a sampling-based method to layout probabilistic graphs. For this, we show how to extend stress majorization by an additional term to anchor the resulting samples to one another (**RQ2**). The ensuing graph embedding reveals that node distribution can be multimodal. To emphasize this visually, we use kernel density estimation and clustering to visualize contours (**RQ1**). Then, in Chapter 5, we show how to adapt principal component analysis to uncertain data (**RQ2**). For this, we generalize the existing method to work on probability distributions. Our approach is general and allows us to perform a sensitivity analysis

of the projection with respect to the amount of uncertainty. This helps users understand the influence of uncertainty on the result (**RQ1**).

In Chapter 6, provide general directions for future research and point out two promising techniques that could be used for uncertainty visualization: Bayesian methods and visual abstraction. We conclude this thesis in Chapter 7, which contains a summary of the methods presented in this thesis and their relation to the visualization pipeline.

2 Uncertainty

The following chapter gives background on uncertainty and lays the mathematical foundations for the methods proposed in this thesis. First, we introduce various types of uncertainty that we encounter in input data. Then, we give an overview of how different domains understand and handle uncertainty. Section 2.2 recaps probability distributions and how they can be used to model the processes that generated the data. In Section 2.3, we identify two approaches to propagating uncertainty through the visualization pipeline. Finally, Section 2.4 gives an overview of various uncertainty visualization methods related to the work presented in this thesis.

2.1 Types and Sources

Several taxonomies exist to describe different kinds of uncertainty. In uncertainty quantification, there are two main categories of uncertainty. *Aleatoric* uncertainty is inherent to the process under observation and cannot be controlled, while *epistemic* uncertainty can be reduced by performing more accurate measurements [KD09].

In the context of information visualization, Skeels et al. [Ske+08] identify three sources of uncertainty: Uncertainty of measurements, aggregations, and uncertain predictions. This thesis will show examples for each of these sources; Table 2.1 lists them in detail. Probability distributions are the mathematical tool to model these types of uncertainty. They allow us to capture the variations and unknowns present in the input data and form the foundation of our visualization techniques.

Table 2.1: Various examples that we provide throughout this thesis, grouped by the source of uncertainty.

Source	Examples
Measurement	CES (3.4) Student grades (5.4)
Aggregation	S&P 500 (3.4) Travel times (4.5) Iris (5.4) Anuran calls (5.4)
Predictions	Protein–protein interactions (4.5)

Set theory is another domain where uncertainty prevails. There is a wide range of work on *fuzzy sets* that tries to model sets where the set membership is described by a function that determines each element's *grade of membership*. Formally, a fuzzy set is a tuple (U, m) of a set U and a membership function[1] $m : U \to [0, 1]$. Hence, an element x can either be not included $m(x) = 0$, partially included $0 < m(x) < 1$, or fully included $m(x) = 1$. Fuzzy sets are often defined in terms of linguistic labels. In Chapter 5, we show an example of this in the context of dimensionality reductions. The dataset contains student grades, some of them provided by common everyday words, such as *good* or *very good*. These assessments are than mapped to their mathematical counterpart in the form of trapezoid distributions. Please note that in this special case, the membership function describes the probability of an element belonging to each of the sets. Fuzzy sets find wide application, among others in decision-making, electrical engineering (system control), and civil engineering [KY95].

1: While m might look similar to probability density functions, which we will introduce in Section 2.2, note that m can, but does not have to be normalized.

Uncertainty is also closely related to risk. However, these terms are often used interchangeably, although they describe different concepts in many domains. In finance, risk and uncertainty are used to quantify the cost associated with future events. In this context, they mean slightly different things [Kni21]: Risk specifies events whose chance of occurrence

can be quantified. Their impact is known, and the goal is to minimize it — for which statistical methods play an essential role. In this context, the term uncertainty is used to describe events with opposite characteristics: They are not known in advance and therefore cannot be quantified. Thus, there is also no way to control their occurrence or minimize their impact. In hindsight, these events were often not deemed possible, and their incidence leads to a large change in beliefs[2]. However, this definition of uncertainty is not very common in the natural sciences, where the term uncertainty describes environments that are only partially observable or inherently stochastic, such as in quantum physics. In the following section, we will look at how to model uncertainty mathematically.

2: Taleb popularized the term *black swan* for these type of events. An example for such an event is the proliferation of personal computers.

2.2 Modeling

When we want to visualize uncertainty, it is crucial to understand the process that generated the data. However, as we discussed earlier, it is impossible to capture the properties of such processes fully. The typical way to deal with this is to make simplifying assumptions. One approach to handle the imprecision of measurements is to perform multiple repeated measurements and aggregate them. In the following section, we briefly recap probability distributions — we will start with the 1D case and then move on to multivariate statistics[3].

3: There are many textbooks on statistics, but for the geometric concepts that we will describe later on, we recommend the book by Wickens [Wic94].

We use *random variables* to describe the values of possible outcomes x of a random phenomenon. Probability distributions model either discrete or continuous random variables. In the former case, we use *probability mass functions* (PMF) defined as $p_X(x_i) = P(X = x_i)$ with $p : \mathbb{R} \to [0, 1]$, to assign a probability to each of the outcomes. The probabilities of each outcome have to sum up to 1, so:

$$\sum_i p_X(x_i) = 1$$

In the later case, where the probability distribution is continuous, the distribution is given by a *probability density function* (PDF) $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$. It maps the value of a continuous random variable $x \in X$ to a normalized density:

$$\int_{\mathbb{R}} f(x)\,dx = 1$$

It is important to note that unlike the PMF p_X, the result of evaluating the PDF f_X at a single location is not a probability. To retrieve the probabilty of for a given value x, we need to consider an infinitesimal interval $[x, x + dx]$. The probability of an outcome falling into this range is then given by:

$$f_X(x) = \frac{d}{dx} F_X(x),$$

where $F_X(x)$ is the cumulative distribution function of X.

So far, we only looked at the one-dimensional case. To extend these concepts to multi-dimensional phenomena, we can group several random variables into a multivariate random variable, which is also called a *random vector*. Throughout this thesis, we denote random vectors by

$\mathbf{x} = [x_1, \dots, x_d]^\mathsf{T}$, with $\mathbf{x} \in \mathbb{R}^d$. Analogously, the corresponding multivariate probability distributions span the same d-dimensional domain. An eminent property arises from the fact that random variables and random vectors form an algebra [Spr79]: They can be manipulated using affine transformations[4]. These transformations can, for example, scale, translate, or rotate $\mathbf{x}$. Generally, an affine transformation has the form $\mathbf{y} = A\mathbf{x} + \boldsymbol{b}$. It consists of a linear transformation A and a translation vector $\boldsymbol{b}$ that together transform an input $\mathbf{x}$ to obtain a new random vector $\mathbf{y}$, which can be described using a modified distribution.

4: We will make extensive use of this property later when we show how to perform dimensionality reduction on random vectors.

Summary Statistics

In many cases, it suffices to consider the general characteristics of random variables or random vectors rather than their full distributions. Using these characteristics, we can make statements about the original distribution's shape and properties — we call them *summary statistics*. The most well-known statistics are the first and second moments, which, in the real-valued case, are also called *mean* μ and *variance* σ^2. In the continuous case, they are defined as:

$$\mu = \int_{\mathbb{R}} x \cdot f_X(x)\, dx, \quad \sigma^2 = \int_{\mathbb{R}} x^2 \cdot f_X(x)\, dx - \mu^2$$

The mean μ is also called the *expected value*. We can think of the variance as the spread of actual realizations $x \in X$ around this value[5].

5: This suggests why error bars and similar visualization techniques are popular: They transfer this metaphor into a geometric encoding. We mimic this choice for the visual variables in our treemap method, which we present in Section 3.3.

Summary statistics also exist for random vectors in the multivariate case. For multi-dimensional data, the mean $\boldsymbol{\mu}$ is a d-dimensional vector, and the variance is replaced by the *covariance* $\mathrm{Cov}(\mathbf{x}, \mathbf{x})$, which reflects correlations between each of the d components. The mean vector $\boldsymbol{\mu}$ describes the expected value of the distribution. Each of its components reflects the mean of the corresponding dimension. $\mathrm{Cov}(\mathbf{x}, \mathbf{x})$ models the variance along each dimension and determines how the different random variables are correlated. Because the covariance describes the pairwise relationships between dimensions, it has the form of a symmetric $d \times d$ matrix. Every covariance matrix is always positive semi-definite [Haz94]. The diagonal of $\mathrm{Cov}(\mathbf{x}, \mathbf{x})$ consists of the variance σ_i^2 of the i-th random variable. And the off-diagonal elements σ_{ij} describe the correlation between the i-th and j-th random variable.

Both $\boldsymbol{\mu}_\mathbf{x}$ and $\mathrm{Cov}(\mathbf{x}, \mathbf{x})$ are defined in terms of the expected value operator $\mathbb{E}[\cdot]$. Let the random vector $\mathbf{x}$ be given by a set of samples $\{\boldsymbol{x}_n\}$, $n \in \{1, \dots, N\}$ from an arbitrary distribution. For the sampling mean $\boldsymbol{\mu}_\mathbf{x}$ we have:

$$\boldsymbol{\mu}_\mathbf{x} = \mathbb{E}[\mathbf{x}] = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$

The covariance $\mathrm{Cov}(\mathbf{x}, \mathbf{x})$ is defined as:

$$\begin{aligned} \mathrm{Cov}(\mathbf{x}, \mathbf{x}) &= \mathbb{E}\left[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^\mathsf{T}\right] \\ &= \mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right] - \boldsymbol{\mu}_\mathbf{x}\boldsymbol{\mu}_\mathbf{x}^\mathsf{T} \end{aligned} \tag{2.1}$$

The term $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right]$ is the expected outer product $\mathbf{x}\mathbf{x}^\mathsf{T}$, which is defined by the following equation:

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right] = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n{}^\mathsf{T}$$

Earlier, we mentioned how to transform a random vector $\mathbf{x}$ using affine transformations. Transforming $\mathbf{x}$ in such a way also influences its summary statistics[6]. Both of the following equations make use of the linearity of the expected value operator $\mathbb{E}[\cdot]$. For the mean, it holds that:

$$\mathbb{E}\left[A\mathbf{x} + \boldsymbol{b}\right] = A\mathbb{E}\left[\mathbf{x}\right] + \boldsymbol{b}$$

In a similar fashion, we can transform the covariance matrix:

$$\mathrm{Cov}(A\mathbf{x} + \boldsymbol{b}, A\mathbf{x} + \boldsymbol{b}) = A\,\mathrm{Cov}(\mathbf{x}, \mathbf{x})A^\mathsf{T} \tag{2.2}$$

6: Please note that only the mean of $\mathbf{x}$ is influenced by the translation $\boldsymbol{b}$. In contrast, the covariance matrix is invariant to translation. The reason for this is that the covariance only captures the relative variance of each component because it is always centered around the sampling mean by the term $\boldsymbol{\mu}_\mathbf{x}\boldsymbol{\mu}_\mathbf{x}{}^\mathsf{T}$.

The Normal Distribution

Arguably the most established probability distribution is the *normal distribution*. It is often used to model errors of measurements or phenomena under the assumptions of the *central limit theorem*. One of this theorem's implications is that the joint distribution of independent, identically distributed random variables with finite variances is normal. We will encounter the d-dimensional, multivariate normal distribution (MVN) multiple times in our work. In the previous section, we introduced mean and covariance as common summary statistics — these first and second moments fully determine an MVN. Therefore, if $\mathbf{x}$ is distributed normally, with mean $\boldsymbol{\mu}$ and covariance matrix $\Psi = \mathrm{Cov}(\mathbf{x}, \mathbf{x})$, we write[7]:

$$\mathbf{x} = \left[x_1, x_2, \ldots, x_d\right]^\mathsf{T} \sim \mathcal{N}(\boldsymbol{\mu}, \Psi).$$

7: Commonly, Σ is used to denote the covariance matrix of an MVN. In Chapter 5, we will apply the expected value operator $\mathbb{E}[\cdot]$ to covariance matrices. To make these equations easier to read, we chose to use Ψ throughout this thesis.

There are a plethora of different probability distributions, but some elegant properties make normal distributions particularly important. For one, they are closed under *marginalization* and *conditioning*, two operations that are very common on multivariate distributions[8], and other basic operations, such as addition and subtraction as well. Furthermore, the density of the MVN is defined as follows:

$$f_X(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\Psi^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k \det\Psi}}$$

Therefore, computing the integral of the density is straight forward. It is also possible to compute values of its corresponding cumulative distribution function analytically in closed-form via the Mahalanobis distance. In addition to that, we derive the integral of a very similar function, the *Gaussian kernel*, in Section 6.2. In the next section and the following chapters, we will see the relevance of these properties.

8: We will go into more details in Chapter 6, where we define Gaussian processes, a versatile approach to regression and machine learning in general.

2.3 Propagation

The crucial step to answering RQ2 is propagating uncertainty through the steps required for creating a visualization. Initially, we start with input data with associated uncertainty. When we transform and preprocess this data, we manipulate values. The hard step is to carry over these manipulations to the uncertainty. As mentioned before, we use probability distributions to model uncertainty. Technically this means that we need to find ways to perform identical operations on probability distributions. As we will see, there are a significant amount of cases where this is not readily possible. Correctly tracing the uncertainty through visualization techniques is fundamental to the tasks that users want to perform, such as identifying areas of substantial variability and understanding the influence of uncertainty on the algorithms that we use to create visualizations.

We identified two principal ways of propagating uncertainty through the work presented in this thesis: Propagation based on Monte Carlo sampling and analytical propagation.

Monte Carlo Sampling

The former is particularly suited when the probability distributions that model the uncertainty are complex[9]. Sampling-based strategies can work in a variety of situations. In Chapter 4, we show how to use such an approach to layout probabilistic networks.

9: This is often the case when analytically integrating the PDF becomes intractable.

Nevertheless, this flexibility also comes with a price. In many cases, it is not clear how to best sample the underlying distributions faithfully. There is always a tradeoff for how many realizations are required: Only a large amount of samples ensures that the given distribution is well-approximated. However, ultimately, all these samples need to be propagated through the visualization pipeline, which may not be affordable depending on the algorithm. On the other hand, with too few samples, there is the chance that we neglect rare yet potentially significant events. Adaptive sampling approaches aim to mitigate this, but they introduce additional complexity and can make it harder to reason about the true distribution of values.

Another problem with sampling-based strategies is that many of the algorithms used for information visualization are not robust, which means that small input variations can lead to vastly different results. Layout algorithms, which are often subject to flipping, are typical examples of this. If we resort to sampling, we can expect different outcomes for the resulting realizations. The problem is that they do not have the same frame of reference[10]. That means we lose the relationship information from the original distribution and cannot respect it in the mapping algorithm.

10: An example of this is PCA, where the numerical computation of eigenvectors can change their sign if one point is moved slightly.

One solution to this problem is anchoring the different outputs to a single source of truth, such as the expected value of all realizations. As we will see in Chapter 4, the results of methods based on anchoring can be challenging to interpret due to this deficiency during propagation. Sampling

also entails that we restore the individual distributions after propagating single instances, for example, through kernel density estimation as shown in Chapter 4.

Analytical Solutions

Although often impossible, finding analytical ways to propagate the uncertainty through the visualization pipeline has a couple of advantages. In contrast to sampling, it is easier to reason about the influence of uncertainty on the resulting representation. Also, we do not lose the characteristic properties of the original distributions during the process. In Chapter 5, we show how to use this to our advantage by analyzing the visualization's sensitivity depending on the amount of uncertainty. The problem with analytical methods is that we can only apply them to a limited set of distributions.

One strategy that can be employed when searching for analytical solutions is looking for objects with suitable properties. In mathematics, a set is *closed* under an operation if the result of the operation is again part of the set. Let us consider multiple random variables that are represented by their mean and variance. Aggregating these random variables by summing them up yields another random variable, again with a mean and associated variance. Both input and output of the operation belong to the same class, so we say that the mean and expected value are closed under addition. We already saw an example of this in the previous section, where we showed that the multivariate normal distribution is closed under affine transformations. In Chapter 5, we use this fact to generalize principal component analysis to uncertain data.

In Chapter 3 we propose a novel treemap method that can also visualize uncertainty information in hierarchies. It can be typical only to have uncertainty information attached to leaf nodes. This comes from the fact that only these elements can be measured; the grouping of values together into categories is often purely virtual. However, in many cases, we are interested in how much uncertainty is present in each subhierarchy. To visualize this information, we need to propagate the uncertainty from the leaves towards the root: During the propagation from children to parent the expected values $\mu_{1,\dots,n}$ add up, while the standard deviations $\sigma_{1,\dots,n}$ aggregate using the Euclidean norm[11]:

$$\mu_{1,\dots,n} = \sum_{i}^{n} \mu_i, \quad \sigma_{1,\dots,n} = \sqrt{\sum_{i}^{n} \sigma_i^2}$$

11: This is true as long as we can assume linear aggregation and statistical independence among nodes.

The above model is just one example of the analytical propagation of uncertainty, and uncertainty propagation does not end here, of course. For a holistic view, we also need to consider the uncertainty introduced during the visualization pipeline's rendering step, for example, during rasterization or color quantization.

2.4 Visualization

This section gives an overview of different ways to visualize uncertainty in general. By no means is this an exhaustive survey, but instead lists relevant techniques or methods that share similarities to ones presented in this thesis[12]. Historically, the representation of uncertainty received broad attention in scientific visualization [BOL12; PWL97]. However, uncertainty can also be present in different data sources of information visualization [AY09] and visual analytics [CCM09]. Examples for this are Feng et al. [Fen+10] who propagate statistical uncertainty to parallel coordinates using a density-based approach. Berger et al. [Ber+11] deal with the uncertainty of predictions during sensitivity analysis of multivariate parameter spaces using scatterplots and parallel coordinates.

12: The individual chapters of this thesis also contain additional related work that is specific to the respective techniques.

Bertin [Ber83] and MacEachren et al. [Mac+12] study the intuitiveness and performance of various visual variables for uncertainty in map reading tasks. These visual variables are further refined by Guo et al. [GHL15] for graph edges. Gschwandtner et al. [Gsc+16] compare different representations of the uncertainty of temporal data in the form of time intervals. Hullman [Hul16] investigates the evaluation of uncertainty visualizations and shows that different study designs have a strong influence on the result. Apart from the visual variables described by Boukhelifa et al. [Bou+12], most visual variables are used for representing areas or volumes and cannot be used to encode uncertainty into shapes. Several methods to encode uncertainty information directly into geometry have been developed in the context of other domains. Khlebnikov et al. [Khl+13] deliberately introduce noise into multivariate volumetric rendering to blend multiple variables.

There is also research in the field of perception and awareness of uncertainty. For example, according to MacEachren et al. [Mac+12], fuzziness is considered a valid visual variable for uncertainty. Tak et al. [TT14] considered the use of color for uncertainty, which we employ in Chapter 4 to depict node stress and color labels. Before we turn to this subject, we will consider uncertainty information in hierarchies in the next chapter.

3 Uncertainty in Hierarchical Data

Hierarchical data plays a significant role in visualization as many datasets are hierarchical by nature or because we can impose a hierarchical structure for analysis. Thus, many different representation methods for hierarchies have been developed [Sch11]. On the one hand, there are node-link diagrams that represent structure through nodes that are connected by edges. On the other hand, implicit representations focus on each node's value and encode the hierarchy through inclusion. Implicit methods, such as treemaps, divide the canvas proportional to the sizes of the respective subhierarchies. They mostly optimize for space, which leads to compact and scalable representations.

In our opinion, however, there is a trade-off between compactness and readability — if a treemap is very compact, the underlying structure is difficult to grasp since the compactness does not leave space for grouping cues or other visual features. With traditional treemaps [JS91], it is not feasible to encode additional information such as uncertainty in a geometric way. Furthermore, inclusion via area suggests additive propagation, which might not be the case for uncertainty. Other methods such as circular treemaps [ZL15] waste too much space, and visual scalability suffers to a point where interaction is mandatory to make sense of the visualized data.

As described in Chapter 2, we model uncertainty as distributions of values instead of absolute values per node. Showing the mean value alone is often not sufficient to describe a distribution. Instead, we need visualizations capable of displaying additional statistical features that help the reader gain a better understanding of the data (**RQ1**). Many visual variables do not work well for illustrating uncertainty [Mac+12]. As noted by Hullman [Hul16], uncertainty visualization is error-prone, mainly when the communication of the underlying statistical model insufficient[1]. This chapter introduces a technique that offers flexibility to choose appropriate encodings, depending on the task and underlying model. An additional benefit is that it also works well in black and white.

1: A common mistake is confusing standard deviation with variance.

We strive to find a good compromise between compactness and readability with our new visualization technique: *bubble treemaps*. They allocate extra space in the layout to geometrically encode values and their variability, similar to error bars (**RQ2**). We encode leaf nodes of a hierarchy as circles enclosed by an arc-based parameterizable contour. Sibling subhierarchies are packed using a force-directed model. Together, they are then enclosed by another parameterizable contour. This process recurses until the whole tree has been traversed. Using our method, we can encode additional group-level information, such as uncertainty, into the visual representation of the contours. Figure 3.10 shows a typical example of a bubble treemap that emphasizes nodes with high uncertainty through deformations of the contour similar to box-plots. Our method also maintains the color-coding of higher-level nodes that is typical for treemaps.

The following are the contributions of our work: First, we propose a layout algorithm based on circle packing that uses space intentionally to achieve a reasonable trade-off between a compact representation of the hierarchy and readability (**RQ2**). Second, we define node contours analytically, which results in a new parameter domain for additional visual variables, in particular, for uncertainty visualization (**RQ1**). Our discussion is rounded up by demonstrations of our technique using three example datasets[2].

2: A prototype of bubble treemaps can be found online at `https://github.com/grtlr/bubble-treemaps`.

3.1 Related Work

In Section 2.4, we list prior work on uncertainty visualization. The following section gives an overview of other work related to our method. First, we review the state-of-the-art in visualizing hierarchical data. Next, we summarize recent work for visualizing set memberships as this topic is closely related to how our method encodes the topology of an underlying tree structure.

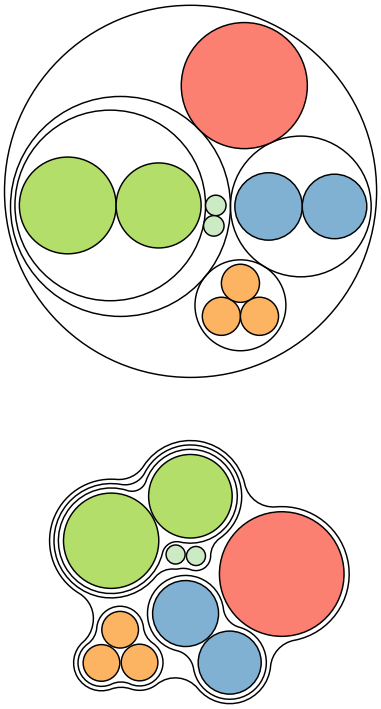


Figure 3.1: Bubble treemaps are initialized using a circular treemap layout and subsequently compacted using a force-based approach.

Visualization of Hierarchies There are many different ways to visualize hierarchies. Schulz et al. [SHS11] provide an extensive survey of implicit visualizations. Most of the treemap approaches, such as squarified treemaps [BHvW00] or Voronoi treemaps [BDL05] follow a top-down strategy, recursively subdividing a given area according to the underlying hierarchy. Similarly, Auber et al. [Aub+13] describe a treemap layout algorithm that produces irregular nested shapes by subdividing the Gosper curve. However, in these methods, the boundaries between areas are not incorporated explicitly into the layout — the contours are inlaid retroactively and not used to encode additional quantitative values.

Circle packing is widely studied in theoretical computer science, especially for its connection to planar graphs [CS03]. Stephenson [Ste05] provides a summary of these findings. Usually, a more pragmatic approach is taken when researchers apply circle packing in visualization: Wetzel [Wet03] and Wang et al. [Wan+06] propose methods based on nesting circles in a bottom-up fashion, which is later refined by Zhao et al. [ZL15]. Viégas et al. [Vié+13] use a combination of circle packing together with a *balloon layout* to visualize information flow in social networks. Several domain-specific works combine dense packing of circles with layout methods from graph drawing such as Bubble Trees or radial layouts [AM11; HBW15]. We utilize the method by Wang et al. to create our initial packings (Figure 3.1).

McGuffin et al. [MR10] provide an extensive study on the space efficiency of different tree representation methods. They introduce a novel metric that aims to measure the distribution of area across nodes in hierarchical visualizations. One conclusion of their work is that a perfect space partitioning might not be ideal when additional information, such as labels, needs to be displayed. Similarly, Schulz et al. [SHS11] argue that packing the space too tightly conceals the underlying structure. In contrast, methods that deliberately leave blank space enable a better perception of the tree structure. These findings were an inspiration for us to develop bubble treemaps.

Bubble charts are often used to visualize three-dimensional data. To create them, we map two dimensions to the x and y axes of the plane. A third dimension can be encoded into the size of a circle at the corresponding position. Multiple websites [Car12; FIN17] employ bubble charts as part of interactive data exploration tools. These examples, however, neglect either the category of the entities or their entire hierarchical structure.

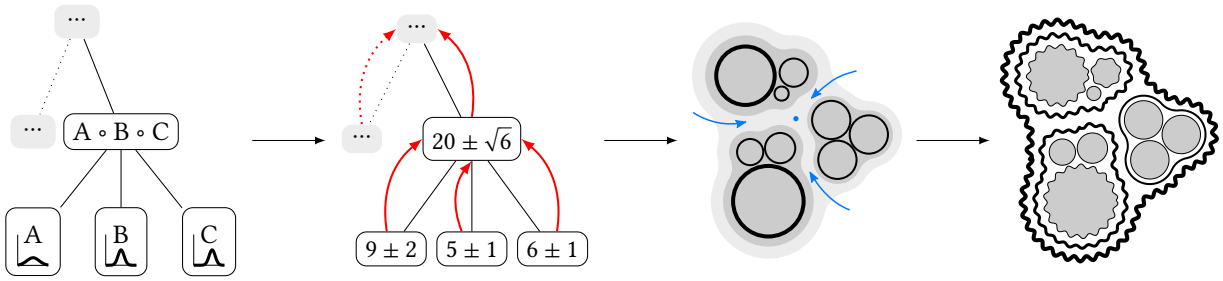
Contours and Set Membership As mentioned above, treemaps encode hierarchies implicitly by aggregating the areas of the child nodes into their parent's area. Our method is similar: All child nodes are enclosed by a contour that represents the current node. As a result, our approach shares similarities with methods that depict set membership for spatial objects. For example, Bubble Sets [CPC09] use marching squares — a 2D version of the marching cubes algorithm [LC87] — to draw contours around embedded objects, while we use arcs. Kelp diagrams [Din+12] and especially the refined version Kelp Fusion [Meu+13] share more similarities with our method, even though they were developed for geographical data. Notably, the authors mention the potential of enclosing areas by contours based on arcs but do not provide further details. Riche et al. [RD10] present a method that builds upon Euler diagrams to visualize set membership by drawing contours around objects of the same logical group while minimizing the number of crossings between contours of different groups. In the last years, several methods have been developed to improve Euler diagrams: Force-directed methods are utilized to optimize their respective layout [MR14b] and to find smoother boundaries [SAS16]. There is also work on drawing area-proportional realizations of Euler-like diagrams [MR14a]. While Euler-like diagrams are similar to our approach, they do not take hierarchical structure into account.

Our method for contouring is similar to the modeling of approximate solvent-accessible surface areas from the field of biomolecules [LR71]. They model the surface area of a molecule that is accessible by a probe with a fixed radius. There are efficient algorithms that compute these surfaces analytically, but they make assumptions on the structure of the molecules that do not hold for the general case of arbitrary intersecting spheres [HO98].

3.2 Method

Figure 3.2 gives an overview of our method. From a hierarchy of values with uncertainty as an additional leaf attribute and an aggregation model, we construct a bubble treemap by first extracting characteristics from distributions for each level of the hierarchy. Then, we map these characteristics to circles for the leaves and analytical arc contours for inner nodes, respectively. To achieve a compact layout, we implement a force-directed model.

A hierarchy with uncertainty is represented by a tree $T = (V, E, A)$, similar to a regular tree in graph theory, with vertices V (nodes), edges E, and attribute vectors $\boldsymbol{a}_i \in A \subseteq \mathbb{R}^n$ associated with each node. Typical attribute vectors of interest would be the mean and standard deviation



(a) Uncertain Hierarchy (b) Uncertainty Propagation (c) Layout Algorithm (d) Rendering

Figure 3.2: Overview of the bubble treemap method. We start with measured distributions organized in a hierarchy (a). We usually only know the distributions at the leaf level, so we need to propagate them towards the root using a suitable uncertainty model (b). Then, we compute the treemap layout using circular arcs and a force-based model (c). Finally, we draw leaf circles and inner-node contours around each level of the hierarchy (d).

$(\mu, \sigma)_i$. Current approaches can only show a single value per node. With our technique, we aim to display multiple statistical properties of the underlying distributions at once. We achieve this by not striving for a perfect partitioning of the space. Instead, we allocate additional space that we then use to encode information about the uncertainty.

In Section 2.2, we have described models of uncertainty. Then, in Section 2.3, we saw two different approaches to propagating uncertainty. Propagating uncertainty from the leaves to the root node falls into the category of analytical methods. In the next sections, we describe how to construct arc contours and compute our treemap layout. Afterward, we discuss the usage of visual variables for uncertainty within bubble treemaps, followed by three example data sets from different domains that demonstrate our method's usefulness. Finally, we discuss the details of our implementation and the limitations of our method.

Circular Arc Contours

We recursively draw contours around inner nodes and leaves to depict parent-child relationships. Instead of using an implicit description of the contour and the marching squares algorithm for rendering, we construct the contour as a parametric curve made of arc segments. This approach benefits from a geometric construction that we can utilize to encode an additional attribute dimension. We do not need to discretize space to draw the contour (which could lead to discontinuities), and it allows us to describe the contour directly using arc segments. Furthermore, bubble treemaps do not require color and can be described entirely using vector graphics. In the following section, we explain the different steps necessary to construct the contour — Algorithm 1 describes the complete procedure.

For a given inner node $n_i \in V$, we need to find an enclosing surface that includes all circles of $\textsc{leaves}(n_i)$. Computing the enclosing contour consists of three parts: First, we need to find the circles that make up the envelope $E_i \subset \textsc{leaves}(n_i)$. The envelope contains the exposed subset of circles at the outside of the set. By iterating over these circles, we can construct the contour as a circular arc spline defined as a sequence of biarc

curves [Bol75]. A biarc curve consists of two arc segments that share the same tangent direction at the connection point, which leads to a smooth transition (G^1 continuous). Therefore, all elements stay circular, simplifying the layout computation.

Parameters Several parameters influence the space requirements of the contour. They can be defined separately for each node n_i of the tree (leaves and aggregate nodes alike). The margin m_i describes the distance of the contour from the underlying structure. The parameter w_i reflects the contour's width, which is essential if we want to encode additional information directly into the contour or use the contour to emphasize the structure of the tree. At last, the padding p_i models how close adjacent objects can be placed. From the layout perspective, we are only interested in the total amount of space that is required for the contour, which we define as $d_i = m_i + w_i + p_i$ for a corresponding node n_i.

In addition to the parameters above, we assign a smoothness parameter s_i for each tree level. This parameter controls how tightly the contour will fit around the CHILDREN(n_i) and represents the radius used for the tangent arc. Varying s allows us to adopt several concepts from computational geometry; we discuss the influence of this parameter on the contour in Section 3.5.

The basic primitive of our method is the construction of the tangent arc. Formally, given two circles a and b with center points at $\boldsymbol{p}_a$, $\boldsymbol{p}_b$ and radii r_a, r_b and the desired radius of the tangent arc $r_t \geq \|\boldsymbol{p}_a - \boldsymbol{p}_b\| - (r_a + r_b)$, we can find the center of t by virtually enlarging a and b by r_t to obtain a' and b':

$$r'_a = r_a + r_t \quad \text{and} \quad r'_b = r_b + r_t$$

The intersection $(a' \cap b')^+$ gives us the center p_t of t. Truncating this circle (with radius r_t) to the length between the two tangent points of t with a and b gives us the desired tangent arc. Figure 3.3 outlines the construction with tangent arcs shown in red.

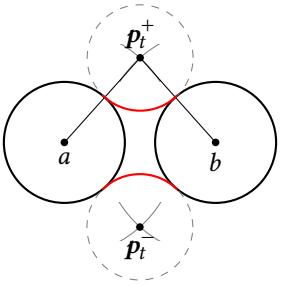


Figure 3.3: The tangent arcs of two circles can be constructed from the intersection points $\boldsymbol{p}_t^{\pm}$ of their enlarged versions.

Finding the Envelope For finding the envelope E_i, we virtually enlarge each circle of LEAVES(n_i) by $d_i + s_i$. The result of this step is the set of circles C'. We can now compute the *intersection graph* of C', a graph that contains an edge (c_i, c_j) with $c_i, c_j \in C'$ iff $c_1 \cap c_2 \neq \emptyset$. For the sake of simplicity, we assume the graph to be connected — a disconnected intersection graph would require a larger smoothness parameter s_i. Next, we find the circle with the leftmost point among all elements of C', which has to be part of E_i by construction. Starting from this element, we can traverse the intersection graph, always choosing the edge that leads to the circle with the leftmost intersection point (as shown in Figure 3.4). The selection procedure is shown in Figure 3.5. Here, the current circle is a and we consider the intersection points $i_1, \ldots, i_4$, comparing their respective angles to $\boldsymbol{v} = \boldsymbol{p}_b - \boldsymbol{p}_a$. We only need to consider the angles that are counter-clockwise to $\boldsymbol{v}$. From those, we choose the largest one (α in this case). Note that it is not sufficient to find the outer face of the embedded intersection graph because there might be small circles that are skipped depending on the specified smoothness s, which is the case for

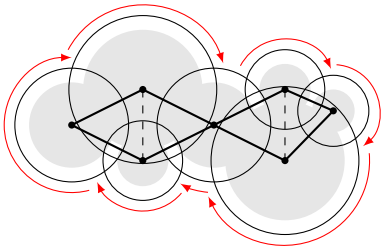


Figure 3.4: The intersection graph of a set of circles. The original circles in gray are enlarged to reflect the smoothness parameter s. The red arrows show the traversal that computes the envelope.

circle c in Figure 3.5 — our proposed selection method solves this problem.

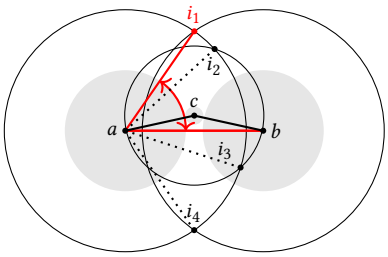


Figure 3.5: Selection of the next circle from the envelope. The leftmost intersection point i_1 is shown in red. Note that the procedure skips circle c in this configuration.

Constructing the Contour We use E_i to construct the contour. To create the circular arc spline, we first add a tangent arc to each neighboring pair of circles. In the second step, we convert these circles to arcs. Then, we set the start angle α and the length θ of the arc segment by converting the left and right neighbors of each circle to polar coordinates centered at the current circle. It is important to note that we need to handle the cases of *inward arcs*, which are oriented clockwise, and *outward arcs*, which turn counter-clockwise.

The described method only works if the intersection graph is connected. Additionally, the smoothness parameter s is constrained by the maximal distance $d = \|\boldsymbol{p}_a - \boldsymbol{p}_b\|$ between two circles $a, b \in C_i$, so that the contour will not intersect itself:

$$s \geq r_a'^2 - \frac{\left(r_a'^2 - r_b'^2 + d^2\right)^2}{4d^2} \tag{3.1}$$

If a and b move further away from each other, the tangent arc will move into the gap between them. By constraining smoothness this way, we prohibit that the tangent arc moves across the line segment connecting the centers $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ of the two circles. Limiting s as described in Equation 3.1 also covers the case where the radius of the tangent arc is too small to find a tangent point for each a and b, which is the case when a' and b' do not intersect.

Algorithm 1: Construction of the contour

```
1  function Contour(n_i)
2      let E be a sequence that represents the envelope
3      C' ← enlarge LEAVES(n_i) by d_i + s_i
4      c ← element from C' with leftmost extent
5      E.push (c)
6      c ← circle with leftmost intersection
7      while c ≠ E[0] and c has unvisited leftmost intersection do
8          E.push (c)
9          c ← circle with leftmost intersection
10     let R be a sequence that will hold the contour
11     forall adjacent pairs c_1, c_2 ∈ E do
12         t ← tangent arc between c_1 and c_2
13         t_1, t_2 ←truncate c_1 and c_2 to t
14         R.push (t_1, t, t_2)
15     return R
```

Layout Algorithm

Similar to circular treemaps [Wan+06; Wet03; ZL15], our layout algorithm maps one attribute dimension, such as the expected value, to the area of the circles. The topology of the underlying tree is encoded implicitly through containment. Therefore, the area of a child lies entirely

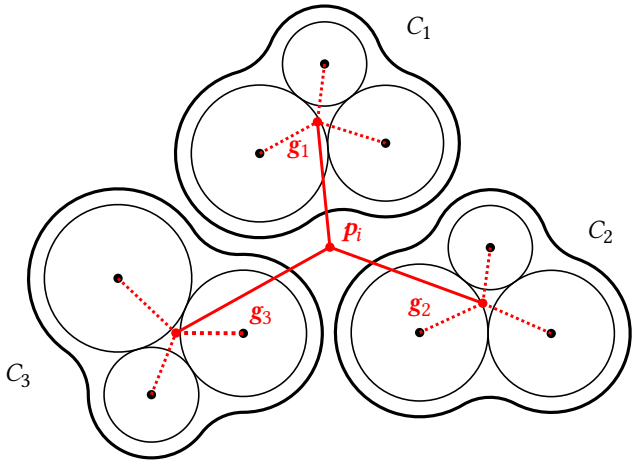


Figure 3.6: Schematic of the force-based method for two levels of the hierarchy. After laying the children out in the first step (dashed), they become fixed and will be moved as a whole in the second step (solid).

within the area of its parent node. To achieve a compact representation, we use an adapted version of the circular treemap algorithm to initialize our layout, similar to the one described by Wang et al. [Wan+06]. Afterward, we traverse the hierarchy bottom-up and perform a force-based circle packing while accounting for the space occupied by the respective contours of the subhierarchies.

Once we have the initial layout, we perform a post-order traversal over the circular treemap and transform it into a bubble treemap. We achieve this by constructing a spring-based system, as shown in Figure 3.6, for each subhierarchy. The post-order traversal that visits each node $n_i \in V$ starts from the leaves, which at first remain in the arrangement determined by the circular layout algorithm. In subsequent steps of the traversal, the elements C_k of each child $k \in \textsc{Children}(n_i)$ are grouped together using a contour G_k, as explained earlier. Depending on the tree's structure, the elements of C_k can either be circles, coming from leaf nodes, or contours constructed in previous steps. We can interpret C_k as a rigid body, with mass distributed according to its area, on which external forces can be applied. Then, we define the center of the current (circular) node n_i as the center of a spring system with a fixed position $\boldsymbol{p}_i$. Next, we compute the center of mass $\boldsymbol{g}_k$ for each k and connect it to $\boldsymbol{p}_i$ using a spring.

Figure 3.6 shows an example of such a setup for two levels of a hierarchy. The springs can be seen as attractors that pull each C_k toward $\boldsymbol{p}_i$. We then simulate the forces in the system using a physics engine, avoiding collisions between each C_k, to create a force-based layout. In practice, we found that approximating the contours by virtually enlarging each circle of $\textsc{leaves}(n_i)$, to the extent of the contour, already yields good results while simplifying the configuration of the physical simulation.

After computing the force-based layout for n_i, we fix the relative positions of the elements of C_k to each other. From this point on, in later steps of the post-order walk, we always transform them as a whole. Algorithm 2 provides a detailed description of the algorithm.

Algorithm 2: Hierarchical bubble treemap layout

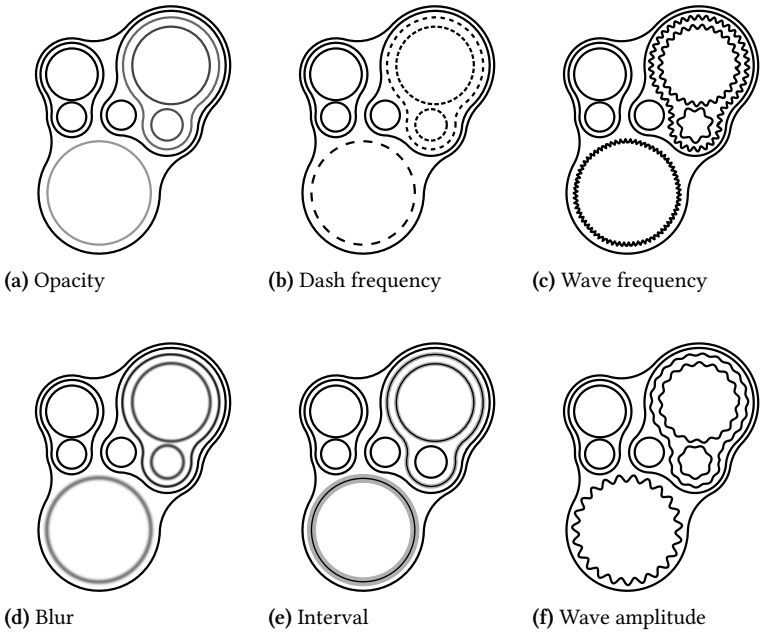
```
1  function LayoutNode(n_i)
       Input: n_i is a node of a tree with circular layout
2      p_i ← center of n_i
3      W[] ← empty list of rigid bodies
4      forall k ∈ CHILDREN(n_i) do
5          LayoutNode(k)
6          C_k ← list of elements for each k ∈ CHILDREN(n_i)
7          forall C ∈ C_k do
8              G ← create rigid body from Contour(C)
9              g_C ← center of mass of G
10             connect g_C to p_i using a spring
11             W.push (g_C)
12     SimulateForces(W)
13 function BubbleTreemap(T)
       Input: T is a tree with circles in the leaves.
14     root ← CircularTreemap(T)
15     LayoutNode(root)
16     return root
```

3.3 Visual Variables

Visual variables in the context of diagrams and maps have been investigated extensively [Ber83]. The expected value, depicted by the node area, adds up from the leaves to the root, analogous to conventional treemaps. Please note that we could also encode uncertainty inside nodes, for example, by using radial gradients like Vehlow et al. [VRW13], at the cost of a design dimension to encode additional information. Instead, we restrict our discussion to the encoding of uncertainty on the contour.

Our technique only requires black and white (cf. Figure 3.7) and offers a comprehensive set of design choices regarding visual variables. Usually, we desire equal saliency between certainty and uncertainty, with the exception of detection tasks being considered an exception. Based on work by MacEachren et al. [Mac+12], we start our discussion using opacity as baseline for uncertainty (Figure 3.7a). Because of the small line width, the difference between various nodes is barely visible. If contrast is an issue, experimenting with more clean and geometric visual variables for uncertainty is an obvious choice. With sketchiness being considered unprofessional [Bou+12], we choose to map clean waveforms on the contours. The first one is dash frequency (Figure 3.7b), resembling a rectangular signal, and the second one is wave frequency (Figure 3.7c), resembling a sinusoidal signal. As expected, both visual variables are easily readable, provide more perceivable levels, and better highlight large values. Despite the visual similarity to *dashing* and *sketchiness*, we refrain from judging intuitiveness based on related work because our application is very different. Dash frequency seems to introduce high-frequent noise. Therefore, the dashes' frequency (and phase) have to be selected carefully to avoid interferences between different lines of the same hierarchy and among siblings of subhierarchies.

To discuss saliency, we present a set of visual variables with varying contour width. We have implemented *fuzziness* [Mac+12] using blur to



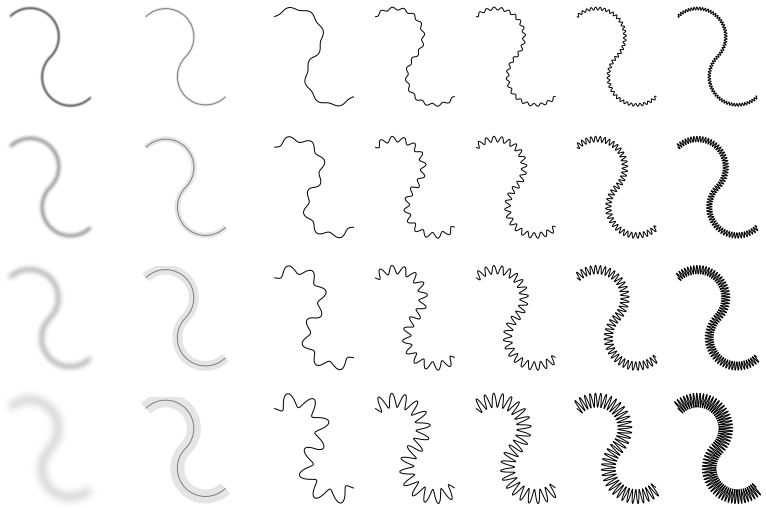
(a) Opacity **(b)** Dash frequency **(c)** Wave frequency

(d) Blur **(e)** Interval **(f)** Wave amplitude

Figure 3.7: Example visual variables applied to the contour. Opacity (a), dash frequency (b), and wave frequency (c) have a constant contour width, whereas blur (d), interval (e), and wave amplitude (f) have variable contour widths per level.

preserve the color and mass of the dissolved lines. Blur (Figure 3.7d) is more readable than opacity and introduces much less noise and saliency than dashed lines or wave frequency. The levels of blur (Figure 3.8a) are difficult to distinguish, which is in line with the findings of Boukhelifa et al., who found that up to four levels of blur can be discerned [Bou+12]. Regarding intuitiveness, Correll et al. [CG14] discuss a binning effect between certainty and uncertainty.

The next one is a representation that is inspired by error bars. To prevent confusion, we call this visual variable interval (Figure 3.7e). Regarding intuitiveness, we expect it to be very close to the well-known error bars. At first glance, smaller levels are more challenging to recognize, whereas higher levels are easier to distinguish (Figure 3.8b). The last visual variable is wave amplitude at a fixed frequency (Figure 3.7f). Please note that there is a dependency between those two variables regarding perception: Low frequencies are detrimental to distinguishability, and high frequencies lead to a Moiré effect (Figure 3.8c). From the same figure, we expect that the perception of frequency depends on the curve's geometry. Nevertheless, sine waves with constant frequency and a variable amplitude seem to work well. We speculate that the amplitude acts as a highlight while frequency allows quantitative encoding. Studying their dependencies is left for future work. We suspect that differences in overall value (cf. Figure 3.7 and Figure 3.8) shift saliency toward certainty or uncertainty, respectively. Therefore, to achieve equal saliency, we suggest counterbalancing based on value, for example, by integrating all pixels of each contour within a certain area and then compensating by adjusting the intensity.



(a) Blur **(b)** Interval **(c)** Wave frequency × amplitude

Figure 3.8: Multiple levels for each of the visual variables for varying contour widths. Please note how the frequency influences the perception of the amplitude.

3.4 Examples

This section aims at demonstrating the usefulness of our technique using exact data as well as uncertain data. In the following examples, we use color to differentiate between categories. In the *Flare* data set, we colorize each group of children with the same color, whereas, in the other datasets, we select a different color for each subhierarchy starting at the children of the root. Our prototype is implemented in C++, using *Box2D* for the force-directed layout (8000 iterations) and *Cairo* for vector graphics output. Table 3.1 provides a summary of the example datasets.

Package Structure of Flare

Figure 3.9a shows the structure of the Flare data visualization software, which comes from the *UC Berkeley Visualization Lab*[3]. The Flare software consists of 10 modules that can contain further submodules. This example contains only certain data and aims to show the structural properties of our approach. We map the size of each module to the area of the circles in the leaves. Figure 3.9b shows a squarified treemap [Bos17] of the same dataset, slightly adapted to fit our color palette better. Please note that the colors of the bubble treemap match the ones of the squarified treemap to allow a better comparison — which is still possible even though such non-uniform colors in subhierarchies impair the readability.

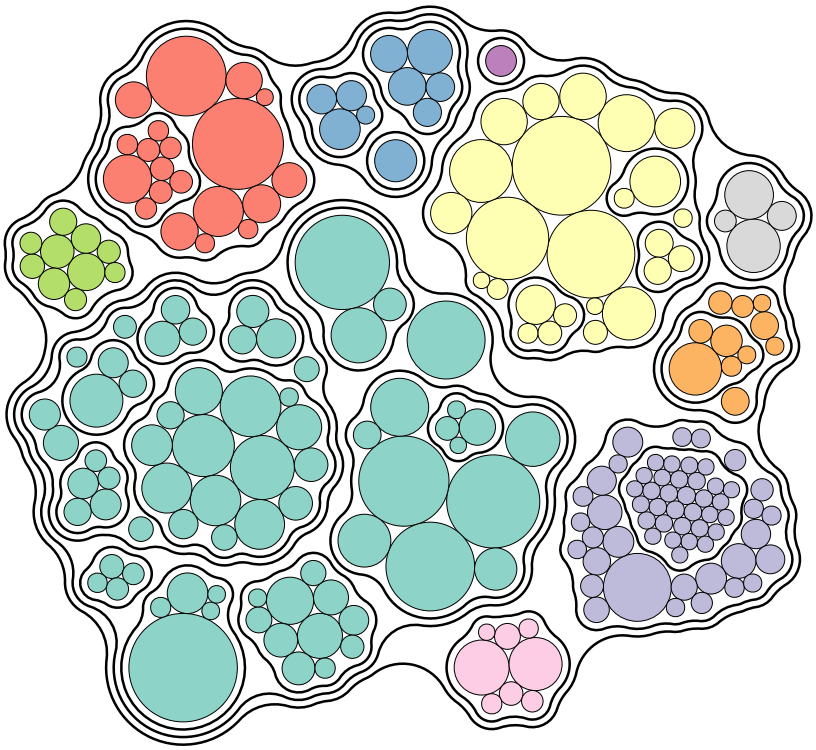
A particular problem of many traditional treemap approaches is that the hierarchy is hard to read and might even be ambiguous. Thus, treemaps use color and different shading styles, as shown in Figure 3.9b. Even though we can also use color to increase readability, it is not necessary

3: The dataset was created by Jeff Heer and is part of the examples of *D3.js*. It can be found at `https://bl.ocks.org/mbostock/4063582#flare.json`.

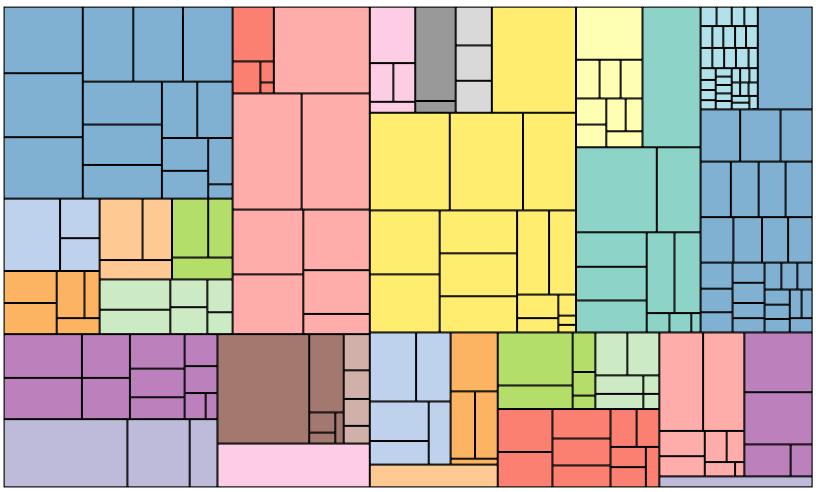
Table 3.1: Summary of example datasets. The runtime was measured on a desktop workstation equipped with an Intel i7–4770 CPU at 3.9 GHz.

Dataset	nodes	leaves	depth	uncertain	time [s]
Flare	252	220	4	no	3.2
S&P 500	639	503	3	yes	6.3
CES	403	295	6	yes	7.7

for our method. In Figure 3.9a, we take the colors from Figure 3.9b and transfer them to our treemap. The resulting coloring is even a bit disadvantageous since now different colors are placed in a subhierarchy. However, our visualization remains readable due to the layering of outlines and the additional blank space. Not being restricted by color means that we can use this strong visual cue for showing additional aspects of the data. It also supports our claim that reserving some extra space offers advantages for visualizing hierarchies.

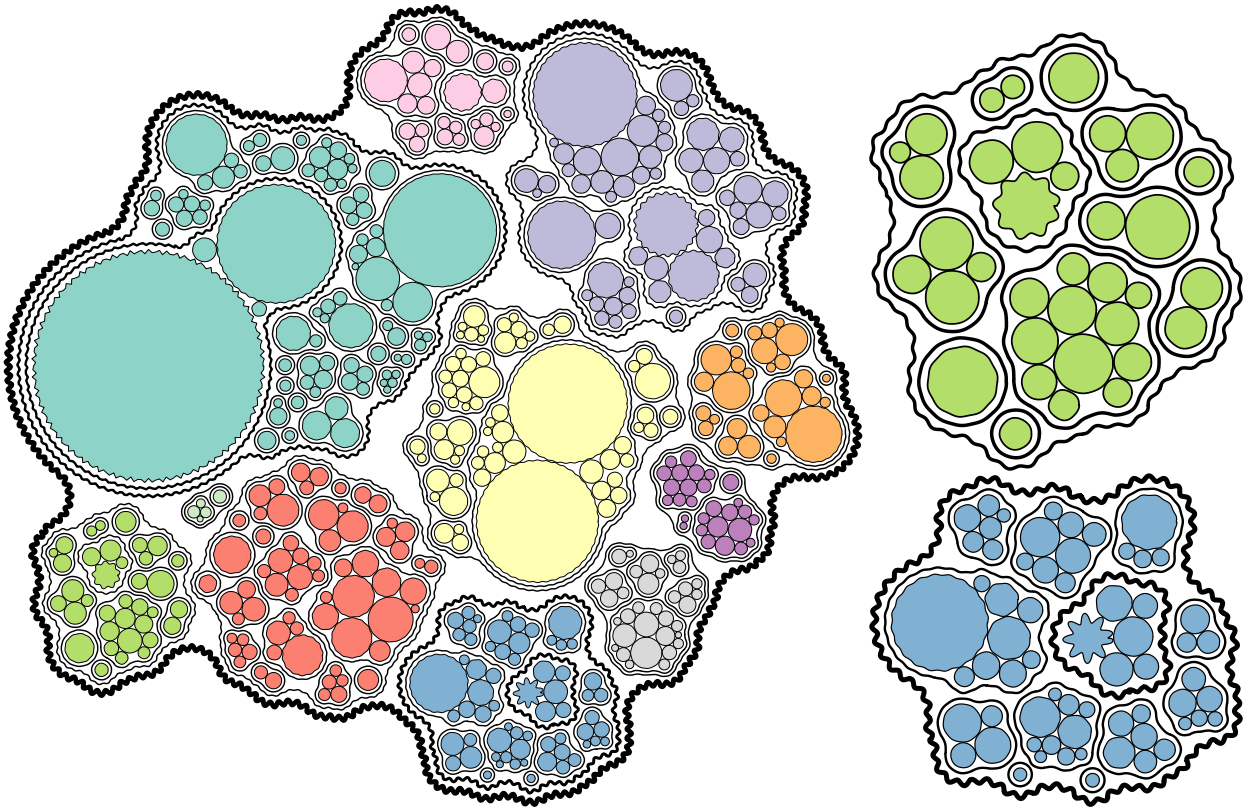


(a) Bubble treemap



(b) Squarified treemap

Figure 3.9: Visualization of the package structure of the Flare software: (a) our method, (b) squarified treemap for comparison. The color coding is the same for both visualizations. However, qualified treemaps, in contrast to bubble treemaps, require color to avoid structural ambiguities.



(a) Full dataset **(b)** Sectors with low/high uncertainty

Figure 3.10: (a) Bubble treemap of the S&P 500 index, decomposed into sectors and companies. Uncertainty arises from aggregating one week of stock data in November 2016. Each circle represents a stock; its area is proportional to the mean closing price, whereas we depict the standard deviation using the outlines. Our visualization helps to discover a medium-sized sector with low uncertainty and assess its composition (b, top), as well as a sector with high uncertainty and the company that mostly introduced it (b, bottom)

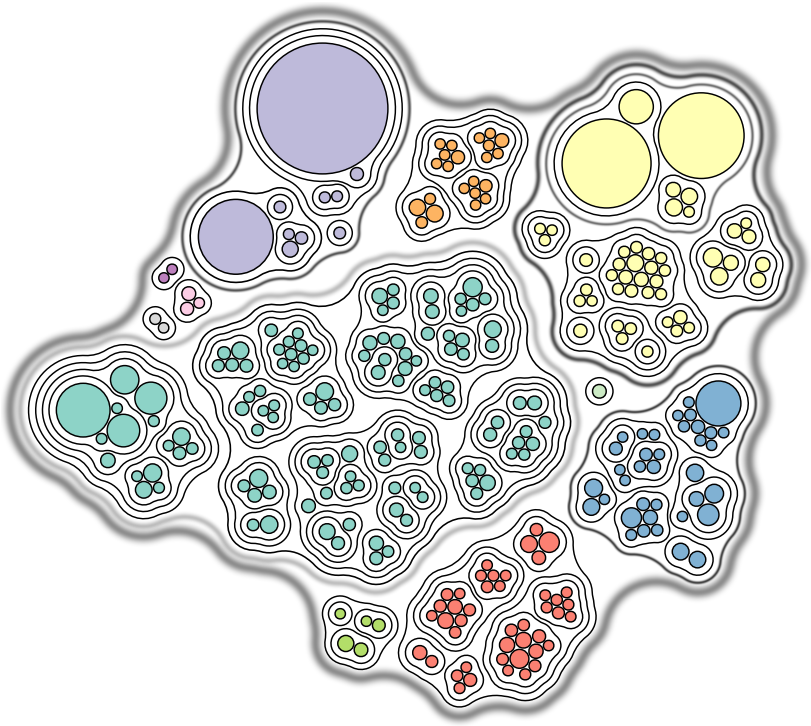


Figure 3.11: Visualization of the data from the *Consumer Expenditure Survey*. The leaf size is proportional to each item's value in the survey; we encode the standard error into the contours' thickness. Also, contours with high uncertainty are blurred to give the impression of uncertainty. The food category (cyan) and the housing category (yellow) have a high standard error and therefore exhibit a larger blur.

S&P 500 Index

Traditionally, financial data has been visualized using treemaps. Analysts are usually interested in a stock's performance over time. Our proposed method can be used to show the current value of a stock as well as supplemental information about its behavior over a given range of dates, providing additional context. In many cases, it is also of interest how well an industry or a sector performs. Figure 3.10 shows a visualization of the companies that are part of the *Standard & Poor's 500* index (S&P 500), grouped by sectors and industries. For this example, we collected data using the *Yahoo Financial API*, for one week in November 2016. The size of the circles represents the mean closing price of the stock for the given week. We use the contour to show the standard deviation σ of each stock. Even though stocks can depend on others, we assume that they behave independently for this visualization, which allows us to use the uncertainty model as described in Section 2.2 to propagate σ toward the root.

Our visualization shows stable stocks during the given period and others with more considerable variations. By looking at the contours' waviness, it is relatively easy to identify the stock with the largest changes since its variance propagates upwards in the contours[4].

4: In this case, the reason for the large variability was a 5-for-1 stock split, which led to single stock only having a fifth of the original price.

Consumer Expenditure Survey

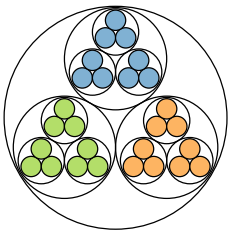
The Consumer Expenditure Survey (CES) is an annual survey by the United States Department of Labor that measures the income and the expenditures of *consumer units*, which are families or households in the United States. Overall, there are about 109 million such consumer units, out of which approximately 30,000 consumer units are sampled [Bla03]. From these samples, the mean expenditures are estimated.

Even though the consumer units are chosen carefully to reflect the population, finding a sample that is perfectly representative is impossible. The sampling error introduced by this method is measured using standard errors and gives information about the uncertainty for each value. The survey uses *stratified random sampling* instead of simple random sampling. Because of this, the usual standard textbook formulas do not apply here[5].

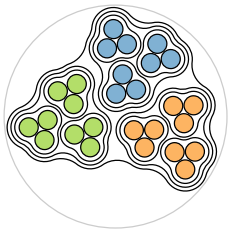
5: Blaha [Bla03] describes the replication methods used to calculate the standard error, namely the *balanced repeated replication*.

We use our method to visualize the diary survey of the 2014 dataset[6], the result can be seen in Figure 3.11. We map the value of each item to the radius of the circles. For our visualization, we use a combination of thickness and blur to show uncertainty. Each of these visual variables alone would suffer from deficiencies: Blur might get hard to read quickly since the contour would become too light, whereas thickness alone would be perceived counter-intuitively (uncertain values would appear very thick). Blur can only be perceived correctly up to four levels [Bou+12]. Hence, we map the uncertainty to four thickness levels and set the blur proportional to each level. Our visualization shows two categories afflicted with a high standard error: the *Food* category, shown in cyan, and the *Housing* category, colored in yellow. Within the housing category, the subcategory with the highest standard error is *Fuel and Utilities*.

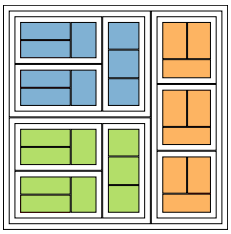
6: The full dataset is available from the United States Bureau of Labor Statistics (BLS) at `https://stats.bls.gov/cex/programs/r14.zip`



(a) Circular treemap



(b) Bubble treemap



(c) Nested treemap

Figure 3.12: Size comparison between circular treemaps (a), a corresponding bubble treemap (b), and a nested treemap (c) for a 3-ary tree where all leaves have the same value. When the branching factor increases, the difference becomes less pronounced between (a) and (b).

3.5 Discussion and Limitations

In comparison to circular treemaps, we can use space more efficiently, especially for k-are trees with a small branching factor. Figure 3.12 shows this problem for a 3-ary tree with four levels and leaves of equal size. In the most extreme case, namely, a binary tree where the left and right children have equal sizes r_c, the parent circle has to have a radius $r_p = 2r_c$. In this case, only 50% of the area of the parent circle is used. Rectangular treemaps are perfectly space-efficient, instances of these are squarified treemaps (Figure 3.9b) and nested treemaps (Figure 3.12c). Note that the leaves can appear unequal when the aspect ratio is not the same.

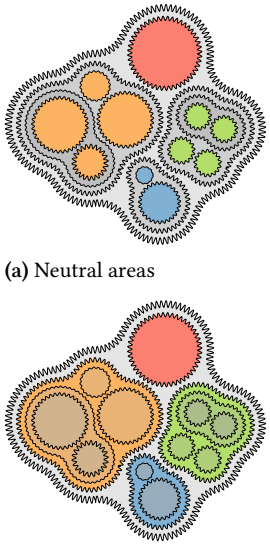
Arc Primitives The main visual characteristic of bubble treemaps is that the inner structure, defined by an arrangement of leaf nodes, reflects outwards through the contours. The layering of contours leads to equidistant curves that capture the underlying tree structure. This effect would be difficult to achieve with energy-based contouring methods such as implicit curves or splines. Using circles and arcs as basic primitives of our method has several advantages: The users already have a good intuition for interpreting bubble treemaps because circle primitives are used widely throughout many different visualizations. Furthermore, adding labels to circles and biarc curves should be simple and visually pleasing because of their clean geometry. For example, labels can be added by either allocating more space per node or, if suitable, using segments of the contours. Another compelling reason to build our method upon circles is that this leads to clear visual outlines and a more engaging visualization. Other circle-based visualizations [HBW14; WZY16] show that arrangements of circles are judged as aesthetic.

Area Perception Like circular treemaps, bubble treemaps do not reflect the aggregate size of a subhierarchy in the area enclosed by a contour. Filling the inner node areas with color would allow us to encode information, such as the hierarchy's topology. There is, however, a risk of shifting saliency regarding aggregation of expected values. This effect is illustrated in Figure 3.13, where the topology of the tree is emphasized using different shades of gray (Figure 3.13a) and color (Figure 3.13b). Filling each level's area in a neutral color improves the perception of groups and the hierarchy's depth. In Figure 3.13b, each subhierarchy was assigned a color, and different levels of depth are emphasized by decreasing the saturation. This highlights the group structure of each node but introduces a bias in area perception: The aggregation of the green group now falsely appears larger than the single red node. The potential error amplifies with increasing contour thickness, which further aggravates the interpretation under uncertainty. Therefore, we advise against filling inner node areas if a visual aggregation of leaf nodes is desired.

Hierarchy Perception In general, the efficiency of treemaps decreases with increasing tree depth because the implicit representation then tends to hide the underlying structure. Bubble treemaps share this characteristic to some degree: It is difficult to grasp the depth of subhierarchies that are not directly adjacent to contours that layer upwards to the root node.

In all other cases, such as the green node to the very left in Figure 3.11, reading the depth is done by counting the number of contours outside a group. As shown in Table 3.1, the datasets in this thesis have a depth that ranges from three to six levels and up to about 500 nodes. Our proposed method works well for such hierarchies, but we expect that deeper hierarchies will pose greater challenges.

Depending on the chosen visual variables and their parameters, uncertain regions can appear more salient than regions without variability. While this helps in some scenarios, for example, when searching for categories with high uncertainty, in the general context of uncertainty visualization, this might be confusing to the user. One way to deal with this problem is to adjust the opacity of regions with uncertainty, giving the user better visual cues to interpret the visualization while retaining the quantitative encoding. Regarding the encoding, we expect frequency and amplitude to behave similarly to gradient as a visual variable in that only a certain amount of levels can effectively be perceived. Our intuition is that a more fine granular distinction between levels of uncertainty should be possible. Figure 3.8 provides some initial evidence for this.



(a) Neutral areas

(b) Colored areas

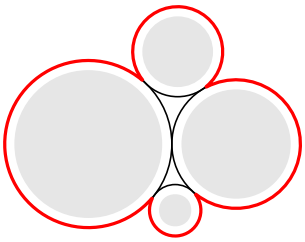
Figure 3.13: Comparison of neutral shading (a) and colored (b) inner node areas. While filling contour areas allows us to encode additional information, it also introduces bias regarding area perception: In (a), the sum of the green leaves correctly appears smaller, while in (b), the sum of the green area erroneously appears more prominent than the red node.

Computational Complexity The complexity of constructing the contour strongly depends on the smoothness s. If s is much greater than the largest radius $\max(r_i)$, this will lead to the degenerate case of the intersection graph where each circle intersects all other circles. The successor of the current circle is the circle with the leftmost intersection point. While traversing the intersection graph, we have to consider all other circles when searching for the next element, which leads to a computational complexity of $O(n^2)$.

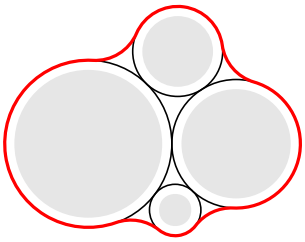
The runtime for finding the layout is bound by the simulation, mainly how many iterations are needed to achieve a good constellation. Finding a good compromise between quality and runtime and predicting the exact number of iterations in advance is difficult. We have found that 8000 iterations usually lead to good results for the datasets that we show in this thesis.

Table 3.1 shows runtimes for different datasets, as well as additional information about each dataset. Even though the force-based simulation determines our algorithm's performance, several characteristics influence the runtime of the computation. The overall number of nodes is the most obvious one. However, since we perform the physical simulation for each subhierarchy, the tree's maximum breadth and depth are also important factors. This reflects in the runtime of the CES dataset, which has fewer nodes than the S&P 500 dataset, but it still takes longer to compute due to the larger maximum depth.

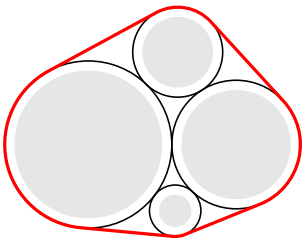
Parameters of the Contour Wrapping the contour around packed circles gives it its visual characteristic. By adjusting parameters that define the contour, namely the smoothness s and the padding p, we can emulate different concepts from computational geometry. As described in Section 3.2, s controls the radius of the tangent arcs and can be used to steer how closely the contour will cling to the underlying circles. The effects of the contour parameters are also shown in Figure 3.14. When



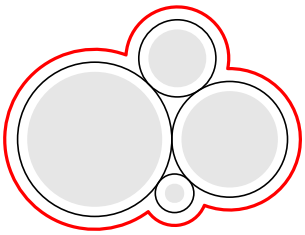
(a) $s \to 0$



(b) $s = 14$



(c) $s \to \infty$



(d) $p = 4, s \to 0$

Figure 3.14: Setting the smoothness factor s toward infinity yields the convex hull of the set of circles. The other parameters of the contour are constant. We can use the same construction to obtain the offset polygon of the circles by introducing an additional padding for each circle.

$s \to 0$, we obtain the concave hull of the set of circles (Figure 3.14a). Increasing s will relax the contour, therefore decreasing its total perimeter (Figure 3.14b). Finally, for $s \to \infty$, our algorithm computes the convex hull of the set of circles (Figure 3.14c). Independent of s, we can also adjust the padding p of the contour, which controls how far the contour will offset from the original circles. By additionally setting $s \to 0$, we simulate the offset polygon (Figure 3.14d). Adjusting these parameters per level, s in particular, can be used to improve readability further. For the provided examples, we slightly reduced s with each level, which leads to contours that resemble isocontours.

We imagine that the generality of the contour description makes it applicable in other visualization contexts too. For instance, bubble charts that visualize hierarchical or categorical data could also benefit from this contouring method. Lately, advances have been made in the field of visualizing set membership for objects embedded in the plane [CPC09; Meu+13]. Our method for constructing contours around objects can be applied to draw contours around arbitrary objects embedded in the plane. For this to work, it should be possible to define a bounding circle for each object. We can then choose a suitable smoothness parameter by analyzing the maximum distance of each cluster's elements.

3.6 Conclusion

We have presented bubble treemaps, a novel method that allows us to visualize hierarchical data afflicted with uncertainty. The main idea is to deliberately allocate extra space that we use to encode additional information. For this, we have presented a method to group circles together using contours based on biarc splines. We have described a hierarchical force-based layout algorithm that we use to transform a circular treemap into a more compact representation. Since encoding uncertainty into visualizations is complicated, our method tries to leave as many design choices as possible. We show how several visual variables can effectively convey uncertainty while still showing the original structure of the hierarchy.

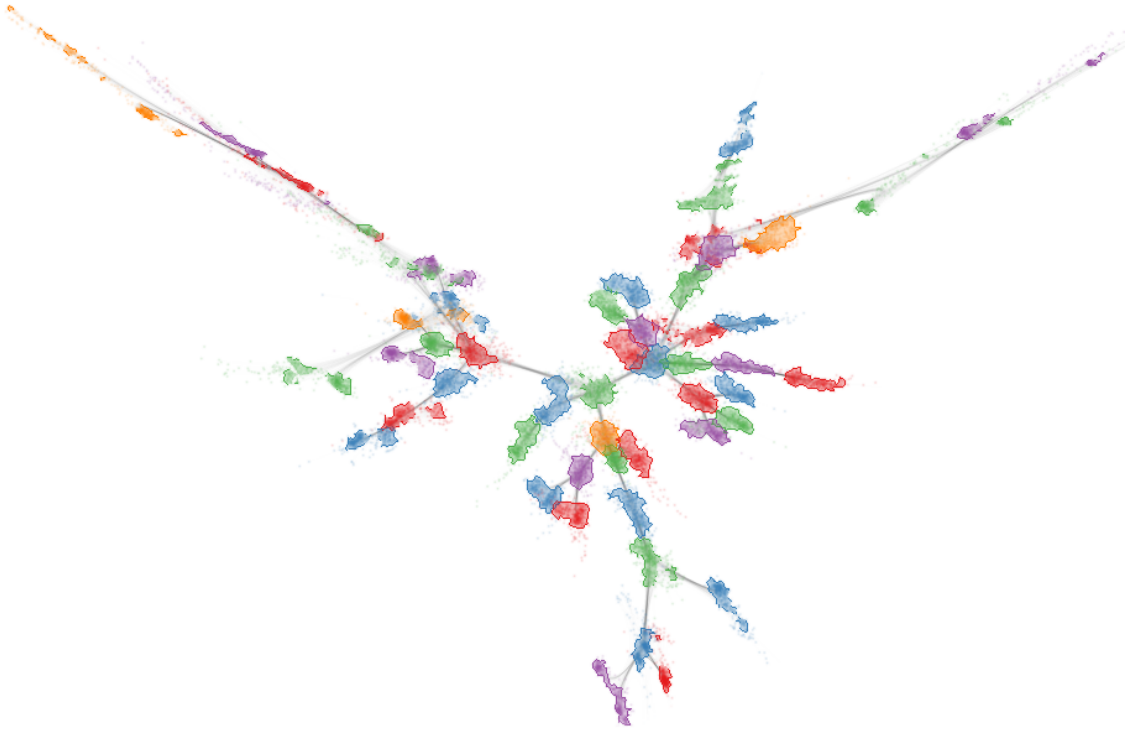
In future work, we want to incorporate temporally changing data into our visualization. Currently, we use the circular treemap algorithm as initialization for our method, so we inherit one of its properties: The initial layout is not stable. However, after performing the force-based layout, our algorithm could be used for dynamic or interactive representation by repeating the force-based layout procedure if the leaves' changes are not too large. We are confident that the subsequent adjustment should allow user interaction. Furthermore, we are interested in incorporating appearing and disappearing hierarchies. To achieve this, we imagine basing the initialization on space-filling curves would create some robustness against changes. Visualizing changes in the topology is especially relevant for depicting topological uncertainty.

4 Layout of Probabilistic Networks

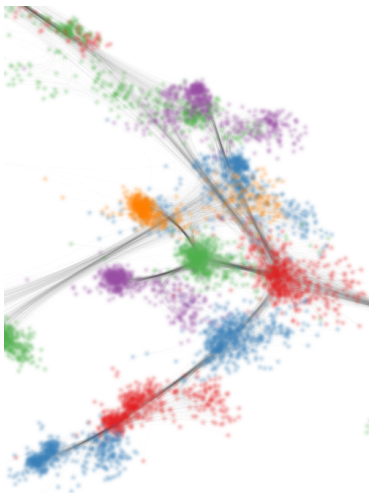
Dimensionality reduction plays an important role in visualization. In Section 2.3, we have identified two different approaches to propagate uncertainty through algorithms: Monte Carlo-based sampling and analytical solutions. The following two chapters show examples for each of those approaches in the context of dimensionality reduction (**RQ2**). This chapter presents a sampling-based technique for probabilistic graph layout and visualization, allowing visual inspection of uncertain networks and their statistical properties (**RQ1**). We aim to visualize the distribution of possible realizations of a probabilistic graph that reflects certainty and uncertainty equally well. We see this as a natural extension of previous work done on uncertainty in the context of graphs, such as knowledge engineering and visual variables.

Various approaches have been developed to visualize graphs that do not contain uncertainty [vLan+11]. However, many applications contain uncertain data due to inaccuracies, incompleteness, and inference. Typically, uncertainty is encoded as a visual variable [GHL15; Mac+12] into an exact graph layout. Although this is a valid strategy, we argue that the nature of a probabilistic graph is hard to understand by inspecting individual elements. The different realizations of the graph, together with their probabilities, also called *possible worlds*, are not encoded into the graph layout. Putting individual relations into context is no longer sufficient with an increasing number of nodes and edges, further increasing the layout's importance for comprehension. Graph mining is bound to help, but getting visual confirmation seems appropriate when dealing with statistics. Our technique reveals statistical properties by transforming probability distributions into a two-dimensional embedding using graph layout techniques. *Uncertain graph* usually means that the edge weights follow a probability distribution. However, uncertainty is a vague concept in visualization, which is why we prefer the term *probabilistic graph* over *uncertain graph*. Figure 4.1 shows our visualization for a simple tree with probabilistic edge weights. A node is not just a single point but an entire point cloud. Hence each node can occur in many regions, which reflects the uncertainty.

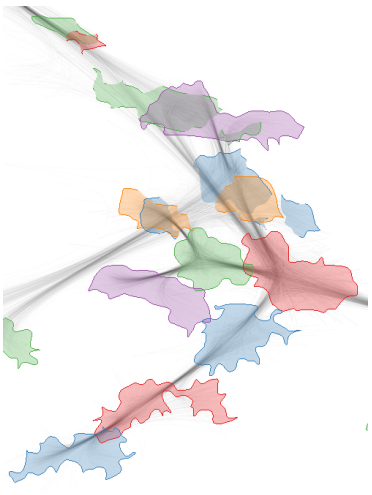
Our contribution is threefold. First, our work provides a model of probabilistic graphs. Second, we propose a formal description of the graph layout problem, along with a practical approximated solution. We present a visualization technique for probabilistic graphs that combines splatting, edge bundling, clustering, and graph coloring. At last, we demonstrate and validate our technique using representative example data sets.



(a) Tree overview



(b) Detail with splatted nodes



(c) Detail with clustered shapes

Figure 4.1: Visualization of an uncertain network. Multiple samples are drawn from a probabilistic graph model and subsequently embedded in a single layout by anchoring the sampled graphs to the expected graph. Splatting and boundary shapes show the spatial distribution of sampled nodes. We also use network topology to bundle edges.

4.1 Related Work

This work builds upon research on graph theory, graph visualization, and uncertainty visualization, which we outline in the following section. In addition to that, we will use concepts from Chapter 2.

Graph Models We use graphs to represent networks that are composed of nodes and edges. Sometimes the existence of the relationship between two nodes is unknown due to inaccuracies, incompleteness, and inference under false assumptions [Ske+08]. For example, in biology, nodes can represent proteins, while edges represent interactions between proteins. Protein–protein interactions are measured or predicted [vMer+05] — both processes are subject to uncertainty. Other examples of uncertainty in graphs are the link-prediction influence [LK07] and obfuscated identities [Bol+12] for social networks.

The term *probabilistic graph* [KF09] is not to be confused with *random graph* [ER59] — the latter is used in conjunction with generative data models. Querying and mining *uncertain graphs* has recently received considerable attention [Par+15]. These graphs differ from exact graphs in that their model expresses possible worlds instead of the actual world. Choosing good representatives that reflect the expected world can be challenging. Nevertheless, most algorithms and people work with instances of these graphs, which can be problematic [Par+15]. Our *probabilistic graphs* are one type of uncertain graphs, and our approach employs probability functions instead of scalar values, leading to a flexible model. Hence, we can represent complex probabilistic processes such as dice experiments on an edge.

Graph Theory Kobourov [Kob13] summarizes force-directed graph drawing algorithms. Brandes et al. [BP08] evaluate different types of graph distance-based drawing algorithms. Our work uses stress majorization [GKN04] and is similar to offline dynamic graph drawing [BM12]. In contrast to linking layouts sequentially over time, we align possible layouts to a reference layout using anchoring and stress majorization. We distinguish ourselves from model estimation approaches, which may seem visually similar [HRH02] since we propagate instead of estimating uncertainty.

One problem was to find a solution to the graph coloring problem because the number of nodes likely exceeds the number of visually distinctive colors like noted by Ware [War13]. We conservatively assume this number to be somewhere between 6 and 12 colors. Our solution is based on the ideas by Gansner et al. [GHK09] and the Welsh-Powell algorithm [WP67]. We provide a discussion of how our probabilistic graph coloring differs from the classic graph coloring problem and provide a heuristic solution.

Graph Visualization and Uncertainty In the following, we outline similarities and differences between graph and uncertainty visualization. We believe that graph structure and uncertainty of the underlying data are

equally important. Hence, we use these two design dimensions to arrange related work.

The field of graph visualization is broad, as indicated by a large number of surveys for different types of graph data: von Landesberger et al. [vLan+11] provide a classification of graphics according to their dependence and structure. Beck et al. [Bec+16] classify the depiction styles for dynamic graphs, which helps evaluate possibilities for the representation of uncertain graphs. Our work adapts graph splatting [VD03] and hierarchical edge bundling [Hol06]. The former conveys the distribution of nodes and edges, whereas the latter emphasizes topology and keep visual clutter at a minimum. While visually similar, we distinguish ourselves from graph bundling [HET12], which uses image processing techniques to bundle generic graphs.

The combination of graph visualization and uncertainty has recently received attention. Wang et al. [Wan+16] studied the uncertainty within graph layouts. Guo et al. [GHL15] investigated the use of visual variables to depict uncertainty of graph edges. Vehlow et al. [VRW13] identified the concept of uncertainty for fuzzy clustering of communities. Lee et al. [Lee+07] visualized structural uncertainty in hierarchies. Our work differs in that we do not try to concentrate on the local features of a graph. Instead, we transform probability distributions using graph layout techniques. We distinguish between uncertainty inherent to data (probabilities) and uncertainty introduced by our visualization technique (stress and distortion) as much as possible during the following discussion.

4.2 Graph Model

Let $G^P = (V, E, F)$ be a probabilistic graph of an uncertain network with V being a set of nodes, E a set of edges, and a set of probability density functions $F = f_{ij}$ with $ij \in E$. These density functions describe the probability that a given edge has a certain weight[1]. For the following discussion, we will assume that the edge weights are *mutually independent*. However, in general, our method does not require this independence assumption and applies to all probabilistic graphs where we can sample from the probability distributions. For numerical computation, we discretize the continuous PDF for edge ij at weight positions $a_{ij,k} \in \mathbb{R}_0^+$, where k indexes the discrete weight. Here, we restrict ourselves to non-negative weights because typical graph layout algorithms require distance metrics that are positive definite. Effectively, we replace the continuous PDFs by discrete probability mass functions (PMFs) for a set of outcomes $A = \{a_{ij,k}\}$[2]:

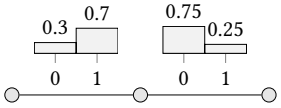
$$f_{ij} : A \to [0, 1]$$

Our model allows us to decompose a probabilistic graph into all possible weighted graphs. In Figure 4.2a, we demonstrate this concept using a simple and discrete probabilistic graph to prevent state explosion. We manage this by sampling the graph's parameter space, drawing edge weights from each PDF. Exhaustive sampling results in all possible realizations of the graph, as shown in Figure 4.2b. Since we assume independence of edge weights in the probabilistic graph, it is possible to compute the

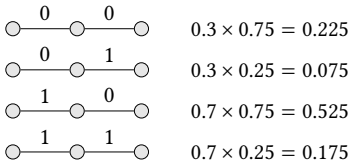
1: We introduce probability density functions in Chapter 2. There, we also detail the differences between probability density functions (PDFs) and probability mass functions (PMFs).

2: This definition is different from the one used by Zou et al. [Zou+10] where $A = \{0, 1\}$ denotes the existence of an edge, because an edge weight of 0 is handled differently than a non-existing edge by graph layout algorithms.

probability for each realization. The probabilities of all possible joint realizations sum up to 1.



(a) Definition of the probabilistic graph.



(b) Probabilities of all possible realizations assuming independence.

Figure 4.2: Decomposition of a simple probabilistic graph into all its realizations and their occurrence probabilities.

4.3 Graph Layout

In this section, we describe our conceptual and numerical approach to probabilistic graph layout. Enumerating all of the graph's realizations becomes more impractical as the number of nodes, edges, and discrete random variables increases. This exponential relationship between a growing number of dimensions and the required amount of samples to sufficiently represent the space is also referred to as *curse of dimensionality*.

The basic idea is to compute a graph layout using a Monte Carlo process by sampling and combining realizations to derive the nodes' probability distributions in a two-dimensional space. Hence, we aim at combining a representative subset of all possible realizations in a single layout. Simple stacking of exact graph layouts would result in confusing and unreadable results due to ambiguities in the procedure to generate the graph layout. Furthermore, and in contrast to linear dimension reduction techniques, such as principal component analysis (PCA), graph layout algorithms are non-linear, aggravating the problem. Therefore, if one considers graph layout as a projection from a high-dimensional space to $\mathbb{R}^2$, the projections must be made coherent. We suspect that there are multiple solutions to this graph drawing problem with different trade-offs.

Our approach combines Monte Carlo methods with dynamic graph drawing techniques: We extend a stress-based force-directed layout method to combine a set of possible realizations in a static visualization. Figure 4.3 shows the main steps of our approach:

1. Sample weighted graphs $G_1^W, \dots, G_k^W$ from G^P by sampling the edge weights independently.
2. For each sample G_i^W: compute its node positions P_i using a force-directed layout with alignment to reference node positions P_R.
3. For each node: use its positions in $P_1, \dots, P_k$ as approximation to its distribution in 2D space.

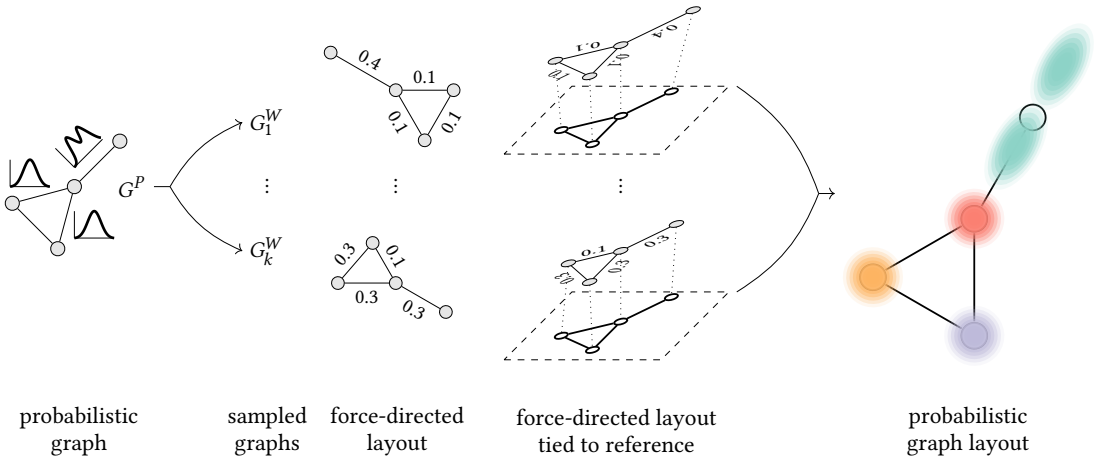


Figure 4.3: Overview of the probabilistic graph layout process.

Similar graphs laid out using a force-directed method may result in similar layouts that are rotated or flipped. The alignment with the reference ensures coherence of sampled layouts and resolves most transformation problems that arise. We now discuss the force-directed layout and the alignment.

Force-directed Layout

The state-of-the-art for drawing general undirected graphs is stress minimization [GKN04; KK89]. It is a variant of multi-dimensional scaling applied to graph-theoretic distances. This approach, in particular, outperforms spring embedder variants [BP08]. Let $G = (V, E)$ be a graph with $n = |V|$ nodes. For any pair of nodes $\{i, j\}$, with $i, j \in V$, there is an ideal distance $d_{ij} \in R^+$. The deviation of a layout $P = (p_1, \dots, p_n) \in \mathbb{R}^{n\times 2}$ from the ideal distances is quantified using the *stress function* [Kru64]

$$\text{stress}(P) = \sum_{i<j} \omega_{ij} \big(\|p_i - p_j\| - d_{ij}\big)^2$$

where the weighting is typically chosen to be $\omega_{ij} = 1/d_{ij}^2$ to better emphasize local distances and $\|\cdot\|$ denotes the Euclidean norm. For graph drawing, the weighted lengths of the shortest paths are the ideal distances. Since we have a weighted graph $G = (V, E, W)$ with $w_{ij} \in W$ denoting the strength of a link, we use the inverted weight with zero mapped to infinity for the shortest path computation. Infinite distances are replaced by a distance that is 1.5 times the maximum of all pairwise finite distances within the collection of sampled networks [BM12]. A single node connected over such an edge will be placed far away from the main graph but will not vanish entirely to infinity. As suggested by Brandes et al. [BP08], we initialize the layout with PivotMDS [BP06] and optimize by using stress majorization [GKN04] until we obtain a local minimum for the stress function.

Alignment with Anchoring

We want to combine layouts of multiple independent samples into one final visualization. Hence, we need to make sure that the layouts are not unnecessarily flipped or rotated. We achieve this by stabilizing each sample layout using anchoring [BM12] on a reference layout. In order to obtain such a reference layout, we first compute an *expected graph*, $\mathbb{E}[G]$, by fixing edge weight to its expected value $\mathbb{E}[f_{ij}]$:

$$w_{ij} = \mathrm{E}[f_{ij}] = \sum_k a_{ij,k}\, f_{ij}(a_{ij,k})$$

We then use the positions P_E from the force-directed layout of $\mathbb{E}[G]$ as the reference positions P_R for the anchoring.

Anchoring incorporates the reference layout into the stress function with control of its influence using a trade-off parameter. Given the layout P_i of a sampled graph G_i^W, the overall stress with respect to the stability

parameter α and reference layout P_R is:

$$\text{stress}(P_i; P_R, \alpha) = (1-\alpha) \cdot \text{stress}(P_i) + \alpha \cdot \sum_{v \in V} \left\| p_i^v - p_R^v \right\|^2$$

This affine combination allows us to do smooth blending between stacked stress-based layouts and the reference layout[3].

3: This formulation also allows incorporating other information such as geographic locations, for example.

4.4 Graph Visualization

Our visualization technique combines node splatting, edge splatting and bundling, graph coloring, and density-based clustering that we will discuss in the following. We base our discussion around one question: how can we convey a distribution for each node and network topology at the same time?

Node Splatting

We obtain a collection of points per node in $\mathbb{R}^2$ from our Monte Carlo based graph layout. The idea is to approximate an underlying continuous distribution or scalar field for each node by applying kernel density estimation (KDE) to the Monte Carlo samples.

Our approach uses graph splatting [VD03], a visual approximation of KDE applied to nodes. Formally, kde_n is defined by a set of samples $x_1, \ldots, x_n$, a kernel k, and a bandwidth parameter h:

$$\text{kde}_n(t) = \frac{1}{nh} \sum_{j=1}^{n} k\left(\frac{t - x_j}{h}\right) \tag{4.1}$$

The choice of kernel k is less important because bandwidth h has more influence. The only requirement is that k is a smooth function because discontinuities would become visible as edges in the final visualization. While kernels like the Epanechnikov kernel might provide slightly better results, we choose a Gaussian kernel for its simplicity. The correct choice of kernel when facing the highly non-linear projections from graph drawing remains to be clarified.

We exploit modern graphics hardware to achieve interactive frame rates. Rendering all nodes is done in a single draw-call by ray casting and blending quads, as shown in Figure 4.4. The bandwidth parameter determines the size of the quad, and each pixel evaluates to a density.

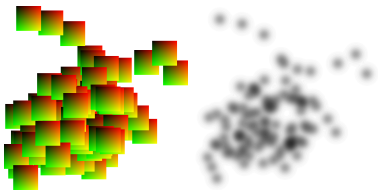
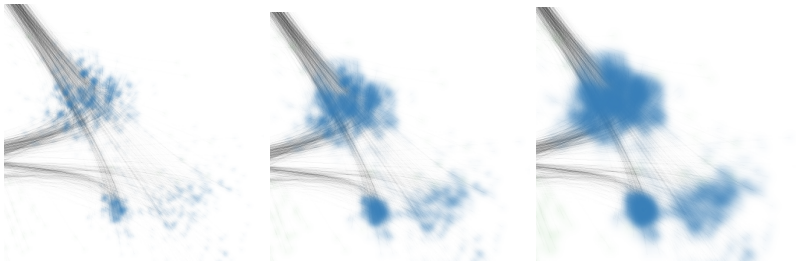


Figure 4.4: Node splatting: For each splat, a rendering primitive is computed (left). Afterward, each splat is ray-casted and blended (right).



(a) (b) (c)

Figure 4.5: Different node bandwidths, from undersmoothed (a), which reveals individual nodes, to oversmoothed (c), which emphasizes low density regions.

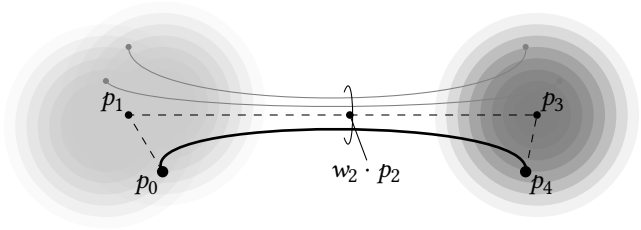


Figure 4.7: Bundled edge connecting two node samples with a quartic Bézier curve. The point p_2 is the midpoint of the segment (p_1, p_3). The bundling effect comes from sharing the control points among all edges. The weight w_2 determines the strength of the bundling.

The splats either undersmooth or oversmooth a node distribution depending on the bandwidth h, as shown in Figure 4.5. An under-smoothed KDE works well, while an over-smoothed KDE appears flat. We presume that under-smoothed KDEs work well because stippled nodes are supported perceptually by edge lines hinting to node locations [TSD09]. Neither edges lines nor node dots work without each other, which supports this presumption.

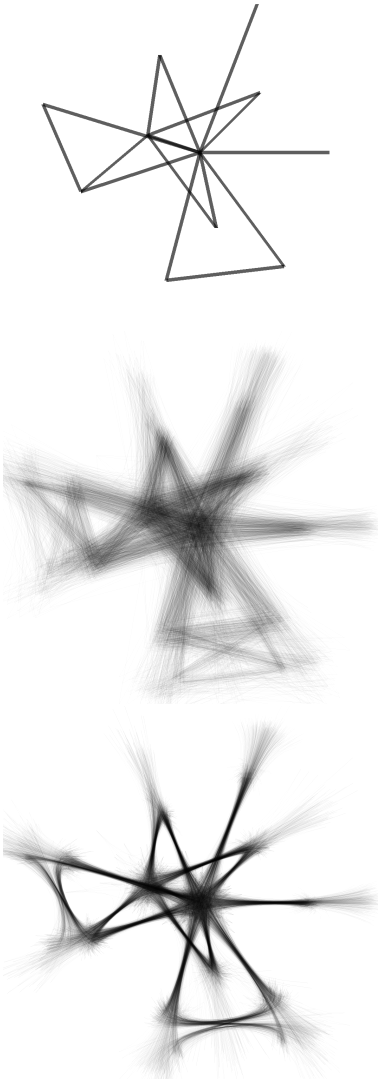


Figure 4.6: Different styles of edge splatting from top to bottom: Straight lines between centroids, straight lines between samples, and bundled lines between samples.

Edge Splatting

We investigated different techniques to depict edges between node instances. This is also shown in Figure 4.6. Drawing one straight line between nodes hampers the mental association of nodes and edges. On the other hand, straight lines cause visual clutter and conceal the topology while revealing the distribution of edges. We believe that the network structure and mental association of node and edges should be visible simultaneously. We achieve this by adopting a form of hierarchical edge bundling [Hol06], even though this conceals the distribution of edges. Presumably, the favored edge style depends on what aspect of the layout is of interest. In terms of hierarchical edge bundling, we define a set of edges between two sets of sampled nodes as one hierarchy level.

Formally, a bundled edge is defined as rational quartic Bézier curve $C(t)$ with points $p_i \in \mathbb{R}^2$ and corresponding weights $w_i > 0$:

$$C(t, p, w) = \sum_{i=0}^{4} \binom{4}{i} t^i (1-t)^{4-i} w_i p_i \tag{4.2}$$

We set p_1 and p_3 to the centroids of the clusters, while p_0 and p_4 represent the source and target positions of sampled nodes, respectively. The remaining point p_3 is set to the midpoint of the segment (p_1, p_3). An example of such a quartic curve is depicted in Figure 4.7. The bundling strength can be controlled using the weight w_2 with other weights set to one. A high bundling strength emphasizes network structure at the cost of the node distribution's visibility, whereas a low bundling strength emphasizes sample-to-sample connectivity.

The rendering of edges works similar to nodes: We splat line primitives and blend them appropriately. The main difference is that we use a thin box kernel instead of a thick Gaussian one.

Node Coloring and Labeling

Node samples spread around their reference node, which poses several challenges for labeling:

Distinctness A node's label should separate a node from other nodes. We have to resort to color labels because text labeling each sample would cause visual clutter. Ware [War13] showed that the number of visually distinctive colors is limited. Hence, we have a classical graph coloring problem.

Distribution Node samples may not be spatially close to each other, which means that there may be gaps. If two nodes get the same color while being close to each other, it can dramatically reduce the user's trust in the coloring. Therefore, we have to extend the classical graph coloring problem to respect spatial proximity.

Overlap The resulting layout heavily depends on the reference layout. Hence, nodes may overlap because node distributions expand into each other. A standard deviation greater than the expected value of the PMF pronounces this problem even further. If colors are blended with less than 180° hue distance, new colors will be generated [CWM09], which is a perceptual problem that we acknowledge and ignore for complexity reasons. After stating the problem, we solve it approximately by reducing it to well-known problems. We compute a *country graph* [GHK09] based on nodes and their samples, like illustrated in Figure 4.8. A country graph G^C consists of probabilistic nodes V^P, node samples V^W, probabilistic edges E^P, and Delaunay edges E^D [Del34] of the sampled points P:

$$G^C = (V^P \cup V^W, E^G \cup E^D) \tag{4.3}$$

We solve the resulting graph coloring problem using the greedy Welsh-Powell algorithm [WP67]. It computes the number of required colors during the assignment step, which we choose from a predefined color palette [HB03]. We also interpolate colors, if required. Note that we assume that the provided color palette is good enough in terms of color perception.

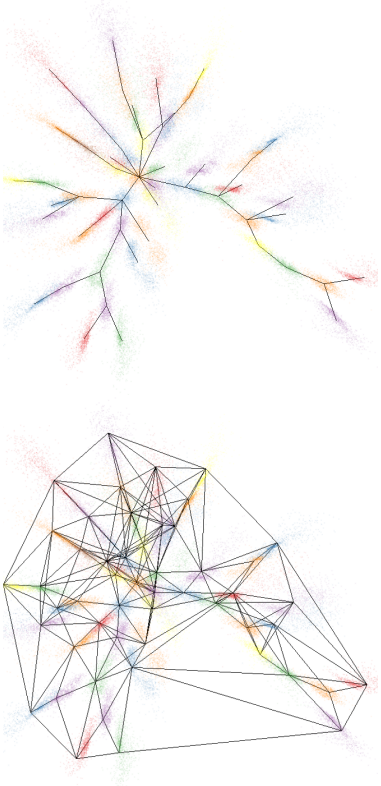
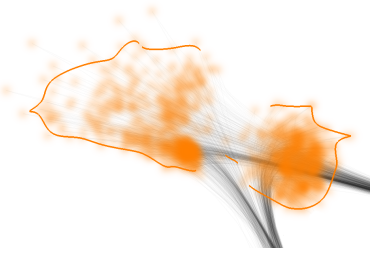


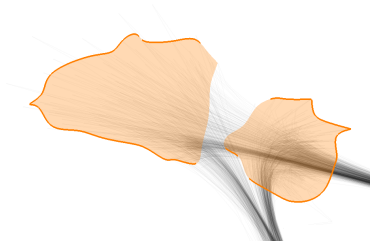
Figure 4.8: (top) Visualization of a tree network and its (bottom) proximity (or country) graph. We use the country graph to determine colors for labeling.

Clustering

We choose to implement an auxiliary approach to analyze nodes, based on the observation that node distributions can have outliers that tear nodes apart into several clusters. We do not make assumptions about the PMFs; notably, they do not have to follow a normal distribution. Arbitrary PMFs and flipping in the layout can lead to a non-coherent distribution of node positions. Because of this, identifying node clusters can be a tedious task for users, which is why we provide automatic support for this process. Clustered node positions allow us to filter and abstract node distributions, as shown in Figure 4.9. We choose contour shapes to depict

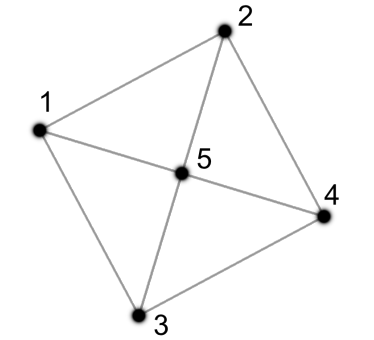


(a) contour and splats



(b) contour and concave hull

Figure 4.9: Clustering of node samples. Split clusters are visually recognized as one cluster by the half-opened contour.



(a) Layout of the expected graph.

(b) Layout of the probabilistic graph.

Figure 4.10: Probabilistic layout of a star-like graph. The corresponding distributions are shown in Figure 4.11.

clusters because they have been proven useful to group sets [CPC09]. Also, color already shows to which nodes the samples belong.

The following are the main steps of our approach. First, we cluster each set of node positions. Then, the next step is to optionally filter the clusters. Afterward, we compute the smooth concave hull of each cluster. The final step is to compute a hull contour with visibility based on cluster-to-cluster geometry.

We use DBSCAN [Est+96] for clustering because it is reasonably fast and insensitive to outliers, depending on the chosen parameters. Other clustering algorithms may also suffice. Next, we calculate the Delaunay triangulation [Del34] to compute the convex hull of each cluster, flex edges inward to achieve the desired concavity [RJB14]. Other algorithms such as α-shapes or marching squares may also work. The concave hull is then fitted with a B-spline and re-sampled to obtain a smooth shape. Last, we compute a contour to convey the togetherness of clusters by opening the line toward other clusters of the same node — a node may have many clusters, each of which has a centroid toward which we open the line. Let c_i, c_j be centroids of a node's clusters and $\tau \in [0..1]$ a visibility threshold. The hull of cluster i consists of line segments with index k. The corresponding outward-pointing unit-length normal vectors are denoted by $\hat{n}_{i,k}$. The visibility of a line segment is then defined as:

$$\bigwedge_{j \neq i} \left(\frac{c_j - c_i}{\|c_j - c_i\|} \cdot \hat{n}_{i,k} < \tau \right) \tag{4.4}$$

The combination of splatting and clustering allows us to analyze the resulting layout top-down and bottom-up simultaneously.

4.5 Results

For evaluation purposes, we use synthetic data, protein–protein interactions (PPI), and travel times by car. The goal of the synthetic examples is to better understand the technique, demonstrate its characteristics, and ultimately gain trust in the visualization technique.

Synthetic Data

Figure 4.10 shows a star-like graph that consists of five nodes and eight edges. The probability mass functions on the edges are either unimodal or bimodal (Figure 4.11, top row), and they have the same expected value. The simple weighted graph layout, shown in Figure 4.10a, does not reveal anything about how distributions might interact within the network — the graph seems uniform, even though it is not. In contrast, the distributions become apparent in our probabilistic graph layout as shown in Figure 4.10b. Grasping whether a node is connected to unimodal or bimodal distributions is easy for outer nodes: We count the clusters within a node. The center node is torn apart into three clusters, which is the result of the interaction between two bimodal distributions. Note that this aggregation is related to the notion of *node strength* for weighted graph layouts, defined as the sum of all incoming weights. While this example

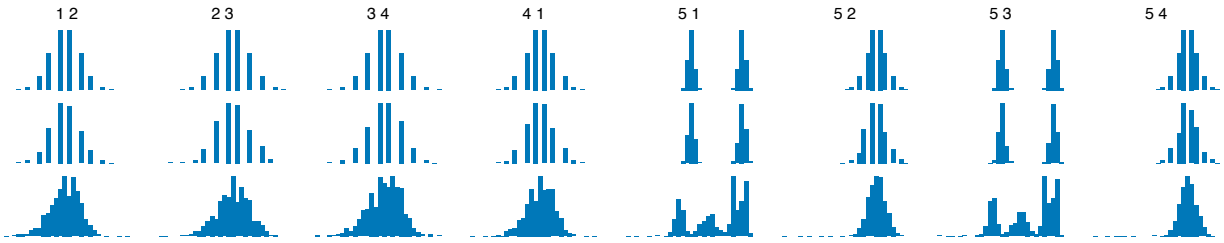
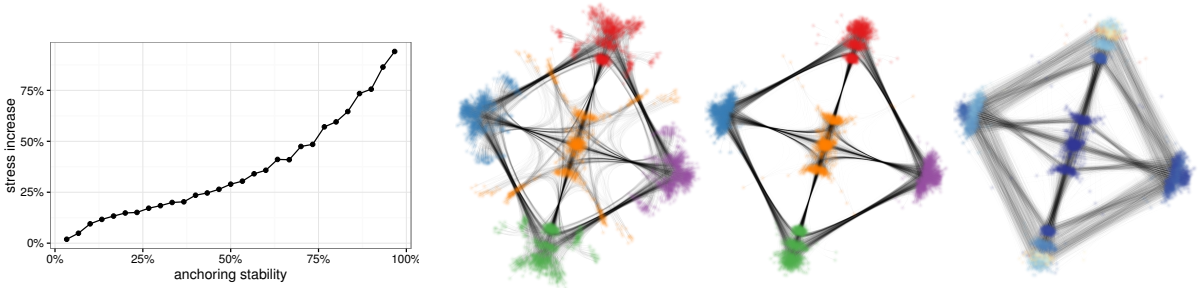


Figure 4.11: Distributions in the star-like graph. For each edge: distribution of PMF (top row), sampled edge weight (center row), and Euclidean edge length for samples (bottom row). While the different PMFs on edges are not visible in (a), they are visible in (b), as the orange node splits into three pieces: one for each possible combination of the bimodal PMF on edges $(5, 1)$ and $(5, 3)$.



(a) stress increase **(b)** $\alpha = 0.05$ **(c)** $\alpha = 0.15$ **(d)** stress at $\alpha = 0.15$

Figure 4.12: Visualization of the star-like graph for various stability values α. (a) Average increase in stress for various stability values. Up to $\alpha = 0.2$, the stress is increased by at most 15%. (b) The flipping and rotation problems of the force-directed layout result in much clutter. Anchoring on the reference layout stabilizes the resulting layout (c) at $\alpha = 0.15$. The stress of each node (d) is well distributed, from low to high stress: blue–yellow–red.

is minimalistic, it demonstrates the potential of our approach because it allows us to see if the layout still reflects the distribution of each edge. Graph drawing is a dimensionality reduction technique, which means we inherit all of its problems. Namely, for most inputs it is impossible to accurately depict all aspects of the data. We consider the PMF of an edge well represented if our layout algorithm does not distort it.

The foremost thing that affects validity is the stability value α shown in Figure 4.12. A low value results in visual clutter, whereas a very high value pins the entire distribution to the expected value. We choose our example stability value by inspecting the stress in the anchored layout S_α over the stress in the reference layout S_R. We can think of this as the relative stress S_α / S_R introduced by anchoring. The basic idea is to find a good trade-off between visual reference and additional stress (Figure 4.12a). Increased stress means that it is more likely that distributions are not well reflected in the final layout. Also, note that the discrete points get visible at low stability values (Figure 4.12b) because of the specified splat size. Retrospectively, there might be better techniques to reconstruct nonlinearly projected continuous distributions. To further quantify this distortion, we compare the PMF, sampled edge weights, and the Euclidean distance between connected nodes in Figure 4.11. The figure reveals that all distributions are well-reflected.

Another possibility to investigate the quality of the probabilistic graph layout is to inspect the stress mapped to nodes, as shown in Figure 4.12d. Stress is relative, so low does not necessarily mean good, and high does not always imply bad, but it allows us to compare two regions relative to each other. We believe that it is crucial to mix relative stress and absolute

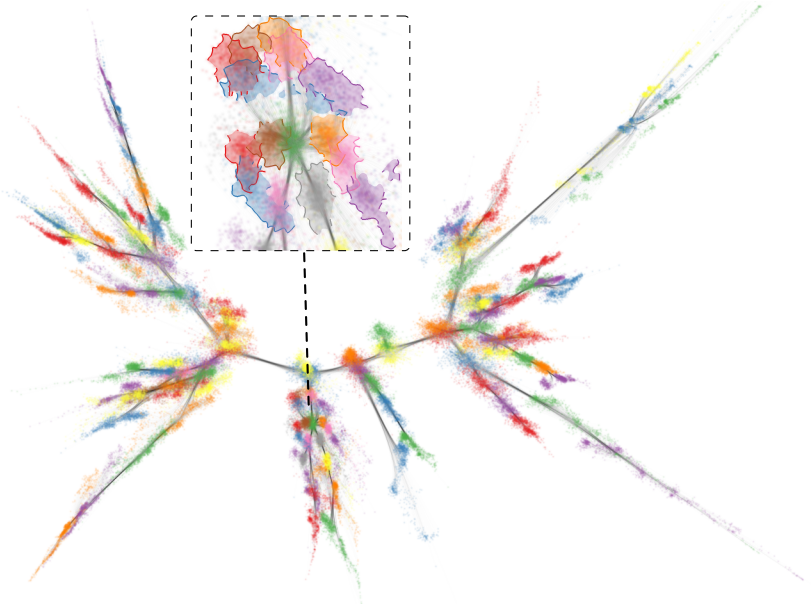


Figure 4.13: Synthetic tree. The zoomed-in view shows abstracted node clusters of competing nodes, resulting in flipping.

function distance for validation. The reduction of the distortion can be made part of the optimization by choosing the right stability value. However, we cannot eliminate it because it is inherent to graph drawing.

Another aspect worth mentioning is flipping, which occurs in more complex graphs, as shown in Figure 4.13. The first aspect to notice is the strong scattering of several leaf nodes due to the tendency toward edge weights of 0 and the high degree of freedom in graph layout, which causes the leaf nodes to "flee" from surrounding nodes. The detail view in Figure 4.13 shows a number of competing nodes. Note that node clusters (shapes) are restricted to nodes with at least two clusters. It is tricky to estimate whether nodes compete because it depends, in order of importance, on the network topology, the reference position, and the PMFs. For example, the distribution of the blue leaf node in the zoomed-in view of Figure 4.13 splits into three major clusters, which we can disambiguate by comparing the PMF to its corresponding Euclidean distance distribution.

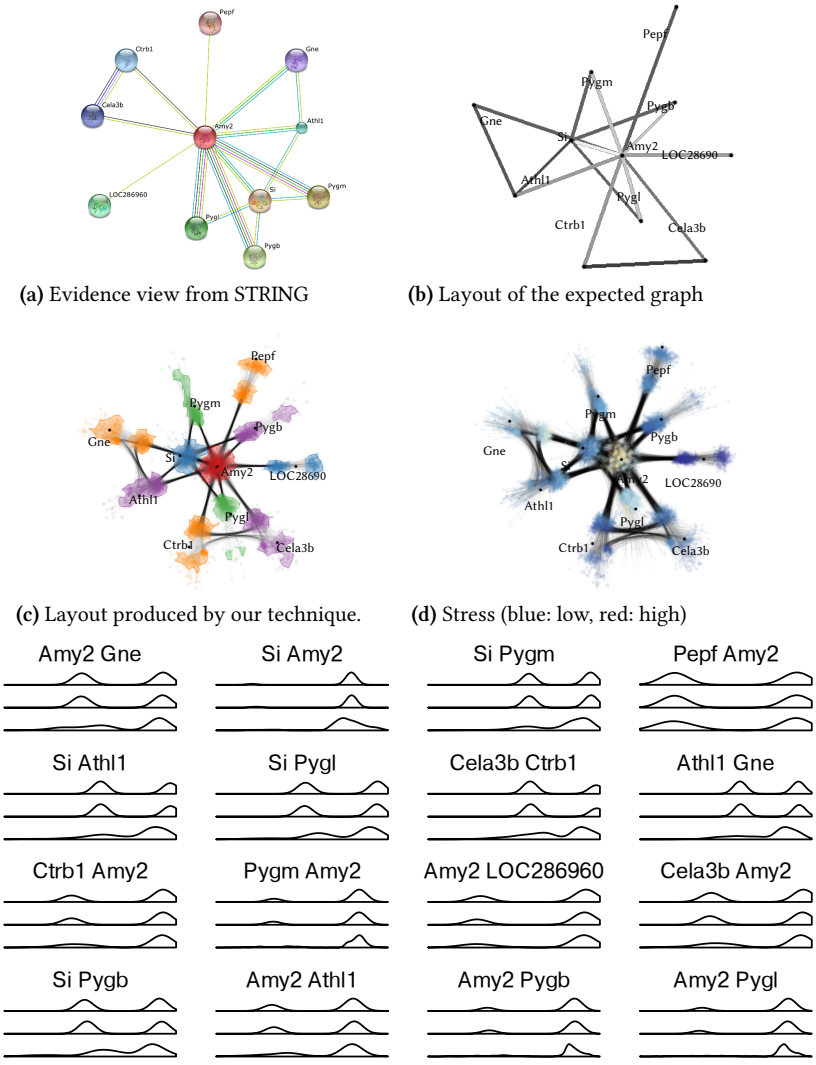
STRING Database

4: The project's website can be found at http://string-db.org/

We examined data from the *STRING database*[4], which contains known and predicted protein–protein interactions. Each interaction is associated with a score $s \in [0, 1]$ that is based on genetic, experimental, and literature data [vMer+05]. We interpret the score as PMF f for the interaction to actually be true:

$$f(w) = \begin{cases} 1 - s & \text{if } w = 0 \\ s & \text{if } w = 1 \end{cases} \tag{4.5}$$

This example is interesting for two reasons: First, two-valued distributions are at the lower end of what our technique can handle in a meaningful way. Second, it exploits zero-replacement heuristics to create separated clusters — torn nodes are less likely than coherent nodes.



(a) Evidence view from STRING

(b) Layout of the expected graph

(c) Layout produced by our technique.

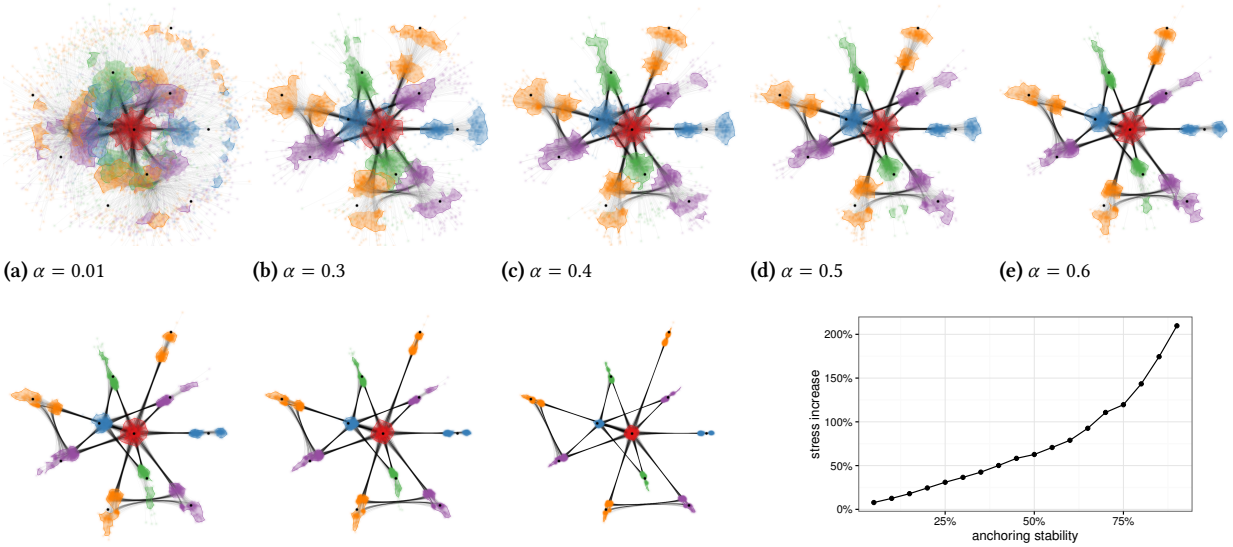
(d) Stress (blue: low, red: high)

(e) PMFs (function plots at the top),sampled edge weights (center plots),Euclidean distances (bottom plots).

Figure 4.14: Protein–protein interaction network Amy2.

Our first example is a query for *pancreatic alpha-amylase (Amy2)*. The evidence view from the STRING database website depicts the amount of information available to a set of links between the nodes (Figure 4.14a), whereas the expected view maps the score to value (Figure 4.14b). When inspecting the number of connections in the evidence view, we can tell that the number of connections does not directly match the connections in the expected graph. There is also a strong link between Amy2 and Si (Score: 0.934), which is hard to make out in the evidence view. In our visualization, these proteins are very close to each other without much deformation. The other proteins can be grouped in descending order by score $\Delta s < 0.05$: group 1 (Pygmy, Pygb, Pygl), group 2 (LOC286960, Athl1, Ctrb1), and group 3 (Cela3b, Gene, Pepf).

This order mainly corresponds to the perceived order by node coherency except for Ath1 (Figure 4.14c). The reason for this is that the network topology keeps Ath1 coherent, which we confirm by inspecting the distortion of distributions for Ath1 shown in Figure 4.14e, and stress mapped to nodes shown in Figure 4.14d. Additionally, we can confirm that this is caused by network topology by inspecting visualizations with varying



(a) $\alpha = 0.01$ **(b)** $\alpha = 0.3$ **(c)** $\alpha = 0.4$ **(d)** $\alpha = 0.5$ **(e)** $\alpha = 0.6$

(f) $\alpha = 0.7$ **(g)** $\alpha = 0.8$ **(h)** $\alpha = 0.9$ **(i)** stress increase

Figure 4.15: Visualization of Amy2 for various stability values α. The flipping and rotation problems of the force-directed layout result in much clutter (a). Anchoring on the reference layout (black dots) with, in this case, $\alpha = 0.3$ stabilizes the layout.

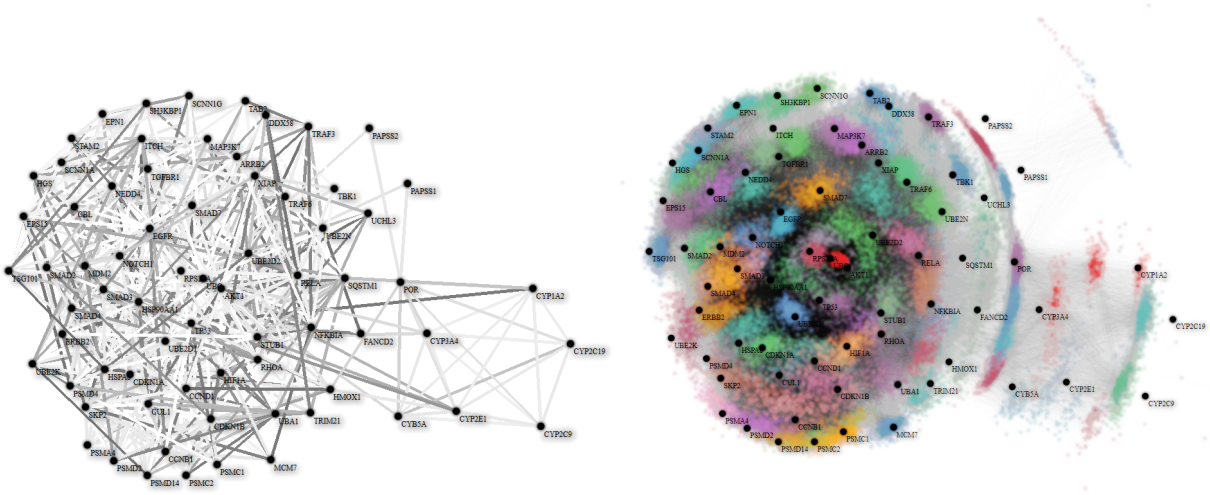
stability values shown in Figure 4.15. Note that orbiting of nodes around each other increases more quickly than nodes falling apart with decreasing stability. A small amount of destabilization of the layout helps to perceive the network topology due to orbiting. Low stability values increase the amount of visual clutter, whereas high stability values match the expected graph's layout (reference for anchoring).

Our second example is a search query for *P450 (cytochrome) oxidoreductase (POR)* with an increased search radius. Increasing the search radius inevitably creates a hairball due to the nature of protein–protein interactions. In these graphs, the considerable amount of edge crossing and coloring in the expected graph conceal most of the structure (Figure 4.16a). Our visualization is different (Figure 4.16b) because the shape of many nodes indicate an orbiting effect that is otherwise not visible.

Whether the aggregation and two-dimensional representation of simple distributions are useful to biologists is a question for future work. We strongly suspect that "piling" information from edges in nodes is a good idea because of the reduced number of visual elements (node versus node–edge–node).

Travel Times by Car

We have experimented with travel times by car between cities. This is interesting because it is geo-referenced, non-trivial real-world data. Here, we interpret cities as nodes that are connected by edges representing large roadways. We then look at the estimated time needed to travel from one city to another. This approximated duration strongly depends on the traffic predicted for this road over the time of day. For example, when traveling from city A to city B, it will make a difference if we drive this route during the rush hour or through the night when overall traffic might below.



(a) Layout of expected graph

(b) Our technique

Figure 4.16: Protein–protein interaction network POR.

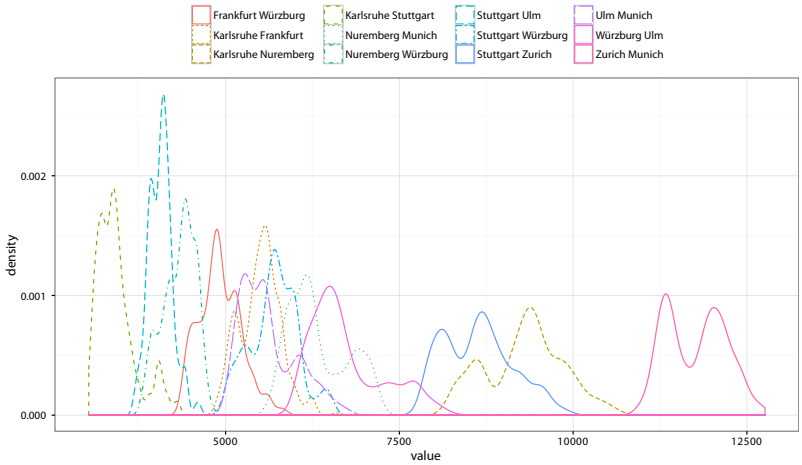
We have extracted a data set of predicted travel durations between the cities using the *Google Directions API* by sampling major roadways for connected cities. Specifically, we have queried the travel time estimates for a coherent week in time intervals of one hour. The extracted network consists of eight cities located in the south of Germany and Switzerland. To get a PMF for each connection, we first build a histogram of the predicted travel times. Normalizing this histogram gives us a discrete PMF over the durations. Note that the durations are distances between two nodes. However, we need weights for our graph model. Therefore we use inverted durations between two cities as weights. Note that the independence assumption is unlikely to be true — travel times are highly correlated — but we ignore this issue here.

Cities in our data set have fixed geographic locations. Hence, we can use each respective reference position for anchoring, as indicated in Section 4.3. As long as this influence is not exaggerated, the graph can still unfold its structure through the layout. Note that when using geographic positions instead of the expected graph, we have to ensure that the average geographic distance between cities is approximately the same as the average shortest path distance in the sampled graph.

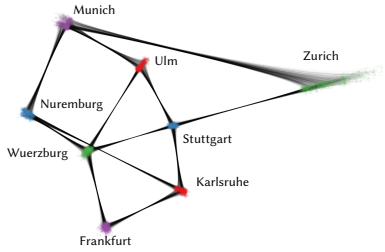
Traffic data usually implies directed edges. Our initial idea was to add virtual nodes and edges to make the graph undirected, without aggregating the travel timings between two cities. Unfortunately, the resulting layout was too confusing for visual inference on travel times. The conversion from a directed graph to an undirected graph by introducing virtual nodes and edges biases the layout.

Therefore, we had to simplify by pivoting on one city — filtering all edges that would lead to the pivot city. The resulting graph is non-starlike and allows us to visually infer travel times from the perspective of one city. Figure 4.17 shows the sampled and filtered PMFs, the graph anchored to the expected graph (like before), and the graph anchored to the geolocations of the cities (experimental). The differences between the expected and geo-referenced layout are small because anchoring with low stability values is more about directions, not about exact reference points.

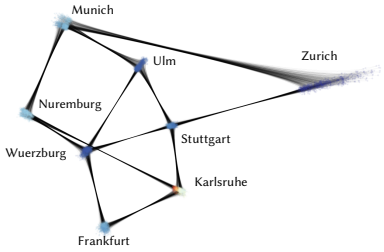
Hence, geo-referencing seems like a valid modification of our technique. Furthermore, the inner nodes depict a preferable direction, roughly reflecting travel times of the entire network, even though Zurich is exaggerated compared to Stuttgart or Karlsruhe.



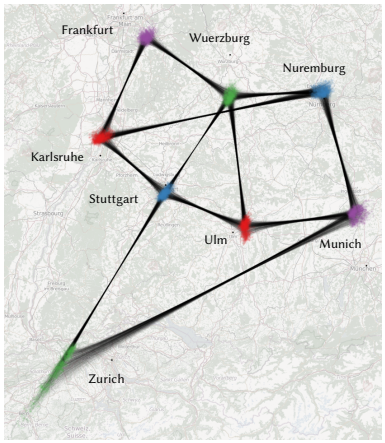
(a) PMFs for travel times between cities



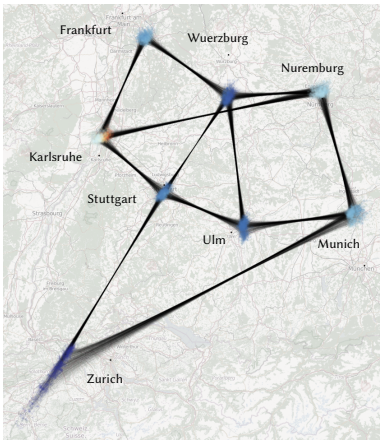
(b) $\alpha = 0.2$



(c) $\alpha = 0.2$



(d) $\alpha = 0.2$, geo-referenced



(e) $\alpha = 0.2$, geo-referenced

Figure 4.17: (a) Travel duration by car in 8 major cities in southern Germany and Switzerland, pivoted to Karlsruhe. The probabilistic graph anchored to the expected layout (b, c) and geo-locations (d, e). Note that the choice of anchor mainly affects rotation.

4.6 Limitations

We summarize the limitations of our approach and discuss issues to address in future work. While our approach benefits from the generality of force-directed methods, it also inherits some of their weaknesses. As we have shown in the previous examples, it is important to consider the

ambiguities that result from the interplay between the uncertainty in the data and the layout algorithm.

Layout

In the force-directed layout approach, nodes can obtain completely different positions through small structural changes. Different local optima may result in reflected or rotated positions for parts of the graph. While this effect can also happen in our approach, it happens less frequently due to the alignment with the reference layout.

As for the force-directed layout, it is often unclear whether the position and shape of a point cloud are due to the edge weights or due to the graph structure. Here, we see the need to incorporate more quantitative measures that would allow us to convey better the reason why a node is in this position. A simple way to do that would be to add noise on top of a single node's incident edges to see which influence this change has on its position. If the layout of a single graph sample does not convey the graph structure — for example, if it looks like a hairball — then our approach is not likely to work either. Nevertheless, techniques [NOB15] for untangling such complicated structures are likely to work for our approach as well.

Scalability

Our implementation of the force-directed method needs $\mathcal{O}(n{\cdot}(n{+}m){+}n^2{\cdot}r)$, where n is the number of nodes, m the number of edges, and r the number of iterations for stress majorization. While this only scales to graphs of medium size, other more scalable force-directed methods could be used as well [Hu05]. The runtime to determine cluster regions for one node is on average $\mathcal{O}(k \log k)$, where k is the number of sampled graphs.

The more problematic limitation is in the visual distinction of the different node regions. The set of distinguishable colors is quickly exhausted, especially if the node regions are not connected. Our half-opened shapes (node clusters) allow us to search and select a limited number of different regions on top of splatting. Here, we see the need for better depiction techniques (set and distribution). In addition to that, tracing edges is hard, and mapping to visual variables other than color is a big challenge. Note that the edge bundles reflect the connection between identical nodes in the graph. Thus, visual identification of single edges in such a bundle is typically not required unless distribution and outliers of edge lines are of interest.

4.7 Conclusion

We have presented a novel approach for uncertain network visualization that maps probability distributions of edges to visually perceivable splats and shapes. In this context, we have explored many aspects of filtering and depicting a set of overlapping clusters of points without obstructing the underlying network topology.

We have applied our approach to several data sets to better understand its characteristics: Synthetic data, protein-protein interaction data, and travel time data with geographic reference. The results are sometimes complicated to interpret because nodes alternate between positions. On the other hand, our method allows us to assess problems of the underlying force-directed layout. Although not developed for this purpose, we think that our visualization approach would help analyze the impact of graph layout techniques.

Other graph layouts, colorings, and visualization methods need to be developed that consider point clouds' full spatial distribution. After examining the influence of our layout and visualization technique on various data sets, it remains to investigate its effectiveness and efficiency with controlled user experiments.

Uncertainty-aware Principal Component Analysis

5

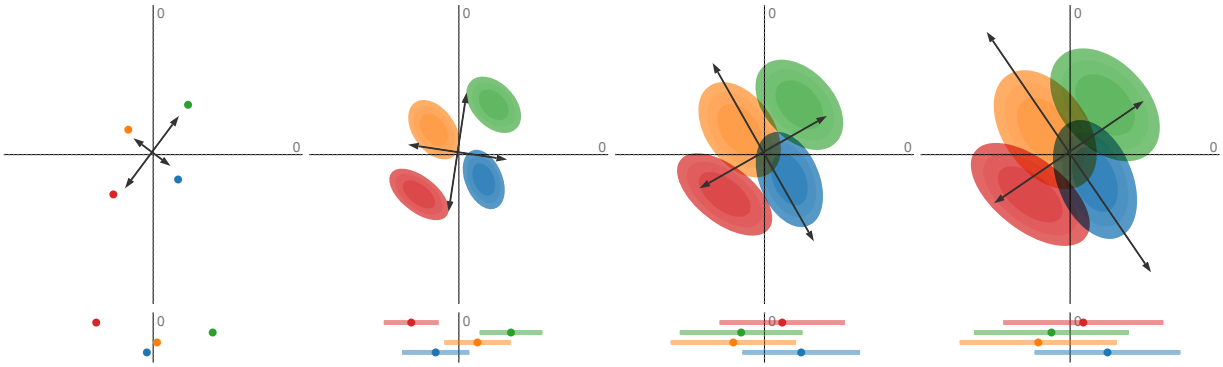
In the previous chapter, we showed how to use a sampling-based approach for dimensionality reduction under uncertainty. The following chapter takes a different approach: It is possible to propagate the uncertainty through the dimensionality reduction method analytically. Often this is the case when dealing with uncertainties that follow a normal distribution.

Dimensionality reduction techniques can be applied to visualize data with more than two or three dimensions, projecting the data to a lower-dimensional subspace. These projections should be meaningful so that the data's characteristics in the high-dimensional space carry over to the low-dimensional representation. In general, dimensionality reduction techniques can either be linear or non-linear [NA18]. Linear dimensionality reduction has the advantage that the input data's properties and invariants are still reflected in the resulting projections. Another advantage of linear over non-linear methods is that they are easier to reason about because the projection's subspace is always a linear combination of the original axes. Also, linear methods usually have efficient implementations [CG15].

Arguably the most frequently used linear method is principal component analysis (PCA). It is most effective if the input data dimensions are correlated, which is commonly the case. PCA uses this property and finds the directions of the data that contain the largest variance by performing eigenvalue decomposition of the sample covariance matrix estimated from the input. The input to conventional PCA is a set of points. However, we often encounter data afflicted with uncertainty or variability. In Chapter 2, we discussed different sources of uncertainty and how to model uncertainty mathematically using probability distributions. In this chapter, we derive a generalization of PCA that directly works on this kind of uncertain data. Like regular PCA, our new method of *uncertainty-aware PCA* solely requires that the expected value and the covariance between dimensions of these distributions can be determined — no higher-order statistics are taken into account. This uncertainty in the input data can substantially impact the resulting projection because it directly influences the magnitude of the eigenvalues of the sample covariance matrix.

In addition to extending PCA, we introduce *factor traces* as a visualization that shows how the projections of the original axes onto the subspace change under varying degrees of uncertainty. This allows us to perform a sensitivity analysis of the dimensionality reduction regarding uncertainty and helps better understand the resulting linear projection. The work of this chapter has four main contributions:

- a closed-form generalization of PCA for uncertain data,
- sensitivity analysis of PCA with regards to uncertainty in the data,
- factor traces as a new visualization technique for the sensitivity of linear projections, and
- establishing a distance metric between principal components.



(a) Regular PCA **(b)** $s = 0.5$ **(c)** $s = 0.8$ **(d)** $s = 1.0$

Figure 5.1: Data uncertainty can have a significant influence on the outcome of dimensionality reduction techniques. We propose a generalization of principal component analysis (PCA) that considers the uncertainty in the input. The top row shows the dataset with varying degrees of uncertainty and the corresponding principal components. In contrast, the bottom row shows the dataset's projection onto the first principal component, using our method. In Figures (a) and (b), with relatively low uncertainty, the blue and the orange distributions are comprised by the red and the green distributions. In Figures (c) and (d), with increasing uncertainty, the projection changes drastically: now the orange and blue distributions encompass the red and the green distributions.

In Figure 5.1, we compare our method to regular PCA and illustrate why it is crucial to consider the uncertainty in the data when determining the projection: It shows a projection of four bivariate probability distributions, each with a varying degree of uncertainty, projected onto a single dimension. When the uncertainty is low, the red and green data points define the extent of the projected data. With increasing uncertainty, the projection looks quite different: Now, the orange and blue data points mark the extent of the projected data. This change in projection follows directly from the shape of the distributions and shows that it is essential to incorporate the uncertainty information adequately into our dimensionality reduction algorithms. Although all distributions in this example are Gaussian, our method works on general probability distributions with expectation and covariance.

5.1 Related Work

The survey by Nonato et al. [NA18] offers a broad overview of dimensionality reduction from a visualization perspective. Principal component analysis [Pea01] is one of the oldest and most popular techniques. It is often applied to reduce data complexity, which is a common task in visualization. By construction, PCA yields the linear projection that retains the most variance of the input data in the lower-dimensional subspace. Probabilistic PCA [TB99] extends traditional PCA by adding a probabilistic distribution model. In contrast to our method, it assumes an unknown isometric measurement error. Likewise, many extensions have been introduced to PCA [Bur09; KC04]. For example, kernel PCA [SSM97] enables non-linear projections by first transforming objects into a higher-dimensional space where a good linear projection can be found. Techniques such as Bayesian PCA [Bis99; NSB11; Nou+02], and the method introduced by Sanguinetti et al. [San+05] focus on estimating the dimensionality of the lower-dimensional space. Robust PCA methods [AO03; TG08; VCB18] target datasets with outliers. Different extensions to PCA have also been developed in the context of fuzzy systems. The technique

described by Denoeux et al. [DM04] applies PCA to fuzzy numbers by training an artificial neural network that incorporates the different possible realizations for each fuzzy number. Giordani et al. [GK06] provide an overview of methods that apply PCA to interval data. In contrast, we extend traditional PCA to an uncertainty-aware linear technique for exploratory visualization that works on general probability distributions.

Besides regular PCA, factor analysis [Spe04] is another well-known linear method. Its goal is to identify latent variables underlying a higher-dimensional space of measurements[1]. Factor analysis models measurement errors, yet constraining the errors to be uncorrelated is common. One reason for this is that modeling correlated errors can be problematic if the actual errors are unknown [Her15]. In our description, we assume that all errors are known or can at least be estimated. Many other linear techniques such as classical multi-dimensional scaling [Tor52] and independent component analysis [HKO01] are covered by Cunningham et al. [CG15]. None of them can deal with data that has explicitly encoded measurement errors to the best of our knowledge.

1: In contrast to PCA, these variables are not necessarily orthogonal.

Liu et al. [Liu+17] provide an overview of the visualization and exploration of high-dimensional data. The star coordinates [Kan00] visualization technique, for example, provides interactive linear projections of high-dimensional data. Recently measure-driven approaches for exploration have gained interest, for instance by Liu et al. [Liu+16] as well as by Lehmann et al. [LT16]. Visualizing the projection matrix of linear dimensionality reduction techniques (instead of projections of the data) can be done with factor maps or Hinton diagrams [Bis99; HS91].

Advances in visualizing uncertainty and errors often originate from the need to represent prediction results [SPS11]. More generally, visualizing Gaussian distributions by a set of isolines is a common practice. In this thesis, we aim at bringing uncertainty-aware dimensionality reduction and visualization together. For example, our technique could extend the approach of Wang et al. [Wan+17] for visualizing large datasets by allowing a fast approximate visualization of clusters. Furthermore, correlated probability distributions are often the result of Bayesian inference, which is widely used in prediction tasks, where the result is always a probability distribution. In this domain, Gaussian processes [RW05] are a prime example of correlated uncertainty.

In Chapter 4, we already proposed a projection for uncertainty networks based on sampling different realizations of the data and investigate potential effects of uncertainty. Lately, there has been a push in the visualization community to understand the intrinsic properties of projection methods better. However, the focus has mainly been on exploring non-linear approaches. With *DimReader*, Faust et al. [FGS19] address the problem of explaining non-linear projections. Their technique uses automatic differentiation to derive a scalar field that encodes the sensitivity of the projected points against perturbations. Wattenberg et al. [WVJ16] examine how the choice of parameters affects the projection results of t-SNE. Similarly, Streeb et al. [Str+18] compare a sample of (non-) linear techniques and influences of their parameters on projections.

5.2 Method

In this part, we describe the necessary adaptions to the PCA framework required to handle uncertainty. For this, we use the definition of random vectors and multivariate normal distributions from Chapter 2. First, we show how to compute the covariance matrix for probability distributions, which is a fundamental part of our technique. Then, we describe how this fits into the context of regular PCA. Afterward, we demonstrate how our method allows us to perform PCA analytically on uncertain data, using multivariate normal distributions as an example. Finally, we show that our approach is a generalization of regular PCA, which allows us to combine certain and uncertain data within the same mathematical framework. It also forms the foundation for sensitivity analysis, as described in Section 5.3.

PCA finds the directions in the data with the largest variance by looking at the covariance of the input. We adopt this concept to arbitrary distributions to handle uncertain data and only require that the expected value and covariance can be determined for all distributions. It is important to note that this does not imply that the input distributions necessarily have to follow a Gaussian distribution[2]. In Section 5.4, we show an example of such a dataset, and in Section 5.6, we discuss its implications on the resulting projection.

2: This can be illustrated with a small example: Consider an input distribution made up of two clusters spread about its mean. Then, the covariance of the distribution still captures the spread of the data alongside the cluster centers. Even though mean and covariance underdetermine the distribution, they still adequately represent its overall shape.

Before performing PCA, we need to establish a relationship between the units of the original axes. Our method requires the same preprocessing step. The usual approach for this is to normalize the input data accordingly, which can be performed for probability distributions using affine transformations, as outlined in Chapter 2.

Uncertain Covariance Matrices

As we have mentioned before, our goal is to perform PCA on a set of N probability distributions that represent the uncertainty. Formally, we represent this collection of distributions as random vectors $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_N\}$. For each of these random vectors, we require that we can determine its expected value $\mathbb{E}[\mathbf{t}_i]$ and its pairwise covariance $\mathrm{Cov}(\mathbf{t}_i, \mathbf{t}_i)$. It is important to note that $\mathbf{T}$ can conceptually be interpreted as a second-order random vector because its components $\mathbf{t}_n$ are random vectors themselves.

Our approach adapts the computation of the covariance matrix to account for uncertainty in the data. Regular PCA works on a set of points. In this setting, the covariance matrix represents the expected products of deviations of these points from the sample mean. In contrast, our approach works on a set of random vectors, which changes the problem in the following way: The actual deviation of each random vector from the overall sample mean is unknown due to the uncertainty. Nevertheless, we can determine the expected deviation for each of the distributions. We do this conceptually by integrating over the deviation of all possible realizations of each probability distribution. The framework of PCA only considers the first- and second-order moments, so it turns out we do not even have to evaluate this integral: It is possible to derive the covariance matrix directly from the summary statistics.

From Equation 2.1, we can derive a property of the covariance matrix that we will need later on: it gives us a way to compute the expected outer product $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right]$ of a particular random vector with itself. We achieve this by solving Equation 2.1 for $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right]$:

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^\mathsf{T}\right] = \mathbb{E}\left[\mathbf{x}\right]\mathbb{E}\left[\mathbf{x}\right]^\mathsf{T} + \mathrm{Cov}(\mathbf{x}, \mathbf{x}) \tag{5.1}$$

We use the following equation for distributions, which is akin to computing the expected products of *expected* deviations. To avoid confusion with the expected value of each random vector $\mathbb{E}\left[\cdot\right]$, we denote the expectation operator that stems from the covariance method with $\hat{\mathbb{E}}\left[\cdot\right]$:

$$\mathrm{Cov}(\mathbf{T}, \mathbf{T}) = \hat{\mathbb{E}}\left[\mathbb{E}\left[\mathbf{T}\mathbf{T}^\mathsf{T}\right] - \mu_\mathbf{T}\mu_\mathbf{T}^\mathsf{T}\right]$$

We can expand this further by making use of Equation 5.1:

$$\begin{aligned}\mathrm{Cov}(\mathbf{T}, \mathbf{T}) &= \hat{\mathbb{E}}\left[\mathbb{E}\left[\mathbf{T}\right]\mathbb{E}\left[\mathbf{T}\right]^\mathsf{T} + \mathrm{Cov}(\mathbf{T}, \mathbf{T}) - \mu_\mathbf{T}\mu_\mathbf{T}^\mathsf{T}\right] \\ &= \hat{\mathbb{E}}\left[\mathbb{E}\left[\mathbf{T}\right]\mathbb{E}\left[\mathbf{T}\right]^\mathsf{T}\right] + \hat{\mathbb{E}}\left[\mathrm{Cov}(\mathbf{T}, \mathbf{T})\right] - \mu_\mathbf{T}\mu_\mathbf{T}^\mathsf{T}\end{aligned} \tag{5.2}$$

The terms in Equation 5.2 have particular interpretations. Algorithm 3 provides the corresponding pseudocode for Equation 5.2. First, we recognize that the term $\hat{\mathbb{E}}\left[\mathbb{E}\left[\mathbf{T}\right]\mathbb{E}\left[\mathbf{T}\right]^\mathsf{T}\right]$ is the same as performing regular PCA on the means of each of the distributions. The second term $\hat{\mathbb{E}}\left[\mathrm{Cov}(\mathbf{T}, \mathbf{T})\right]$ computes the average covariance matrix over all random vectors:

$$\hat{\mathbb{E}}\left[\mathrm{Cov}(\mathbf{T}, \mathbf{T})\right] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{Cov}(\mathbf{t}_i, \mathbf{t}_i) \tag{5.3}$$

It reflects the uncertainty that each random vector has and how these uncertainties influence the overall covariance in the dataset — it is also the major difference between our method and regular PCA, which cannot handle probability distributions. The last term is called centering matrix and also part of regular PCA. It consists of the outer product of the empirical mean $\mu_\mathbf{T}$ of our dataset. The empirical mean of our dataset can be computed as follows:

$$\mu_\mathbf{T} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\mathbf{t}_i\right]$$

Algorithm 3: Covariance matrix of random vectors

Input : Set of d-variate distributions $\mathbf{T}$, scaling factor $s = 1$
Output: Covariance matrix $K_{\mathbf{TT}}$

$\mu_\mathbf{t} \leftarrow d$-dimensional vector initialized to 0
foreach $\mathbf{t} \in \mathbf{T}$ **do**
  $\mu_\mathbf{t}$ += $\mathbf{t}$.*mean*()
$\mu_\mathbf{t}$ /= $\mathbf{T}$.*length*()
$K_{\mathbf{TT}} \leftarrow d \times d$ matrix initialized to 0
foreach $\mathbf{t} \in \mathbf{T}$ **do**
  $\boldsymbol{m} \leftarrow \mathbf{t}$.*mean*()
  $K_{\mathbf{TT}}$ += $\boldsymbol{m}\boldsymbol{m}^\mathsf{T} + s^2 \cdot \mathbf{t}.cov() - \mathbf{t}$
$K_{\mathbf{TT}}$ /= $\mathbf{T}$.*length*()
return $K_{\mathbf{TT}}$

We now show that our method, provided by Equation 5.2, yields a covariance matrix by looking at the different terms of this equation. A matrix K is positive semi-definite if $\boldsymbol{u}^\mathsf{T} K \boldsymbol{u} \geq 0$, for every non-zero vector $\boldsymbol{x}$.

Theorem 5.2.1 *The outer product $\boldsymbol{x}\boldsymbol{x}^\mathsf{T} \in \mathbb{R}^{d\times d}$ of a vector $\boldsymbol{x} \in \mathbb{R}^d$ with itself always results in a symmetric, positive semi-definite matrix.*

Proof. Let $\boldsymbol{u} \in \mathbb{R}^d$ be a nonzero vector. Using the definition of positive semi-definiteness from above,

$$\boldsymbol{u}^\mathsf{T}(\boldsymbol{x}\boldsymbol{x}^\mathsf{T})\boldsymbol{u} = (\boldsymbol{x}^\mathsf{T}\boldsymbol{u})^2 \geq 0.$$

The symmetry follows from the definition of matrix multiplication. □

Our method differs from regular PCA in one term, which we can see in Equation 5.3: In essence, this term computes the arithmetic mean of the covariance matrices $\mathrm{Cov}(\mathbf{t}_i, \mathbf{t}_i)$ of each distribution $\mathbf{t}_i$. A matrix is a covariance matrix if and only if it is *symmetric* and *positive semi-definite*. By definition, $\mathrm{Cov}(\mathbf{t}_i, \mathbf{t}_i)$ always satisfies this property.

Theorem 5.2.2 *Let $\boldsymbol{K} = \{K_1, \ldots, K_N\}$ with $K_n \in \mathbb{R}^{d\times d}$ be a set of covariance matrices, then the arithmetic mean of this set $\frac{1}{N}\sum_{n=1}^{N} K_n$ is a covariance matrix.*

Proof. Let $\boldsymbol{u} \in \mathbb{R}^d$ be a nonzero vector and $A, B \in \mathbb{R}^{d\times d}$ positive semi-definite matrices. Both addition $A + B$, and multiplication with a scalar $kA, k \geq 0$ result in positive semi-definite matrices:

$$\boldsymbol{u}^\mathsf{T}(A + B)\boldsymbol{u} = \boldsymbol{u}^\mathsf{T} A\boldsymbol{u} + \boldsymbol{u}^\mathsf{T} B\boldsymbol{u}$$
$$\boldsymbol{u}^\mathsf{T}(kA)\boldsymbol{u} = k(\boldsymbol{u}^\mathsf{T} A\boldsymbol{u})$$

Because of this and the properties of symmetric matrices, it follows that the arithmetic mean of $\boldsymbol{K}$ is a symmetric and positive semi-definite matrix and therefore also a covariance matrix. □

PCA Framework and Diagonalization

Now that we have constructed the covariance matrix while respecting the uncertainty, we can look at the remaining steps of PCA. After setting up the covariance matrix, we retrieve its eigenvalues λ_d and corresponding eigenvectors $\boldsymbol{v}_d$. We use *eigenvalue decomposition* to do this:

$$\mathrm{Cov}(\mathbf{T}, \mathbf{T})\boldsymbol{v} = \lambda\boldsymbol{v}$$

Let q be the desired number of dimensions for the low-dimensional space. We then choose the q largest $\boldsymbol{v}_d$ by their corresponding eigenvalue λ_d, yielding q principal components $\boldsymbol{W} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_q\}$. We can then project each distribution onto the subspace $\langle \boldsymbol{W} \rangle$ that is spanned by these principal components $\Phi(\mathbf{t}_n) \in \langle \boldsymbol{W} \rangle$, where $\Phi(\cdot)$ is a linear projection that can be described using a linear transformation.

It is important to note that eigenvalues and eigenvectors have specific characteristics that complicate their analysis. The orientation of $\boldsymbol{v}_d$ is not completely defined, therefore $\boldsymbol{v}_d \hat{=} -\boldsymbol{v}_d$. In practice, $(\lambda_d, \boldsymbol{v}_d)$ are computed

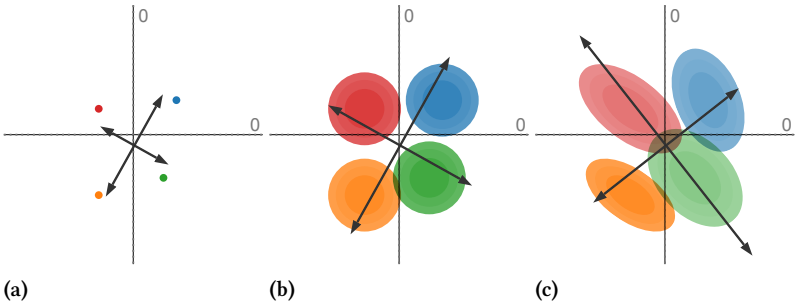


Figure 5.3: PCA on different types of input data: (a) Regular PCA without uncertainty. (b) Isometric error model as used by previous work that describes PCA as an optimization problem; the directions of the principal components are the same, but the lengths differ. (c) Our method works on arbitrary distributions. Here it results in drastically different principal components.

numerically, which can lead to instabilities and rounding errors. We will discuss the impact of these characteristics on the analysis of linear projections in Section 5.3.

Linear Transformation of MVN Distributions

Now that we have defined the projection $\Phi(\cdot)$, we need to transform each distribution into the subspace $\langle \boldsymbol{W} \rangle$. In the following, we will describe how this can be carried out for multivariate normal distributions, as they are often used to model errors or uncertainty in the data. As mentioned in Section 5.1, several existing techniques already model uncertainty using MVN distributions. They usually describe the distributions using an error model, which means that a measurement $\boldsymbol{x}$ is disturbed according to an error term $\boldsymbol{\epsilon}$:

$$\mathbf{t}_n = \boldsymbol{x}_n + \boldsymbol{\epsilon}_n, \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \Psi_n)$$

Sometimes it is easier to retrace our derivation of the covariance matrix by thinking of this error in terms of a single random vector $\mathbf{t}_n$ that can be equivalently defined as follows:

$$\mathbf{t}_n \sim \mathcal{N}(\boldsymbol{x}_n, \Psi_n).$$

Figure 5.3 shows examples of different error models that can be created depending on the shape of Ψ_n. We also visualize the corresponding principal components of the dataset, determined by using our method.

The dimensionality of $\Phi(\mathbf{t})$ is $\dim(\langle \boldsymbol{W} \rangle)$. To perform the actual projection, we assume that $\boldsymbol{w}_q$ are unit vectors, and write them in a column matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{w}_1 & \dots & \boldsymbol{w}_q \end{bmatrix}$$

It is important to note that a projection Φ is an affine transformation, as defined in Chapter 2. Accordingly, we can project a normal distribution as follows:

$$\Phi(\mathbf{t}_n) = \mathcal{N}(\boldsymbol{A}^\mathsf{T}\boldsymbol{\mu}_n,\ \boldsymbol{A}^\mathsf{T}\Psi_n\boldsymbol{A})$$

The resulting distribution remains multivariate normally distributed. Figure 5.2 shows an example of this for 2D.

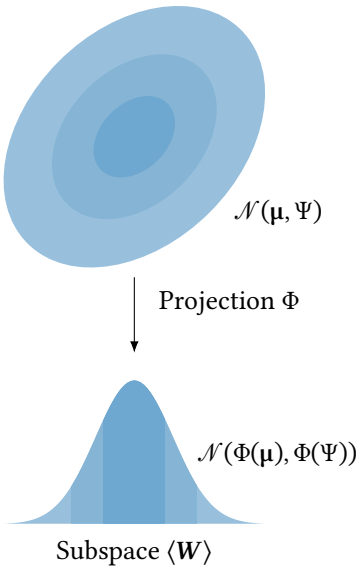
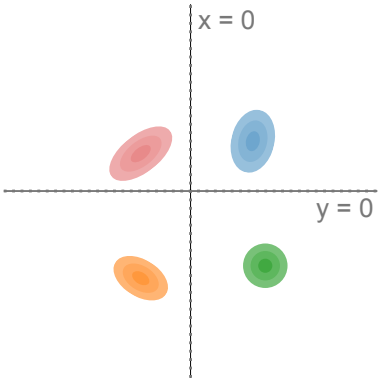


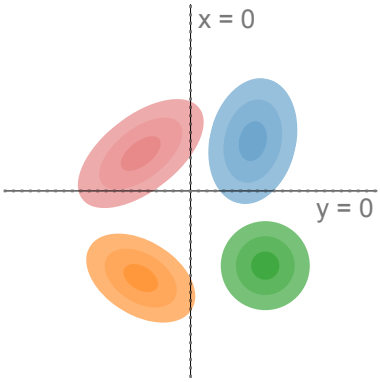
Figure 5.2: A linear projection $\Phi(\cdot)$ of a normal distribution $\mathbf{t} \sim \mathcal{N}(\boldsymbol{\mu}, \Psi)$ results in a modified multivariate normal distribution $\Phi(\mathbf{t})$ in the lower dimensional subspace $\langle \boldsymbol{W} \rangle$. Because of this, we can propagate the uncertainty directly through linear dimensionality reduction techniques.

Reduction to Regular PCA

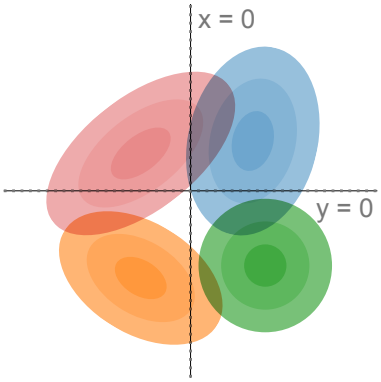
In this section, we will show that our method is a mathematical generalization of conventional PCA. The main difference between the two algo-



$s = 0.25$



$s = 0.50$



$s = 0.75$

Figure 5.4: Different levels of uncertainty can be achieved by scaling the covariances of each distribution with a factor s. By letting $s \to 0$ we can emulate traditional principal component analysis, as the distributions converge toward single points.

rithms lies in the covariance matrix setup, as described by Equation 5.2. Our method includes an additional term that reflects each input's uncertainty (Equation 5.3). To reduce our formulation to regular PCA, we will scale each distribution's covariance by a constant factor s, which decreases its spread, implicitly reducing the amount of uncertainty within each distribution.

To scale the covariance matrices, we will again make use of the properties of affine transformations for covariance matrices, as discussed in Chapter 2. Let S be a scale matrix that has the form $S = \text{diag}(s)$. We can now use Equation 2.2 to scale $K_{\mathbf{TT}} = \hat{\mathbb{E}}\left[\text{Cov}(\mathbf{T}, \mathbf{T})\right]$:

$$S\left(K_{\mathbf{TT}}\right) S^{\mathsf{T}}$$

In practice, we can make use of the fact that a scale matrix S is always a diagonal matrix. In our case, each diagonal entry is equal to s, therefore $S = \text{diag}(s)$, which allows us to simplify the equation above even further:

$$S\left(K_{\mathbf{TT}}\right) S^{\mathsf{T}} = s^2 \cdot K_{\mathbf{TT}}$$

Figure 5.4 shows a set of multivariate normal distributions, all scaled with different weights. Another property of this description is that we can use s to interpolate between the absolute and uncertain representation of our data. Algorithm 3 shows how to compute the covariance matrix with this scaling factor. In the next section, we will use this fact to investigate how much the uncertainty influences the resulting projection.

5.3 Sensitivity Analysis

In previous sections, we have shown that uncertainty in the input can strongly influence the resulting set of principal components. To better understand this relationship, we investigate the degree to which the dimensionality reduction depends on the shape of the probability distributions. In Section 5.2, we have shown that our method is a generalized formulation of conventional PCA by scaling each distribution's covariances with a factor s that describes the importance of the uncertainty. Now, we will leverage this model to show how the fitted projection varies for different scaling factors in the interval $s \in [0, \infty[$. This interval can be split up into two parts to investigate two different scenarios. For $0 \leq s \leq 1$, we can interpolate between uncertainty-aware PCA and regular PCA. Conversely, by choosing $1 < s < \infty$, we can extrapolate what the projection would look like under more substantial uncertainty. In the following, we propose a novel visualization technique tailored to analyzing the effects of different scaling factors s and hence influences of different levels of uncertainty.

Factor Traces

Factor analysis shares many similarities with PCA, and it is a standard tool for the explanatory analysis of multi-dimensional datasets. The individual latent factors, similar to principal components, are usually represented using *factor maps*. To create a factor map, we project the unit vec-

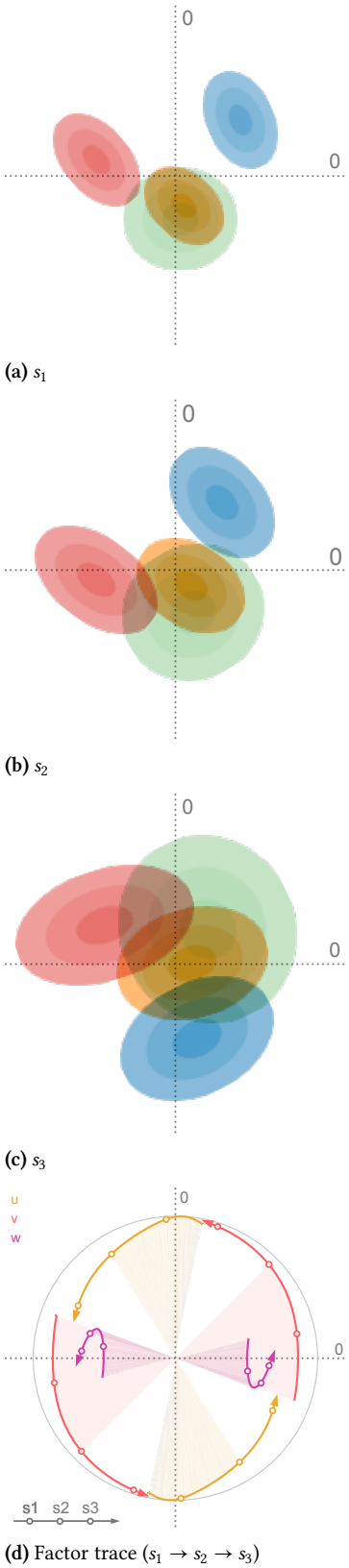
tors of the feature space according to the latent factors of the data [Ste15]. We extend this technique to enable the exploration of the effects of uncertainty on PCA.

Factor maps visualize static latent information of the input data. However, we are interested in visualizing the progression of uncertainty. We do this by looking at the *factor traces* described by the change of principal components under varying degrees of uncertainty. In particular, we perform sensitivity analysis by continuously scaling the distributions' covariances from the original dataset using s as a scaling factor, as described above. Figure 5.5 shows an example of factor traces of a three-dimensional dataset. For each $s \in [0, \infty[$ a different subspace is chosen. As a result, the projected unit vectors describe a trace in the image space. Thereby, we obtain a compact representation of the analogous transformation of the feature space coordinate system. As we mentioned before, there are two intervals for s that are of interest for analyzing the sensitivity regarding the uncertainty. The interval $0 \leq s \leq 1$ is highlighted by shading the area under the trace. In contrast, for the interval $1 < s < \infty$, we only show the trace to avoid visual clutter, and we use an arrowhead to represent $s \to \infty$.

In practice, we systematically sample s in the interval using a hyperbolic function. At the heart of principal component analysis lies the eigendecomposition of the covariance matrix, which entails various challenges for the interpretation of the projection. While the eigenvectors of a positive semi-definite matrix are always orthogonal to each other, their orientation is ambiguous as their sign can change from $\boldsymbol{v}_i$ to $\boldsymbol{v}_{i+1}$. Such a change in sign then leads to a flipped projection. Factor traces account for this by projecting both unit vectors of the original axes. This symmetry becomes apparent in the purple trace of Figure 5.5. We discuss the limitations of this approach in Section 5.6.

Interpretation

Factor traces simultaneously visualize different properties of the original dataset for the corresponding projection: The length of each trace describes how strongly each original axis is affected by the uncertainty. In contrast, the distance from each point of a trace to the center depicts the linear combination of the original unit vectors that define the projection. Factor traces also offer a way to analyze the robustness of the resulting projections against uncertainty. The covariance matrices and the overall shape of the data determine the corresponding eigenvalues. Because their eigenvalues sort the principal components and only the q largest eigenvalues are chosen, their respective values also have a large effect on the resulting projection. Figure 5.6 shows factor traces of two different datasets, together with plots of their eigenvalues. With increasing s, sometimes the distance between two eigenvalues λ_i, λ_j decreases more and more. In some cases, it appears that the eigenvalues will cross, but instead, they will eventually start to move away from each other again. This effect closely resembles *avoided crossings*, a quantum phenomenom [vNW91]. The reason for this effect is that two eigenvalues coalesce as they end up with the same length [SKM05]. Eigenvalues that avoid crossing manifest in distinctive bumps in their corresponding



(a) s_1

(b) s_2

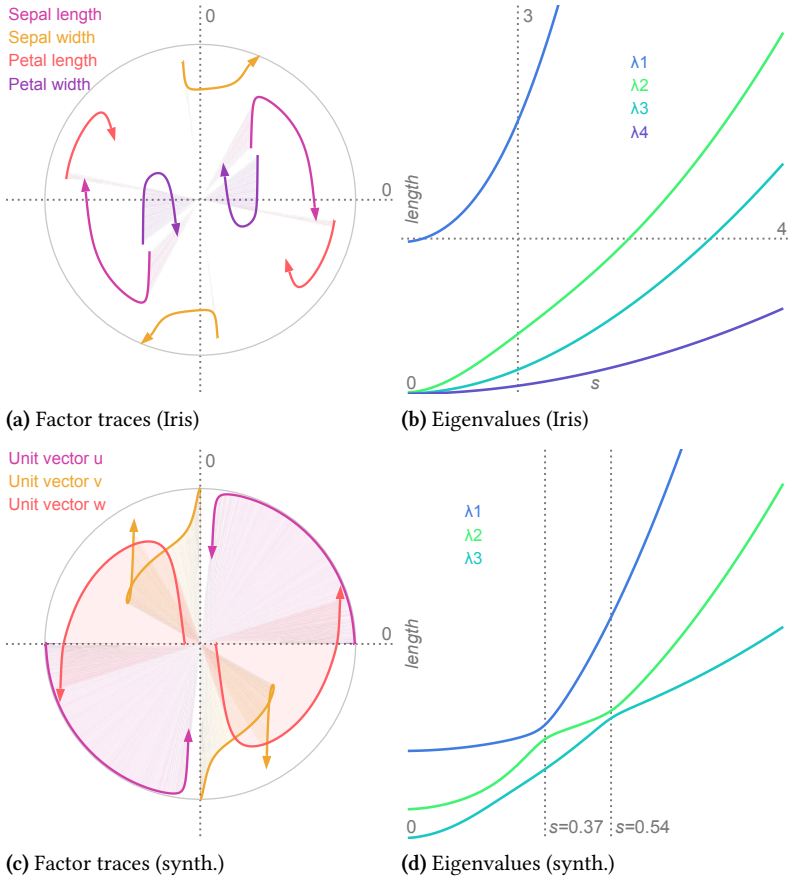
(c) s_3

(d) Factor trace ($s_1 \to s_2 \to s_3$)

Figure 5.5: Progression of a factor trace (bottom) with an increasing amount of uncertainty. Both unit vectors u and v rotate around w. Accordingly, the projected distributions (top) rotate around the mean as well.

eigenvalue plots, which can be seen in Figure 5.6d. The first dataset in Figure 5.6 does not contain any avoided crossings. By contrast, Figure 5.6c and Figure 5.6d show a three-dimensional dataset with two bumps (highlighted by the dotted lines). Avoided crossings make it difficult to reason about the eigenvectors' behavior and, consequently, the resulting projection at these points. In some cases — Figure 5.6c, for example — we can observe sharp turns in the corresponding factor traces. Here, the avoided crossing is between λ_2 and λ_3.

In conjunction with PCA, factor traces aid the exploratory analysis of datasets by giving insights into the principal components under uncertainty. Apart from showing how the projection changes, factor traces can assess the projection's robustness and trustworthiness. Still, the visualization of high-dimensional data involving a large variety of distributions remains a difficult challenge. Generally, factor traces work well for datasets with up to six original dimensions. Above this limit, overplotting makes the representation challenging to understand. As shown in Figure 5.6, the interpretation of factor traces can be further enhanced by taking the corresponding eigenvalue plot into account. Depending on the dataset, we see the possibility to encode this information directly onto the factor trace, either by thickness or color.



(a) Factor traces (Iris)

(b) Eigenvalues (Iris)

(c) Factor traces (synth.)

(d) Eigenvalues (synth.)

Figure 5.6: Factor traces for two different datasets: (a) The 4D Iris dataset with (b) corresponding plot of the eigenvalues. (c) A 3D synthetic dataset, also with eigenvalues (d). Whereas the Iris dataset has no avoided crossing eigenvalues, the synthetic dataset has two avoided crossings (d) represented by bumps in the plot ($s \in \{0.37, 0.54\}$). The factor traces are projected onto a 2D subspace — as a consequence only the second bump manifests in the traces: at $s \approx 0.54$ the orange trace v forms a loop, while the purple trace u curves inward.

5.4 Examples

Our method can handle various types of data uncertainty. Following the classification of Skeels et al. [Ske+08], we will look at examples from the measurement precision level and the completeness level. Measurement precision can play a substantial role in analyzing datasets, especially for qualitative studies and experiments, where it is hard to assign absolute values to responses. One way to deal with this uncertainty is to assign fuzzy numbers or even explicitly encoded probability distributions to each of the data points. Furthermore, we will look at different types of aggregations as sources for uncertainty on the completeness level. Apart from these examples, we see potential use cases for our method in visualizing preprocessed data for real-time analysis or aggregated data that protects the privacy of individuals[3]. Regarding aggregation, Section 5.6 gives more details about our approach's computational complexity. Please note that we use different representations for the distributions in the following examples to highlight the projections found by our method.

3: Aggregating data is common practice for medical data. A common approach is to ensure k-anonymity, which means that any combination of attributes applies to at least k people. This property prevents the identification of single users that have a rare combination of attributes. One way to achieve this is to aggregate values into ranges or distributions.

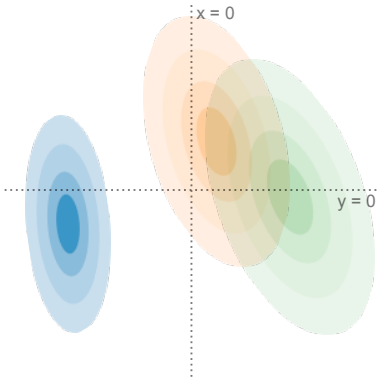
4: Published by the UCI machine learning repository at `https://archive.ics.uci.edu/ml/datasets/iris`

Iris Dataset

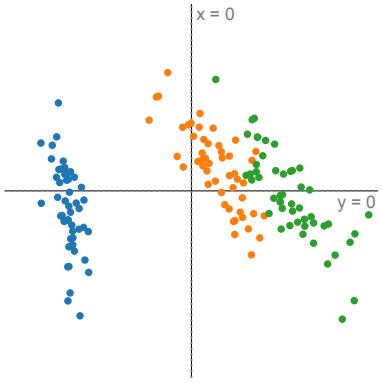
The Iris dataset[4] has widely been used to study projection and machine learning algorithms. It is four-dimensional and consists of 150 specimens of the Iris plants. Additionally, each instance belongs to one of three classes, and the instances are distributed equally among the classes. The Iris dataset's three classes are modeled accurately using multivariate normal distributions, on which we then perform uncertainty-aware PCA. The result of this can be seen in Figure 5.7a. For comparison, we also perform conventional PCA and color each point according to its class label — Figure 5.7b shows the result. Both projections are almost identical. This shows that our method can find projections with only a fraction of the original 4D data: three multivariate normal distributions instead of 150 points.

This example also illustrates two different ways to visualize data that has additional labels. To convey the class information, we need to support the visual aggregation of each cluster. When using conventional projection methods, this aggregation is usually performed in image space. Figure 5.7b, for example, uses color. Another aggregation technique in image space is kernel density estimation. For clusters that roughly follow a normal distribution, our method provides a different approach: It allows aggregation in feature space, where all the information is still present, and subsequent projection of the aggregated information. As a result, no further aggregation is needed in the image space. In Section 5.5, we provide a more detailed comparison to sampling-based strategies.

Figure 5.6a shows the factor traces for the Iris dataset. Here, we can see that *petal width* moves closest to the center of our visualization. This means that the dimensionality reduction projects increasingly along this axis when $s \to \infty$. Furthermore, *sepal width* and *petal length* have almost no shaded area. Because we use the shaded area to encode and highlight the interval $s \in [0, 1[$, this illustrates that the projection of these two axes



(a) PCA on distributions



(b) PCA on orginal points

Figure 5.7: Comparison between (a) our approach and (b) performing PCA on the original set of points of the Iris datasets. In (a), the aggregation has been performed before projection, while in (b), the aggregation into the different clusters is performed visually by the observer.

remains almost the same while interpolating between regular PCA and our method.

Student Grades

Our uncertainty-aware PCA method performs dimensionality reduction on data with explicitly encoded uncertainty. Amongst others, such data is common in the domain of fuzzy systems. As an example, we adopt the synthetic student grade dataset established by Denoeux et al. [DM04]. It consists of four test results (*M1*, *M2*, *P1*, *P2*) for each of six students. The possible marks for the tests range from 0 to 20, and the dataset is highly heterogeneous: grades can be represented either as real numbers, such as 15, without any uncertainty, or as intervals, such as $[10, 12]$. Furthermore, many grades are given by qualitative statements like *fairly good* or *bad*. Both intervals and linguistic labels contain uncertainty, modeled using uniform distributions and trapezoidal distributions, respectively. Table 5.1 shows the mapping from linguistic labels to distributions. The original paper also contains one *unknown* value. We model the missing value using a normal distribution $\mathcal{N}(14, 5.7^2)$, which we extract from prior information: the mean is similar to previous test results, and the variance represents realistic deviations in both directions from this mean. Figure 5.9 shows PCA on this dataset. It is important to note that PCA performed solely on the input's expected values fails to capture important uncertainty information. Figure 5.9a shows an example for this. Our method (Figure 5.9b) faithfully depicts the uncertainty that is present in *P1* of *Tom* and *Bob*, drawing a very different picture from the result of regular PCA because the topology changes: it is quite possible that *Tom* performed similar to *Jane* — a fact that is not readily visible from Figure 5.9a. The importance of *P1* on the resulting projection can also be seen in the factor trace (Figure 5.8) for this dataset: The trace of *P1* moves toward the outside of the unit circle, with an increasing amount of uncertainty. The interpretation for this is that the projection preserves most of the information along this axis.

Table 5.1: The trapezoidal distributions for the qualitative labels of the student grade dataset. A trapezoidal distribution $T(a, b, c, d)$ is defined by its bounds a, d and its discontinuities b, c.

Description	Distribution
very bad	$T(0, 0, 2, 6)$
bad	$T(2, 4, 6, 7)$
fairly bad	$T(5, 7, 9, 10)$
fairly good	$T(10, 11, 13, 14)$
good	$T(13, 14, 16, 18)$
very good	$T(14, 18, 20, 20)$

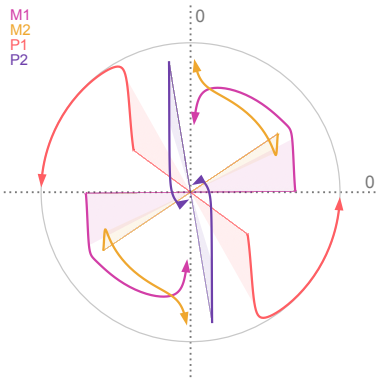
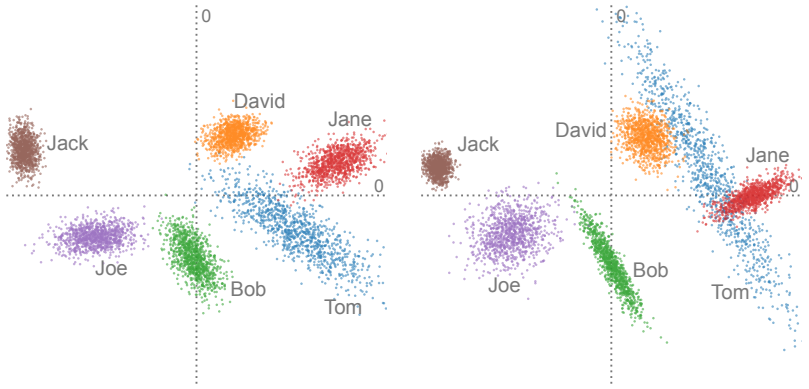


Figure 5.8: The factor traces corresponding allow us to analyze the role of the original axes. In this case, *P1* approaches unit length, which means that its information is present even after projection.



(a) Traditional PCA **(b)** Uncertainty-aware PCA (ours)

Figure 5.9: Importance of respecting the uncertainty in the input data. (a) Traditional PCA on the mean values of the student grades. (b) The projection found by our method shows the uncertainty in the data more faithfully.

Name	M1	M2	P1	P2
Tom	15	fairly good	$\mathcal{N}(14, 5.7^2)$	$[14, 16]$
David	9	good	fairly good	10
Bob	6	$[10, 11]$	$[13, 20]$	good
Jane	fairly good	very good	19	$[10, 12]$
Joe	very bad	fairly bad	$[10, 14]$	$[14]$
Jack	1	$[4, 6]$	9	$[6, 9]$

Table 5.2: The synthetic student dataset: Six students were assessed by four exams (M1, M2, P1, P2). The grades are given either by constant values, qualitative values (linguistic labels), or probability distributions. Table 5.1 defines the conversion from qualitative labels such as *fairly good* to probability distributions.

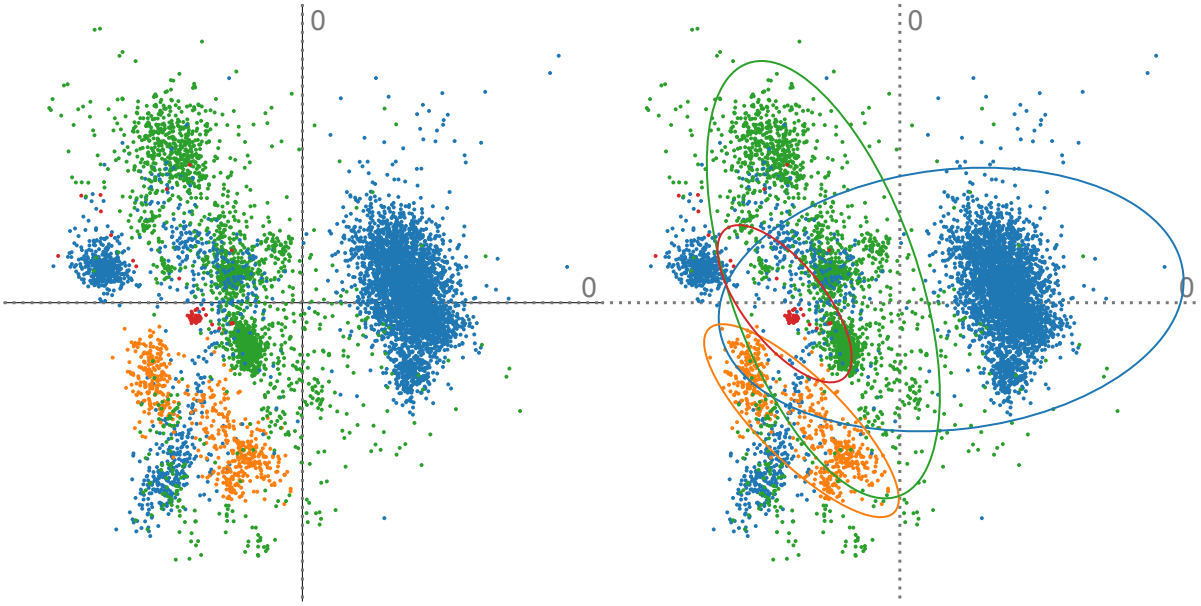
Anuran Calls Dataset

The Anuran Calls dataset[5] consists of acoustic sound features extracted from frog recordings. In total, there are 7195 instances of such calls. They are grouped by family, genus, and species labels. Again, we perform aggregation of the instances, in this case, by looking at the family class label. However, the difference of this dataset to previous examples is that there are a different number of instances per class. There are calls from four different frog families in this dataset — the numbers of instances per class are 4420, 2165, 542, and 68. These families can further be subgrouped by genus, yielding eight distinct clusters. Furthermore, it is essential to note that many groups do not follow a normal distribution and exhibit varying modality, as shown in Figure 5.10.

5: Published as part of the UCI machine learning repository [DG17].

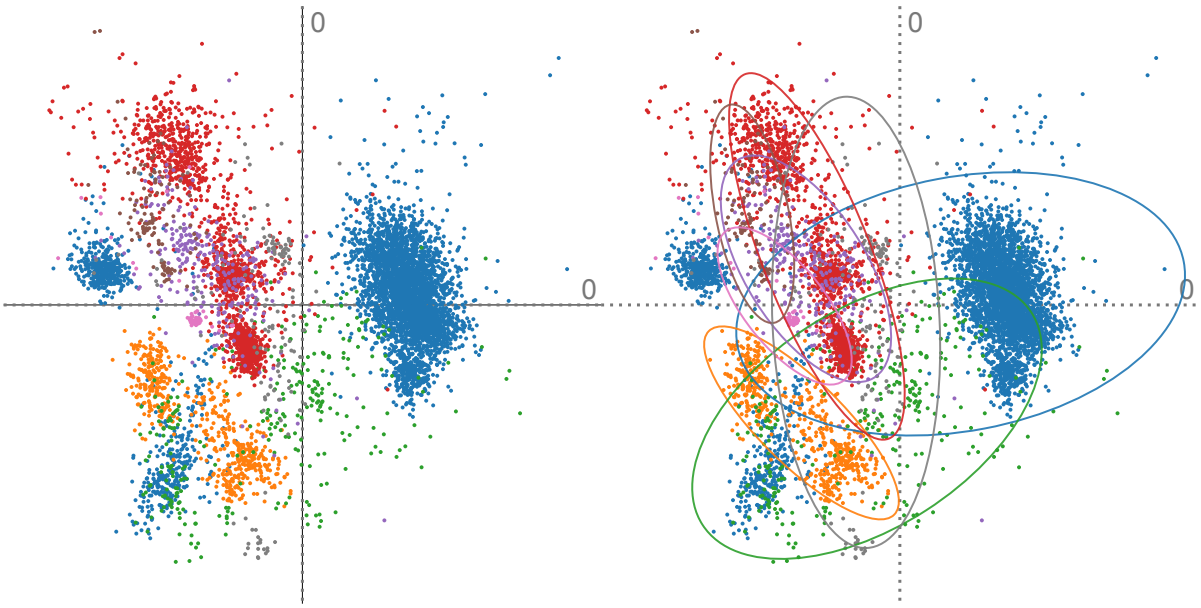
So far, we have assumed that all aggregated distributions represent the same amount of instances. If a cluster has only a few samples, its influence can become disproportionate. For the Anuran calls dataset, this would mean that the family with 4420 instances would receive the same amount of weight as the family with 68 instances. To achieve a better fit to the actual data that these distributions stand for, we can adapt our method to take class weights into account by slightly modifying Equation 5.2. In particular, it suffices to use the weighted average to evaluate $\hat{\mathrm{E}}[\cdot]$. The computation of the sample mean is adjusted accordingly.

Figure 5.10 shows the comparison of our method, adapted to handle cluster weights, to regular PCA on the original set of points. For Figure 5.10a and Figure 5.10b, the data is clustered by *family*, yielding four distributions. Figure 5.10c and Figure 5.10d were aggregated by *genus*, which results in eight distinct discrete probability distributions. We show the covariances extracted from each of the different clusters for the projections created with our method. The results demonstrate that even if clusters do not follow simple distributions, such as the blue cluster in Figure 5.10b, our technique can still reconstruct the original PCA. The projections for the point data and the aggregated data are visually the same. Assigning weights to each cluster according to the amount of data that it represents is an obvious application of this extension to our method. However, we can also imagine that this technique can be used in a more exploratory setting, for example, by investigating the effect of one cluster on the resulting principal components.



(a) Original points colored by family

(b) Distributions clustered by family



(c) Original points colored by genus

(d) Distributions clustered by genus

Figure 5.10: Comparison of projections resulting from conventional PCA and our method. Ellipses show the extracted covariances of each distribution. Clustering by family leads to four clusters, while clustering by genus results in eight clusters. Although the clusters have a large variance in the number of instances (a), our weighted approach matches the original dataset's projection well. The projection also remains stable for clustering by a different class label, here, by genus (b). Overall, our method (d) performs well, even though not all the original dataset clusters follow a normal distribution (c).

5.5 Comparison to Sampling

This section compares our method with another strategy to construct the covariance matrix for uncertain data: sampling[6]. Instead of directly computing $\mathrm{Cov}(\mathbf{T}, \mathbf{T})$ on the distributions, we can draw samples from each of them. If we concatenate the resulting set of points, we can use the conventional way for computing the covariance matrix as specified by Equation 2.1.

6: In Chapter 2, we talked in depth about different strategies for propagating uncertainty through the visualization pipeline. Here, we can actually do both without further assumptions, which lets us compare the two approaches.

To compare the resulting covariance matrices, we need a suitable distance metric. We choose the Hellinger distance, which is commonly used to compare the results of linear models [Tor91]. This distance metric compares two multivariate normal distributions p and q. It is based on the Bhattacharyya coefficient[7] that describes the overlap between p and q:

$$BC(p, q) = \int \sqrt{p(x)q(x)}\, dx$$

7: This coefficient is the basis of the Bhattacharyya distance $D_B(p, q)$:

$$D_B(p, q) = -\ln(BC(p, q))$$

It measures the amount of overlap between the two distributions. One problem with this measure is that it does not adhere to the triangle inequality. This is why we use the Hellinger distance. Furthermore, it is interesting to note that the Mahalanobis distance is a particular case of the Bhattacharyya distance for distributions that share the same covariance.

Using the definition of the Bhattacharyya coefficient, the Hellinger distance is defined as:

$$H(p, q) = \sqrt{1 - BC(p, q)}$$

To apply this distance metric to compare the results from principal component analysis, we recall that PCA is entirely defined by its sample mean and overall covariance matrix. Together, we interpret these two artifacts as a multivariate normal distribution, which we can then compare using the Hellinger distance. In contrast to a description based on eigenvalues and eigenvectors, our method is invariant against flipping, and no further preprocessing has to be performed.

For our experiment, we applied PCA to a synthetic dataset with 10 distributions $\mathbf{T}_{Syn} = \{\mathbf{t}_1, \ldots, \mathbf{t}_{10}\}$, each following a normal distribution $\mathbf{t}_i \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{i}, \Psi_i)$. All the means $\boldsymbol{\mu}_i$ are drawn from another overarching multivariate normal distribution:

$$\boldsymbol{\mu}_\mathbf{i} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

The covariance of each of the distributions Ψ_i is constant across the dataset. It is created by reversing the elements of Σ. Because of this, all covariances also share the same determinant.

Figure 5.11 shows the results of our experiment. For each data point, we performed 40 runs and chose the median outcome. We can draw several conclusions from our experiment. First, it shows that the sampling approach converges to our method's result with an increasing number of samples, which indicates that our method is a valid way to compute PCA on probability distributions. Second, it shows that our method scales far better than the sampling-based approach with a growing number of dimensions. We expect the curse-of-dimensionality to be the reason for this.

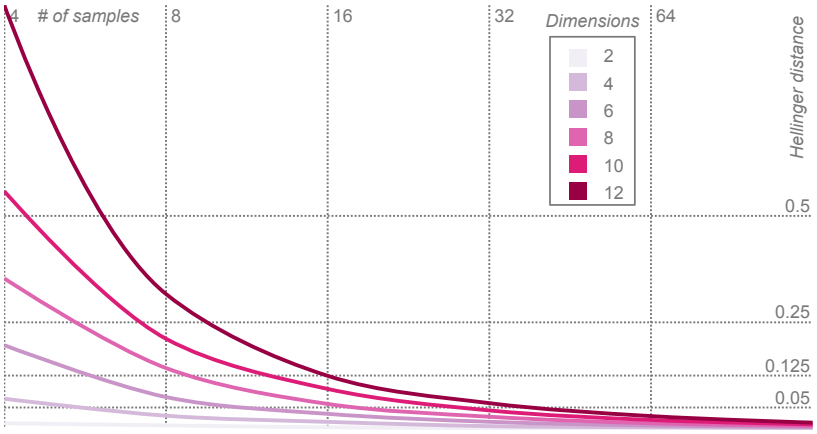


Figure 5.11: Multiple comparison of our method to a sampling-based approach using the Hellinger distance for input data with two to 12 dimensions. The x axis shows the increasing number of samples used for the sampling strategy, while the y axis shows the distance to the result from our method. The required number of samples for a good result grows with the number of dimensions.

5.6 Discussion

In the following, we discuss our extension of PCA. To begin with, we compare its computational complexity to traditional PCA. We follow up with more details on its application to interactive visualization, especially concerning scalability. Finally, we discuss the general limitations of PCA and how these carry over to our method.

Computational Complexity

In general, our method has the same computational complexity as regular PCA. For a dataset with N samples and D features, regular PCA has a computational complexity of $O(ND^2)$ for the computation of the covariance matrix. Retrieving the eigenvalues and eigenvectors has a complexity of $O(D^3)$.

With our method, the samples N are D-dimensional probability distributions instead of points. In many cases, the probability density function of a random vector $\mathbf{t}_n$ is known analytically, and $\mathbb{E}[\mathbf{t}_n]$ as well as $\mathrm{Cov}(\mathbf{t}_n, \mathbf{t}_n)$ can be looked up in constant time $O(1)$. Our adapted computation of the global covariance matrix has complexity $O(2 \cdot ND^2)$ because we additionally need to compute the average covariance matrix over all N distributions. Asymptotically, however, the constant factor 2 can be neglected. The result is a complexity of $O(ND^2)$ for determining the covariance matrix. We share the extraction of the eigenvalues and eigenvectors with regular PCA. As mentioned above, this can be performed in $O(D^3)$. Thus, our technique is of similar complexity as standard PCA[8]. In the following section, we provide details on why preparing clusters is of particular importance for applying our technique to data visualization.

8: Please note that we consider the aggregation of clusters as a preprocessing step. Its complexity would add to the total complexity but is not considered here. We provide more details in the next section.

Interactive Visualization and Scalability

Big data is gaining relevance, and the amount of data that can be acquired and stored grows rapidly[9]. At the same time, it often is critical to visualize such data for exploration, analysis, and knowledge generation [Sac+14]. Processing latencies are of significant concern for interactive visualization regarding big data. We tackle this problem by separating the computationally complex task of data aggregation from the pro-

9: The Large Hadron Collider (LHC) at CERN exceeded 200 Petabytes of collected sensor data already in 2017 [Gai17].

jection and visualization tasks. Since our method is aware of the shape of the distributions, we can approximate the projection of clustered datasets by the projection of their respective distributions. For a large number of samples N in a D-dimensional feature space, this aggregation step is computationally costly since the computing covariance matrices requires $O(ND^2)$. The advantage of our method is that the aggregation can be done instantly during data acquisition, and, in case memory demands are of concern, there is even no need to store raw data persistently [Wan+17]. In some fields, it is already common practice to aggregate data as a preprocessing step, for example, the *in-situ* analysis in large-data visualization [Dut+17]. Our method preserves the characteristics of the data during the complete pipeline, and their influence on the projection can still be taken into account during the analysis process. Estimating the covariance matrix is an inevitable step when we fit a normal distribution to multiple data points. To do so, the number of data points needs to be sufficient concerning the number of dimensions, and there must not be problems with (local) outliers [SS11]. Similarly, a small number of clusters can be a problem in high-dimensional space [JL09]. By scaling the uncertainty of each cluster depending on the number of data points, it contains, our method compensates for differences in cluster sizes, as outlined in the Anuran Calls example in Section 5.4. However, more research needs to be conducted to assess whether the additional information provided by each clusters' weight and error covariance matrix can fully counter this problem.

Limitations of PCA

PCA is applied to all kinds of datasets, mainly as a tool for exploratory analysis. Conceptually, our approach yields a projection operator that is more aware of the uncertainty in the data. Like other linear methods, important information in the non-principal components is discarded due to the orthographic projection, which can guide the analysis in the wrong direction. Our method inherits this limitation. For regular PCA, methods have been developed to mitigate these effects[10]. For one, this is because one of the terms of our method essentially performs PCA on the expected values of each of the distributions. A second limiting factor can arise concerning the uncertainty in the data: If the fraction of the covariance introduced by the uncertainty in the data is small compared to the covariance introduced by the expected values, and if the uncertainty happens to be orthogonal to the projection, it can also remain covert in the final representation. Future research may investigate how non-linear methods, which could alleviate this problem, can be generalized to probability distributions.

10: We provide a more general overview of these methods in Section 5.1.

Several other factors pose challenges to finding the correct principal components. The presence of outliers in the data can strongly influence the resulting projection, which stems from the quadratic term in the computation of the covariance matrix. When outliers are of concern, forms of *Robust PCA* (see Section 5.1), which rely on solving optimization problems, can be applied. It remains to be seen how such approaches can be adapted to uncertainty-aware PCA. Although PCA requires real-valued data, it is often also used on datasets where axes represent ordinal or

even categorical values. Naturally, these axes can contain uncertainty information as well. Furthermore, as of now, we do not explicitly model missing values. In the context of regular PCA, several techniques have been developed to deal with this — Dray et al. [DJ14] provide a summary of approaches that can be applied in this case. One straightforward way to handle these inputs in our framework is imputation, as we have done in the student grade example provided in Section 5.4. With our method, these imputed values can even take the form of more complex distributions, which is why we see this as a practical workaround.

5.7 Conclusion

We have presented a technique for performing principal component analysis on probability distributions. Unlike previous work, which was concerned with non-correlated error models, our method works on arbitrary distributions. We achieve this by incorporating the first and second moments of the uncertain input data into the global covariance matrix calculation. Our formulation of the global covariance matrix offers the potential for various extensions to traditional PCA. Particularly, in this thesis, we have shown the application to aggregated datasets (Figure 5.7 and Figure 5.10) and datasets with explicitly encoded errors (Figure 5.9).

In general, PCA and other linear dimensionality reduction techniques have the advantage that the projections remain interpretable, in contrast to non-linear methods. The principal components found by PCA are linear combinations of the axes from the original data space. With our technique, scaling the influence of each distributor's covariances allows us to perform sensitivity analysis concerning uncertainty. Our proposed factor traces are a visual method to assess how uncertainty in the data is reflected by each original dimension's contributions to the principal components. Further, our technique preserves the low computational complexity and clear algorithmic structure of traditional PCA. This enables the assessment of uncertainty induced differences to the projection by sampling different parameters for scaling the uncertainty. As a result, our technique constitutes a next step towards the earnest consideration of uncertainty in analyzing high-dimensional data and forms the foundation for straightforward extensions in numerous directions.

6 Directions for Further Research

There are several research opportunities that could build upon the work in this thesis. We have already identified concrete extensions to the presented methods in the individual chapters. The following chapter takes a step back and reflects upon future challenges in handling uncertainty in visualizations and looks at more general research topics. Although there has been growing interest in visualizing uncertainty over the last couple of years, the area is still far from being subject to wide interest. With the increasing maturity of a field usually come better tools and frameworks for other researchers to build upon. If we look at the area of machine learning, for example, there is a rich ecosystem of various open-source tools, such as *Scikit-learn* [Ped+11] that researchers and practitioners can access[1]. One thing that could greatly help the proliferation of uncertainty methods in visualization would be a collection of well-understood datasets that can be used to benchmark different methods or visualization systems against each other. Furthermore, having better, more expressive tools for handling uncertainty could significantly lower the barrier to entry into the field.

1: Of course, machine learning has drawn exceptional attention over the last decade, and its success would be hard to replicate. Still, the accessibility for researchers to get started in the field is worth pointing out.

Declarative specifications are a powerful tool to capture the important information and characteristics of a problem. The domain-specific query language *SQL* is arguably the most well-known specimen. For general purpose visualization, Satyanarayan et al. [Sat+17] developed *Vega-Lite* as a high-level grammar for creating interactive visualizations. It follows a declarative approach to specify what a visualization should look like, which is then auto-generated by that specification[2]. In statistics, *Stan* is a probabilistic programming language that allows describing models for inference declaratively [Car+17]. Similarly, finding a way to specify the uncertainty that is present in the data, either in the form of probability distributions or as quantitative uncertainty, could be an interesting direction for future research. In general, there is a gap between the tools that are used to describe uncertainty mathematically and JavaScript, the language in which most interactive visualizations are written today. Closing this gap would certainly benefit both domains. The challenging part of such an endeavor is allowing the flexibility to define statistical dependencies between the different variables — which leads directly to the next interesting problem of visualizing uncertainty that has not gained much attention yet.

2: This is also the main idea behind declarative approaches: Describe *what* we want to accomplish instead of *how* the actual computations are performed. The latter represents the traditional style of imperative programming.

6.1 Conditional Probabilities

Many of the uncertainty visualization techniques currently used — static ones in particular — encode the uncertainty for each of the data variables, usually as an additional annotation. Error bars are one example of this. However, this approach cannot capture important information about dependent variables. In Chapters 3 and 5, we assumed conditional independence between variables. We made similar assumptions to demonstrate our method to layout probabilistic graphs — the method itself can handle

arbitrary probability distributions because it is based on sampling. In the following, we take a closer look at sampling-based techniques that can show dependent variables.

Hypothetical outcome plots (HOPs) [HRA15], are capable of showing the dependencies between variables. Based on sampling, they respect the uncertainty by randomly drawing possible realizations of the data and then visualize them as individual frames of an animation. In the multivariate case, the different dimensions of the input data are usually visualized side by side. Because the samples are drawn directly from the underlying distribution, HOPs faithfully represent dependence and correlations. Nevertheless, the method also comes with some drawbacks: For one, it is harder to estimate the density of probability distributions, than for example, in *Violin plots* [HRA15]. Another problem arises when the distribution contains many modes[3]. This can cause visual clutter in the form of flipping and therefore results in incoherent representations of the variables. This is problematic because it makes it harder to identify patterns in the data. To make matters worse, humans tend to see patterns in random data, which is also known as *gambler's fallacy*. Therefore, it could be possible that some relationships or patterns perceived by users are purely virtual and not actually present in the underlying distributions. Also, and this is the case with many sampling-based approaches, it is not guaranteed that the tails of the distribution are faithfully represented, simply because it is less probable to draw samples in these regions. This is particularly true for higher-dimensional data — which we will describe in more detail in the next paragraph.

3: Modes describe the most frequent values in a dataset. The normal distribution has only a single mode — its mean – in contrast to bimodal or multimodal distributions with multiple "peaks".

The problems described above are not only inherent to hypothetical outcome plots. They can arise with any method that is based on sampling. Events that have a very low probability of occurrence belong to the *tails* of a probability distribution. For a normal distribution in the 1D case, these are the areas towards the left and right assymptotes[4]. Naturally, sampling from such distributions will yield only a few draws from those tails, which often leaves these areas under-represented. It is easy to see how this can get even worse for multivariate distributions: There might exist tails with rare events in each of the dimensions. By definition, their joint probability is even lower — this is another manifestation of the curse of dimensionality. Current research in mathematics is concerned with developing techniques that are able to create samples from these low-probability regions. These methods are often based on *Markov chain Monte Carlo* methods [Bio15]. It will be interesting to see this work carried over to the field of visualization.

4: Mathematically, tails are described as the intervals $P(X < -x)$ and $P(X > x)$ for large values of x.

Understanding the tails of distribution is essential when creating visualizations, particularly ones aimed at supporting decisions. The reason for this is that rare events from the tails often impose an unproportionate risk. Therefore, proper risk management should be a fundamental feature of such systems. In Chapter 2, we discussed the different meanings of the terms "uncertainty" and "risk" in decsion theory. To summarize: Decision-making under risk means that all alternatives and probabilities are known in advance, which is why we can use statistics to guide the process. On the other hand, decision-making under uncertainty means that not everything can be determined or factored into the decision. Following this definition, it is impossible to quantify the uncertainty fully. In their current state, visualization systems can help to quantify the risk

associated with given outcomes. Ensemble methods come to mind, which show potential outcomes under varying assumptions. To deal with the uncertainty that cannot be modeled statistically, users have to resort to *heuristics* to make decisions [MG14]. We imagine that future visualization systems can help the user identify, test, and refine their heuristics. Such a system will interactively help explore the *bias-variance tradeoff*[5] for various decision policies. In the next section, we will look at a general approach to model assumptions and domain knowledge in statistical methods, which complements the ideas from the previous paragraphs.

5: The bias-variance tradeoff is a dilemma from statistics, which states that a model cannot capture patterns from training data and still generalize well to all unseen data at the same time. In the context of decision theory, heuristics can introduce bias while at the same time minimizing the variance.

6.2 Bayesian Methods

A very promising avenue for future research in information visualization focusing on uncertainty is the adaption of Bayesian methods. As we will see, these methods are also closely related to conditional probabilities and widely used in statistical modeling. They are applied to a wide variety of tasks ranging from prediction to regression. Their ability to encode assumptions about the world lends them their flexibility. These assumptions are called *priors* and modeled by probability distributions. If we observe additional information, these priors are updated by conditioning them on the observed data. This process yields the *posterior* distribution of possible outcomes. As we will see later, this setup makes them very suitable for decision making under uncertainty. Their fundamental building blocks are Bayesian statistics and Bayes' rule of conditional probabilities, in particular. Given a hypothesis H, we describe the probability of H being true by its prior distribution $P(H)$. Once we observe new evidence E in the form of new data, we can use Bayes' rule to update our beliefs and obtain the posterior distribution $P(H \mid E)$:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Here, $P(E \mid H)$ describes the *likelihood* of observing the evidence E given the hypothesis H and $P(E)$ denotes the general probability of observing the evidence.

Bayesian methods belong to the family of generative models. This means that they allow us to generate potential outcomes by sampling the associated probability distributions, which makes them well suited for communicating uncertainty as seeing individual outcomes helps us estimate the variability intuitively. In a similar vein, Kay et al. [Kay+16] use a Bayesian model to communicate the uncertainty in bus departure times, helping the user in their decision-making process of when to leave for their connection. For this, they investigate and compare different visualization techniques and conclude that counting-based techniques, such as carefully crafted dot plots[6], show promising results in comparison to continuous visualizations. These methods make the uncertainty easy to grasp: The user counts the number of desirable outcomes and pits this information against the number of unfavorable events.

6: The authors propose *quantile dot plots*, which derive samples directly from intervals in the cumulative distribution function to overcome noise [Kay+16].

The fact that Bayesian methods belong to generative models is not the only reason why they are compelling from a visualization perspective. If we want to support the decision-making process, we need to incorporate

the user by allowing them to model their knowledge and iteratively test their hypothesis[7]. Of course, this is very challenging not only from an interaction perspective but also because user interaction can be an additional cause of uncertainty that is orthogonal to the uncertainty in the data that we have considered up to this point. Bayesian methods offer direct solutions to these problems. By adapting the prior distributions for the potential outcomes, domain experts can encode their knowledge into these models. However, this process is not always straightforward from an interaction standpoint, which leaves many opportunities for future research.

7: This paradigm is often called *human-in-the-loop*. Endert et al. [End+14] argue that many algorithms, particularly ones supporting exploratory analysis, have to be redesigned from the ground up to achieve this. Our work on predicting the users' intent [Gad+21], which we outline later in this section, is one step in that direction.

In the course of this chapter, we show the potential of Bayesian methods for varying tasks in information visualization. First, we introduce Gaussian processes and talk about strategies that could make it easier for domain experts to see the influence of their prior knowledge on the final prediction. Next, we show how to use Bayesian methods for modeling uncertainty in the user input. Such models, in turn, can be used to improve the user's interaction with visualization systems. Finally, we show how we can use a *Bayesian classifier* to predict the current task that the user wants to perform.

Gaussian Processes

This section features excerpts from our article „A visual exploration of Gaussian processes" [GKD19]. It provides a more in-depth discussion of this topic and explains our methodology.

Gaussian processes are a powerful tool in the machine learning toolbox [Ras04]. They allow us to make predictions about our data by incorporating prior knowledge. Their most obvious area of application is regression — *fitting* a function to the data[8]. Because Gaussian processes are based on Bayesian inference, they also contain a generative model of the data. This means that we can draw samples from them that allow us to understand the uncertainty in the resulting predictions. In that sense, Gaussian processes are very similar to hypothetical outcome plots.

8: Gaussian processes are often applied in robotics and time series forecasting, but they can also be extended to other tasks such as classification or clustering tasks [Kap+10; KL07].

For a given set of training points, there are potentially infinitely many functions that fit the data. Gaussian processes offer an elegant solution to choosing the right one by assigning a probability to each of them [Ras04]. The mean of this probability distribution then represents the most probable characterization of the data. Furthermore, using a probabilistic approach allows us to incorporate the confidence of the prediction into the regression result. Together with kernels, the multivariate normal distribution (MVN) forms the mathematical foundation for Gaussian processes.

Marginalization and Conditioning

Multivariate normal distributions have the nice algebraic property of being *closed* under conditioning and marginalization. Being closed under an operation means that the resulting distributions are also distributed normally, which makes many problems in statistics and machine learning tractable. Marginalization and conditioning both work on subsets of the original distribution. We will use the following notation, with X and

Y representing subsets of original random variables:

$$P_{X,Y} = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX}\, \Sigma_{XY} \\ \Sigma_{YX}\, \Sigma_{YY} \end{bmatrix}\right)$$

Marginalization gives us a way to extract partial information from multivariate probability distributions. In particular, for a normal probability distribution $P(X, Y)$ over vectors of random variables X and Y, we can determine their marginalized probability distributions in the following way:

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX})$$
$$Y \sim \mathcal{N}(\mu_Y, \Sigma_{YY})$$

The interpretation of this equation is that each partition X and Y only depends on its corresponding entries in μ and Σ. To marginalize a random variable from an MVN, we can drop the variables from μ and Σ.

$$p_X(x) = \int_y p_{X,Y}(x, y)dy = \int_y p_{X\,|\,Y}(x\,|\,y)p_Y(y)dy$$

The way to interpret this equation is that if we are interested in the probability density of $X = x$, we need to consider all possible outcomes of Y that can jointly lead to the result[9].

9: Or, put in other words, marginalization is the same as integrating along the remaining dimensions of the joint distribution. The corresponding Wikipedia article has a very detailed description of the marginal distribution, including several examples (https://en.wikipedia.org/wiki/Marginal_distribution).

The other necessary operation for Gaussian processes is *conditioning*, which determines one variable's probability given another variable. Similar to marginalization, this operation is also closed and yields a modified distribution. This operation is the cornerstone of Gaussian processes since it allows Bayesian inference. Conditioning is defined by:

$$X\,|\,Y \sim \mathcal{N}(\,\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y)\,,\; \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\,)$$
$$Y\,|\,X \sim \mathcal{N}(\,\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_X)\,,\; \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\,)$$

Note that the new mean only depends on the conditioned variable, while the covariance matrix is independent from this variable[10].

10: Conditioning also has a nice geometric interpretation — we can imagine it as making a cut through the multivariate distribution, yielding a new Gaussian distribution with fewer dimensions.

Modeling Processes as Distributions

Stochastic processes, such as Gaussian processes, are a set of random variables, each of which has a corresponding index i. To perform the regression, we will move from the continuous view of a function to its discrete representation: Rather than finding an implicit function, we are interested in predicting the function values at concrete *test points* X. How do we derive this functional view from the multivariate normal distributions that we have considered so far? We will use the index of the process i to refer to the i-th dimension of our n-dimensional multivariate distributions.

Now, the goal of Gaussian processes is to learn this underlying distribution from *training data* Y. As we have mentioned before, Gaussian processes model the joint probability distribution $P_{X,Y}$ as an MVN in the space of possible function values for the function that we want to predict[11]. This joint distribution of test and training data has $|X| + |Y|$ dimensions. To perform regression on the training data, we will treat this

11:

problem as Bayesian inference. The essential idea of Bayesian inference is to update the current hypothesis as new evidence becomes available. In the case of Gaussian processes, this information is the training data. Thus, we are interested in the conditional probability $P_{X|Y}$, which itself follows a normal distribution. In Gaussian processes, it is common to set $\mu = 0$, which simplifies the necessary equations for conditioning — it gets more interesting when we look at Σ, the other parameter of the distribution. The covariance matrix describes the distribution's shape and ultimately determines the function's characteristics. In Gaussian processes, Σ is determined by its covariance function k, which can vary depending on the data at hand. It is also called *kernel*.

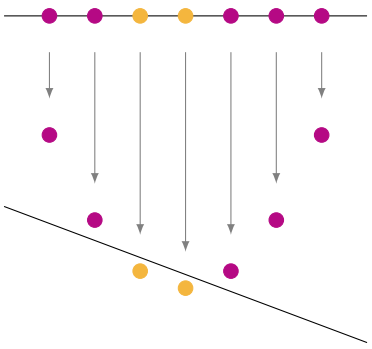


Figure 6.1: Sparked by better hardware and GPU implementations, training deep neural networks has received a large amount of attention recently. Before this "revolution" set in, learning algorithms based on kernels achieved state-of-the-art results. The idea behind the kernel trick: It transforms points that cannot be separated linearly (top) into a higher-dimensional space — here onto 2D — where this is possible (bottom).

Kernels

In their basic form, many machine learning methods are based on linear methods. To make them applicable to a wider range of problems, they use kernels, which make it possible to apply these linear methods to problems that are non-linear. There are a variety of different kernels that we can use to map points from an input space onto a higher-dimensional manifold. Although this transformed set of points has more dimensions, the problem that we want to solve can actually be easier in this new space. In binary classification, for example, we could have a set of points that cannot be separated linearly into half-spaces. But, by using a clever transformation — a suitable kernel — this becomes possible in the higher-dimensional space. Figure 6.1 shows an example of this.

We generate the covariance matrix of the Gaussian process by pairwise evaluating the kernel k for all the points. The kernel receives two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$ as an input and returns a similarity measure between those points in the form of a scalar:

$$k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \Sigma = \mathrm{Cov}(X, X') = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

We evaluate this function for each pairwise combination of the test points to retrieve the covariance matrix. In order to get a better intuition for the role of the kernel, we need to think about what the entries in the covariance matrix stand for. The entry Σ_{ij} describes how much influence the i-th and j-th point have on each other. This follows from the definition of the multivariate normal distribution, which states that Σ_{ij} defines the correlation between the i-th and the j-th random variable. Since the kernel describes the similarity between the values of our function, it controls the possible shape that a fitted function can adopt[12]. This is where the flexibility of Gaussian processes comes from. By choosing a suitable kernel function, domain experts can directly encode their knowledge into a model, which lets Gaussian processes capture the relevant patterns in the training data.

Another significant benefit that kernels provide is that they can be combined, resulting in a more specialized kernel. The decision on which kernel to use is highly dependent on prior knowledge about the data. Examples of this are stationary behavior, global trends, or other patterns. *Stationary* kernels are translation invariant and therefore independent of the index i. This also means that we cannot model global trends using a strictly stationary kernel. On the other hand, *non-stationary* kernels

12: One of the most prominent kernels is the radial basis function kernel:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma}\right)$$

When defining new kernels, we need to make sure that the resulting matrix adheres to the properties of a covariance matrix.

do not share this property. An example of such a kernel is the linear kernel, which mimics regular linear regression. Remember that the covariance matrix has to be positive semi-definite. When choosing the optimal kernel combinations, all methods that preserve this property are allowed. The most common kernel combinations are addition and multiplication[13]. Let's consider two kernels, a linear kernel k_{lin} and a periodic kernel k_{per}, for example. This is how we would multiply the two:

$$k'(x_i, x_j) = k_{\text{lin}}(x_i, x_j) \cdot k_{\text{per}}(x_i, x_j)$$

If we add a periodic and a linear kernel, the global trend modeled by the linear kernel carries to the combined kernel. The result is a periodic function that rises over time[14]. When combining the same kernels through multiplication, the result is a periodic function with a linearly growing amplitude away from a center point. In the next section, we will describe how to perform predictions using Gaussian processes.

13: There are more possibilities such as concatenation or composition with a function [Mac03].

14: Such a model could be useful for analyzing weather and climate data.

From Prior to Posterior

We will now shift our focus back to the original task of regression. As we have mentioned earlier, Gaussian processes define a probability distribution over possible functions, where each sample of our multivariate normal distribution represents one realization of the function's values. For now, we consider the case where we have not yet observed any training data. In the context of Bayesian inference, this is called the prior distribution $P(X)$, which will have the same dimensionality as the number of test points $N = |X|$. Accordingly, we use the kernel to set up the covariance matrix, which has dimensions $N \times N$ and determines the type of functions, from the space of all possible functions, that are more probable. Figure 6.2 shows samples of potential functions from a prior distribution.

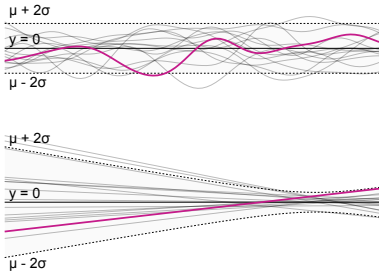


Figure 6.2: Multiple samples drawn from prior distributions with an RBF kernel (top) and a linear kernel (bottom). Over time, it is possible to see that functions are distributed normally around the mean μ.

So what happens when we observe training data? According to Bayesian inference, we can incorporate this additional information into our model, which yields the posterior distribution $P(X \mid Y)$. For Gaussian processes, this procedure is straightforward: First, we form the joint distribution $P(X, Y)$ between the test points X and the training points Y by concatenation. The result is a multivariate Gaussian distribution with dimensions $|X| + |Y|$. Then, we use *conditioning* to determine $P(X \mid Y)$ from $P(X, Y)$. The dimensions of this new distribution match the number of test points N — and the new distribution is also normal, albeit with different mean and standard deviation: $X \mid Y \sim \mathcal{N}(\mu', \Sigma')$. The intuition behind this step is that the training points constrain the set of functions to those that pass through the training points. Figure 6.3 shows an example of this. Visualizing the covariance matrix next to the prediction makes the influence of the points on another apparent.

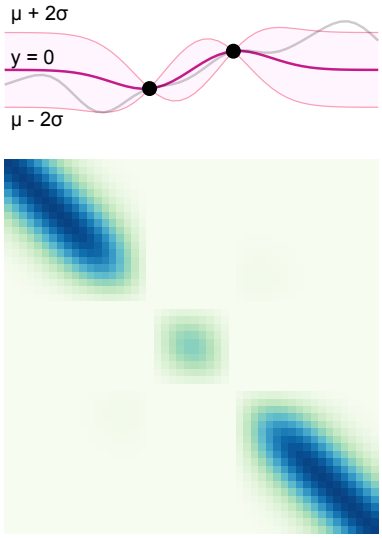


Figure 6.3: Gaussian process with two training points (top). The grey line is one sample from this process. Conditioning on the given points restricts the covariance matrix (bottom). Note how the training points constrict the covariance matrix.

Gaussian Processes for Uncertainty Visualization

There are different ways to visualize the resulting predictions from Gaussian processes. Analogous to hypothetical outcome plots, which we described earlier, we could obtain a set of possible predictions for the function values by sampling from $P(X \mid Y)$. Another approach would be to *marginalize* over each dimension of $P(X \mid Y)$ to extract the mean μ'_i and

standard deviation $\sigma_i' = \Sigma_{ii}'$ for the i-th test point. Here lies one of the strengths of Gaussian processes that we can use for uncertainty visualization: Instead of a single value, we get a probability distribution of potential outcomes for each test point — a richer and more interpretable output. An interesting research direction would be to use this expressiveness to support model-peeling and hypothesis testing. Ideally, such a system would also provide a user interface designed to help domain exports encode their knowledge intuitively. There has been recent research on directly synthesizing suitable combinations of kernels using a domain-specific language [Saa+19]. Extending their work with a visual interface that allows quick comparison of predictions would be another exciting research opportunity.

Gaussian processes handle uncertain observations gracefully. As mentioned before, the conditional distribution $P(X \mid Y)$ forces the set of functions to precisely pass through each training point. In many cases, this can lead to fitted functions that are unnecessarily complex. Interpreting the training points Y to be perfect measurements makes this effect even worse. Gaussian processes offer a simple solution to this problem by allowing to model the error of the measurements[15].

15: To achieve this, we can add an error term $\epsilon \sim \mathcal{N}(0, \psi^2)$ to each of our training points:

$$Y = f(X) + \epsilon$$

Then we can slightly modify the setup of the joint distribution $P_{X,Y}$:

$$P(X, Y) \sim \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\Sigma_{XX} & \Sigma_{XY}\\ \Sigma_{YX} & \Sigma_{YY} + \psi^2 I\end{bmatrix}\right)$$

Again, we can use conditioning to derive the predictive distribution $P_{X \mid Y}$. In this formulation, ψ is an additional parameter of our model.

In Chapter 5, we discussed how to perform dimensionality reduction on multivariate normal distributions. As we have seen in this chapter, Gaussian processes are well suited to model time series data. We believe that in conjunction, these methods could be used to find projections onto 2D. The goal for such a projection is that similar time series end up in similar locations, essentially clustering them. A way to approach this would be to model the time series as Gaussian processes, with a mean μ corresponding to the steps of the time series. There is some flexibility in setting up the covariance matrix: By choosing from various kernels, the user could steer which features in the input data are important — for example, periodicity or local similarities. The MVN that results from this process can then be used for dimensionality reduction and would readily work with our uncertainty-aware PCA approach. We expect the similarities between time series to be non-linear in most cases. Because of this, it would also be worth investigating non-linear projections techniques in this context[16].

16: Kernel PCA [SSM97] in combination with the Hellinger distance (Chapter 5) could be a good starting point for this.

Probabilistic Selection of Primitives

If we want to embrace uncertainty in visualization fully, we also need to consider the variability and ambiguity that arises from user interactions. We believe that Bayesian methods can help achieve this. In this section, we propose a method that models uncertainty when selecting visual primitives — in this case, lines — to demonstrate the approach. First, let us motivate the problem with a more concrete example: Selections are a typical user interaction in exploratory visualization systems. They are usually used to drill down into the data by providing additional information or details about the data that corresponds to a visual primitive. In some cases, clicking on a primitive triggers further processing steps, such as filtering the data. Many visualization systems are web-based, so they often use the build-in events of HTML[17]. However, this comes with certain problems: An action is triggered only when the cursor is accurately positioned over an element. The problem is that this does not scale to

17: In particular, the `onMouseover` and `onClick` events are the culprit here.

datasets that contain overplotting, where many elements overlap. Furthermore, positioning the mouse pointer over an element becomes increasingly tricky as the primitives get smaller. To mitigate these problems, we show how we can model user input — the position of the mouse pointer — as a probability distribution and how this influences the selection using Bayesian statistics.

Formally, we are interested in finding the probability that the user wants to select a particular shape S, given its uncertain mouse position C that can be subject to uncertainty. According to Bayes' rule, the probability of selection $P(S \mid C)$ is given by:

$$P(S \mid C) = \frac{P(C \mid S)P(S)}{P(C)}$$

In this example, we will mainly focus on computing $P(C \mid S)$ under the assumption that the uncertainty that governs the input is distributed normally. In other words, this means that we assume a Gaussian error on the position of the mouse pointer. Without loss of generality, we will restrict ourselves to an isometric error model[18] that only depends on the parameters a and c. The probability density function for the error centered at the position of the pointer $\boldsymbol{p}$ is given by:

$$G(\boldsymbol{x}) = c \cdot \exp\big(-a^2\|\boldsymbol{x}-\boldsymbol{p}\|^2\big)$$

18: An isometric error model means that the expected error is the same in each dimension. This is reflected in a diagonal covariance matrix of the normal distribution that models the error.

For now, we only consider line segments as visual primitives. However, our technique applies to other primitives over which we can integrate as well. We now show how to compute $P(C \mid S)$ by making use of the error function $\operatorname{erf}(x)$. A similar technique is also used in computer graphics for raytracing clouds and fog [Zho+07]. In the following, we adapt their notation. First, we model the line segment with length d as a ray starting at $\boldsymbol{v}$ with normalized orientation $\hat{\boldsymbol{r}}$:

$$\boldsymbol{x}(t) = \boldsymbol{v} + t\hat{\boldsymbol{r}}$$

Our goal is to integrate the Gaussian error function over this ray:

$$\boldsymbol{y} = \int_0^d c \cdot \exp\big(-a^2\|\boldsymbol{x}(t)-\boldsymbol{p}\|^2\big)\,dt$$

Let $\boldsymbol{p'} = \boldsymbol{p} - \boldsymbol{v}$ and $\hat{\boldsymbol{p}}'$ its normalized version from the start of the line segment $\boldsymbol{v}$. Further, ϕ is the angle between $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{p}}'$, $\cos\phi = \hat{\boldsymbol{r}} \cdot \hat{\boldsymbol{p}}'$, and $\sin\phi = \|\hat{\boldsymbol{r}} \times \hat{\boldsymbol{p}}'\|$. We can then substitute the ray equation and use trigonometry to write:

$$\begin{aligned} y &= \int_0^d c \cdot \exp\big(-a^2\|t\hat{\boldsymbol{r}} - \boldsymbol{p'}\|^2\big)\,dt \\ &= \int_0^d c \cdot \exp\big(-a^2((t - \|\boldsymbol{p'}\|\cos\phi)^2 + \|\boldsymbol{p'}\|^2\sin^2\phi)\big)\,dt \\ &= c \cdot \exp(-a^2\|\boldsymbol{p'}\|^2\sin^2\phi) \int_0^d \exp\big(-a^2(t - \|\boldsymbol{p'}\|\cos\phi)^2\big)\,dt \end{aligned}$$

Finally, we can approximate this integral using the error function $\operatorname{erf}(x)$:

$$y = c \cdot \exp(-a^2\|\boldsymbol{p'}\|^2\sin^2\phi)\frac{\sqrt{\pi}}{2a}\big(\operatorname{erf}(a(d - \|\boldsymbol{p'}\|\cos\phi)) - \operatorname{erf}(a(-\|\boldsymbol{p'}\|\cos\phi))\big)$$

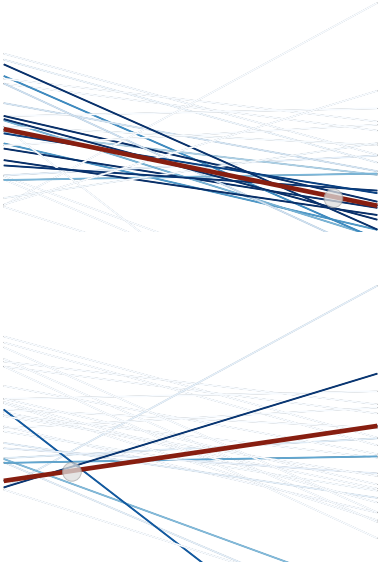


Figure 6.4: Probabilistic selection of lines for dense (top) and sparse (bottom) regions. Each line is shaded according to its probability from light-blue to blue. The best match is highlighted in red.

This solution is in closed form, so the probability for a line segment can be computed in constant time $O(1)$.

Now that we can compute $P(C \mid S)$ and know $P(C)$, the final term that we need to consider is the prior $P(S)$. Again, this prior can be chosen to model the behavior of the user. The simplest assumption would be to make the probability of the shape proportional to its area, as it is more likely for the user to select shapes with larger areas. In this case, a $P(S)$ can be computed by integrating over uniform distribution defined over the drawing area. However, the flexibility of the Bayesian formulation also allows us to tweak the prior to better match the task that we expect the user to perform with the visualization. For instance, if we assume that one primitive is more likely to be selected — like the median of an ensemble — $P(S)$ could be chosen to reflect this.

Figure 6.4 shows an example of using this method to find the probability of selection for each line. This method is general and easy to adapt for other primitives and more complicated selections such as brushing by recording the brush positions and placing a kernel at each position. The resulting algorithm would be very similar to kernel density estimation.

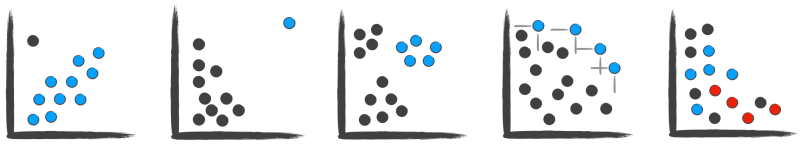
Predicting User Intent

When experts interact with a visual analysis system, they are frequently guided by a domain-specific analysis goal, such as identifying a gene that could be a drug target. To answer this question, they execute a series of tasks, such as selecting a set of correlated items for detailed analysis. In contrast to the high-level goal of answering a domain-specific question, these intermediate tasks are often based on patterns in the data: For example, selecting outliers, clusters, or items that correlate. Such a carefully constructed selection of items based on a higher level but domain-agnostic structure reflects a reasoning process — the analyst's intent. The following section shows how we can use Bayesian methods to infer such intents for brushes in scatterplots.

This section is a shortened version of our article „Capturing user intent when brushing in scatterplots“ [Gad+21]. In the article, we propose different ways to predict the users' intent — here, we only focus on the Bayesian aspects of our method.

Why is capturing intents important? First, inferring intents based on partial selections can be used to *auto-complete* selections. For example, to select outliers, analysts would only have to brush a few examples and then auto-complete the selection instead of painstakingly brushing them individually. Auto-complete can also be used to correct a selection. For example, if an analyst intended to select a cluster, reviewing the predicted cluster might reveal points that should be added to the selection.

Second, making intents available in provenance data improves *recall and reproducibility* of analysis processes conducted with visualization tools. By capturing such processes at a higher abstraction than just low-level interactions, they become more transparent when revisited either by the original analyst or a collaborator. Hence, analysis sessions that capture intents are more justifiable and likely to increase trust in the process. Down the line, such rich provenance data also has the potential to enable re-using visual analysis sessions on modified or updated data. For example, when an analyst first removes outliers before proceeding with an analysis, that action could be translated into a rule which could be used to automatically remove outliers from an updated dataset.



(a) Correlation **(b)** Outliers **(c)** Clusters **(d)** Skyline **(e)** Categories

Figure 6.5: Common data patterns that can be found in scatterplots. The *skyline* (d) operator is a form of non-parameteric multivariate optimization.

Mapping Patterns to Intents

When analyzing data, users have intentions at different levels of abstraction. We are specifically interested in the intents behind brushes or selections of data items in scatterplots. By surveying the literature and performing interviews with domain experts, we identified the following data patterns that match user intents when analyzing data in scatterplots[19]. Figure 6.5 gives an overview of these patterns.

19: Our article provides additional patterns and more details on the methodology we used to find them. Here, we only present the subset of patterns that can be predicted from user selections using Bayesian inference.

Correlations
Describe associations between two or multiple dimensions. Frequently, analysts want to identify correlations in the data and find points that do not fit the correlations. We found that participants did an approximate visual regression analysis, identifying both linear and non-linear trends.

Outliers and Inliers
Outliers are data points that differ significantly from other items[20]. Frequently, analysts wanted to understand what causes the data points to be outliers, relying on their background knowledge. We consider both outliers and "inliers", i.e., the set of points that are not outliers, as target patterns.

20: Outliers are also related to, but distinct from the points that do not fit a correlation: An item can be an outlier in its magnitude but perfectly fit the correlation.

Clusters and Groups
Clusters or groups of data points are items that are similar to each other but distinct from the rest of the dataset. They were mentioned as a pattern that analysts look for in three out of six interviews. Clusters were frequently not well-defined in the data they analyzed.

Multivariate Optimization
One goal when analyzing data is to find data points that are dominant over multiple dimensions. A typical example is to find a hotel that is both close to the city center and affordable. The set of such points is often called a skyline [BKS01]. Hotels in the skyline are such that no other hotel is both cheaper and closer to the center.

Categories
An observed pattern can sometimes be traced back to the items being of distinct categories. Four out of our six expert participants mentioned that they intend to select elements by category. For example, one expert wanted to separate the experiment from controls.

These patterns are also commonly targeted in data mining, which implies that various algorithms can identify them. We leverage this to calculate a broad set of patterns using different algorithms, combinations of dimensions, and parameters. We then compare the computed patterns with user

selections and rank them according to that match. While our initial step creates a large set of patterns, the subsequent ranking makes these predictions manageable. We now explain the details of the algorithms used and a Bayesian ranking approach.

Bayesian Ranking

All the described patterns result in a classification for each item in the dataset. To rank the predictions in our system, we compare these patterns with a binary classification representing an analyst's selection. Like clustering, some algorithms produce a multi-class prediction, which we first transform into a set of binary classifications. We can then use a similarity metric to rank each of the predictions. We use a pre-processing step to remove duplicate predictions for the same pattern from the set of predictions to rank. Duplicate predictions frequently occur if a pattern is robust to different parameterizations of the same algorithm.

A straightforward way to rank an algorithm's output to the user selection would be to apply the Jaccard index[21], which considers each possible pattern independently. However, an analyst's intent is rarely independent, and some predicted patterns are more likely than others. To address this, we propose a probabilistic framework that models these effects. We denote predicted patterns with $C_i \in \boldsymbol{C}$ and the boolean vector representing the users' selection as S. Finding a probabilistic ranking of the predicted patterns is the same as determining the conditional probability $P(C \mid S)$ for each pattern. Framing the problem using probabilities also gives us more interpretability as it relates the different intents to one another. The probabilities for each intent add up to one:

$$\sum_{C_i \in \boldsymbol{C}} P(C_i \mid S) = P(C_{\text{Outliers}} \mid S) + P(C_{\text{Clusters}} \mid S) + \ldots = 1$$

To compute $P(C_i \mid S)$ we can use Bayes' theorem.

$$P(C_i \mid S) = \frac{P(S \mid C_i) P(C_i)}{P(S)}$$

$P(S \mid C_i)$ models how a particular intent explains the current selection of the user. It is scaled by the prior term $P(C_i)$ that describes the probability of each intent without considering additional information. Later, we will see how we can use this prior to capture domain knowledge about intents. Finally, $P(S)$ acts as a normalizing constant that ensures that the result is a probability. To make this equation computationally tractable, we make use of two observations. First, if we do not consider the order of selections, the problem that we are trying to solve is very similar to text classification. Our description of the users' selection is almost identical to a *bag-of-words* model, which is often used in this domain. The difference is that typically in text classification, the bag-of-words model describes the frequency of each word. In our method, this simplifies to a constant frequency of one if a point is part of the selection. Second, by assuming that each feature (selected point) is independent of another, we can compute $P(S \mid C_i)$ using the *naive Bayes* method. In particular, we use a *multinomial naive Bayes* classifier to compute the conditional probabilities: For each user's selection, we train such a classifier on the output

21: The Jaccard index $J(S, C)$ between selection S and pattern C is defined as:

$$J(S, C) = \frac{|S \cap C|}{|S \cup C|} = \frac{|S \cap C|}{|S| + |C| - |S \cap C|}.$$

When $J(S, C) = 1$, it signals a perfect match, while a value of 0 indicates no overlap.

vector of each of the intents C_i. Given a selection S as input, the classifier yields the corresponding probability. Our prediction is then the intent that maximizes this probability. Sometimes selected points are not part of any of the training samples, which leads to zero probabilities for the intents. This is a common problem when using naive Bayes classifiers — we use *Laplacian smoothing* to avoid this effect.

We believe that the work presented in this section can form the foundation of many future projects. For this, we would first have to further evaluate the effectiveness of our method[22]. Immediate next steps are the application to different visualization techniques and data types. Other prospects include learning from interactions and integrating the output of interactions in visualizations into computational workflows. Because we use Bayesian methods, our method readily allows this by adapting the priors accordingly.

22: We performed initial experiments using crowd-sourcing, which are described in [Gad+21]. Further studies should help better understand the cases in which our method works well or fails and what benefits it can bring to the user's workflow.

6.3 Visual Abstractions

We now want to shift our focus away from Bayesian methods and look at another method that we think can be very useful for visualizing uncertainty. Artists often use visual abstractions to capture scenes of the real world. Their goal is only to capture the relevant details while omitting all the unnecessary information. Generating such abstraction has also extensively been researched in computer graphics — often in the context of non-photorealistic rendering. Visual abstraction has also been identified as one of the top visualization research problems by Johnson [Joh04]. Viola et al. [VI18] provide an extensive survey of current methods and their advantages. They conclude that a promising opportunity for abstraction is visualizing uncertainty. Their argument is that a representation can be more general when we omit details. The new representation can then act as a proxy for a variety of similar, concrete instances. Additionally, when we omit details, we often reclaim some of the space as we have fewer visual primitives to draw. Similar to bubble treemaps, we can use this new-found space to encode additional information about the uncertainty. Figure 6.6 shows the result of our method applied to the elevation data of Australia.

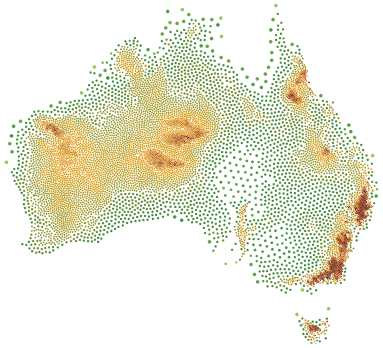


Figure 6.6: Stippled elevation data of Australia. Our method uses less space than traditional methods, such as elevation maps. This reclaimed space could then be used to encode additional information about the uncertainty.

When it comes to the visual representation of univariate data, we usually think of bar charts, line charts, or possibly dot plots, where abstraction is often achieved through binning or smoothing. Although binning reduces the amount of visualized data, task performance for continuous geospatial data does not necessarily decrease [Pad+17]. However, it has been shown that dot representations significantly outperform contour representations for memorization tasks [TSD09]. In the following section, we want to summarize our work on generating stippled abstraction of two-dimensional scalar fields and show how this technique can help visualize uncertainty.

Stippling of 2D Scalar Fields

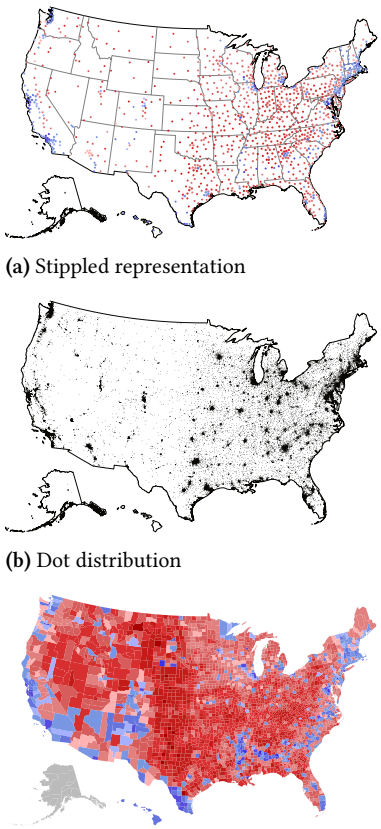
Stippling is a form of abstraction, where stipples (dots) are carefully distributed to approximate shading. In that sense, stippling of scalar fields is

This section is an adapted version of our article with the same name [Gör+19].

a controlled simplification or non-linear smoothing of the dataset. Traditional algorithms for stippling are hard to steer and therefore challenging to use in a visualization context. We aim to close this gap by modifying a technique from computer graphics to support binned and continuous data simultaneously.

Stippling has also been explored in the visualization community. Biological cell maps have been represented using stipples [Mey+10] and share visual similarities to our visualization. However, the point positions for these maps are part of the input and do not need to be sampled. Lu et at. [Lu+02; Lu+03] propose a stipple and feature enhancement method for volume rendering. Stippling has also been used to visualize brain fibers [Gol+11] using diffusion tractograms. These approaches sample stipples in predefined cells of a grid, whereas we use an algorithm based on adaptively changing Voronoi cells.

Generating random point sets with almost uniform point-to-point distances is necessary for sampling, halftoning, remeshing, and artistic rendering methods such as stippling [Mar+17]. Lloyd's method [Llo82] is one widely used optimization method based on Voronoi diagrams to generate such point sets. For a given point set, it iteratively moves each point to the centroid of its corresponding Voronoi cell until the points are distributed equally. The method has been extended to create different point densities, for example, based on underlying image information [Sec02]. Stippling with varying point sizes has also been explored [Deu+01]. A recent stippling method is also based on Voronoi diagrams: Weighted Linde-Buzo-Gray (LBG) stippling [DSZ17] provides more intuitive parameters to control the final result and can also handle variable point sizes. Some stippling algorithms have been proposed to recreate or introduce controlled patterns [Kim+08; LM11]. However, these algorithms focus on specific artistic effects and are limited in their applicability for other purposes. We generalize the algorithm of Deussen et al. [DSZ17] to work on scalar fields and extend it to encode additional structure in the results. Figure 6.7 shows our method applied to election data in the US.



(a) Stippled representation

(b) Dot distribution

(c) Choropleth representation

Figure 6.7: Stipple illustration of election data from the United States (a) that combines the population density as the distribution of stipples (b), and the vote difference for each administrative region (c). The stippled abstraction visualizes both aspects of the data simultaneously. In contrast to dot distributions, our method does not suffer from over-plotting.

Overview

Given 2D data, we aim to find a stippled representation of the distribution of its values. To achieve this, we adapt the Linde-Buzo-Gray (LBG) stippling algorithm [DSZ17] to continuous and discrete scalar fields. The algorithm is based on Voronoi diagrams: By iteratively moving each stipple to the centroid of its Voronoi cell, global point-to-point distances become more and more equalized. In addition to merely moving the points, the LBG algorithm also splits and merges cells based on the corresponding density function.

Figure 6.8 provides an overview of the individual steps of our approach. First, the input data is transformed to the target density using a mapping function. This density allows us to establish a relation between the data that falls into the region of a stipple — its Voronoi cell — and the conceptual amount of ink that a stipple carries. This is analogous to assigning a color to each value when using color maps. Additionally, non-linear

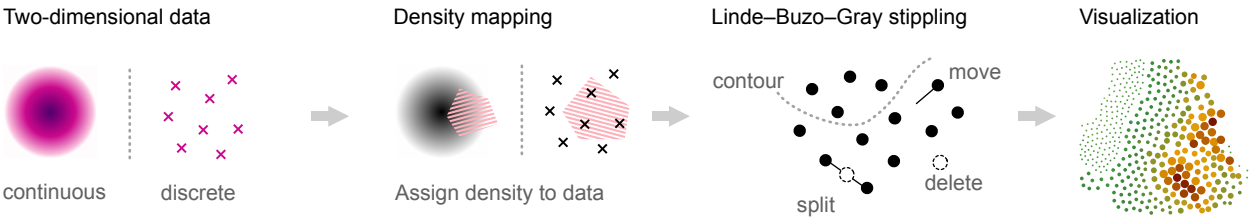


Figure 6.8: Overview of our technique. Our approach can handle scalar fields as well as discrete points as input. We use a modified version of the Linde-Buzo-Gray stippling algorithm that provides more control over the final appearance. By locally modifying the stipple distribution, we can encode additional information such as contour lines into the final representation. Our method encodes distributions of the underlying data into corresponding distributions of stipples, leaving other visual variables, such as color, free for encoding additional attributes.

mapping functions can highlight the relevant information. For discrete input points in 2D, this mapping determines how multiple data points are aggregated. Of course, the concrete mapping function depends on the particular data at hand.

We apply our modified version of the LBG stippling algorithm to compute the final representation. The resulting point sizes can either be determined adaptively or directly specified by the user. The final stippled result reflects the underlying data with the possibility to distinguish density changes locally. Moreover, it can display contours to provide an overall impression of the data simultaneously.

Stippling

The Linde-Buzo-Gray stippling algorithm has mainly been described in the context of computer graphics, where it is used to abstract images or for resampling meshes. In the following, we want to generalize this method to scalar fields by means of formalization. Generally, 2D data can be discrete or continuous[23] and potentially comprise a high dynamic range. Therefore, we define a 2D scalar field as a function $\phi : \mathbb{R}^2 \to \mathbb{R}$. Similar to color scales, we require a mapping $\rho : \mathbb{R} \to [0, 1]$ to transform the co-domain of the scalar field to unit range. We call this the derived scalar field Φ:

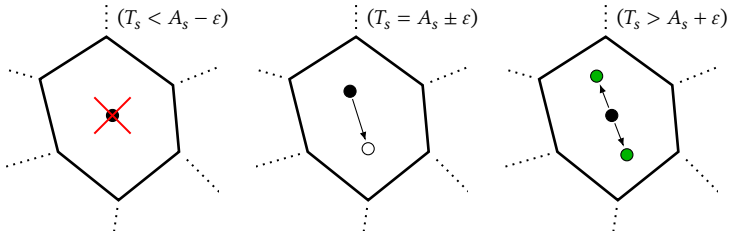
$$\Phi : \mathbb{R}^2 \to [0, 1], \quad \Phi = \rho \circ \phi$$

In NPR, different rendering primitives and shapes have been proposed for stippling [Dal+06; HHD03]. Theoretically, our proposed method can handle arbitrary convex shapes, but we will use circles with a given extent and position for now. The goal of the LBG stippling algorithm is to arrange stipples according to a given measure, in our case, the values of the transformed scalar field Φ. The algorithm starts with a random initial distribution of stipples. Then, it evaluates how well each stipple represents its proximity in Φ. The neighborhood of each stipple $s \in S$ is found by computing the Voronoi diagram over the complete set of stipples.

Conceptually, we want to relate the required amount of ink for a stipple to the values of the scalar field in its vicinity. For continuous scalar fields, we achieve this by integrating over its corresponding Voronoi cell V_s. The target density for a stipple T_s is therefore given by:

$$T_s = \iint_{V_s} \Phi(x, y)\, dA$$

23: In practice, as a result of simulations, scalar fields are often only defined for a fixed, regular grid of points — in which case the domain is $\mathbb{N}^2$, indexing into this grid.



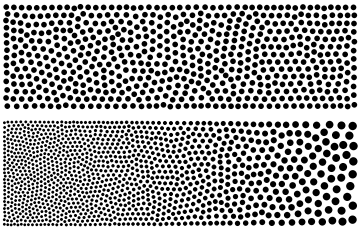
(a) Delete **(b)** Move **(c)** Split

Figure 6.9: Schematic illustration of the different steps of the LBG stippling algorithm. Based on the comparison between the density contained in each cell and the area of the stipple, cells are either deleted, moved to the centroid of the cell, or split.

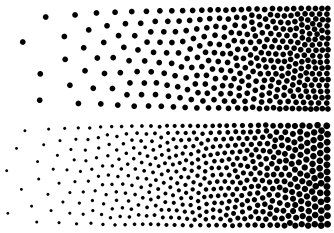
In the case of discrete scalar fields and uniform grids, this is the same as accumulating the overall density points or grid cells that belong to the associated Voronoi cell:

$$T_s = \sum_{i \in V_s} \Phi(i)$$

According to the Linde-Buzo-Gray algorithm, we compare T_s with the area occupied by the stipple A_s. More precisely, with the area-weighted by the values of the scalar field. Three different cases can occur, as shown in Figure 6.9: If a stipple represents the target density reasonably well, up to a specified error threshold ε ($T_s \in [A_s - \varepsilon, A_s + \varepsilon]$), a relaxation step is performed. This moves the stipple toward the weighted centroid of its Voronoi cell (Lloyd relaxation). In the other two cases the stipple is either split ($T_s > A_s + \varepsilon$) into two separate cells or deleted ($T_s < A_s - \varepsilon$). This procedure is repeated iteratively until no more splits and merges occur, and the algorithm converges. The threshold ε is slightly increased with each iteration to increase the convergence rate. This marginally affects the resulting quality in later steps, where very few changes are performed, but guarantees convergence due to the steadily increasing allowed error.



(a) constant density



(b) varying density

Figure 6.10: (a) Stipples distributed with a constant density and either constant point size or variable size. (b) Increasing density, again with constant point size and variable size.

The main parameter of the algorithm is the stipple size — the required number of stipples is automatically determined. Choosing a smaller size results in more points and thus preserves more details, whereas a larger size yields a more abstract representation with fewer points. Because the split and merge operations compare the contained density of a cell to the density represented by the generating stipple, the stipple size can vary locally. It can be set according to its position, attributes of the scalar field, or directly provided by the user. The original LBG stippling algorithm supports variable point sizes by adjusting the size of a stipple (within a provided range) according to T_s. This, however, does not enforce the point size, and there are no guarantees that all sizes within the given range are reached. Our proposed method changes this step of the algorithm: we set the point size according to the scalar field and then compare its area to the target density.

Through this modification, information from the scalar field can be encoded independently in the point size as well as the point density. Figure 6.10a shows how a constant density can be established using constant and variable point sizes. Varying the point size but keeping the density constant does not create a constant perception (the density of the left looks different compared to the right), even though the density is still accurately reflected. Figure 6.10b shows a density gradient stippled using constant and variable point sizes. Increasing the point size can be used to boost the perception of the density. It is important to note that these two channels are not independent: For a given density, we might have only a

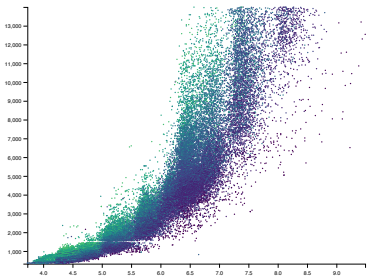
subset of the preferred point sizes and we restrict the number of required points by enforcing a certain point size. Therefore, it is not readily possible to use these two channels to encode two independent variables of the data[24].

24: Our original article [Gör+19] provides more details on the implementation of the method.

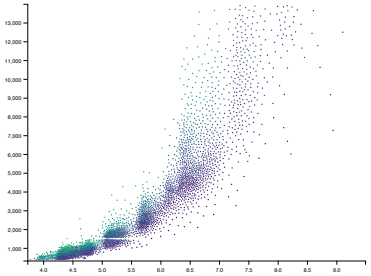
Usefulness for Uncertainty Visualization

Traditional scatter plots often suffer from over-plotting: many points are plotted on top of each other in dense regions, obscuring the underlying distribution. A standard technique to tackle this problem employs kernel density estimation or transforms an existing continuous data domain, such as velocity, to the image domain [BW08]. The resulting density field is then colored using a transfer function that conveys an impression of depth and opacity. However, this approach has a significant drawback: Areas with low point density suffer from low contrast, making it challenging to spot outliers.

Our proposed technique can be used to avoid both of these problems by sampling the data points. Figure 6.11a shows a traditional scatter plot of diamond properties, namely *width* and *price*. Color is used to encode the *clarity*. In this example, the representation suffers from a lot of overplotting in the lower regions. The stippled representation behaves differently, as can be seen in Figure 6.11b. Please note how our technique reveals clusters in the data which were concealed by visual clutter in the traditional scatter plot. With our method, we were able to reduce the number of points from 51,772 points in the scatter plot down to 3,389 points. Admittedly, the points in the stippled scatter plot do not necessarily fall onto points from the original dataset. However, other methods from information visualization, such as kernel density estimation, make the same compromise. Nevertheless, each point is representative of the underlying distribution. Recently, *core sets* have been applied to information visualization by Zheng et a. [Zhe+19]. Similarly, a possible extension for our method could be to incorporate actual data locations into our algorithm. Sparse regions could use the actual data positions, whereas regions with higher density could maintain the intended stipple distribution, and everything in-between could use a compromise of both. This presents a tradeoff between the accuracy of individual points and their overall distribution.



(a) Scatter plot



(b) Stippled scatter plot

Figure 6.11: Visualization of the diamond dataset, showing *price* over *width*. Color represents *clarity* of each diamond. In (a), the dataset is depicted using a regular scatter plot with all 51,772 data points, while (b) shows the result of our stippling technique, reducing the number of points to 3,389, showing the actual distribution more clearly.

7 Conclusion

In this thesis, we presented different methods to handle uncertainty in information visualization, aimed at answering the research questions, **RQ1** and **RQ2**[1], from Section 1.1. While uncertainty is omnipresent, it is often not communicated to the user or — arguably worse — omitted in the early stages of the visualization pipeline. If we recall the pipeline, there are four distinct steps: Data analysis, filtering, mapping, and rendering [HM90]. The methods presented in the main part of this thesis touch upon the first three stages.

For the data analysis step, we showed in Chapter 3 how to propagate uncertainty information in the form of standard errors through hierarchies (**RQ2**). In many hierarchical datasets, uncertainty information is only present in the leaves, as the intermediate nodes of the hierarchy are often categories that do not have associated measurements. Therefore, we need to estimate the uncertainty for each of the inner nodes to allocate the required space in the final representations. In the context of uncertainty-aware PCA (Chapter 5), we devised a method that allows storing aggregates of measurements in the form of probability distributions while retaining the important information that is required to perform dimensionality reduction faithfully (**RQ2**). We showed that it suffices to capture the mean and the covariance of each of the categories of points, which can greatly reduce the amount of data that needs to be stored. Moving on to the second stage of the pipeline, the filtering step, where parts of our preliminary work from Section 6.3 contribute to. We proposed a method to generate abstractions of 2D scalar fields. By perceptually filtering out information, we can make patterns visible that would otherwise be hidden behind overplotting — we demonstrated this for scatterplots (**RQ1**). Ultimately, leaving out information can also make room for additional attributes such as the uncertainty for each region in the representation (**RQ1**). Investigating proper ways to do this is left for future work.

Finally, all presented methods impact the visualization pipeline's mapping stage, which transforms data to visual primitives. For bubble treemaps (Chapter 3), this is done in the layout by allocating additional space to encode uncertainty information geometrically into node contours. We also devised new mappings in the form of visual variables that encode uncertainty (**RQ1**). Our technique to layout probabilistic networks maps the uncertainty to possible node positions in the embedding (**RQ1**). The resulting distributions over node positions are visualized using kernel density estimation. To emphasize the characteristics of the distributions, we cluster the nodes and compute their α-shapes to show their outlines (**RQ1**). In our work on uncertainty-aware PCA, we find a way to perform dimensionality reduction directly on probability distributions (**RQ2**), projecting potentially high-dimensional multivariate distributions onto a lower-dimensional subspace. We also investigate the influence that the uncertainty has on the final projection by generalizing factor maps to factor traces. This allows the user to understand the influence of the uncertainty on the computed projection (**RQ1**).

1: The two research questions are:

RQ1 How can we communicate uncertainty with its statistical properties?

RQ2 How to adapt visualization methods to uncertainty?

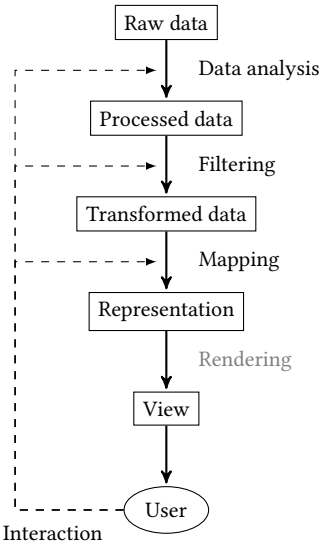


Figure 7.1: Overview of the visualization pipeline with the stages relevant for this thesis.

The methods presented in this thesis are by no means exhaustive and should be seen as steps towards a more holistic approach to dealing with uncertainty in information visualization. Even if new techniques are developed to better handle uncertainty, in the end, visualization is not a means by itself. The goal of visual representations is always to communicate a message to the viewer.

Therefore, we need to very carefully consider how they receive this message. Many examples for ambiguous visualizations can be found in our day-to-day life, for instance, in the media[2]. Or even worse, visualizations can deliberately point the user in the wrong direction or change an argument. If used improperly, for example, with wrong parameters or incorrect assumptions, it is not difficult to manipulate facts using statistics. The same is true for visualizations. Correll et al. [CH17] pick up the message analogy and coin this a visualization as man-in-the-middle attack, a terminology the originates in the area of cybersecurity. This problem exists for all kinds of visualization, one of the described ways to mitigate it is by shifting the burden onto the user, asking for visualization literacy [BBG19][3]. We are not sure if offloading the responsibility onto the user is the right approach, as the problem becomes even worse if we additionally communicate uncertain information. The work by Hullman [Hul20] has shown that communicating uncertainty can alienate users and decrease trust in point the authors of visualization try to communicate. We believe that a big part of this is that the concept of uncertainty very unintuitive the understand for people. The lack of knowledge about probability distributions certainly plays a role in this, as our hunches about the outcomes of probabilistic processes are usually wrong. This has also extensively been shown in behavioral sciences, which has recently been made accessible to a wider audience by authors such as Kahneman [Kah12], and Taleb [Tal05]. We believe sensitivity analysis approaches play a significant role in understanding the implications of uncertainty. Not only do they help to build trust in the visualization, but they also support exploratory analysis. We showed examples of this in Chapters 4 and 5 — but these methods form just the tip of the iceberg.

In Chapter 6 we proposed another set of potential approaches to incorporate the uncertainties that arise from user interactions. Modeling user interactions using a probabilistic framework allows the visualization system to "reason" about what the user tried to achieve with their actions. Another strength of the probabilistic approach is that it allows them to encode their assumptions or beliefs flexibly. We believe that this is an exciting and very important ongoing area of research.

2: Typical examples of this are choropleth maps for election results, which disproportionately emphasize regions with low population density.

3: The main idea is that users should be taught or develop intuitions to better understand and interpret visualizations.

Bibliography

[AY09] C. C. Aggarwal and P. S. Yu. „A survey of uncertain data algorithms and applications“. In: *IEEE Transactions on Knowledge and Data Engineering* 21.5 (2009), pp. 609–623. DOI: 10.1109/tkde.2008.190 (see p. 15).

[AO03] J.-H. Ahn and J.-H. Oh. „A constrained EM algorithm for principal component analysis“. In: *Neural Computation* 15.1 (2003), pp. 57–65. DOI: 10.1162/089976603321043694 (see p. 52).

[AM11] G. Aisch and D. McCandless. *Bubble tree (Open Knowledge Foundation)*. Last Accessed: 2021-07-15. 2011. URL: http://okfnlabs.org/bubbletree/ (see p. 18).

[Aub+13] D. Auber et al. „GosperMap: Using a Gosper curve for laying out hierarchical data“. In: *IEEE Transactions on Visualization and Computer Graphics* 19.11 (2013), pp. 1820–1832. DOI: 10.1109/tvcg.2013.91 (see p. 18).

[BW08] S. Bachthaler and D. Weiskopf. „Continuous scatterplots“. In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (2008), pp. 1428–1435. DOI: 10.1109/TVCG.2008.119 (see p. 85).

[BDL05] M. Balzer, O. Deussen, and C. Lewerentz. „Voronoi treemaps for the visualization of software metrics“. In: *Proceedings of the ACM Symposium on Software Visualization*. 2005, pp. 165–172. DOI: 10.1145/1056018.1056041 (see p. 18).

[Bec+16] F. Beck et al. „A taxonomy and survey of dynamic graph visualization“. In: *Computer Graphics Forum* 36.1 (2016), pp. 133–159. DOI: 10.1111/cgf.12791 (see p. 36).

[Ber+11] W. Berger et al. „Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction“. In: *Computer Graphics Forum* 30.3 (2011), pp. 911–920. DOI: 10.1111/j.1467-8659.2011.01940.x (see p. 15).

[Ber83] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983 (see pp. 15, 24).

[Bio15] G. Biondini. „An introduction to rare event simulation and importance sampling“. In: *Handbook of Statistics*. Elsevier, 2015, pp. 29–68. DOI: 10.1016/b978-0-444-63492-4.00002-2 (see p. 70).

[Bis99] C. Bishop. „Bayesian PCA“. In: *Advances in Neural Information Processing Systems*. Vol. 11. MIT Press, 1999, pp. 382–388 (see pp. 52, 53).

[Bla03] J. L. Blaha. *Standard Errors in the Consumer Expenditure Survey*. Last Accessed: 2021-07-15. 2003. URL: https://www.bls.gov/cex/anthology/csxanth5.pdf (see p. 29).

[Bod+16] S. Bodenstedt et al. „Superpixel-based structure classification for laparoscopic surgery“. In: *Medical Imaging, SPIE*. 17. 2016. DOI: 10.1117/12.2216750 (see p. 6).

[Bol+12] P. Boldi et al. „Injecting uncertainty in graphs for identity obfuscation“. In: *Proceedings of the VLDB Endowment* 5.11 (2012), pp. 1376–1387. DOI: 10.14778/2350229.2350254 (see p. 35).

[Bol75] K. M. Bolton. „Biarc curves“. In: *Computer-Aided Design* 7.2 (1975), pp. 89–92. DOI: 10.1016/0010-4485(75)90086-x (see p. 21).

[BBG19] K. Börner, A. Bueckle, and M. Ginda. „Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments“. In: *Proceedings of the National Academy of Sciences (PNAS)* 116.6 (2019), pp. 1857–1864. DOI: 10.1073/pnas.1807180116 (see p. 88).

[BKS01] S. Borzsony, D. Kossmann, and K. Stocker. „The skyline operator“. In: *Proceedings 17th International Conference on Data Engineering*. IEEE, 2001. DOI: 10.1109/icde.2001.914855 (see p. 79).

[Bos17] M. Bostock. *Treemap*. Last Accessed: 2021-07-15. Mar. 2017. URL: https://bl.ocks.org/mbostock/4063582 (see p. 26).

[Bou+12] N. Boukhelifa et al. „Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty“. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2769–2778. DOI: 10.1109/tvcg.2012.220 (see pp. 3, 15, 24, 25, 29).

[BM12] U. Brandes and M. Mader. „A quantitative comparison of stress-minimization approaches for offline dynamic graph drawing“. In: *Graph Drawing, 19th International Symposium*. 2012, pp. 99–110. DOI: 10.1007/978-3-642-25878-7_11 (see pp. 35, 38).

[BP06] U. Brandes and C. Pich. „Eigensolver methods for progressive multidimensional scaling of large data“. In: *Graph Drawing, 14th International Symposium*. 2006, pp. 42–53. DOI: 10.1007/978-3-540-70904-6_6 (see p. 38).

[BP08] U. Brandes and C. Pich. „An experimental study on distance-based graph drawing“. In: *Graph Drawing, 16th International Symposium*. 2008, pp. 218–229. DOI: 10.1007/978-3-642-00219-9_21 (see pp. 35, 38).

[BOL12] K. Brodlie, R. A. Osorio, and A. Lopes. „A review of uncertainty in data visualization“. In: *Expanding the Frontiers of Visual Analytics and Visualization*. 2012, pp. 81–109. DOI: 10.1007/978-1-4471-2804-5_6 (see p. 15).

[BHvW00] M. Bruls, K. Huizing, and J. J. van Wijk. „Squarified treemaps“. In: *Proceedings of the Joint EUROGRAPHICS and IEEE TCVG Symposium on Visualization*. 2000, pp. 33–42. DOI: 10.1007/978-3-7091-6783-0_4 (see p. 18).

[Bur09] C. J. C. Burges. „Dimension reduction: A guided tour“. In: *Foundation and Trends in Machine Learning* 2.4 (2009), pp. 275–365. DOI: 10.1561/2200000002 (see p. 52).

[Car+17] B. Carpenter et al. „Stan: A probabilistic programming language“. In: *Journal of Statistical Software* 76.1 (2017). DOI: 10.18637/jss.v076.i01 (see p. 69).

[Car12] S. Carter. *Four ways to slice Obama's 2013 budget proposal*. Accessed: 2017-03-24. Feb. 2012. URL: https://www.nytimes.com/interactive/2012/02/13/us/politics/2013-budget-proposal-graphic.html (see p. 19).

[CWM09] J. Chuang, D. Weiskopf, and T. Möller. „Hue-preserving color blending“. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1275–1282. DOI: 10.1109/TVCG.2009.150 (see p. 41).

[CPC09] C. Collins, G. Penn, and S. Carpendale. „Bubble sets: Revealing set relations with isocontours over existing visualizations“. In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1009–1016. DOI: 10.1109/TVCG.2009.122 (see pp. 19, 32, 42).

[CS03] C. R. Collins and K. Stephenson. „A circle packing algorithm“. In: *Computational Geometry* 25.3 (2003), pp. 233–256. DOI: 10.1016/S0925-7721(02)00099-8 (see p. 18).

[CCM09] C. Correa, Y.-H. Chan, and K.-L. Ma. „A framework for uncertainty-aware visual analytics“. In: *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*. 2009, pp. 51–58. DOI: 10.1109/VAST.2009.5332611 (see p. 15).

[CG14] M. Correll and M. Gleicher. „Error bars considered harmful: Exploring alternate encodings for mean and error“. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2142–2151. DOI: 10.1109/TVCG.2014.2346298 (see p. 25).

[CH17] M. Correll and J. Heer. „Black hat visualization“. In: *Workshop on Dealing with Cognitive Biases in Visualisations (DECISIVe), IEEE VIS*. 2017 (see p. 88).

[CMH18] M. Correll, D. Moritz, and J. Heer. „Value-suppressing uncertainty palettes“. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, pp. 1–11. DOI: 10.1145/3173574.3174216 (see p. 2).

[CG15] J. P. Cunningham and Z. Ghahramani. „Linear dimensionality reduction: survey, insights, and generalizations“. In: *Journal of Machine Learning Research* 16.1 (2015), pp. 2859–2900 (see pp. 51, 53).

[Dal+06] K. Dalal et al. „A spectral approach to NPR packing“. In: *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*. 2006, pp. 71–78. doi: 10.1145/1124728.1124741 (see p. 83).

[Del34] B. Delaunay. „Sur la sphère vide. A la mémoire de Georges Voronoï“. In: *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* 7 (1934), pp. 793–800 (see pp. 41, 42).

[DM04] T. Denoeux and M.-H. Masson. „Principal component analysis of fuzzy data using autoassociative neural networks“. In: *IEEE Transactions on Fuzzy Systems* 12.3 (2004), pp. 336–349. doi: 10.1109/tfuzz.2004.825990 (see pp. 53, 62).

[DSZ17] O. Deussen, M. Spicker, and Q. Zheng. „Weighted Linde–Buzo–Gray stippling“. In: *ACM Transactions on Graphics* 36.6 (2017), 233:1–233:12. doi: 10.1145/3130800.3130819 (see pp. 4, 82).

[Deu+01] O. Deussen et al. „Floating points: a method for computing stipple drawings“. In: *Computer Graphics Forum* 19.3 (2001), pp. 41–50. doi: 10.1111/1467-8659.00396 (see p. 82).

[Din+12] K. Dinkla et al. „Kelp diagrams: Point set membership visualization“. In: *Computer Graphics Forum* 31.3 (2012), pp. 875–884. doi: 10.1111/j.1467-8659.2012.03080.x (see p. 19).

[DJ14] S. Dray and J. Josse. „Principal component analysis with missing values: A comparative survey of methods“. In: *Plant Ecology* 216.5 (2014), pp. 657–667. doi: 10.1007/s11258-014-0406-z (see p. 68).

[DG17] D. Dua and C. Graff. *UCI Machine Learning Repository*. Last Accessed: 2021-07-15. 2017. url: http://archive.ics.uci.edu/ml (see p. 63).

[Dut+17] S. Dutta et al. „In-situ distribution guided analysis and visualization of transonic jet engine simulations“. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 811–820. doi: 10.1109/tvcg.2016.2598604 (see p. 67).

[End+14] A. Endert et al. „The human is the loop: new directions for visual analytics“. In: *Journal of Intelligent Information Systems* 43.3 (2014), pp. 411–435. doi: 10.1007/s10844-014-0304-9 (see p. 72).

[ER59] P. Erdös and A. Rényi. „On random graphs“. In: *Publicationes Mathematicae* 6 (1959), pp. 290–297. doi: 10.2307/1999405 (see p. 35).

[Est+96] M. Ester et al. „A density-based algorithm for discovering clusters in large spatial databases with noise“. In: *Second International Conference on Knowledge Discovery and Data Mining* (1996), pp. 226–231. doi: 10.1.1.71.1980 (see p. 42).

[FGS19] R. Faust, D. Glickenstein, and C. Scheidegger. „DimReader: Axis lines that explain non-linear projections“. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 481–490. doi: 10.1109/TVCG.2018.2865194 (see p. 53).

[Fen+10] D. Feng et al. „Matching visual saliency to confidence in plots of uncertain data“. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 980–989. doi: 10.1109/TVCG.2010.176 (see p. 15).

[FIN17] FINVIZ.com. *Stock market bubbles*. Last Accessed: 2021-07-15. 2017. url: http://finviz.com/bubbles.ashx (see p. 19).

[Gad+21] K. Gadhave et al. „Capturing user intent when brushing in scatterplots“. In: *OSF Preprints (Submitted to SIGCHI)* (2021). doi: 10.31219/osf.io/mq2rk (see pp. 5, 72, 78, 81).

[Gai17] M. Gaillard. *CERN data centre passes the 200-petabyte milestone*. https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone. [Online; accessed 22. Mar. 2019]. 2017 (see p. 66).

[GHK09] E. R. Gansner, Y. Hu, and S. G. Kobourov. „GMap: Drawing graphs as maps“. In: *Graph Drawing, 17th International Symposium*. 2009, pp. 405–407. doi: 10.1007/978-3-642-11805-0_38 (see pp. 35, 41).

[GKN04] E. R. Gansner, Y. Koren, and S. C. North. „Graph drawing by stress majorization“. In: *Graph Drawing, 12th International Symposium*. 2004, pp. 239–250. DOI: 10.1007/978-3-540-31843-9_25 (see pp. 35, 38).

[GK06] P. Giordani and H. A. L. Kiers. „A comparison of three methods for principal component analysis of fuzzy interval data“. In: *Computational Statistics & Data Analysis* 51.1 (2006), pp. 379–397. DOI: 10.1016/j.csda.2006.02.019 (see p. 53).

[Gol+11] M. Goldau et al. „Fiber stippling: An illustrative rendering for probabilistic diffusion tractography“. In: *2011 IEEE Symposium on Biological Data Visualization (BioVis)*. 2011, pp. 23–30. DOI: 10.1109/biovis.2011.6094044 (see p. 82).

[GKD19] J. Görtler, R. Kehlbeck, and O. Deussen. „A visual exploration of Gaussian processes“. In: *Distill* (2019). DOI: 10.23915/distill.00017 (see pp. 5, 72).

[Gör+18] J. Görtler et al. „Bubble treemaps for uncertainty visualization“. In: *IEEE Transactions on Visualization and Computer Graphics* 24 (1 2018), pp. 719–728. DOI: 10.1109/TVCG.2017.2743959 (see p. 4).

[Gör+19] J. Görtler et al. „Stippling of 2D scalar fields“. In: *IEEE Transactions on Visualization and Computer Graphics* 25 (6 2019), pp. 2193–2204. DOI: 10.1109/TVCG.2019.2903945 (see pp. 4, 81, 85).

[Gör+20] J. Görtler et al. „Uncertainty-aware principal component analysis“. In: *IEEE Transactions on Visualization and Computer Graphics* 26 (1 2020), pp. 822–831. DOI: 10.1109/TVCG.2019.2934812 (see p. 5).

[Gsc+16] T. Gschwandtner et al. „Visual encodings of temporal uncertainty: A comparative user study“. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 539–548. DOI: 10.1109/TVCG.2015.2467752 (see p. 15).

[GHL15] H. Guo, J. Huang, and D. H. Laidlaw. „Representing uncertainty in graph edges: An evaluation of paired visual variables“. In: *IEEE Transactions on Visualization and Computer Graphics* 21.10 (2015), pp. 1173–1186. DOI: 10.1109/TVCG.2015.2424872 (see pp. 15, 33, 36).

[HO98] D. Halperin and M. H. Overmars. „Spheres, molecules, and hidden surface removal“. In: *Computational Geometry* 11.2 (1998), pp. 83–102. DOI: 10.1016/s0925-7721(98)00023-6 (see p. 19).

[HM90] R. B. Harber and D. A. McNabb. „Visualization idioms: A conceptual model for scientific visualization systems“. In: *Visualization in Scientific Computing*. IEEE Computer Society, 1990 (see p. 87).

[HB03] M. Harrower and C. A. Brewer. „ColorBrewer.org: An online tool for selecting colour schemes for maps“. In: *The Cartographic Journal* 40.1 (2003), pp. 27–37. DOI: 10.1179/00087040323500 2042 (see p. 41).

[Haz94] M. Hazewinkel, ed. *Covariance Matrix (Encyclopedia of Mathematics)*. Encyclopedia of Mathematics: Springer Science+Business Media B.V. / Kluwer Academic Publishers, 1994 (see p. 11).

[Her15] R. Hermida. „The problem of allowing correlated errors in structural equation modeling: concerns and considerations“. In: *Computational Methods in Social Sciences* 3.1 (2015), pp. 05–17 (see p. 53).

[HHD03] S. Hiller, H. Hellwig, and O. Deussen. „Beyond stippling — methods for distributing objects on the plane“. In: *Computer Graphics Forum* 22.3 (2003), pp. 515–522. DOI: 10.1111/1467-8659.00699 (see p. 83).

[HS91] G. E. Hinton and T. Shallice. „Lesioning an attractor network: investigations of acquired dyslexia“. In: *Psychological Review* 98.1 (1991), pp. 74–95. DOI: 10.1037/0033-295X.98.1.74 (see p. 53).

[HBW15] M. Hlawatsch, M. Burch, and D. Weiskopf. „Visual analysis of eye movements by hierarchical filter wheels“. In: *Proceedings of the 19th International Conference on Information Visualisation*. 2015, pp. 107–113. DOI: 10.1109/iV.2015.29 (see p. 18).

[HBW14] M. Hlawatsch, M. Burch, and D. Weiskopf. „Bubble hierarchies“. In: *Proceedings of the Workshop on Computational Aesthetics*. 2014, pp. 77–80. DOI: 10.1145/2630099.2630107 (see p. 30).

[HRH02] P. D. Hoff, A. E. Raftery, and M. S. Handcock. „Latent space approaches to social network analysis“. In: *Journal of the American Statistical Association* 97.460 (2002), pp. 1090–1098. DOI: 10.1198/016214502388618906 (see p. 35).

[Hol06] D. Holten. „Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data“. In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 741–748. DOI: 10.1109/TVCG.2006.147 (see pp. 36, 40).

[Hu05] Y. Hu. „Efficient, high-quality force-directed graph drawing“. In: *Mathematica Journal* 10.1 (2005), pp. 37–71 (see p. 49).

[Hul16] J. Hullman. „Why evaluating uncertainty visualization is error prone“. In: *Proceedings of the 6th Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*. 2016, pp. 143–151. DOI: 10.1145/2993901.2993919 (see pp. 15, 17).

[Hul20] J. Hullman. „Why authors don't visualize uncertainty“. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 130–139. DOI: 10.1109/tvcg.2019.2934287 (see pp. 2, 88).

[HRA15] J. Hullman, P. Resnick, and E. Adar. „Hypothetical outcome plots outperform error bars and violin plots for inferences about reliability of variable ordering“. In: *PLOS ONE* 10.11 (2015), e0142444. DOI: 10.1371/journal.pone.0142444 (see p. 70).

[HET12] C. Hurter, O. Ersoy, and A. Telea. „Graph bundling by kernel density estimation“. In: *Computer Graphics Forum* 31.3 (2012), pp. 865–874. DOI: 10.1111/j.1467-8659.2012.03079.x (see p. 36).

[HKO01] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley and Sons, 2001 (see p. 53).

[JS91] B. Johnson and B. Shneiderman. „Tree-Maps: A space-filling approach to the visualization of hierarchical information structures“. In: *Proceedings of the 2nd Conference on Visualization*. 1991, pp. 284–291 (see p. 17).

[Joh04] C. R. Johnson. „Top scientific visualization research problems“. In: *IEEE Computer Graphics and Applications* 24.4 (2004), pp. 13–17. DOI: 10.1109/MCG.2004.20 (see p. 81).

[JL09] I. M. Johnstone and A. Y. Lu. „On consistency and sparsity for principal components analysis in high dimensions“. In: *Journal of the American Statistical Association* 104.486 (2009), pp. 682–693. DOI: 10.1198/jasa.2009.0121 (see p. 67).

[Kah12] D. Kahneman. *Thinking, Fast and Slow*. Penguin Books, May 10, 2012. 512 pp. (see p. 88).

[KK89] T. Kamada and S. Kawai. „An algorithm for drawing general undirected graphs“. In: *Information Processing Letters* 31.1 (1989), pp. 7–15. DOI: 10.1016/0020-0190(89)90102-6 (see p. 38).

[Kan00] E. Kandogan. „Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions“. In: *Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*. 2000, pp. 9–12 (see p. 53).

[Kap+10] A. Kapoor et al. „Gaussian processes for object categorization“. In: *International Journal of Computer Vision* 88.2 (2010), pp. 169–188. DOI: 10.1007/s11263-009-0268-3 (see p. 72).

[Kat+13] D. Katić et al. „Context-aware augmented reality in laparoscopic surgery“. In: *Computerized Medical Imaging and Graphics* 37 (2 2013), pp. 174–182. DOI: 10.1016/j.compmedimag.2013.03.003 (see p. 6).

[Kay+16] M. Kay et al. „When (ish) is my bus?: User-centered visualizations of uncertainty in everyday, mobile predictive systems“. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016. DOI: 10.1145/2858036.2858558 (see p. 71).

[Khl+13] R. Khlebnikov et al. „Noise-based volume rendering for the visualization of multivariate volumetric data“. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2926–2935. DOI: 10.1109/TVCG.2013.180 (see p. 15).

[Kim+08] D. Kim et al. „Feature-guided image stippling“. In: *Computer Graphics Forum* 27.4 (2008), pp. 1209–1216. DOI: 10.1111/j.1467-8659.2008.01259.x (see p. 82).

[KL07] H.-C. Kim and J. Lee. „Clustering based on Gaussian processes“. In: *Neural Computation* 19.11 (2007), pp. 3088–3107. DOI: 10.1162/neco.2007.19.11.3088 (see p. 72).

[KD09] A. D. Kiureghian and O. Ditlevsen. „Aleatory or epistemic? Does it matter?“ In: *Structural Safety* 31.2 (2009), pp. 105–112. DOI: 10.1016/j.strusafe.2008.06.020 (see p. 9).

[KY95] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Pearson, 1995 (see p. 9).

[Kni21] F. H. Knight. *Risk, Uncertainty and Profit*. 32nd ed. Houghton Mifflin, 1921 (see p. 9).

[Kob13] S. Kobourov. „Force-directed drawing algorithms“. In: *Handbook of Graph Drawing and Visualization*. CRC Press, 2013, pp. 383–408 (see p. 35).

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques — Adaptive Computation and Machine Learning*. MIT Press, 2009 (see p. 35).

[KC04] Y. Koren and L. Carmel. „Robust linear dimensionality reduction“. In: *IEEE Transactions on Visualization and Computer Graphics* 10.4 (2004), pp. 459–470. DOI: 10.1109/tvcg.2004.17 (see p. 52).

[Kru64] J. B. Kruskal. „Nonmetric multidimensional scaling: A numerical method“. In: *Psychometrika* 29.2 (1964), pp. 115–129 (see p. 38).

[LR71] B. Lee and F. M. Richards. „The interpretation of protein structures: Estimation of static accessibility“. In: *Journal of Molecular Biology* 55.3 (1971), pp. 379–400. DOI: 10.1016/0022-2836(71)90324-x (see p. 19).

[Lee+07] B. Lee et al. „CandidTree: Visualizing structural uncertainty in similar hierarchies“. In: *Human-Computer Interaction*. INTERACT'07. Springer, 2007, pp. 250–263. DOI: 10.1007/978-3-540-74800-7_20 (see p. 36).

[LT16] D. J. Lehmann and H. Theisel. „Optimal sets of projections of high-dimensional data“. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 609–618. DOI: 10.1109/TVCG.2015.2467132 (see p. 53).

[LM11] H. Li and D. Mould. „Structure-preserving stippling by priority-based error diffusion“. In: *Proceedings of Graphics Interface 2011*. 2011, pp. 127–134 (see p. 82).

[LK07] D. Liben-Nowell and J. Kleinberg. „The link-prediction problem for social networks“. In: *Journal of the American Society for Information Science and Technology* 58 (2007), pp. 1019–1031. DOI: 10.1002/asi (see p. 35).

[Liu+17] S. Liu et al. „Visualizing high-dimensional data: Advances in the past decade“. In: *IEEE Transactions on Visualization and Computer Graphics* 23.3 (2017), pp. 1249–1268. DOI: 10.1109/TVCG.2016.2640960 (see p. 53).

[Liu+16] S. Liu et al. „The Grassmannian atlas: a general framework for exploring linear projections of high-dimensional data“. In: *Computer Graphics Forum* 35 (3 2016), pp. 1–10. DOI: 10.1111/cgf.12876 (see p. 53).

[Llo82] S. Lloyd. „Least squares quantization in PCM“. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489 (see p. 82).

[LC87] W. E. Lorensen and H. E. Cline. „Marching cubes: A high resolution 3D surface construction algorithm“. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. 1987, pp. 163–169. DOI: 10.1145/37401.37422 (see p. 19).

[Lu+02] A. Lu et al. „Non-photorealistic volume rendering using stippling techniques“. In: *IEEE Visualization, 2002. VIS*. IEEE, 2002, pp. 211–218. DOI: 10.1109/visual.2002.1183777 (see p. 82).

[Lu+03] A. Lu et al. „Illustrative interactive stipple rendering“. In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 127–138. DOI: 10.1109/tvcg.2003.1196001 (see p. 82).

[Mac+12] A. M. MacEachren et al. „Visual semiotics & uncertainty visualization: An empirical study“. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2496–2505. DOI: 10.1109/TVCG.2012.279 (see pp. 15, 17, 24, 33).

[Mac03] D. J. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003 (see p. 75).

[Mar+17] D. Martín et al. „A survey of digital stippling“. In: *Computers & Graphics* 67 (2017), pp. 24–44. DOI: 10.1016/j.cag.2017.05.001 (see p. 82).

[MR10] M. J. McGuffin and J.-M. Robert. „Quantifying the space-efficiency of 2D graphical representations of trees“. In: *Information Visualization* 9.2 (2010), pp. 115–140. DOI: 10.1057/ivs.2009.4 (see p. 18).

[Meu+13] W. Meulemans et al. „KelpFusion: A hybrid set visualization technique“. In: *IEEE Transactions on Visualization and Computer Graphics* 19.11 (2013), pp. 1846–1858. DOI: 10.1109/tvcg.2013.76 (see pp. 19, 32).

[Mey+10] M. Meyer et al. „MulteeSum: A tool for comparative spatial and temporal gene expression data“. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 908–917. DOI: 10.1109/tvcg.2010.137 (see p. 82).

[MR14a] L. Micallef and P. Rodgers. „eulerAPE: Drawing area-proportional 3-Venn diagrams using ellipses“. In: *PLoS ONE* 9.7 (July 2014), pp. 1–18. DOI: 10.1371/journal.pone.0101717 (see p. 19).

[MR14b] L. Micallef and P. Rodgers. „eulerForce: Force-directed layout for Euler diagrams“. In: *Journal of Visual Languages & Computing* 25.6 (2014), pp. 924–934. DOI: 10.1016/j.jvlc.2014.09.002 (see p. 19).

[MG14] S. Mousavi and G. Gigerenzer. „Risk, uncertainty, and heuristics“. In: *Journal of Business Research* 67.8 (2014), pp. 1671–1678. DOI: 10.1016/j.jbusres.2014.02.013 (see p. 71).

[NSB11] S. Nakajima, M. Sugiyama, and S. D. Babacan. „On Bayesian PCA: Automatic dimensionality selection and analytic solution“. In: *International Conference on Machine Learning*. 2011, pp. 497–504 (see p. 52).

[NOB15] A. Nocaj, M. Ortmann, and U. Brandes. „Untangling the hairballs of multi-centered, small-world online social media networks“. In: *Journal of Graph Algorithms and Applications* 19.2 (2015), pp. 595–618. DOI: 10.7155/jgaa.00370 (see p. 49).

[NA18] L. G. Nonato and M. Aupetit. „Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment“. In: *IEEE Transactions on Visualization and Computer Graphics* 25.8 (2018), pp. 2650–2673. DOI: 10.1109/TVCG.2018.2846735 (see pp. 51, 52).

[Nou+02] M. N. Nounou et al. „Bayesian principal component analysis“. In: *Journal of Chemometrics* 16.11 (2002), pp. 576–595. DOI: 10.1002/cem.759 (see p. 52).

[Pad+17] L. Padilla et al. „Evaluating the impact of binning 2D scalar fields“. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 431–440. DOI: 10.1109/tvcg.2016.2599106 (see p. 81).

[PWL97] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. „Approaches to uncertainty visualization“. In: *The Visual Computer* 13.8 (1997), pp. 370–390. DOI: 10.1007/s003710050111 (see p. 15).

[Par+15] P. Parchas et al. „Uncertain graph processing through representative instances“. In: *ACM Transactions on Database Systems* 40.3 (2015), pp. 1–39. DOI: 10.1145/2818182 (see p. 35).

[Pea01] K. Pearson. „On lines and planes of closest fit to systems of points in space“. In: *Philosophical Magazine* 2 (1901), pp. 559–572 (see p. 52).

[Ped+11] F. Pedregosa et al. „Scikit-learn: Machine learning in Python“. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830 (see p. 69).

[Ras04] C. E. Rasmussen. „Gaussian processes in machine learning“. In: *Advanced Lectures on Machine Learning*. Springer, 2004, pp. 63–71. DOI: 10.1007/978-3-540-28650-9_4 (see p. 72).

[RW05] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Nov. 23, 2005 (see p. 53).

[RD10] N. H. Riche and T. Dwyer. „Untangling Euler diagrams“. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1090–1099. doi: 10.1109/tvcg.2010.210 (see p. 19).

[RJB14] E. Rosén, E. Jansson, and M. Brundin. „Implementation of a fast and efficient concave hull algorithm“. Project report. University of Uppsala, 2014 (see p. 42).

[Saa+19] F. A. Saad et al. „Bayesian synthesis of probabilistic programs for automatic data modeling“. In: *Proceedings of the ACM on Programming Languages* 3.37 (2019), (37):1–32. doi: 10.1145/3290350 (see p. 76).

[Sac+14] D. Sacha et al. „Knowledge generation model for visual analytics“. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1604–1613. doi: 10.1109/TVCG.2014.2346481 (see p. 66).

[San+05] G. Sanguinetti et al. „Accounting for probe-level noise in principal component analysis of microarray data“. In: *Bioinformatics* 21.19 (2005), pp. 3748–3754. doi: 10.1093/bioinformatics/bti617 (see p. 52).

[Sat+17] A. Satyanarayan et al. „Vega-Lite: A grammar of interactive graphics“. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 341–350. doi: 10.1109/tvcg.2016.2599030 (see p. 69).

[SSM97] B. Schölkopf, A. Smola, and K.-R. Müller. „Kernel principal component analysis“. In: *Artificial Neural Networks — ICANN'97*. Springer, 1997, pp. 583–588 (see pp. 52, 76).

[Sch+17] C. Schulz et al. „Probabilistic graph layout for uncertain network visualization“. In: *IEEE Transactions on Visualization and Computer Graphics* 23 (1 2017), pp. 531–540. doi: 10.1109/TVCG.2016.2598919 (see p. 4).

[Sch11] H. J. Schulz. „Treevis.net: A tree visualization reference“. In: *IEEE Computer Graphics and Applications* 31.6 (2011), pp. 11–15. doi: 10.1109/MCG.2011.103 (see p. 17).

[SHS11] H. J. Schulz, S. Hadlak, and H. Schumann. „The design space of implicit hierarchy visualization: A survey“. In: *IEEE Transactions on Visualization and Computer Graphics* 17.4 (2011), pp. 393–411. doi: 10.1109/TVCG.2010.79 (see p. 18).

[Sec02] A. Secord. „Weighted Voronoi stippling“. In: *Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering (NPAR)*. 2002, pp. 37–43. doi: 10.1145/508530.508537 (see p. 82).

[SKM05] A. P. Seyranian, O. N. Kirillov, and A. A. Mailybaev. „Coupling of eigenvalues of complex matrices at diabolic and exceptional points“. In: *Journal of Physics A: Mathematical and General* 38.8 (2005), pp. 1723–1740. doi: 10.1088/0305-4470/38/8/009 (see p. 59).

[SS11] G. Shevlyakov and P. Smirnov. „Robust estimation of the correlation coefficient: an attempt of survey“. In: *Australian Journal of Statistics* 1 & 2 (2011), pp. 147–156 (see p. 67).

[SAS16] P. Simonetto, D. Archambault, and C. Scheidegger. „A simple approach for boundary improvement of Euler diagrams“. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 678–687. doi: 10.1109/TVCG.2015.2467992 (see p. 19).

[Ske+08] M. Skeels et al. „Revealing uncertainty for information visualization“. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, 2008, pp. 376–379. doi: 10.1145/1385569.1385637 (see pp. 9, 35, 61).

[Spe04] C. Spearman. „General intelligence, objectively determined and measured“. In: *American Journal of Psychology* 15 (1904), pp. 201–293 (see p. 53).

[SPS11] D. Spiegelhalter, M. Pearson, and I. Short. „Visualizing uncertainty about the future“. In: *Science* 333.6048 (2011), pp. 1393–1400. doi: 10.1126/science.1191181 (see p. 53).

[Spi+18] T. Spinner et al. „Towards an interpretable latent space“. In: *Proceedings of the Workshop on Visualization for AI Explainability (VISxAI)*. https://thilospinner.com/towards-an-interpretable-latent-space/. Last Accessed: 2021-07-15. IEEE VIS, 2018 (see p. 6).

[Spr79] M. D. Springer. *The Algebra of Random Variables*. Wiley, 1979 (see p. 11).

[Ste15] J. H. Steiger. *Principal components analysis*. [Online; accessed 9 Mar 2019]. Feb. 2015. URL: https://statpower.net/Content/312/R%20Stuff/PCA.html (see p. 59).

[Ste05] K. Stephenson. *Introduction to Circle Packing: The Theory of Discrete Analytic Functions*. Cambridge University Press, 2005 (see p. 18).

[Str+18] D. Streeb et al. „Distances, neighborhoods, or dimensions? Projection literacy for the analysis of multivariate data". In: *Proceedings of the Workshop on Visualization for AI Explainability (VISxAI)*. https://visxprojections.dbvis.de. Last Accessed: 2021-07-15. IEEE VIS, 2018 (see p. 53).

[TT14] S. Tak and A. Toet. „Color and uncertainty: It is not always black and white". In: *EuroVis – Short Papers* (2014). DOI: 10.2312/eurovisshort.20141157 (see p. 15).

[Tal05] N. N. Taleb. *Fooled by Randomness: The Hidden Role of Chance in Life and in the Markets*. Random House, 2005 (see p. 88).

[Tal08] N. N. Taleb. *The Black Swan: The Impact of the Highly Improbable*. Penguin Books, 2008 (see p. 10).

[TB99] M. E. Tipping and C. M. Bishop. „Probabilistic principal component analysis". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61.3 (1999), pp. 611–622. DOI: 10.1111/1467-9868.00196 (see p. 52).

[Tor91] E. Torgersen. „Comparison of linear models". In: *Comparison of Statistical Experiments*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1991, pp. 411–504. DOI: 10.1017/CBO9780511666353.009 (see p. 65).

[Tor52] W. S. Torgerson. „Multidimensional scaling: I. Theory and method". In: *Psychometrika* 17.4 (1952), pp. 401–419 (see p. 53).

[TSD09] M. Tory, C. Swindells, and R. Dreezer. „Comparing dot and landscape spatializations for visual memory differences". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009), pp. 1033–1040. DOI: 10.1109/tvcg.2009.127 (see pp. 40, 81).

[TG08] S. Tripathi and R. S. Govindaraju. „Engaging uncertainty in hydrologic data sets using principal component analysis: BaNPCA algorithm". In: *Water Resources Research* 44.10 (2008). DOI: 10.1029/2007WR006692 (see p. 52).

[VD03] R. Van Liere and W. De Leeuw. „GraphSplatting: Visualizing graphs as continuous fields". In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 206–212. DOI: 10.1109/TVCG.2003.1196007 (see pp. 36, 39).

[VCB18] N. Vaswani, Y. Chi, and T. Bouwmans. „Rethinking principal component analysis (PCA) for modern data sets: Theory, algorithms, and applications". In: *Proceedings of the IEEE* 106.8 (2018), pp. 1274–1276. DOI: 10.1109/JPROC.2018.2853498 (see p. 52).

[VRW13] C. Vehlow, T. Reinhardt, and D. Weiskopf. „Visualizing fuzzy overlapping communities in networks". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2486–2495. DOI: 10.1109/TVCG.2013.232 (see pp. 24, 36).

[Vié+13] F. Viégas et al. „Google+Ripples: A native visualization of information flow". In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pp. 1389–1398. DOI: 10.1145/2488388.2488504 (see p. 18).

[VI18] I. Viola and T. Isenberg. „Pondering the concept of abstraction in (illustrative) visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 24.9 (2018), pp. 2573–2588. DOI: 10.1109/tvcg.2017.2747545 (see p. 81).

[vLan+11] T. von Landesberger et al. „Visual analysis of large graphs: State-of-the-art and future research challenges". In: *Computer Graphics Forum* 30.6 (2011), pp. 1719–1749. DOI: 10.1111/j.1467-8659.2011.01898.x (see pp. 33, 36).

[vMer+05] C. von Mering et al. „STRING: Known and predicted protein-protein associations, integrated and transferred across organisms". In: *Nucleic Acids Research* 33 (2005), pp. D433–7. DOI: 10.1093/nar/gki005 (see pp. 35, 44).

[vNW91] J. von Neumann and E. P. Wigner. „Über merkwürdige diskrete Eigenwerte“. In: *The Collected Works of Eugene Paul Wigner*. Springer, 1991, pp. 291–293. DOI: 10.1007/978-3-662-02781-3_19 (see p. 59).

[Wan+06] W. Wang et al. „Visualization of large hierarchical data by circle packing“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2006, pp. 517–520. DOI: 10.1145/1124772.1124851 (see pp. 18, 22, 23).

[Wan+16] Y. Wang et al. „AmbiguityVis: Visualization of ambiguity in graph layouts“. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 359–368. DOI: 10.1109/TVCG.2015.2467691 (see p. 36).

[Wan+17] Z. Wang et al. „Gaussian cubes: Real-time modeling for visual exploration of large multidimensional datasets“. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 681–690. DOI: 10.1109/TVCG.2016.2598694 (see pp. 53, 67).

[War13] C. Ware. „Foundations for an applied science of data visualization“. In: *Information Visualization*. Elsevier, 2013, pp. 1–30. DOI: 10.1016/B978-0-12-381464-7.00001-6 (see pp. 35, 41).

[WVJ16] M. Wattenberg, F. Viégas, and I. Johnson. „How to use t-SNE effectively“. In: *Distill* (2016). DOI: 10.23915/distill.00002 (see p. 53).

[WP67] D. J. A. Welsh and M. B. Powell. „An upper bound for the chromatic number of a graph and its application to timetabling problems“. In: *The Computer Journal* 10.1 (1967), pp. 85–86. DOI: 10.1093/comjnl/10.1.85 (see pp. 35, 41).

[Wet03] K. Wetzel. *Pebbles — using circular treemaps to visualize disk usage*. Last Accessed: 2021-07-15. 2003. URL: http://lip.sourceforge.net/ctreemap.html (see pp. 18, 22).

[Wic94] T. D. Wickens. *The Geometry of Multivariate Statistics*. Psychology Press, 1994 (see p. 10).

[WPL96] C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. „Glyphs for visualizing uncertainty in vector fields“. In: *IEEE Transactions on Visualization and Computer Graphics* 2.3 (1996), pp. 266–279. DOI: 10.1109/2945.537309 (see p. 3).

[Woo+12] J. Wood et al. „Sketchy rendering for information visualization“. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2749–2758. DOI: 10.1109/tvcg.2012.262 (see p. 3).

[WZY16] T. Wu, L. Zhang, and J. Yang. „Automatic generation of aesthetic patterns with cloud model“. In: *Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. 2016, pp. 1077–1084. DOI: 10.1109/FSKD.2016.7603328 (see p. 30).

[ZL15] H. Zhao and L. Lu. „Variational circular treemaps for interactive visualization of hierarchical data“. In: *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)*. 2015, pp. 81–85. DOI: 10.1109/PACIFICVIS.2015.7156360 (see pp. 17, 18, 22).

[Zhe+19] Y. Zheng et al. „Visualization of big spatial data using coresets for kernel density estimates“. In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 524–534. DOI: 10.1109/tbdata.2019.2913655 (see p. 85).

[Zho+07] K. Zhou et al. „Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media“. In: *15th Pacific Conference on Computer Graphics and Applications*. IEEE, 2007. DOI: 10.1109/pg.2007.48 (see p. 77).

[Zou+10] Z. Zou et al. „Mining frequent subgraph patterns from uncertain graph data“. In: *IEEE Transactions on Knowledge and Data Engineering* 22.9 (2010), pp. 1203–1218. DOI: 10.1109/TKDE.2010.80 (see p. 36).