Bachelor's Thesis

VisRecly

Towards a Visualization Recommender Tool
for Non-Visualization Experts

Author

Péter Ferenc Gyarmati

Sought Degree

Bachelor of Science (BSc)

Vienna, 2022

Study Code:	521 [4]
Subject:	Computer Science - Data Science
Supervisor:	Univ.-Prof. Torsten Möller, PhD
Co-supervisor:	Univ.Ass. Manfred Klaffenböck, MSc

Abstract

There is a story behind every single set of data, waiting to be told. Using visualizations to tell those stories is a terrific choice; however, the way how the data analyst authors these visualizations vastly affects how the audience receives the story concealed behind the raw data. Just as people are not fond of reading prose filled with spelling and grammar mistakes or out-of-tone compositions, they also do not enjoy looking at visualizations violating fundamental design guidelines or being out of context regarding the analytical task at hand.

The system introduced in this thesis aims to be a grammar and spelling checker for stories written with visualizations. *VisRecly* strives to generate visualization recommendations in which non-negotiable visualization design guidelines are enforced by default, just as a writing assistant would dictate correct spelling and grammatical structures. Furthermore, it aims to help its users to find the tone of their visualizations by ranking recommendations across visualization tasks and highlighting the most suitable ones, just as a writing assistant would propose synonyms and alternative sentence formation based on the assumed needs of the target audience. *VisRecly* seeks to achieve these feats while being a highly accessible, easy-to-use system even for non-visualization expert users.

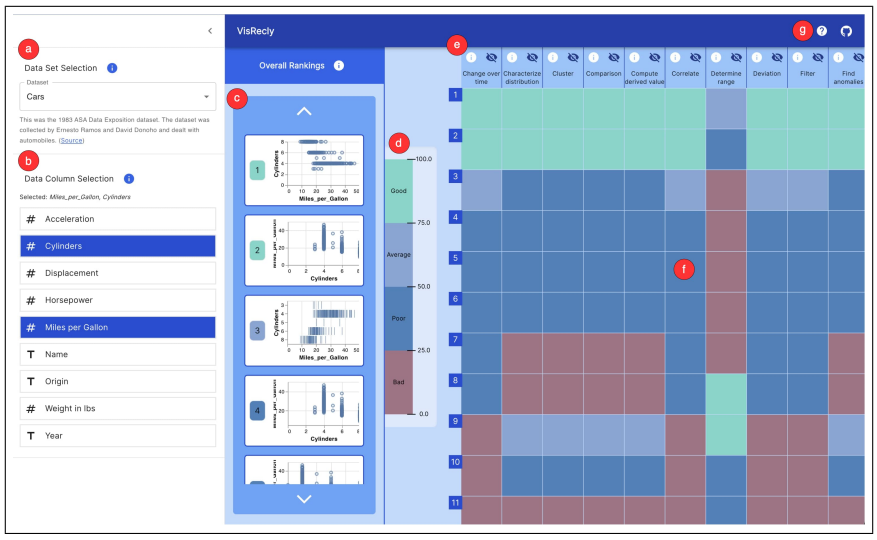

Figure 1: The *VisRecly* user interface. The side panel allows for selecting dataset (a) and data columns (b) via dedicated selector components. Recommendations are generated and ranked automatically, displayed in the overall ranking list (c). The overall rank of recommendations gets categorized and color-coded into four categories based on their overall score (Good, Average, Poor and Bad), displayed by a color scale (d). Related visualization tasks are presented as the header (e) of the central heatmap (f). Each heatmap cell refers to a recommendation item of the overall ranking list (c). The color of a cell conveys the overall rank of the recommendation associated with the cell, while its vertical position in the heatmap represents its task-specific rank. Users can access in-app explanations of the UI on-demand via the help button (g).

Contents

1 Introduction

At the time of writing, big data being a ubiquitous buzzword, data analysis – especially exploratory data analysis – is becoming increasingly important, as it aims at asking questions about raw data and extracting knowledge from it. Visualizations, a common tool in the toolbox of exploratory data analysis, have the primary objective to aid users in exploring, understanding and analyzing data as well as helping them to answer questions they did not even know they would need to ask in the first place. They can do so by leveraging the power of the human visual system, our perception channel with one of the highest bandwidths to our brain.

1.1 Motivation & Problem Statement

Since visualization is such a pivotal tool when it comes to performing data analysis, visualization recommendation (VisRec) systems also have been developed extensively, so that generating visualizations gets accelerated and facilitated. Several independent studies [2][46][50] have concluded that the effectiveness of visualizations depends on the user's specific goals and tasks to a significant degree. An equally pivotal aspect to consider when interacting with VisRec systems is their usability, interpretability and the overall user experience with which results are presented to users.

Yet, as Shen et al. [55] outline in their recent TaskVis short paper *"most of the existing systems lack detailed modeling of analysis tasks, so they are only able to prune meaningless visualizations but fail to recommend targeted results"*. A further issue one may discover after gaining familiarity with relevant works is that even if some recommendation systems have considered analytical tasks in the recommendation process, most of these systems are realized with the prior assumption that users have familiarity with VIS. As Shen et al. put it in a more recent paper of theirs, *"[existing VisRec systems] suffer from high requirements for domain expertise"* [56]. As a direct consequence, these tools make heavy use of VIS-specific technical terms and they do not guide users to a sufficient extent with regards to how the application should be operated; hence they are not accessible to novice users, who are not familiar with the formal concepts of visualization. [1]

Analogously to Shen et al. [56], Vartak et al. [62] also argue that existing visualization recommendation systems are limited in that *"[they] require substantial manual effort and tedious trial-and-error"*. They also outline that *"current tools lack the means to specify what the analyst is looking for"*.

As the importance of making big data accessible to non-data scientists too is ever-growing, the need for a visualization recommendation system that considers visualization tasks and is accessible to novice users is also not a matter of

[1] In the context of this thesis, the term "novice user" refers to a person who can interpret quantitative data, however, does not possess any prior knowledge about formal VIS concepts.

dispute. Since generating visualizations tends to be the first step in exploring data, a VisRec tool accessible to a wider audience that allows for rapid iterations might prove to be valuable. Hereby, I dedicate my thesis to the design-space exploration and development of a high-fidelity prototype of such a tool, named *VisRecly*[2], while seeking answers to the questions introduced in the upcoming *Thesis Objective* section.

1.2 Thesis Objective

In light of the above-introduced problem statement, the objective of my thesis is to answer the below-posed research questions as well as to explore the related design space.

RQ1 How can a VisRec user interface support novice users in exploring visualization recommendations relevant to their specific goals?

RQ2 Along which main principles should a VisRec user interface be designed to make the ranked recommendations' interpretation straightforward to a novice user?

RQ3 To what extent can a user-experience-centered VisRec system take advantage of an existing VisRec engine's capabilities to produce visualization recommendations for a novice user?

Note that RQ1 is the guiding research question of my thesis, the proceeding ones serve as sub-questions, helping me to explore the topic in a fine-grained manner. I will strive to answer RQ2 as a part of the low-fidelity prototyping phase of my research, while I will attempt to provide a satisfying answer to RQ3 as a part of my work's high-fidelity prototyping phase. The extent to which I managed to satisfy the above-defined objectives will be discussed in the concluding section of my thesis.

1.2.1 Metrics of Success

Considering the depth of the research questions and the scope of a Bachelor's thesis, finding adequate, well-reasoned answers to the posed questions will already constitute a major part of my work, hence the granularity of the proposed answers can be considered as a metric of success. Isenberg et. al [29] discuss this metric in their work in which they investigate the practice of evaluation in visualization research as a form of *Qualitative Result Inspection*. Apart from this, a significantly more objective - but not exclusive - metric will be the results of the user test sessions conducted on the high-fidelity prototypes, with regards to the extent to which novice users find *VisRecly* accessible, usable and useful.

I dedicate the upcoming section to providing an overview of the exact process and methodology I will utilize to fulfill the aforementioned objectives.

[2] A combination of the terms "VisRec" and "Grammarly", hinting at the inspirative sources of this thesis.

1.3 Process & Methodology Overview

Given the problem-oriented nature of this work, I utilized the design study methodology introduced by Sedlmair et al. in their *Design Study Methodology: Reflections from the Trenches and the Stacks* paper, providing a well-defined framework for problem-driven visualization research. The authors' definition of a *design study* captures the core aspects of my research:

> *"A design study is a project in which visualization researchers analyze a specific real-world problem faced by domain experts, design a visualization system that supports solving this problem, validate the design, and reflect about lessons learned in order to refine visualization design guidelines."*[53]

The original framework proposes nine stages, split into three main sections:

1. Precondition
   (a) Learn: Visualization Literature
   (b) Winnow: Select Promising Collaborations
   (c) Cast: Identify Collaborator Roles
2. Core
   (a) Discover: Problem Characterization & Abstraction
   (b) Design: Data Abstraction, Visual Encoding & Interaction
   (c) Implement: Prototypes, Tool & Usability
   (d) Deploy: Release & Gather Feedback
3. Analysis
   (a) Reflect: Confirm, Refine, Reject, Propose Guidelines
   (b) Write: Design Study Paper

My thesis is built around the very same nine steps. The listing below provides an overview of the concrete actions I took at each step.

- **1a - Learn**: Gaining familiarity with Visualization, VisRec (Visualization Recommendation), user study literature and similar tools.
- **1b - Winnow**: Identifying collaborations by conducting initial meetings with my thesis supervisors, both of them being well-versed in my topic of research.
- **1c - Cast**: Defining personas and identifying collaborators accordingly to help me in participating in test sessions of my prototypes. Collaborators include persons from within the target audience (novice users) and outside of it (users with prior expertise in VIS).
- **2a - Discover**: Collecting initial ideas using the *Five Design Sheet* methodology and iterating on them.
- **2b - Design**: Creating three vastly different low-level prototypes.
  - Conducting test sessions on them to evaluate their usability.
  - Iterating on the prototypes based on user feedback.
  - Synthesizing the prototypes into a single one, to be implemented in high-fidelity.

  The low-level prototyping phase constituted a major part of my work, as it was crucial to investigate a wide design space of possible solutions. I have conducted twelve user interviews with a focus on obtaining qualitative feedback on low-fidelity prototypes.
- **2c - Implement**: Implementing a high-fidelity prototype from the synthetized low-level prototype in the form of a self-contained web application.

  The full development time of the high-fidelity prototype - involving the adjustment & customization of third-party libraries - required one month of software engineering.
- **2d - Deploy**: Releasing the implementation.
  - Deploying the high-fidelity prototype as a web application.
  - Conducting a usability study on the prototype with thirteen participants.
- **3a - Reflect**: Summarising user feedback, reflecting on the findings, takeaways and limitations of my work.

  For the evaluation of the high-fidelity prototype, usability studies have been conducted with thirteen participants, representing users from the predefined persona types as well as users with prior VIS knowledge to gain feedback from a broader perspective too.
- **3b - Write**: Creating this paper.

A more detailed discussion of the prototyping phases is provided in the *Design: Low-fidelity Prototyping* and *Implementation: High-fidelity Prototyping* sections of my thesis.

1.4 Main Contributions

In summary, the main contributions of my thesis are:

1. Finding out which user interface elements facilitate the interpretation of ranked visualization recommendations and their relation to visualization tasks for a novice user
2. Determining the main principles along which a VisRec user interface should be designed to maximize the interpretability of the produced recommendations and their relation to visualization tasks for a novice user
3. Exploring the extent to which a VisRec system focusing on being accessible for non-visualization experts can take advantage of an existing VisRec engine's capabilities to produce recommendations to a novice user
4. Development of a high-fidelity prototype that encapsulates the findings of the above-listed contributions

In order to establish a solid foundation for my research and allow me to focus on a more restricted set of challenges, a thorough analysis of the existing state-of-the-art literature has been conducted, discussed in the upcoming, *Related Work* section.

2 Related Work

Due to the growing role and significance of visualization recommendation systems, an abundance of research has been done both on principles and ideas along which such tools shall be created as well as on concrete implementations. While my thesis focuses on contributing a user-task-oriented VisRec tool, it is essential to survey a wider set of tools in order to understand their benefits, limitations and how their proposed ideas might be advantageous for my research. In what follows I will elaborate on the state-of-the-art strategies and design considerations as well as on existing tools which are similar to my work while historically reflecting on the past two decades' research progress in the field of visualization recommendation.

Influenced by the most distinguishing factors identified by Vartak et al. [62], Kaur and Owonibi [31] have proposed to classify visualization recommendation approaches into four categories, based on their contribution to research:

1. **Data Characteristics Oriented**: recommendations are created under the consideration of the data characteristics.
2. **Task Oriented**: recommendations are created under the consideration of the user's representational goals as well as the data characteristics.
3. **Domain Knowledge Oriented**: recommendations are created under the consideration of domain knowledge specific to the processed dataset.
4. **User Preferences Oriented**: recommendations are created under the consideration of the user's presentation goals as well as preferences inferred by the way of real-time user interaction with the tool.

Given the scope of this piece of research, I will focus on the first two categories, that is, I will analyze literature in more detail relevant to data-characteristics-oriented recommenders and task-oriented ones.

2.1 Data Characteristics Oriented Works

Studies about visualization recommendations focusing on this category have been aiming at improving the extent to which data and relationships existing within the data are understood. As Kaur and Owonibi outline, *"The choice of variables to represent different aspects of the same information can greatly influence the perception and understanding of the presented information. Therefore, the research under this category focuses on: the definition of new data dimensions or attributes, the formalization of the process of visual mapping from data attributes to visual marks, and the introduction of new techniques for visual mapping."* [31].

One of the earliest studies dedicated to automating visualization generation based on data characteristics have been conducted in 1981 by Gnanamgari in

his dissertation [8]. He proposed a set of rules based on heuristics to determine a mapping between specific types of data attributes and visualization primitives suitable for their display. Five years later, Mackinlay contributed the APT system [39] with a formalized graphical design specification. Thanks to the formalization, this approach made it possible to programmatically generate graphical design specifications, hence achieving automation. Mackinlay also came up with a mapping between data attributes and visual marks, considering criteria such as expressiveness and effectiveness, as Table 1 testifies.

	Nominal	**Ordinal**	**Quantitative**
Size	-	*	*
Saturation	-	*	*
Texture	*	*	
Color	*	*	
Orientation	*		
Shape	*		

Table 1: *"Expressiveness of retinal techniques. The - indicates that size and saturation should not be used for nominal measurements because they will probably be perceived to be ordered. The * indicates that the full color spectrum is not ordered. However, parts of the color spectrum are ordinally perceived."* [39]

Further work has built upon the contributions of Mackinlay. Design specifications based on the heuristics he laid down were fused into the development of Polaris, a system for query, analysis, and visualization of multidimensional relational databases [59]. Four years later a formal declarative visual language called VizQL [11] has been developed, incorporating a revised version of the specifications used in Polaris. VizQL is used by the well-known visualization software, Tableau's [24] "Show Me" module to automatically recommend visualizations to its users.

While all of the aforementioned studies contributed desktop-based, offline VisRec implementations, in 2007 Viegas et al. broke this trend with the development of a system called Many Eyes [65] which was fully web-based. It also considers the data characteristics when generating recommendations: based on a predefined mapping scheme visualization techniques are matched with the dataset specified by the user.

The approach taken by Many Eyes was popular in that it was easily accessible to the public because it was deployed as a web application. More contemporary academic contributions by Wongsuphasawat et al., Voyager [68] and its successor Voyager 2 [69] are also web-based VisRec tools, built by using Vega-Lite [52], a high-level grammar that enables rapid specification of interactive data visualizations. As Kaur and Owonibi argue, *"[a] Vega-lite specification is a JSON object that describes a single data source, a mark type, visual encodings of data variables, key-value, and data transformations including filters and*

aggregate functions." [31]. Apart from Vega-lite, these tools make use of Compass [68] as a recommendation engine too, which defines a partial specification that describes enumeration constraints, and methods for choosing, ranking, and grouping recommended visualizations. *"The Compass Recommendation Engine first suggests a list of visualizations based on the univariate summary of each variable in the dataset. Then the user can exclude or include variables from the list to focus on a particular variable set of interest."* [31]

As apparent from the discussion above, substantial contributions have been made in the past two decades toward better data characteristics-oriented visualization recommendations. APT [39] and Many Eyes [65] defined a set of rules and heuristics to map data characteristics to visual marks and idioms. VizQL [11] and Vega-Lite [52] are language formalizations to facilitate the definition of specifications and to automate the process of visual mapping. Finally, Voyager [68] contributes a statistics-based approach, utilizing statistical and exploratory data analytics procedures to recommend visualization.

2.2 Task Oriented Works

As discussed in the previous section, the first pieces of research dedicated to automatic visualization recommendations were mainly focusing on the data characteristics and the principles based on which a sensible mapping can be found between data attributes and visual marks. In 1990, Roth and Mattis [49] contributed the idea of considering the tasks and goals of the user to produce more relevant recommendations. As they put it, they proposed *"a taxonomy of information characteristics which would need to be provided to either human or computer designers for them to create presentations reflecting the individual needs of a diverse group of users"*. In their work, they abstracted away the domain-specific objectives and identified domain-independent ones, such as comparison, distribution and correlation.

Based on the research conducted by Kerpedjiev et al. in 1997 [32], producing more relevant visualization recommendations is possible by considering specific domain-level tasks too. They proposed a model (Figure 2) to decompose domain-specific goals into domain-independent ones and to map these high-level tasks to visual marks.

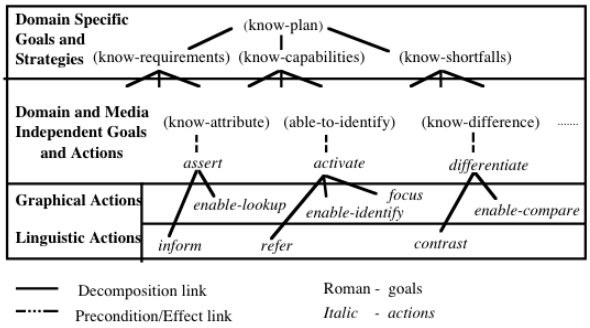

Figure 2: Goals, Actions and Tasks by Kerpedjiev et al. (1997) [32]

More recent works have also been dedicated to formalizing a taxonomy of common categories to enhance task-oriented visualization recommendations. Lee et al. [35] contributed a taxonomy of common analytical actions (Figure 3) and used this in the implementation of Frontier, an automated assistant for visual data exploration [34].

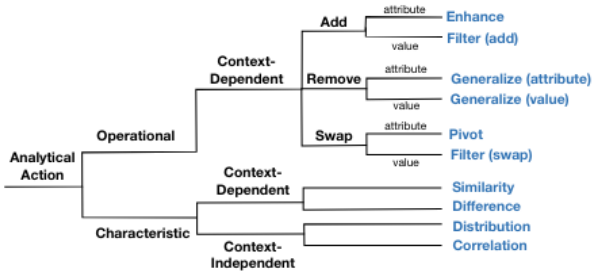

Figure 3: A taxonomy of common analytical actions used in recommending visualizations for visual analysis. [35]

Yet another notable contribution in the field of categorizing visualization tasks has been submitted by Brehmer and Munzner [2], proposing a multi-level typology of visualization tasks (Figure 4) to bridge the gap between low-level and high-level visualization tasks. The proposed typology is built upon three guiding questions:

1. **Why** is the task performed?
2. **How** is the task performed?
3. **What** does the task pertain to?

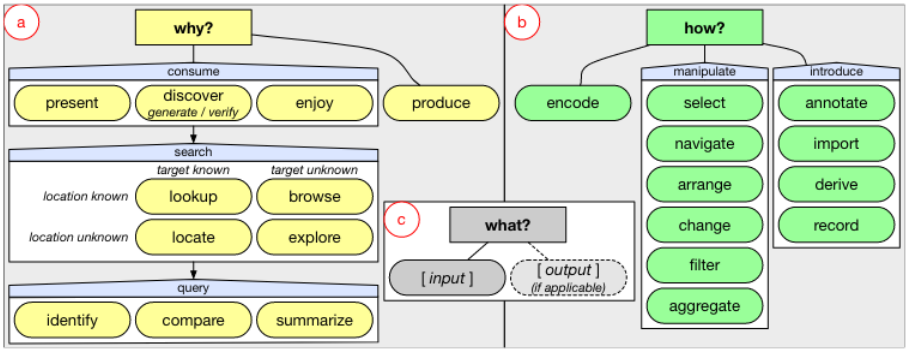

Figure 4: *"The typology spans why, how, and what; task descriptions are formed by nodes from each part: a) why [is a task performed], from high-level (consume vs. produce) to mid-level (search) to low-level (query). b) how [is a task executed] in terms of methods, defined as families of related visual encoding and interaction techniques. c) what are the [task's] inputs and outputs."* [2]

A task-oriented tool of particular interest in the context of my thesis is TaskVis, developed by Shen et al. [56]. It is a web-based, task-oriented visualization recommendation system that allows users to specify their tasks via the user interface. It supports eighteen different analytic tasks and maintains an empirical-wisdom-based rule base of them to enumerate candidate visualizations with the help of answer set programming. TaskVis aims at providing a robust but at the same time user-friendly VisRec system which does not demand high domain expertise, such as other tools.

2.3 Visualization Recommendation Systems

As per the work of Saket et al. [51], visualization recommendation systems can be categorized into three classes, representing the source of knowledge used to produce recommendations:

- **Knowledge-based systems**: *"A large body of existing automated visualization design tools focuses on suggesting visualizations based on user-defined constraints (such as which data attributes to visualize) and design constraints."* [51]
- **Data-driven systems**: *"A machine learning model tries to best predict what visualization is most appropriate based on the given inputs (e.g., tasks, data attributes, etc.). Developers of visualization tools need to handcraft informative, discriminating, and independent features for learning such models."* [51]
- **Hybrid systems**: *"Hybrid recommender systems are both knowledge-based and data-driven"*, that is, *"the system designer provides the knowledge about visualization design"* and *"the system learns a recommendation*

> *model from data". "The system designer has full control over the recommendation process but can augment the knowledge base with machine learning."* [51]

Table 2 below serves to provide a better overview of existing VisRec systems along with the recommendation method they utilize, the language they use in the background to create graphics, as well as the visual idioms they support.

System	Method	Language	Bar	Pie	Line	Scatter	Area	Histogram	Tick	Gantt
TaskVis (2022) [56]	Knowledge-based	Vega-Lite	*	*	*	*	*	*	*	
Dziban (2020) [37]	Hybrid	Vega-Lite	*	*	*	*	*	*	*	
Ask Data (2019) [60]	Knowledge-based	VizQL	*	*	*	*				*
VizML (2019) [28]	Hybrid	D3	*	*	*	*				
Data2Vis (2018) [6]	Data-driven	Vega-Lite	*	*	*	*	*		*	
DeepEye (2018) [38]	Hybrid	Vega-Lite	*	*	*	*				
Draco (2018) [44]	Hybrid	Vega-Lite	*	*	*	*	*	*	*	
Keshif (2017) [70]	Knowledge-based	D3	*	*	*	*				
Voyager (2017) [68]	Knowledge-based	Vega-Lite	*	*	*	*				
Zenvisage (2016) [58]	Knowledge-based	ZQL	*	*	*	*				
SeeDB (2014) [63]	Knowledge-based	-	*	*	*	*				
Profiler (2012) [30]	Knowledge-based	JS	*	*	*	*				
VizDeck (2012) [33]	Knowledge-based	JS	*	*	*	*				
HARVEST (2009) [9]	Knowledge-based	JS	*	*	*	*				
Show Me (2007) [40]	Knowledge-based	VizQL	*	*	*	*				*
Polaris (2000) [59]	Knowledge-based	VizQL	*	*	*	*				
SAGE (1994) [48]	Knowledge-based	-	*	*	*	*				
APT (1986) [39]	Knowledge-based	-	*	*	*	*				

Table 2: Overview of existing VisRec systems

As Table 2 shows, knowledge-based approaches have been the most common in the past years, with hybrid approaches becoming more popular in recent contributions, as it allows for harnessing the advantages of both the knowledge-based and data-driven approaches. When it comes to the languages used for specifying visualizations, Vega-Lite has been used most frequently, while visual query languages such as VizQL and ZQL also have been utilized for some tools. Note that VizML, Keshif, Profiler, VizDeck and HARVEST opted for using technologies native to web browsers, namely JavaScript (JS) and D3, a low-level visualization library written in JavaScript. It is also notable that most of these systems only support generating bar charts, pie charts, line charts and scatter plots as well as their variants. This might lead to limitations when it comes to expressing users' data.

2.4 System Architectural Considerations

Generating visualization recommendations in real-time for users is computationally very expensive, therefore, it is essential to take system architectural

considerations into account. As Vartak et al. [62] argue, *"[while] traditional disk-resident databases can accomodate large datasets, they cannot provide the interactive speeds necessary for visualization recommendation. As a result, a VISREC system must take advantage of main-memory using techniques such as operating on samples, pre-materializing views and using efficient indexes."* Vartak et al. identify three main strategies to guide the architecting of a VisRec system.

Pre-computation Performing complex, expensive computations offline is preferred, as it allows for using these results for fast predictions during the online phase. *Since VISREC systems must employ knowledge-based filtering and the set of potential visualization is not known upfront, opportunities to perform complex computations offline may be limited. (...) a visualization recommender can also perform offline computation of various statistics and correlations that can inform subsequent explorations and construction of visualizations.* [62] Furthermore, the authors also identify traditional caching as a viable option to improve performance, that is, to save the results computed to a specific input, then returning them for the subsequent user's requesting recommendations for the same input.

Online Computation *"[Visual] recommenders are in the unique position of having to produce the space of potential recommendations on-the-fly. To avoid latencies in the hours, online computation must perform aggressive optimizations while evaluating visualizations."* [62] Vartak et al. discuss four main strategies for online optimizations.

- **Parallelism**: the evaluation and ranking of visualizations can happen in parallel to reduce the latency of the system.
- **Multi-query optimization**: *"the computations used to produce candidate visualizations are often very similar; they perform similar operations on the same or closely related datasets. Consequently, multi-query optimizations techniques (...) can be used to intelligently group queries and share computation"* [62]
- **Pruning**: Reducing the search space of visualizations by discarding low-utility views using pruning techniques such as confidence-interval pruning [66] or bandit resource allocations [64] also contribute to the system's performance.
- **Better algorithms**: using more efficient algorithms to compute statistical properties of the data can also contribute to the system's performance.

Approximate Computation Vartak et al. propose computing query results only in an approximative manner in order to reduce the latency when processing large datasets. Sampling-strategy-based techniques might be also used to further speed up processing, *"users may be satisfied with imperfect results: both*

imperfect visualizations (...) and imperfect recommendations of visualizations." [62]

2.5 Takeaways

The above-introduced survey of relevant literature allowed me to assess the landscape of visualization recommendation systems in detail and to identify tools and ideas I can leverage to develop a high-fidelity prototype of *VisRecly*. Various works have made it clear that considering the user tasks is essential when it comes to evaluating the relevance of the generated visualization recommendations. The fact that only a single tool, TaskVis [56] has been contributed as a VisRec system focusing mainly on user tasks confirmed the viability of *VisRecly*; it is clear that research aiming at contributing a novice-user-first visualization recommendation system is an area yet to be ventured.

The analysis of related work also made it unequivocal that using a formal language or specification such as Vega-Lite to define visualizations programmatically is a crucial part of VisRec systems. Looking at the trend that more modern systems make use of a hybrid strategy when it comes to sourcing knowledge to produce recommendations made me realize that benefitting both from the knowledge-based and data-driven approaches might lead to more eloquent results.

Finally, analyzing the complexity of a recommendation engine's architecture and all the performance challenges it needs to conquer, I came to the conclusion that the engine picked for my project will influence the quality of the final product to a significant extent, just as the orchestrated system architecture when it comes to the development of the high-fidelity prototype.

3 Design: Low-fidelity Prototyping

Given that a fundamental objective of *VisRecly* is to make it accessible for users new to visualization and visual data analysis, conducting a comprehensive low-fidelity prototyping phase was essential in order to explore the design space of user interface variations and to identify the most suitable candidates. Moving forward, I elaborate on the low-fidelity prototyping process I followed to arrive at a synthesis of the most promising design ideas.

3.1 Approach

The approach I followed in the design phase was made up of creating user personas and user stories, creating initial prototypes in accordance with the *Five Design-Sheets* method [47] and gathering qualitative feedback on them through test sessions with real users. The paragraphs below give insight into the process in finer detail.

3.1.1 Personas

As the very first step of the low-fidelity prototyping phase, I utilized the HCI technique of defining personas, in order to identify details about the target users & to be able to better comprehend their characteristics, hence their needs.

I came up with four different personas, in an attempt to cover as many details about my target audience as possible. I discuss the differences and the motivation behind these personas below.

Veronika: BSc Student Represents persons with basic formal education having some sense about interpreting quantitative information. Users in the group identified by this persona would make use of a polished user interface on which most of the options are displayed already, as they would likely come up with their VIS goals on the fly while interacting with the UI. Visual elements of the tool's user interface & interactions might have an intermediate complexity due to the prior familiarity with non-trivial pieces of technology gained throughout their scientific Bachelor studies.

Albert: Ph.D. Student Represents persons with a more advanced formal education having a strong command of working with quantitative data. Similar users will already have an idea about their VIS tasks and they would be able to communicate these informally. Hence, the UI must let them select the task type and the granularity in a quick & straightforward way. Visual elements of the UI and interactions might have an upper-intermediate complexity due to the prior experience with various non-trivial pieces of technology.

Reinhard: Professor Represents persons who are professionals in a specific field of science, however, are still novices when it comes to big data and

VIS. Users belonging to the group identified by this persona have a very limited amount of time that they can invest in familiarizing themselves with new concepts and tools, as they tend to need tangible results as soon as possible. Therefore, they need a tool to do the heavy lifting for them in a field they are not proficient at.

Judith: Marketing Employee Represents persons with no formal education but some sense of quantitative data, often driven only by intuition. Similar users will already have some idea about the type of analytics or VIS they wish to see, but they will use the tool primarily in an explorative manner, adjusting tasks on the fly. Visual elements of the user interface and the interactions should be highly intuitive and user-friendly, due to the lack of experience with academic or technical pieces of software.
A common characteristic of the four personas is that they all have a sense of how to interpret quantitative data, however, they have no formal VIS knowledge. The difference between them is primarily their progress in academia, as well as the prior familiarity they possess with advanced tools which would impact their ability to interact with a dashboard such as *VisRecly* more seamlessly.

I focused on these specific traits of the personas, as I assumed that the academic level of a person and the extent of familiarity they have with digital tools directly influence their needs when it comes to the nature of the application I am developing. By sketching the above-introduced personas, I hoped to cover the relevant spectrum decently in an attempt to proceed with the low-level prototypes by having different needs in mind.

The actual, more detailed persona definitions can be found in the *Personas* section of the appendix.

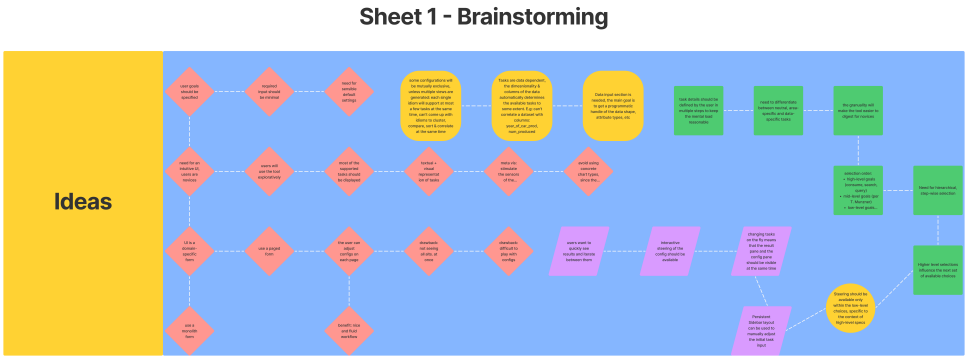

Figure 5: Design sheet used for brainstorming about the design space of possible solutions.

3.1.2 Considered Alternatives

I aimed to come up with vastly different approaches for the prototypes, to cover various aspects of the problem domain. I tried to grasp most of the ideas presented in the design sheet dedicated to brainstorming as illustrated by *Figure 5*. In what follows, I describe the considered alternatives.

Sheet 2 From the very beginning of the brainstorming phase, I tried to come up with an intuitive way to capture the visualization task selection's hierarchy such that it is easy for a novice user to interpret. I designed this prototype so that the aforementioned hierarchy is represented in the form of fractals which can be opened and explored on demand. I assumed that the interaction with this kind of UI will be familiar for the users since the popular presentation creator tool Prezi [23] also takes advantage of this sort of representation.

See Appendix, *Figure 28*.

Sheet 3 Keeping in mind the origin of my idea behind this thesis, I came up with this prototype resembling the user interface of the online writing assistant Grammarly [27]. The interface models the layers of the visualization task selection, and it gives users the chance to steer their selections interactively, seeing the results of their actions immediately.

See Appendix, *Figure 29*.

Sheet 4 Showing empathy towards my target audience, I realized that when being exposed to a previously unknown field - VIS in our exact case - it is important to take one step at a time and to be guided throughout the computer-aided exploration of the problem domain. I created this stepper-based prototype exactly for this purpose: on every single screen, users need to choose from a very restricted set of options, this way, the amount of new information they need to process is minimized.

See Appendix, *Figure 30*.

Sheet 4+1 This prototype is the follow-up of an idea that my supervisor gave me about an approach, inspired by the *LineUp* paper, that would allow me to better visualize multi-attribute ranking [10]. My initial interpretation of how a LineUp interface could be utilized gave birth to this sheet with a tree-based visualization of the task selection hierarchy. The interface was meant to be more complex but at the same time capable of more advanced features such as displaying the generated visualization recommendations automatically.

See Appendix, *Figure 31*.

Sheet 4+2 After discussions with my supervisors, I realized that the LineUp interface could have been leveraged in a potentially less-complex way, namely,

by pinning the first column of the grid and displaying the recommendations in those cells, while using the remaining screen real-estate to visualize the ranking of the recommendations based on their usefulness with regards to a specific user-goal.

See Appendix, *Figure 32*.

3.1.3 Winnowing Ideas

As apparent from what has been discussed in the previous section, contrary to the guidelines introduced by the *Five Design-Sheet* methodology [47], I came up with more than three non-synthetized ideas for the user interface of *VisRecly*. We deemed this deviation as necessary after recognizing that the design space we started to explore was even broader than we anticipated, making the development of more low-fidelity designs justified. As explained in more detail below, after several in-detail discussions with my supervisors, a decision has been made about which three prototypes should be presented to test users to gather constructive feedback from members of the tool's target audience.

Ultimately, *Sheet 2*, *Sheet 4* and a slightly modified version of *Sheet 4+2* have been chosen as the prototypes to be used for testing. *Sheet 3*, the prototype inspired by the user interface of Grammarly [27] has been thrown away since it proved to be the least interesting approach among the introduced alternatives in the sense that it featured the same grouping-based approach like *Sheet 2* did while providing a less consistent user experience by displaying similar or identical user interface components with different responsibilities based on the protoype's state. The sketched user interface *Sheet 4+2* has been moderately reworked to support the additional feature of weighing specific user-tasks as well as to make the comparison of different charts' ranking more intuitive.

As I finalized the initial designs, I craved feedback from real, fully unbiased users. To do so, I decided to adapt the representation of my hand-sketched prototypes, to make them more suitable for conducting test sessions.

3.2 Towards Testability

When it came to planning how I would conduct test sessions about the low-fidelity prototypes, I had to decide between two alternatives regarding their presentation:

- **Paper-based Prototype**: a low-fidelity prototype that is presented and "operated" using a physical paper representation
- **Computer-based Prototype**: a low-fidelity prototype that is presented and "operated" through a digital user interface

As Sefelin et al. outline it in their work, *"... the comfort of subjects is one of the major factors of a successful (...) test, one may argue that these two results mean that a design team should always prefer a computer-based prototype."* [54]. Based on this finding, and based on the fact that I would prefer to conduct the sessions online for time-efficiency reasons, I concluded that creating a computer-based prototype would be the ideal solution.

I ported my sketched ideas to interactive computer-based prototypes with the help of Figma [26], a collaborative interface design tool.

3.3 Testable Prototypes

Screenshots of the prototypes presented to users to gather qualitative feedback are depicted below.

3.3.1 Prototype A

Computer-based realization of *Sheet 2*.

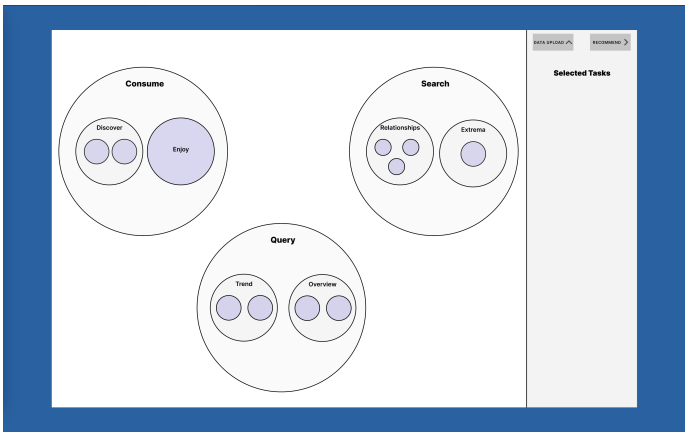

Figure 6: Prototype A: *Sheet 2* as an interactive Figma prototype

3.3.2 Prototype B

Computer-based realization of *Sheet 4*.

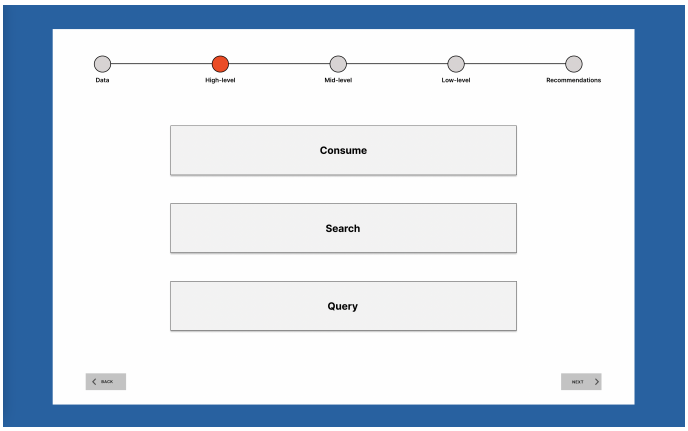

Figure 7: Prototype B: *Sheet 4* as an interactive Figma prototype

3.3.3 Prototype C

Computer-based realization of *Sheet 4+2*'s variation.

Figure 8: Prototype C: *Sheet 4+2* as an interactive Figma prototype

3.4 Method of Evaluation

While qualitative studies are built around collecting and analyzing non-numerical data, I decided to use a semi-structured interview format to gather textual, audiovisual as well as numerical data from participants to scrutinize and use it to shape *VisRecly* into an accessible and useful application. The sections below shed light on the details of the conducted qualitative study.

3.4.1 Nature of The Study

At this stage of the project, I wished to hear critical, constructive feedback from users to be able to recognize the advantages and disadvantages of the proposed prototypes. Therefore, I conducted a series of tests with a focus on capturing qualitative data. This approach seemed to be a reasonable choice, as the work of Adams et al. *A qualitative approach to HCI research* also outlines that *"[with] qualitative research, the emphasis is not on measuring and producing numbers but instead on understanding the qualities of a particular technology and how people use it (...), how they think about it and how they feel about it"*. [1]

3.4.2 Study Structure

The study was structured around real-life interactions with the prototypes. When testing a specific prototype, users had to try to accomplish a specific task, such as uploading data, configuring visualization objectives or observing & interpreting the resulting recommendations. The participants were asked to do so by thinking out loud, and by narrating their actions, their interpretation

of the purpose of user interface elements as well as their expectations upon operating the said elements.

This made it possible to obtain a more detailed picture of the prototypes in the form of fluid, natural, non-guided feedback. After performing a specific user action, the participants were prompted to rate their experience and to share how performing a given flow would have been more ideal for them.

3.4.3 Used Questionnaire

Instead of exclusively focusing on capturing structured, quantitative information, I created a survey with a mixture of open-ended questions and multiple-choice questions, all serving to stimulate the testers to express their notions about specific details of the prototypes. In addition, users had the chance to rate specified aspects of the prototypes on a linear scale.

After exploring literature relevant to creating questionnaires, I decided to incorporate the idea presented by Tullis and Stetson in their journal article *A Comparison of Questionnaires for Assessing Website Usability* [61], namely, to include a multiple-choice-section in the questionnaire in which each option is an adjective. Users are asked to select a few adjectives that - in their opinion - apply to the given prototype. Using this approach, participants have the chance to associate prototypes with characteristics implied by these adjectives, without the need to come up with qualifiers on their own, therefore, the chance of capturing a more nuanced picture of the prototypes is increased.

3.4.4 Study Participants

The study was split into two phases: a pilot phase, to gather very early feedback both about the prototypes as well as about the way how I am conducting the study and a standard phase, in which user's faced the finalized version of the study questionnaire and the computer-based prototypes.

I had the chance to conduct the pilot tests with two non-novice participants who already had familiarity with VIS concepts and studies in general. They provided me with constructive feedback on how could I enhance the questionnaire to better query users' opinions.

In the second phase of the study, I conducted six individual sessions with participants belonging to my target audience, novice users. Participants have been recruited through the dissemination of digital flyers about the study on online forums with a thorough description of the project and the role of a study participant in it. Five male and one female participant have been interviewed, their age ranging from 19 to 26 years with a mean of 23 years. All of the participants have a strong command of interpreting quantitative data. Four of the six participants were university students actively studying Computer Science and concepts of Human-Computer Interaction. The other participants conduct research in the field of digital humanities and astronomy respectively.

3.4.5 Study Details

Each session has been conducted online via Zoom. The participants were asked to share their screens with me, so that I could observe their actions and their reactions to the prototypes.

The prototypes were operated in real-time by users, using Figma's built-in prototyping functionality. The participants were asked to think out loud and to narrate their actions, their interpretation of the purpose of user interface elements as well as their expectations upon operating the said elements.

All aspects of the session were recorded with the explicit consent of the participants, including audio, video as well as their computer's screen while interacting with the prototypes.

The questionnaire was presented as a Google Form document. Submitted answers have been exported and post-processed[3] by me using a tool I coded for this specific purpose.

3.5 Qualitative Study Results

After conducting all test sessions, I consulted the notes I took during each of them, took a look at the quantitative results submitted through the questionnaires, as well as I listened to critical parts of the recordings, to make sure that I do not miss any crucial piece of feedback the prototypes received. I would claim that the sessions were fruitful, as I received unexpected feedback from the participants and they expressed their thoughts and ideas in great detail.

As a result of re-watching the session recordings, I identified ten core values that were outlined by all of the participants either in the form of complimenting a specific aspect of a prototype or by calling my attention to some insufficiencies. In what follows, I introduce these identified aspects and – if possible – associate them with the usability heuristics presented by Nielsen [45] as well as with those presented by Shneiderman [57] also listed below.

[3] "Post-processing" in this context refers to generating graphics from the collected structured data to be presented in this thesis

Usability Heuristics for User Interface Design by Nielsen

N1 Visibility of system status

N2 Match between system and the real world

N3 User control and freedom

N4 Consistency and standards

N5 Error prevention

N6 Recognition rather than recall

N7 Flexibility and efficiency of use

N8 Aesthetic and minimalist design

N9 Help users recognize, diagnose, and recover from errors

N10 Help and documentation

Usability Heuristics for User Interface Design by Schneiderman

S1 Strive for consistency

S2 Seek universal usability

S3 Offer informative feedback

S4 Design dialogs to yield closure

S5 Prevent errors

S6 Permit easy reversal of actions

S7 Keep users in control

S8 Reduce short-term memory load

3.5.1 Core Values for Users

- **Hierarchy is easy to interpret**: It is easy to recognize the task hierarchy and all its nodes at a glance and into which category a specific low-level goal belongs.
  Related usability heuristics: None
- **Hierarchy is easy to navigate**: It is easy to explore all nodes of the hierarchy, and requires a minimum number of user interactions.
  Related usability heuristics: None

- **Task selection path is easy to inspect and reproduce**: After the selection of VIS goals, the user can inspect their selection & can see their action history clearly, based on which they would be able to adjust their subsequent selections with ease.
  Related usability heuristics: None

- **Results of user actions are predictable**: When a user presses a button or makes a choice via some UI interaction, they have some sense about the outcome of their action even on their very first encounter with the tool.
  Related usability heuristics: N1, N4, S1

- **User actions are strongly guided**: The user is not overloaded mentally and always can see a well-defined, minimal set of choices they should take to progress on the dashboard.
  Related usability heuristics: N6, S7

- **No tool usage hints would be required**: Denotes how much sense a given prototype makes without a brief onboarding or an optionally accessible info section in the UI, explaining how to use the tool.
  Related usability heuristics: N10, S3, S8

- **Tasks are explained clearly**: VIS tasks and task categories are explained clearly, providing base information for novice users about their possibilities.
  Related usability heuristics: None

- **Little input, high output**: Refers to the amount of "out-of-the-box" results which are displayed in reaction to a minimal number of user actions or no actions at all.
  Related usability heuristics: N3,

- **All features are accessible on a single screen**: Features and functionalities such as data upload, task reconfiguration, and result inspection are available on a single page, eliminating the need to keep navigating around, hence dropping users out of their workflow.
  Related usability heuristics: N1, N3, N8, S2

- **Clarity, simplicity & familiarity of the UI**: The user interface is not busy nor cluttered, the UI elements are familiar, and their functionality can be deduced by intuition and based on encounters with mainstream tools.
  Related usability heuristics: N7, N8, S2

Note that identified core values related to the interpretation and navigation of task hierarchy and the explanation of the tasks could not be explicitly mapped to any of the heuristics defined by Nielsen or Schneiderman, since these aspects are highly specific to the context of VIS. However, the other core values are closely related to those heuristics. Users emphasized their desire of having a consistent user experience with universal usability as well as informative, guiding feedback while also longing for an aesthetically pleasing application with a minimalistic design.

I distilled the quantitative user feedback corresponding to the above-listed aspects of evaluation and illustrated the results in *Figure 9* as a matrix of histograms, where each histogram summarizes the relevant linear-scale ratings of the participants.

As *Figure 9* shows, each prototype was successful in some aspects, however, there is room for improvement in each case. *Prototype A* was received relatively poorly: users thought that its operation was not intuitive enough and it was not powerful enough to make up for this. Several participants expressed that they are not familiar with the fractal-based user interface, in contrast to my initial assumption, that it is going to be a well-known approach for them based on the popularity of Prezi. When it came to *Prototype B*, users appreciated the guided experience provided by the stepper-based interface, however, they thought that it was difficult to navigate through the hierarchy, as they cannot peek into the next steps. The productivity granted by *Prototype C* was well-received, however, user's felt the UI to be complex and overwhelming.

Yet, it can be stated both based on the qualitative and the quantitative feedback, that *Prototype C* was the most promising candidate, therefore, it will be the base prototype I will use for the synthesis.

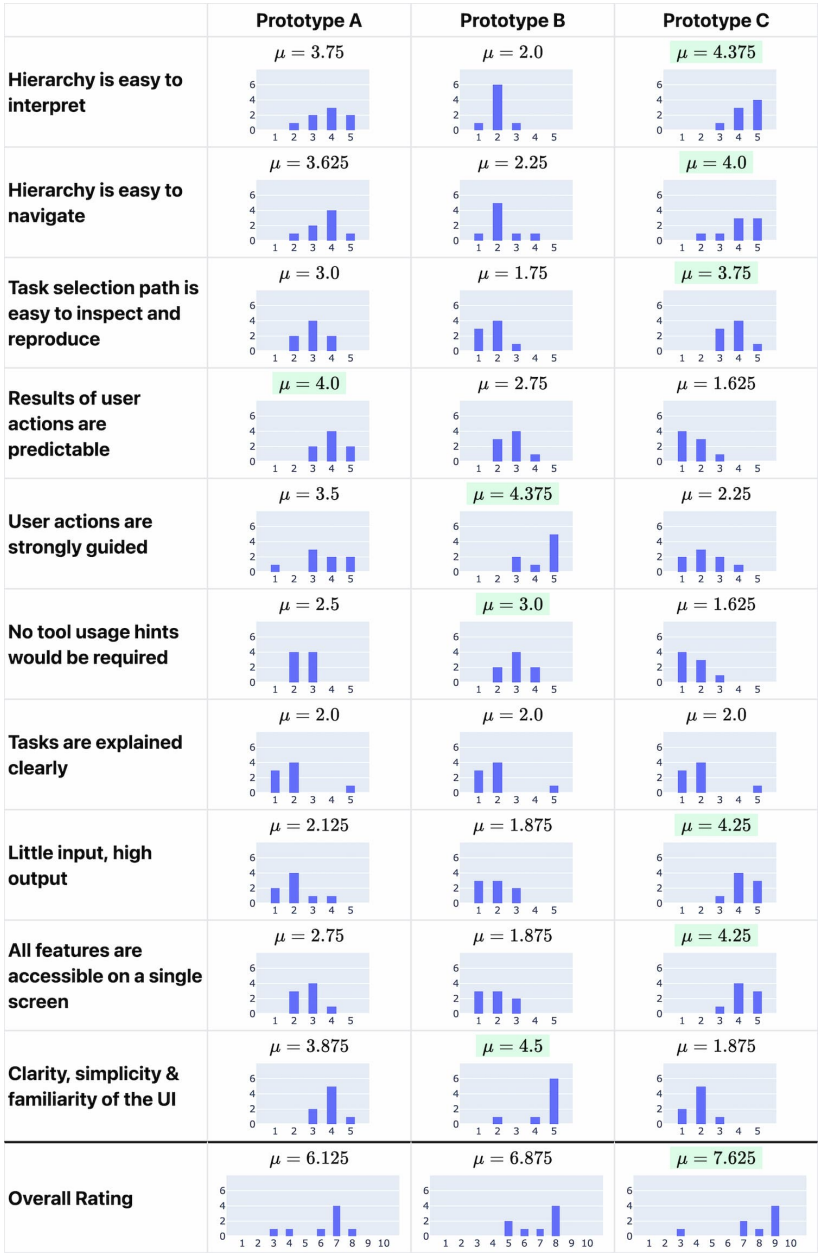

Figure 9: Histogram matrix of the distilled results of the qualitative interviews after the initial prototyping phase

3.6 Result Synthesis

After the collection and evaluation of the qualitative feedback on the initial low-level prototypes, as *Figure 9* shows, *Prototype C* triumphed over the other ones to a notable extent thanks to its single-page-application nature. As a result of this, when synthesizing the prototypes, it was central to preserve the core characteristics of *Prototype C* while focusing on remedying its highlighted shortcomings, such as the absence of usage hints and unpredictable user action results. Two main ideas have been developed and evaluated with users, presented below.

To address the lack of in-app guidance which was pointed out by the participants testing the initial low-level prototypes, both of the below-presented ideas involved a step-by-step onboarding section, guiding the user through the most pivotal parts of the dashboard.

Users also expressed that they would prefer to see possible visualizations of their data first and they would consider VIS tasks only after being presented with the recommendations. They justified this with their lack of familiarity with VIS concepts, claiming that they are often not aware of what task would be sensible for them in the first place. Therefore, they envisaged a feature to help them explore their possibilities first and only then consider how suitable the recommendations are for specific VIS tasks, giving an educative flavor to the tool.

The qualitative feedback provided by the users gave the project a clear vision of the direction in which the tool should be heading. We proposed two variations of this vision, having different levels of complexity. The reason for developing two synthesized prototypes instead of one is that we wanted to test how different approaches with regards to encoding ranking and making use of screen real estate would be received by the users. In what follows, I elaborate on the variations in detail.

3.6.1 Synthesis 1

The rationale behind this prototype was to simplify the user interface and make it less crowded, hence easier to navigate. To do so, instead of displaying the recommendations in the matrix cells repeatedly, color coding has been introduced to refer to visualizations via categorical mapping. Another motive for this decision was to prevent wasting screen real estate caused by encoding the same idioms over and over again.

3.6.2 Synthesis 2

Similarly to *Synthesis 1*, the main objective of the design below is to provide the intended features as a single-page application while refraining from making the user interface unnecessarily complex. The presented prototype encodes the recommended visualizations and their ranking with regards to the VIS tasks as

a heatmap, where the color coding is based on the ranking score of the visualizations. The color of a cell encodes the overall ranking of a recommendation, while its vertical position encodes the task-specific rank belonging to the task displayed in the column header.

Figure 10: Synthetized Prototype 1

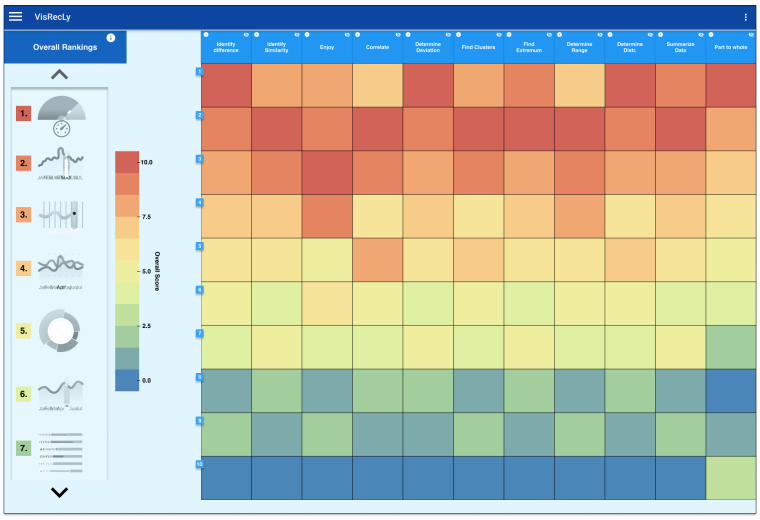

Figure 11: Synthetized Prototype 2

Qualitative feedback about the synthesized prototypes was collected from four participants, using a questionnaire structured similarly to the interviews conducted for the initial low-level prototypes. As *Figure 12* testifies, the feedback of the participants was straightforward, *Synthesis 2* was their preferred alternative. While the participants argued that *Synthesis 1* was simpler and thought that the features are more accessible to the user, they recognized the advantages provided by *Synthesis 2* in that one can observe all the recommendations in a single list as well as it is possible to get a more detailed overview about the VIS-task-specific rank of the recommendations.

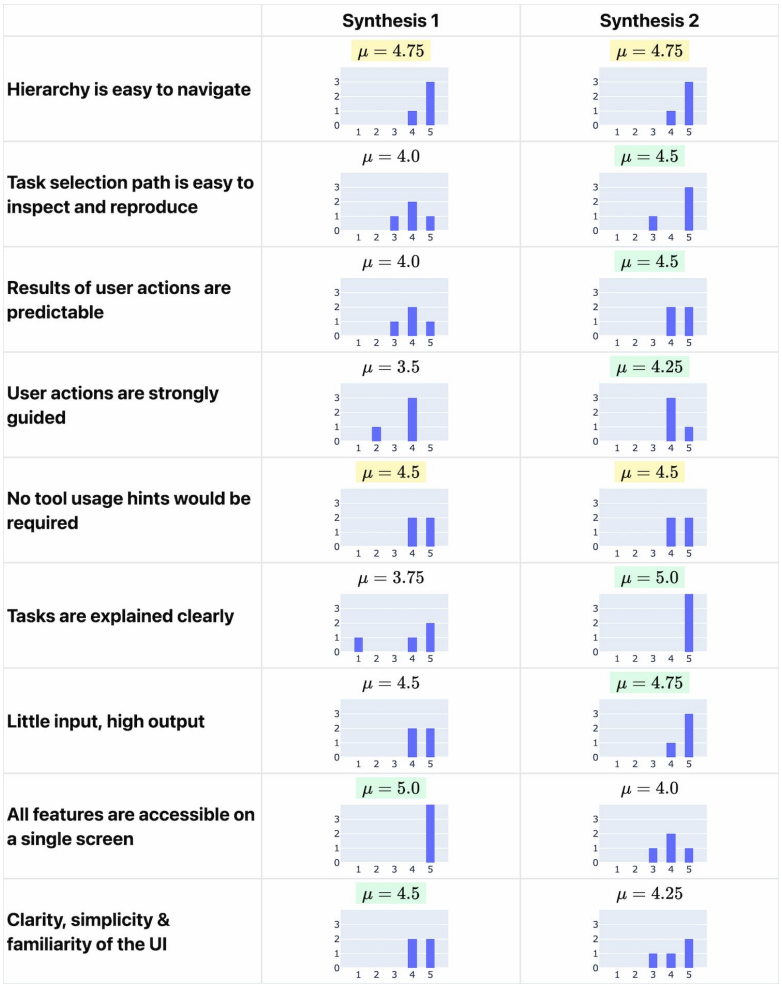

Figure 12: Histogram matrix of the distilled results of the qualitative interviews after the prototype synthesis phase. Green highlighting is used to indicate that a prototype obtained a higher average rating than the other one. Yellow highlighting indicates that the average rating of the two prototypes is the same.

As a final synthesis step, following the recommendations of the participants, the preferred *Synthesis 2* prototype has been altered such that the configuration sidebar is open by default - making all the app features accessible through a single page - and the continuous heatmap color scale has been simplified to an easier-to-perceive alternative, to a four-tier color coding, indicating four distinct rank categories: *Good*, *Average*, *Poor* and *Bad*.

The low-level prototyping phase got completed after these concluding alterations, making it possible to transition to the implementation of a high-fidelity prototype.

3.7 Outcomes & Reflection

The low-level prototyping phase comprised a substantial amount of effort and accordingly, it yielded valuable learnings and insights on understanding what novice VIS users would appreciate in a task-based VisRec system, - and as RQ2 also queries - which user interface design principles should be followed to maximize the interpretability of the generated & ranked recommendations.

Users expressed general sympathy towards truly single-page prototypes which exposed all their features in a single, interactive view. Participants highlighted that multi-page alternatives were easier to operate due to the reduced number of simultaneously available interaction options, nevertheless, they imposed a heavier cognitive load on them, as they had to remember their configurations and past actions to fully understand the system's output. This characteristic made multi-page alternatives less appealing for novice users, as trying to remember recent actions in an unfamiliar context is inherently challenging.

Furthermore, a crucial finding extracted from the interviews was that novice VIS users prefer being presented with recommendations out of the box, with the least amount of configuration needed on their part. They were also inclined towards a more didactic, educational approach, wishing for a unified overview of all recommendations and their usefulness for VIS tasks, over the ability to configure the recommendation output by pre-selecting tasks. Study participants recognized that this would come with the drawback of being more limited when it comes to recommendation output customizations, however, they emphasized that they would appreciate keeping the complexity of the system as low as possible.

As a piece of reflection, it is crucial to note that the above-mentioned takeaways are not completely generalizable to all user groups as the number of participants was limited and their background was specific to the personas predefined for this project out of scope-restricting purposes.

4 Implementation: High-fidelity Prototyping

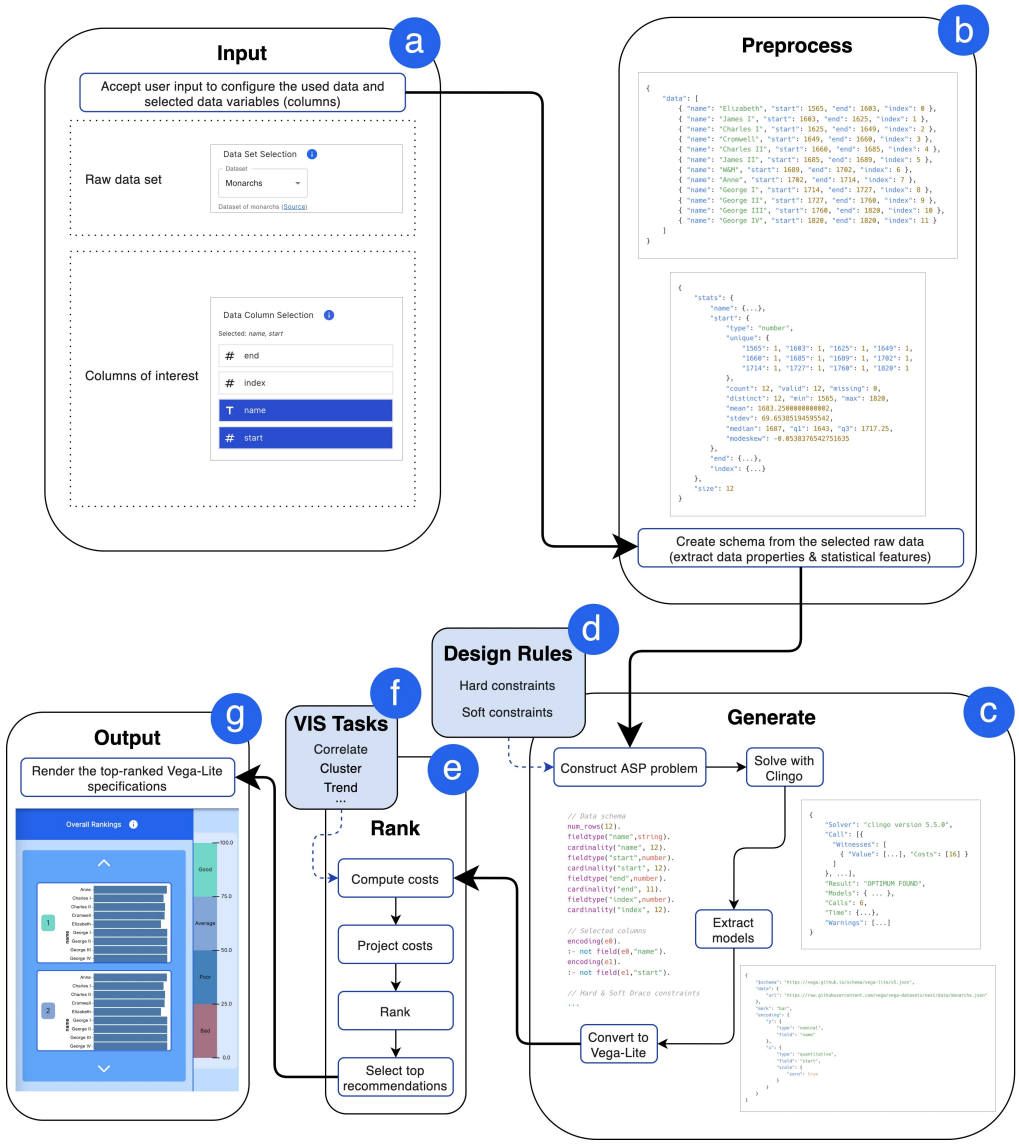

Figure 13: Architectural overview of *VisRecly*. **(a)** Input: accepts user input. **(b)** Preprocess: prepare data schema. **(c)** Generate: Solve ASP problems and produce Vega-Lite specifications. **(e)** Rank: Compute recommendation costs and rank visualizations. **(f)** VIS Tasks: The knowledge base declaring the mark preferences of analytical tasks, used for computing costs. **(g)** Output: Render the ranked visualizations. **(d)** and **(e)** are knowledge sources for design guidelines and VIS tasks.

4.1 Approach

As one of the most important takeaways from the low-fidelity prototyping phase was that users prefer simplicity and accessibility over all other considered aspects, the high-fidelity version of the system was designed to support these principles. The introduced tool was built around the principle that novice users cannot be expected to really know what they need and what configurations might be the most useful for them in a completely new, unfamiliar domain. Therefore, *VisRecly* attempts to minimize the user's involvement with the input's specification, letting them channel their full attention to the system's output.

Due to the fact that virtually all members of the tool's target audience have access to the internet and to a browser on a device, *VisRecly* was implemented as a web-based visualization recommendation system in order to make accessing and using the application as easy as possible.

As *Figure 13* illustrates, the system is built with a modular architecture, where the user needs to interact only with the *Input* (**(a)**) and *Output* (**(g)**) modules. The only configurations a user needs to specify are the data selection and the data column selection settings to start generating recommendations. Solely a general knowledge about the chosen dataset is foreseen on users' part, they are not expected to have any prior knowledge or understanding about VIS tasks in order to generate visualizations, making the system more accessible to novices.

In contrast with *TaskVis* [56], presented by Shen et al., *VisRecly* does not expose advanced configuration options such as choosing from available visualization ranking schemes or defining tasks explicitly, as users during the low-fidelity prototyping phase made it clear that such functionalities with their level of expertise would be more confusing than useful. Instead, *VisRecly* makes use of sensible default configurations in the background, hiding the inner workings of the system from the end-user, thus contributing to a more seamless and novice-user-friendly experience.

Also opposed to *Voyager 2* [69], developed by Wongsuphasawat et al., *VisRecly* handles the visual encoding - and the selection of the corresponding channels - of selected data columns automatically, without imposing additional cognitive load to novice users by presenting them with options to encode their data using channels of their choice.

A further crucial aspect considered during the implementation of the high-fidelity prototype was performance. As a result of the nature of this tool - and visualization dashboards in general - high interactivity with low execution overhead is desired in order to keep users focused and prevent dislocating them out of their flow. In the case of *VisRecly*- a tool that primarily targets novice users - this is even more critical, as most likely they are more impressionable by an unfamiliar tool with poor performance.

4.2 Architecture

While *Figure 13* exhibits the conceptual architecture of the system in some detail, in what follows, I introduce each module comprehensively along with their software counterparts.

The software implementation of the prototype is made up of six packages:

- *libs/data*: Houses example datasets and related utilities.
- *libs/draco*: The core of the underlying recommendation engine, defining learning and visualization design guidelines as Answer Set Programming (ASP) problems. It is a custom fork of Draco by UW Interactive Data Lab [43].
- *libs/draco-web*: Custom web-API leveraging the core API introduced in *libs/draco* and *clingo-wasm* [42] to solve ASP programs in the browser, eliminating the need for a server component.
- *libs/ranking*: Defines the ranking algorithm and aggregator utilities, considering both data-oriented costs (obtained from *libs/draco-web*) and VIS-task-relevant preferences.
- *libs/vis-tasks*: Defines the VIS tasks and their associated mark preferences.
- *apps/dashboard*: The actual client of the above modules, the dashboard that allows users to steer their desired tasks and marvel at the generated Vega-Lite-based visualizations.

4.2.1 Input

The *Input* module is responsible for accepting user input based on which recommendations can be generated. Users have the chance to choose from a predefined list of datasets. The rationale behind the decision not to support custom data upload for the presented high-fidelity prototype was twofold: on the one hand, I wanted to facilitate the flow of the usability study interviews by not expecting to bring properly formatted data and on the other hand, I wanted to restrict the - already wide - scope of this thesis.

The below-listed sets of data are selectable by the user, all sourced from *vega-datasets* [20], a common repository for example datasets used by Vega-related projects. They were hand-picked to provide the users of *VisRecly* with a variety of options regarding the data size as well as the available data column types.

- Barley: The result of a 1930s agricultural experiment in Minnesota, this dataset contains yields for 10 different varieties of barley at six different sites.
- Cars: This was the 1983 ASA Data Exposition dataset. The dataset was collected by Ernesto Ramos and David Donoho and dealt with automobiles.

- Monarchs: Dataset of monarchs.
- Movies: Dataset of movies. The dataset has well-known and intentionally included errors, making it possible to test the system's robustness.
- Wheat: In an 1822 letter to Parliament, William Playfair, a Scottish engineer who is often credited as the founder of statistical graphics, published an elegant chart on the price of wheat. This dataset was the underlying source of the visualization.

After making a selection, the user also gets the chance to select data columns to be encoded in the generated visualization recommendations. It is possible to select up to five data columns at the same time, selecting more than this threshold is not supported to make sure that the Clingo-solver running in the user's browser does not run out of memory, potentially leading to freezing the used browser window.

The available datasets and their metadata are stored in the *libs/data* package, while the user interface through which the selection is made possible is implemented in the *apps/dashboard* package.

4.2.2 Preprocess

After the user selects raw data, a new Draco instance gets created, by tapping into *libs/draco* and *libs/draco-web*. The data schema gets generated automatically, describing the data column types as well as properties such as cardinality, number of distinct values, minimum value, maximum value, mean, standard deviation and median. This step is essential to generate recommendations since the characteristics of the raw data, the output of the preprocessing phase is used as the input to construct ASP problems in the generating phase.

4.2.3 Generate

Building upon the hard constraints and soft constraints defined based on empirical data as well as machine learning techniques, defined in Draco [43] (also denoted as module **(d) Design Rules**), the full ASP problem is generated with two dynamic parts: definitions of the selected data columns as well as characteristics of the selected dataset. A selected data column called would be transformed into an ASP program chunk as illustrated below.

```
encoding(e0).
:- not field(e0, "name").
```

The logic program above declares that *e0* is an encoding and that it cannot be the case that *"name"* is not a field encoded onto *e0*, essentially enforcing to find a visual encoding for the *"name"* data column.

This dynamically-constructed ASP problem is piped into Clingo [7], a grounder and solver for logic programs.

The models of the problem generated by Clingo get extracted and converted into a Vega-Lite JSON specification, to make them easy to consume and render by a client program. *Figure 14* presents the previously mentioned ASP input, Clingo's output, as well as the resulting Vega-Lite specification.

Operations in this phase are handled by the *libs/draco-web* and *libs/draco* software packages. The Vega-Lite JSON outputs and the source Clingo models generated by this module get piped into the ranking module in order to compute costs based on the marks used by the Vega-Lite recommendations and the costs associated with the Clingo models.

```
// Data schema
num_rows(12).
fieldtype("name",string).
cardinality("name", 12).
fieldtype("start",number).
cardinality("start", 12).
fieldtype("end",number).
cardinality("end", 11).
fieldtype("index",number).
cardinality("index", 12).

// Selected columns
encoding(e0).
:- not field(e0,"name").
encoding(e1).
:- not field(e1,"start").

// Hard & Soft Draco constraints
...
```

```
{
    "Solver": "clingo version 5.5.0",
    "Call": [{
      "Witnesses": [
        { "Value": [...], "Costs": [16] }
      ]
    }, ...],
    "Result": "OPTIMUM FOUND",
    "Models": { ... },
    "Calls": 6,
    "Time": {...},
    "Warnings": [...]
}
```

```
{
    "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
    "data": {
        "url": "https://raw.githubusercontent.com/vega/vega-datasets/next/data/monarchs.json"
    },
    "mark": "bar",
    "encoding": {
        "y": {
            "type": "nominal",
            "field": "name"
        },
        "x": {
            "type": "quantitative",
            "field": "start",
            "scale": {
                "zero": true
            }
        }
    }
}
```

Figure 14: Textual representation of the ASP problem, its solution and the yielded Vega-Lite specification for the scenario when the *name* and *start* columns of the *Monarchs* dataset are selected by the user in module **(a) Input**.

4.2.4 Rank

The ranking module is responsible for assigning VIS-task-specific costs and an overall cost to each recommendation represented as a Vega-Lite JSON produced by module **(c) Generate**. The task-specific costs are computed based on the knowledge base recorded in module **(f) VIS Tasks**. Each VIS task has a list of favored mark types. For each recommendation, an individual task-specific cost is computed for each task by checking whether the Vega-Lite JSON representing a visualization makes use of a visual mark that is preferred by the current task. The supported tasks and their predefined mark preferences are listed in *Table 3*.

VIS Task	**Description**	**Favored Mark(s)**
Change over time	Analyse how the data changes over time series	line, area
Characterize distribution	Characterize the distribution of the data over the set	bar, point
Cluster	Find clusters of similar attribute values	bar, point
Comparison	Give emphasis to comparison on different entities	line, point, bar
Compute derived value	Compute aggregated or binned numeric derived value	line, point, bar
Correlate	Determine useful relationships between the columns	bar, line
Determine range	Find the span of values within the set	tick
Deviation	Compare data with certain values like zero or mean	bar, rule
Filter	Find data cases satisfying the given constraints	rect, bar, arc
Find anomalies	Identify any anomalies within the dataset	bar, point
Find extremum	Find extreme values of data column	bar, point
Magnitude	Show relative or absolute size comparisons	arc, bar
Part to whole	Show component elements of a single entity	arc
Retrieve value	Find values of specific columns	rect
Sort	Rank data according to some ordinal metric	bar
Trend	Use regression or LOESS to show the variation trend	point, line

Table 3: Overview of supported VIS tasks and their favored marks, as compiled by Shen et. al [56]

The overall cost is computed by a function taking the data-oriented cost (the cost of the Clingo model from which the investigated Vega-Lite specification was generated) and the task-oriented cost. The function computes the mean cost of all tasks and sums this up with the data-oriented cost, yielding the overall cost. These computed costs get projected so that each value falls into the range between 0 and 100 inclusively.

The projection step is introduced in order to present users with a familiar scoring scheme. As a final step, the recommendations are sorted in ascending order by cost (indicating that a lower cost represents a better recommendation) and the top 15 recommendations are returned. These top-ranked VisRecs get piped into module **(g) Output** to be displayed to the user. The entire ranking logic is implemented in the *libs/ranking* package of the high-fidelity prototype.

4.2.5 Output

The visualization recommendations generated by module **(c) Generate** and ranked by module **(e) Rank** are presented to the user via the web component of *VisRecly*, implemented in *apps/dashboard*. As *Figure 15* illustrates, the recommendations appear in a scrollable list, enriched with a color coding to convey the rank-category of a recommendation.

Note that the costs of the recommendations get inverted for this step so that a cost of 100 represents the best recommendation and a cost of 0 stands for the worst one. This inversion was introduced to make the ranking logic more intuitive to the user.

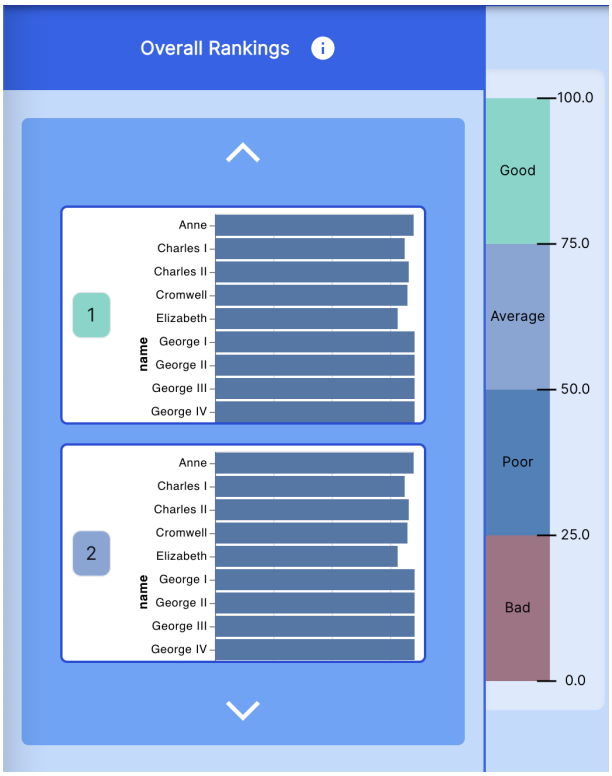

Figure 15: User interface segment of *VisRecly*, featuring a scrollable list of the ranked recommendations, along with a color scale conveying the rank-category of each visualization. Possible categories are: *Good*, *Average*, *Poor* and *Bad*.

4.3 Considered Alternatives

The high-fidelity prototyping phase began by defining the modules introduced by *Figure 13* as well as their responsibilities on a conceptual, framework- and library-independent level. The most crucial decisions I had to make were about the technologies to be used for the presentation layer (the user-facing application itself) as well as the recommendation engine. In what follows, I introduce the considered alternatives along with their trade-offs in detail.

4.3.1 Presentation Layer

Due to the fact that *VisRecly* targets novice VIS users as its audience and aims to be as accessible as possible, the primarily considered target platform was the web. On the one hand, this makes it possible to use an arbitrary web browser to interact with the application online and on the other hand - thanks to existing modern web technologies - it also makes an entirely offline version of the tool feasible in the form of a Progressive web application (PWA) [13] as future work on the project.

Challenges related to web development involve handling responsive design, that is, displaying the dashboard of *VisRecly* in an aesthetically pleasing manner on a variety of browser sizes without the need to hide indispensable components of the user interface. Furthermore, coming up with an implementation that not only is highly interactive but also performant on the user's side but at the same time can be developed in rapid iterations also required careful planning.

The considered web frameworks providing a solution for the above-mentioned demands were:

- **Nuxt.js** [5]: Vue.js-based framework
- **SvelteKit** [12]: Svelte-based framework
- **Next.js** [21]: React.js-based framework

All of these frameworks have a vibrant developer community behind them, offering both statically-generated Single-page-application-like options as well as an opt-in possibility for executing server-side operations.

Frameworks built on top of Vue.js and React.js come with performance degradations compared to the Svelte-based framework, because the former libraries make use of the virtual DOM (an in-memory representation of the Document Object Model) to rerender pieces of the web application reactively, while Svelte uses native browser features to achieve the same functionality.

On the other hand, Vue.js and React.js are both more mature & developed than Svelte, having more third-party libraries available for them as well as an ampler community helping developers in seeking answers to common technical problems with the libraries.

A distinguishing advantage of using React.js for this project would be that a stable renderer library [19] for Vega-Lite specifications is available for it as well as SVG-based visualization components are supported by the *airbnb/visx* [15] library, facilitating the implementation of *VisRecly*'s heatmap component.

4.3.2 Recommendation Engine

Since Vega-Lite is a well-supported, lightweight and easy-to-use tool to create an expressive range of visualizations for data analysis and presentation, I was sure that I will make use of a recommendation engine base that produces Vega-Lite specifications as its output. In order to keep the scope of this Bachelor's thesis as reasonable as possible, it was also a hard requirement to take advantage of an already developed, well-supported open source project to handle generating recommendations. I explored three options in detail:

- **CompassQL** [67] [18]: the query language used to produce visualizations in Voyager 2 [69].
- **Draco** [44] [43] & **draco-vis** [17]: a library defining VIS design knowledge as constraints as well as utility functions to convert constraint solution sets to Vega-Lite specifications accompanied by a web-based wrapper to solve ASP problems in the browser.
- **Draco 2** [16]: the official successor of Draco, being actively developed as an open source project by Dominik Moritz et. al at the Carnegie Melon University Data Interaction Group.

CompassQL is a proven, robust piece of software, powering chart specifications in Voyager 2. It allows constructing queries represented by JSON objects, specifying a collection of queried visualizations, wildcards, grouping and nesting methods as well as ordering preferences. The drawbacks of using CompassQL in a new project are that the latest release of the package at the time of writing dates back to 2019, leading to several dependency conflicts and that it lacks detailed, easy-to-follow documentation about its API, leading to an initial development overhead.

Draco is a library built around the idea of expressing visualization design guidelines formally as constraints interpretable as logic program sections, making it so expressive that it can be used to reimplement CompassQL [44]. It comes with a rich set of design constraints, and at the same time, the used constraints can be extended flexibly on demand. Draco has no extensive external documentation, however, its source code is self-documenting thanks to the meaningful comments, the usage of TypeScript and the presence of unit tests. *draco-vis* is a dedicated library with the promise of providing web-friendliness for Draco, written in TypeScript, providing a bundled WebAssembly Clingo module, which sounds promising for projects targeting the web as their primary platform.

Unfortunately, *draco-vis* does not deliver on its promise, as it is also not actively

maintained anymore and *wasm-clingo* [41], the WebAssembly Clingo module it uses is not maintained anymore, it has been officially deprecated and archived in 2021 in favor of *clingo-wasm* [42], for which - at the time of writing - no standalone Draco-integrations are available in the open source community. Considering Draco, similarly to CompassQL, a notable drawback of it is that it is not usable out-of-the-box for new projects with fresh dependencies, since its latest release also dates back to 2019. Furthermore, Draco shows signs of deprecation, as it is made clear in the repository's *README.md* file that the authors are actively working on an improved version of the project, namely on Draco 2.

Draco 2 is a highly promising successor, in that it is written uniformly in Python but is still built around the initial idea of using ASP for the representation of knowledge bases. It provides a more extensible, generalized and extended chart specification format inspired by Vega-Lite and it supports multiple views and view compositions, allowing for generating highly sophisticated visualizations. Another advantage of Draco 2 is that it has a more advanced development tooling and a dedicated documentation website [14] making the work with it highly developer-friendly.

The most significant drawback of choosing Draco 2 as a recommendation engine is that it is still a work in progress, meaning that essential features are still not implemented, such as having a default renderer for its Vega-Lite-inspired (but not backward-compatible) visualization specification format. Furthermore, since the project is written in pure Python, it cannot be used in an entirely web-based application, without implementing a server component to expose its API.

4.4 Final Design

After several careful planning sessions and drawing from the experience I gained from developing throw-away proof of concept prototypes, striving to find the most optimal stack for *VisRecly* the final design of the system presented itself. In what follows, I elaborate and justify which combination of the previously introduced alternatives I picked as well as describe *VisRecly*'s design from a software engineering point of view.

4.4.1 Chosen Alternatives

After a few rapid prototyping sessions with the web frameworks - as the advantages listed in the *Presentation Layer* section also implied - Next.js with React.js had clear leverage over the other considered alternatives. React has first-class support for particular types of libraries I planned to use to implement *VisRecly*while Next.js handles code-splitting and optimization tasks out of the box. I managed to conclude that my pick's performance drawback relative to Svelte is negligible compared to the value it provides through its exceptional developer experience and the rapid development speed it offers.

Committing myself to a recommendation engine was a more challenging task

than I anticipated, because - opposed to my initial assumptions - none of the alternatives I investigated in the *Recommendation Engine* section were usable for a fresh project out of the box, without the need to tap into their source code and handle package updates and other related adjustments. As soon as I realized that adjustments will be necessary regardless of the alternative I choose, I evaluated which engine would be the best pick for *VisRecly* from a conceptual perspective.

I excluded CompassQL from the considered alternatives first, since it is a tool developed specifically to solve the problem of generating charts from partial specifications, allowing the user to specify some of the encoding channels, but not all of them. Since *VisRecly* is meant for novice VIS users, it is not realistic to expect them to specify encodings at all. All in all, while CompassQL is a powerful tool for the specific goal it is meant to achieve, it is not possible to extend & customize it to a sufficient extent for the needs of *VisRecly*.

At this point, deciding between Draco and its - powerful, yet relatively immature - successor Draco 2 was straightforward: I chose Draco for its maturity and stability, accepting that I need to adjust its source code to support the most recent package versions and that I need to find a way to make it web-compatible by integrating *clingo-wasm* into it. The choice of Draco 2 would have imposed more difficult tasks on me, making the scope of this thesis a lot more ample: the development of a stable default renderer for its custom chart specification and the creation of a server component to expose its Python API in an easy-to-consume manner would have been inevitable.

4.4.2 Software Design

Given the highly modular nature of *VisRecly*, I did not even consider a monolithic design, I planned the actual software implementation to have reasonably low coupling between the modules, advocating the well-known Single-responsibility principle. In order to facilitate the build tasks and to accelerate the development process while using robust solutions, I decided to set up the repository of *VisRecly* as a monorepo[4], using Nx [22]. The final software module structure of the app is as follows:

- *libs/data*: Houses example datasets and related utilities.
- *libs/draco*: The core of the underlying recommendation engine, defining learning and visualization design guidelines as Answer Set Programming (ASP) problems. It is a custom fork of Draco by UW Interactive Data Lab [43].
- *libs/draco-web*: Custom web-API leveraging the core API introduced in *libs/draco* and *clingo-wasm* [42] to solve ASP programs in the browser, eliminating the need for a server component.

[4] A repository that contains multiple sub-repositories, each being responsible for a specific part of the software implementation

- *libs/ranking*: Defines the ranking algorithm and aggregator utilities, considering both data-oriented costs (obtained from *libs/draco-web*) and VIS-task-relevant preferences.
- *libs/vis-tasks*: Defines the VIS tasks and their associated mark preferences.
- *apps/dashboard*: The actual client of the above modules, the dashboard that allows users to steer their desired tasks and marvel at the generated Vega-Lite-based visualizations.

Note the absence of a server component of the final software design: it is made possible by delegating the task of solving ASP programs with Clingo to the user's browser by leveraging WebAssembly. This provides additional performance benefits, as no networking is necessary for solving the ASP programs between the web client and a server exposing an API for the logic grounder.

Following software engineering best practices, continuous delivery has been set up for *VisRecly* using Vercel [25] as the deployment platform, making it possible to access preview deployments and production deployments automatically after opening a pull request in *VisRecly*'s GitHub repository and after pushing contributions to the *main* branch.

4.5 Functionality Overview

The following sections present an overview of *VisRecly*'s features accessible through a web module, realized by implementing the previously explained architecture and software design.

4.5.1 Core Features

The dashboard's core functionality includes accepting user input, rendering the ranked recommendations and making it possible to explore both the overall and the task-specific ranks of recommendations. Furthermore, users also get the chance to discover recommendations in detail and export them in image or vector graphic format.

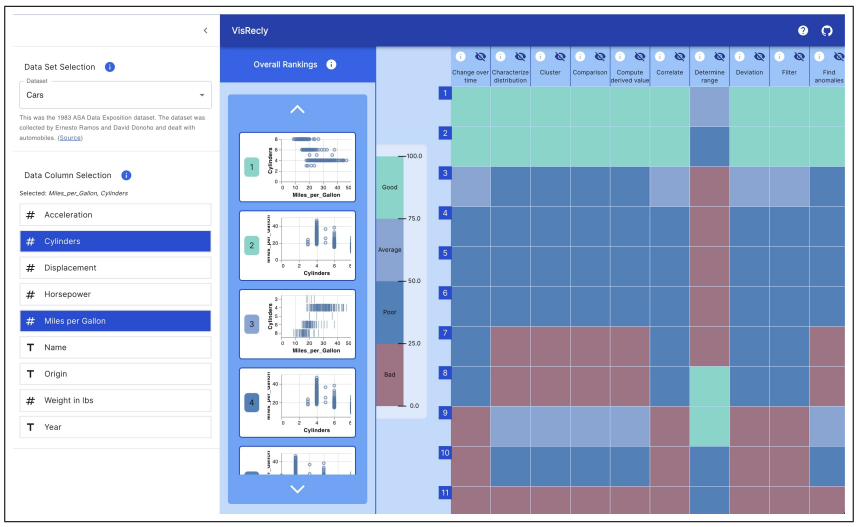

Figure 16: User interface of *VisRecly* with having the *Cylinders* and *Miles per Gallon* columns of the *Cars* dataset selected.

Figure 16 illustrates the user interface of the tool presented to the user upon navigating to the app with their browser. The side panel allows for selecting a dataset and associated data columns via dedicated selector components. Recommendations are generated and ranked automatically, displayed in the overall ranking list. The overall rank of recommendations gets categorized and color-coded into four categories based on their overall score (Good, Average, Poor and Bad), displayed by a color scale. Related visualization tasks are presented as the header of the central heatmap. Each heatmap cell refers to a recommendation item on the overall ranking list. The color of a cell conveys the overall rank of the recommendation associated with the cell, while its vertical position in the heatmap represents its task-specific rank. Users can access in-app explanations of the UI on-demand.

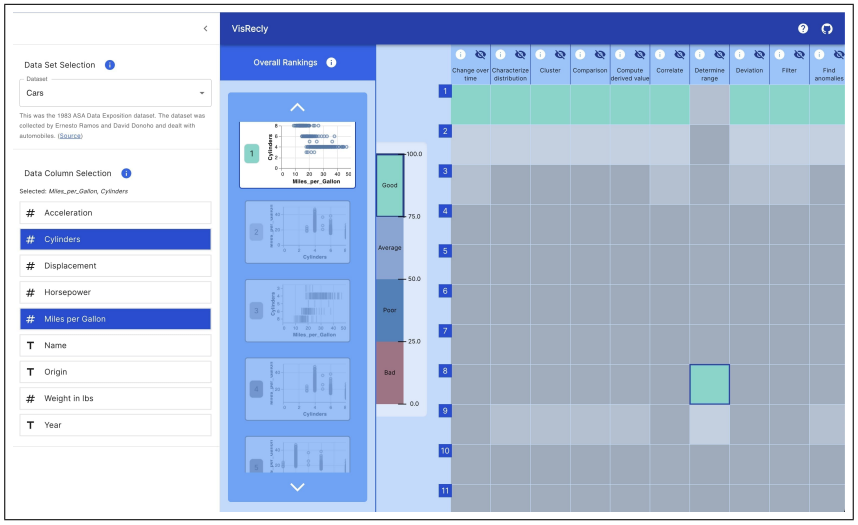

Figure 17: Exploring the recommendations with the help of linked highlighting. Hovering over the 8th cell of the heatmap in the *Determine range* task's column highlights the first item in the overall ranking list, implying, that the chart ranked as the 1st overall is the 8th best solution for the *Determine range* task relative to the other recommendations.

Figure 17 presents the linked highlighting technique featured by *VisRecly* to allow for making logical connections between the overall rank of recommendations and their task-specific ranks. Upon hovering over an item in the overall ranking list, all the heatmap cells associated with the activated recommendation get highlighted, while the other cells fade out to assist the user in focusing on a reduced number of visual inputs. As the vertical positions of heatmap cells represent the task-specific rank of a recommendation, the user can quickly explore how the activated recommendation performs across the different tasks.

The linked highlighting works from the other direction too: the user can answer questions such as *"Which recommendation is the 8th best solution for the Determine range task?"* by hovering over the 8th cell of the heatmap in the *Determine range* task's column. Upon activating the cell, the recommendation in the overall ranking list gets highlighted automatically, clearly identifying the chart associated with the hovered cell.

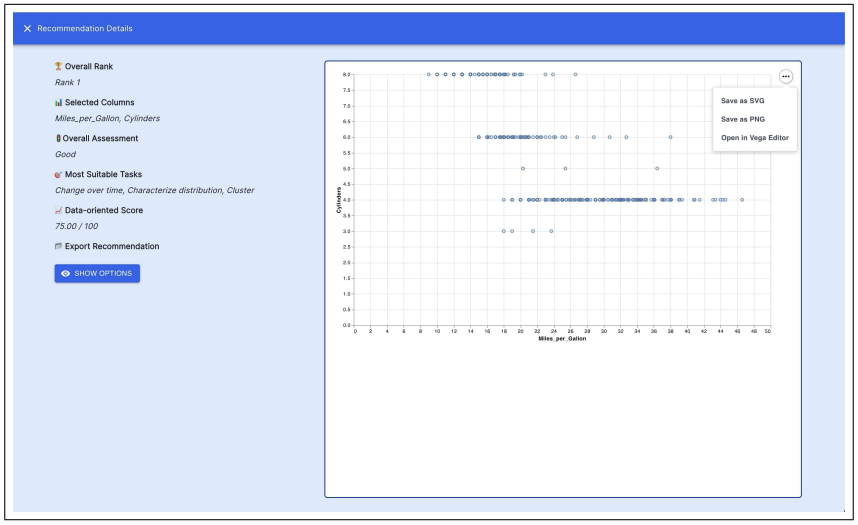

Figure 18: Detail view of a visualization recommendation, appearing after the user clicks on a heatmap cell of an item in the overall ranking list. Additional recommendation details and export options are available in this view.

Apart from the quick view presented in the form of the overall ranking list's items, a full-screen detail view is also available for each recommendation, as *Figure 18* depicts. Apart from featuring the recommendations in a reasonably-sized viewport, an overview of the user's data column selection is also exhibited along with the overall assessment, a list of the top three most suitable tasks for the active chart as well as its score in a numerical representation. In an attempt of adding an additional layer of usefulness to *VisRecly*, exporting visualization recommendations is also supported.

4.5.2 Assistive Features

Besides the array of core functionalities, *VisRecly* also exhibits features to make a novice user's experience more streamlined, to make the system overall more accessible and easy to use for experienced and novice users alike. In what follows, such pieces of functionality implemented for the high-fidelity prototype are described.

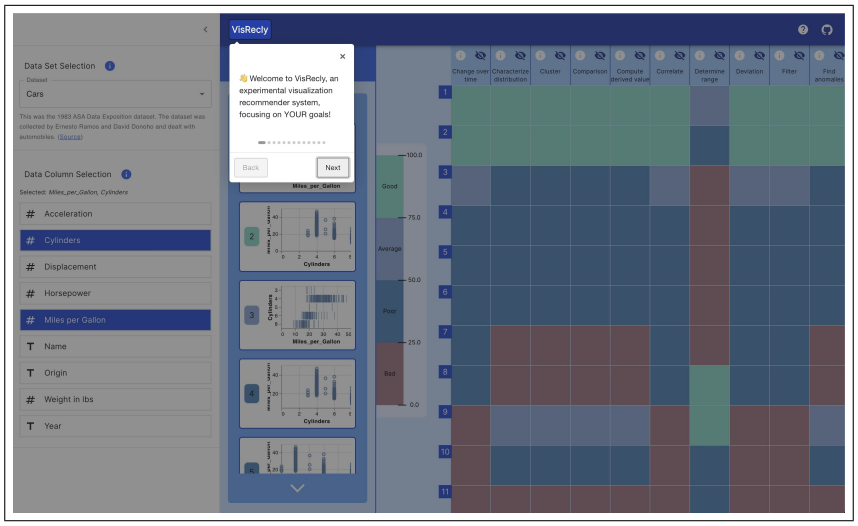

Figure 19: User interface of *VisRecly* featuring the first step of a detailed onboarding flow, appearing automatically on a user's first encounter with the system.

Illustrated by *Figure 19*, *VisRecly* has a built-in onboarding section, helping first-time users to familiarise themselves with the concepts and ideas of the dashboard in a simple, step-by-step way, explaining the purpose of each user interface component. Each step highlights an actual UI section and describes how to interact with it, and how they should be interpreted in general.

While users are encouraged to not skip the introduction, nothing stops them to do so. In case of changing their mind, they can access the system hints whenever they wish to by clicking the question mark icon in the top right corner of the screen. They can either explore the steps in sequential order by using the *Back* and *Next* buttons, or they can jump between arbitrary steps by operating the dotted progress indicator provided by the intro component.

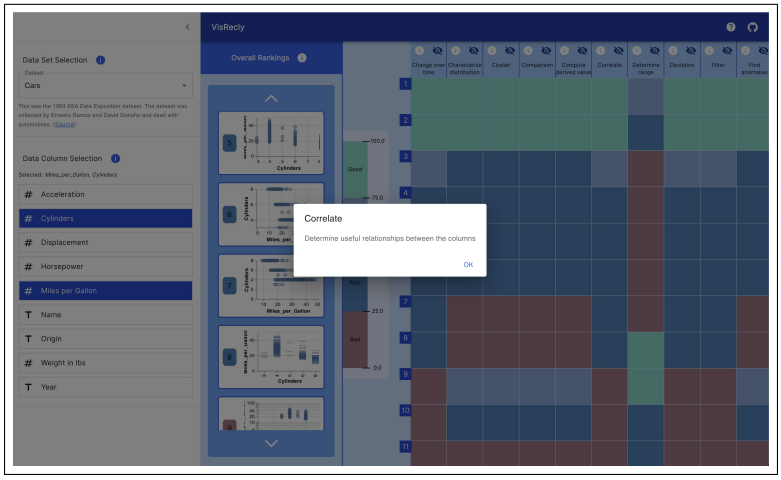

Figure 20: Additional information prompt about the *Correlation* visualization task. Task explanations are accessible on-demand for users by clicking the information icon of a task header cell.

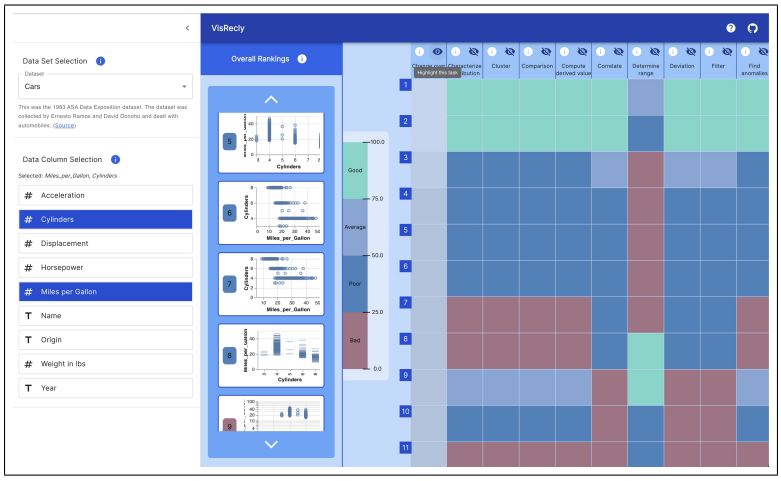

Figure 21: The *Change over time* task's column blended out from the heatmap, letting the user concentrate on other tasks. Task columns can be blended out and blended in by clicking the eye icon in the corresponding header cell.

Figure 20 and *Figure 21* reveal the task-specific accessibility operations: users get the chance to access explanations of tasks and they also have the possibility to blend them in and out if they wish.

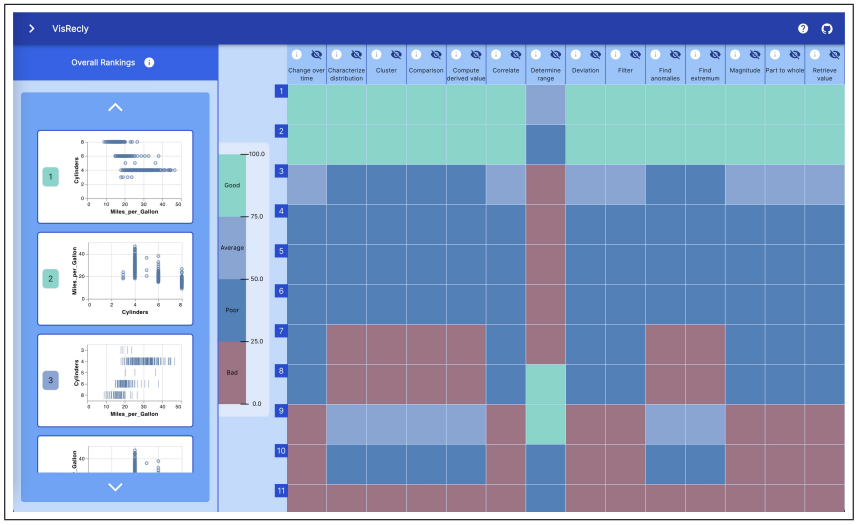

Figure 22: The side panel is hidden from the user interface, giving users more screen real estate to interact with the dashboard. The side panel's visibility can be toggled by clicking the chevron icon in the top-left corner.

Finally, *Figure 22* demonstrates *VisRecly*'s ability to let users adapt the dashboard's layout based on their viewport size needs. While *VisRecly* was developed with responsive web design in mind, users visiting the application with smaller screen sizes might still face inconveniences while interacting with it. Unless the user's screen size is not wide enough, a portion of the heatmap will overflow and will be visible only through scrolling.

By making it possible to hide the side panel, space is freed up on the horizontal axis of the layout, giving room for displaying more tasks without the need for horizontal scrolling, as well as making the chart previews exhibited in the overall ranking list bigger, hence more legible.

4.6 Method of Evaluation

VisRecly has been evaluated through a usability study, with the primary goal of investigating how usable and accessible the tool feels in the hands of novice users & to shed light on the viability of the approach implemented for the high-fidelity prototype.

4.6.1 Study Participants

Similarly to the study conducted for the low-level prototypes' evaluation, participants have been recruited through the dissemination of digital flyers about the study on online forums with a thorough description of the project and the role of a study participant in it.

15 persons have been invited to participate in formal study sessions (5 female, 10 male). The participants include 8 persons actively pursuing academic degrees in a scientific field [5], 5 persons categorizable as domain experts and 2 persons qualifying as VIS experts. The age of participants ranges from 22 to 32 years with a mean of 26 years. All of the participants have a strong command of interpreting quantitative data, and - with the exception of the VIS experts - match a novice VIS user's persona. The experts have been interviewed in order to provide more detailed feedback on the usability of the system as well as the usefulness and sensibility of the generated recommendations.

4.6.2 Usability Study Protocol

The study sessions were conducted as one-on-one Zoom calls with 13 participants, entirely remotely, while 2 of the participants preferred an on-site session. Participants have been informed about the nature and structure of the study in detail. Each of them granted their explicit, written consent to participate in the study.

Each session had three main parts: guided interaction, free exploration and comparison with a similar VisRec tool. Before commencing the guided interaction part of the study, participants were asked to perform a color blindness test[6] to make sure that their color vision is not impaired.

The guided interaction part involved the completion of five pre-designed real-life scenarios, with prompts formulated as listed below.

- **Scenario 1 - Onboarding**: *Please open the application and go through the onboarding steps which should appear automatically.*
- **Scenario 2 - Targeted Generation of Recommendations**: *Let's assume that you are interested in the US Gross and IMDB Rating variables of the Movies dataset. Please generate recommendations accordingly.*

[5] Computer science, Food engineering, Electrical engineering
[6] The following tool was used: https://enchroma.com/pages/color-blindness-test

- **Scenario 3 - Recommendation Export**: *From the previous scenario, please save the recommendation at overall rank 10 as a PNG image.*
- **Scenario 4 - Linked Rank Identification**: *Let's assume that you are interested in the Miles per Gallon and Horsepower variables of the Cars dataset. Please generate recommendations accordingly. Please identify the overall rank of the recommendation that is the best suited for the Determine Range task.*
- **Scenario 5 - VIS Task Descriptions**: *Let's assume that you are not familiar with the meaning of the Magnitude task. Please try to find a way in the application to get to know more about it.*

Directly after performing a scenario, users were asked to evaluate their experience by filling out an After-Scenario Questionnaire introduced by James R. Lewis in his article *IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use* [36]. This made it possible to get an impression of how satisfied users are with the ease and quickness of scenario completion as well as the in-app support information available to streamline their experience.

In the free exploration of the study, users were requested to interact with *VisRecly* freely, as if they were conducting a natural session with the application, without a time limit. After users indicated that they have explored the tool to a sufficient extent, they were presented with a SUS questionnaire [3], developed by John Brooke, providing a quick way to assess the system's usability scale.

Finally, users were requested to perform **Scenario 2 - Targeted Generation of Recommendations** again, however, using Voyager 2[7] instead of *VisRecly*, providing a baseline for comparison with another tool. The comparison and its results are discussed in finer detail in the upcoming comparison section.

4.6.3 Comparison

While TaskVis [56] would have been a great point of comparison for *VisRecly*, as both systems put emphasis on user tasks, at the time of writing, a fully working version of the tool developed by Shen et. al was not publicly accessible, rendering me unable to include it in the usability study. Therefore, the comparison was conducted with Voyager 2 [69], which is also a visualization tool for data exploration, letting its users generate recommendations.

Users were requested to perform **Scenario 2 - Targeted Generation of Recommendations** again using Voyager 2 and were asked to complete an After-Scenario Questionnaire afterward. Users were also encouraged to provide qualitative feedback on their experience with Voyager 2 as well as to compare the tools from a usability perspective.

[7] https://vega.github.io/voyager/

4.7 Usability Study Results

Each participant had a normal color vision, according to the results of the online color blindness test that was conducted before users started to interact with *VisRecly* in a study session.

4.7.1 Pre-designed Scenarios

As *Figure 23* illustrates, participants were generally satisfied with their user experience throughout performing the pre-designed scenarios. 12 of the participants outlined that they appreciated the presence of the interactive onboarding feature of the tool, giving an initial overview of the UI components as well as the main ideas of *VisRecly*. At the same time, some users revealed that they tend to skip introductory tutorials and prefer to discover the capabilities of an application on the fly using a trial-and-error approach.

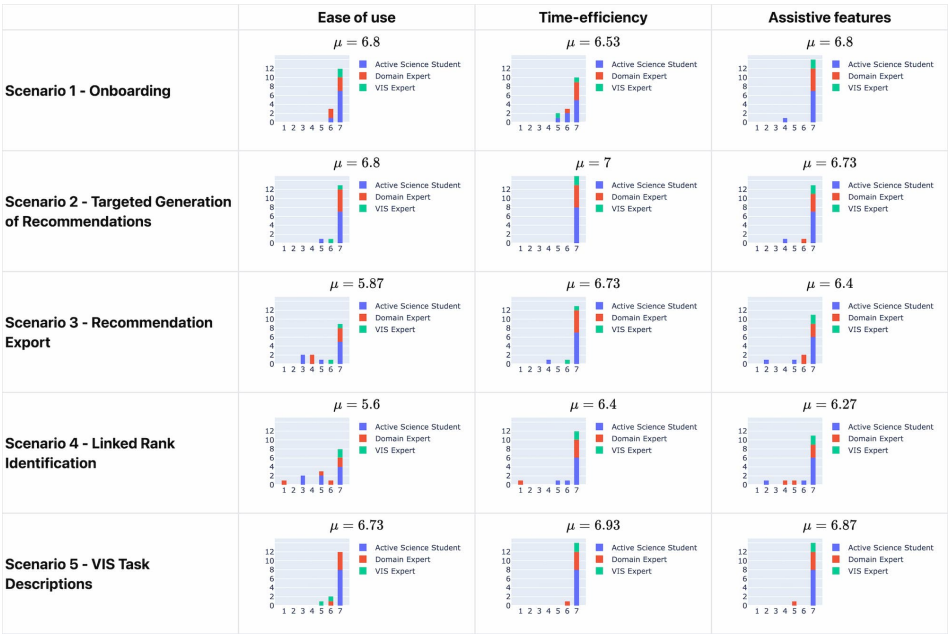

Figure 23: Summary of user responses to the After Scenario questionnaires presented to them after each pre-designed scenario. User types are visualized using stacked bar charts. The rating scale ranges from 1 to 7, corresponding to the *strongly disagree* and *strongly agree* statements.

While participants found the method of generating recommendations straightforward, users with less experience with dashboards and more complex web applications in general faced difficulties when they were asked to export a specific visualization recommendation into an image file. This was attributed to

the fact that the button triggering the context menu for a visualization on the detail screen was not easy to spot for everyone.

Scenario 4 - Linked Rank Identification proved to be the most challenging task for users. They were asked to identify the overall rank of the visualization which is the most suitable for a specific task. Performing this use-case correctly using *VisRecly* demands an understanding of the difference between the overall rank and the task-specific rank. Some participants confused the overall rank with the task-specific one indicated by the heatmap label, however, most of the users managed to state the correct answer after consulting the help feature of the system again. Users who spent more time on the initial onboarding were able to perform this scenario without problems. Participants who decided to only skim through the initial descriptions or skip them altogether had more trouble performing the use-case and rated the scenario from the *Ease of use* aspect more poorly.

Participants found accessing descriptions of VIS tasks intuitive and useful in general. However, some of the users who accessed the web application via a device with a smaller viewport had some difficulties performing the scenario, due to the fact that they had to scroll the heatmap horizontally in order to find information for the task mentioned in the actual scenario prompt.

Participants voiced that they are in general satisfied with the amount of time required to perform real-life scenarios using *VisRecly* and they also expressed their appreciation towards the performance and the reactivity of the system, allowing them to see the results of their actions almost instantaneously.

4.7.2 General Usability

The general overview *Figure 24* illustrates that *VisRecly* obtained an overall SUS score of 89.5, with the highest scores received by the *Active Science Student* group, followed by *Domain Experts* and *VIS Experts*.

As the detailed overview given by *Figure 25* also depicts, participants were generally satisfied with *VisRecly* from a usability point of view. Twelve of the participants agreed or strongly agreed that they would use the system frequently. Users gave voice to their view that the system is somewhat complex, but they also outlined that the introduced complexity is not at all unnecessary, it is inherent to the nature of the problems *VisRecly* aims to provide a solution for.

Attendants of the study were generally satisfied with the system's ease of use, articulating that they would not require a technical person's support when interacting with the app in order to be productive. They also appreciated the functionalities supported by the app and they indicated that they did not discover significant inconsistencies either from an aesthetical or an operational point of view.

Some of the participants asserted that the linked highlighting and automatic scrolling functionality of the dashboard made interactions somewhat awkward and requested the possibility to make this feature optional on the side pane to remedy this.

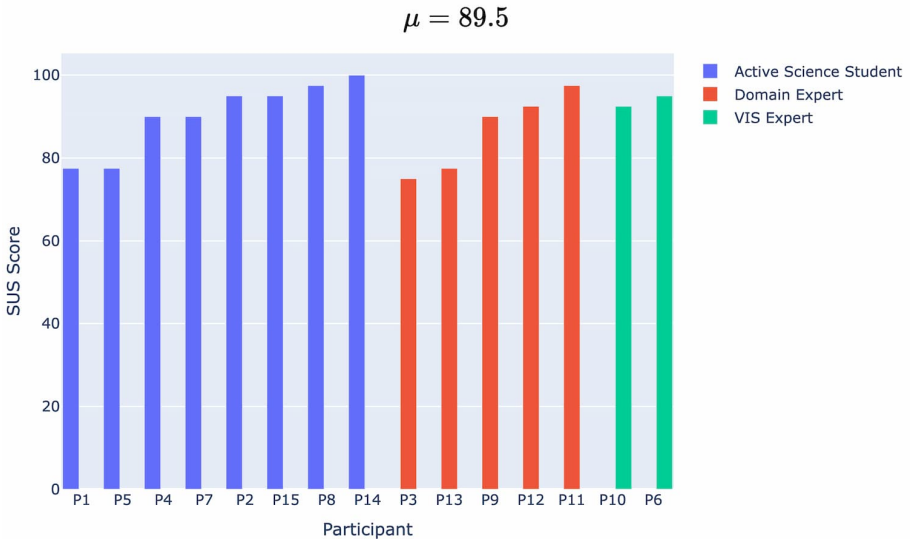

Figure 24: Summary of converted user responses to the System Usability Scale questionnaire presented to them after interacting with *VisRecly* freely. User types are visualized using stacked bar charts. The rating scale ranges from 0 to 100, corresponding to the overall value of SUS, as defined by Brooke [4].

Figure 25: Summary of converted user responses to the System Usability Scale questionnaire presented to them after interacting with *VisRecly* freely.

4.7.3 Comparison with Voyager 2

After performing **Scenario 2 - Targeted Generation of Recommendations** with Voyager 2, participants made it clear that the user interface of *VisRecly* is more user-friendly, making it easier to operate. Users voiced that the absence of introductory hints and helping information in Voyager 2 made their interaction with the tool more difficult. *Figure 26* displays the usability scores awarded to Voyager 2 by the participants.

While users recognized how powerful Voyager 2 is by making it possible to explicitly specify which encoding channel to use for specific variables, they also voiced that they are more satisfied with the approach of *VisRecly* in that it does not impose this task on a novice user.

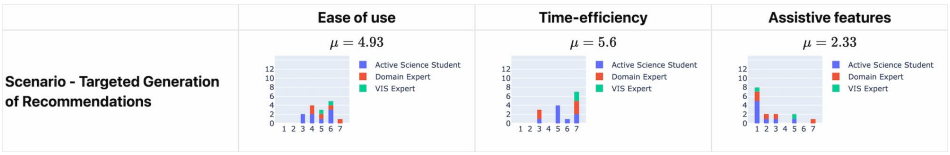

Figure 26: Summary of user responses to the After Scenario questionnaire item presented to them after the pre-designed scenario with Voyager 2. User types are visualized using stacked bar charts. The rating scale ranges from 1 to 7, corresponding to the *strongly disagree* and *strongly agree* statements. See *Figure 23* for the usability scores awarded to *VisRecly*.

The feedback of the study attendants, also illustrated by *Figure 26*, let one come to the conclusion that novice VIS users prefer simplicity and in-app guidance over a more complex - but also more powerful - alternative when it comes to exploring data visually.

5 Conclusion

I dedicated this thesis to the design space exploration of task-oriented VisRec systems as well as to the implementation of an actual high-fidelity prototype featuring a fully functional recommendation pipeline exposed through a novice-user-friendly, highly reactive web-based dashboard. In what follows, I discuss the initial objectives of the thesis along with the extent to which they have been achieved as well as the system's limitations and future work associated with it.

5.1 Thesis Objectives Revisited

In the low-level prototyping phase, I was seeking an answer to the question raised in RQ2, namely, I was looking into the main principles along which a visualization recommendation user interface should be designed to maximize the interpretability of the produced results. Through a series of interviews conducted with novice VIS users, I came to the conclusion that users express general sympathy towards single-page prototypes exposing all their features in a single, interactive view.

Building on top of the synthesis of my findings from the low-level prototyping phase, I immersed myself in the high-fidelity implementation of the synthesized candidate. Throughout the high-fidelity prototyping phase, I had the chance to explore RQ3, namely the extent to which the capabilities of an existing VisRec engine can be leveraged to produce visualization recommendations for novice users through a user-experience-centered application. While various powerful engines have been developed as the product of prior work of researchers, I found that no existing solution matched the needs of *VisRecly*. Throughout the progress of trying to find a remedy for this, I contributed to multiple projects[8] and ended up integrating a customized, extended version of Draco into my codebase.

Drawing from the formal user study sessions conducted for both prototyping phases, I managed to reach an answer to the guiding research question of my thesis, RQ1, querying how a usability-centered VisRec user interface can support novice users in choosing visualizations relevant to their specific goals. Novice VIS users are seeking a usable tool in the first place with a streamlined, guided user experience, featuring implicit educational aspects in the topic of visualization. The formal, academic rigor of the system's output proved to be secondary for them. Novice VIS users look for a tool helping them to quickly produce a variety of alternative visual representations of their data and to explicitly label (rank) the recommendations so that they can come to a decision with more confidence.

[8] Contributed to *uwdata/draco* and became an official contributor at *cmudig/draco2* after a series of contributions to it

5.2 Limitations of the Introduced Approach

Even though a significant effort was given to this piece of research, there are undeniable limitations associated with its outcomes that need to be mentioned. Most of these limitations stem from the fact that the *VisRecly* is the product of a Bachelor's Thesis, imposing a scope restriction on the research. The below-described aspects are the most prominent limitations of the presented work.

The high-fidelity prototype is limited in that visualizations are recommended purely based on the characteristics of the data, the provided visualization tasks only influence the ranking of the recommendations by inspecting the used marks and assigning a favorable or unfavorable cost to them based on the type of the ranked chart. Giving users the possibility to select single or multiple tasks for which they would like to generate recommendations would make *VisRecly* more powerful.

An apparent limitation of *VisRecly* is that currently, it does not allow for uploading custom datasets, all it supports is the use of the built-in datasets. This is a limitation that can be easily overcome by implementing an isolated module responsible for accepting user uploads and preprocessing them to make them consumable by the dashboard in a uniform format.

Another shortcoming of the tool is that it does not put constraints on what data column combinations are allowed for selection for specific tasks. As a concrete example, the *Cluster* task is not sensible when having two categorical variables selected such as *Name* and *Origin* from the *Cars* dataset.

Considering the output of the ranking pipeline, an associated limitation is that similar recommendations are not pruned away automatically, resulting in fairly similar recommendations being ranked next to each other. Given the fact that *VisRecly* should make data exploration easy and efficient, presenting users with a variety of reasonable recommendations, the absence of a pruning mechanism is suboptimal.

5.3 Future Work

In what follows, a discussion is presented on how *VisRecly* can emerge to be a more usable and overall more effective system.

User interface enhancements: While users were overall satisfied with the usability of *VisRecly*, some of them expressed that they had a difficult time with the interpretation of the ranking types (overall and task-specific). Enhancing the heatmap user interface component to make it simpler to comprehend the ranking conveyed by the tool would be a great improvement. Adjusting the color scheme of the application as well as providing a variety of color scheme choices to users would also be a sensible addition to the system to make it more accessible.

Support uploading of custom datasets: Providing users with the possibility to upload their own datasets would increase the usefulness of *VisRecly* to a significant extent. A module responsible for accepting user uploads, sanitizing the data and converting it into a common format would be necessary.

Introduce ASP constraints for VIS tasks: As mentioned in the limitations section, the recommendations generated by *VisRecly* are not influenced by task selection, they play a role only in the ranking phase of the application's workflow. Extending the ASP base of Draco with task-specific constraints would improve the recommendations' quality to a significant extent. Hard constraints could be introduced to declare which column types are supported by specific tasks, while soft constraints could dictate the encoding preferences associated with given tasks.

Prune away indistinguishable recommendations: Introducing an automatic pruning mechanism would provide users with a wider visualization recommendation space to explore based on their data selection.

Generate more sophisticated recommendations: It would be beneficial to support generating visualization recommendations presenting multiple views and view compositions. Migrating *VisRecly*'s recommendation module to use Draco 2 would make these possible and this could be a sensible addition to the system as soon as Draco 2 reaches a stable state with a default renderer for its specifications.

6 Appendix

A Personas

A.1 Veronika: BSc Student

Gender	Female
Age	24
Location	Vienna
Career Focus	About to finish her Statistics BSc studies
Relationship Status	Single

Table 4: Veronika Demographics

General Background Veronika has been an exemplary student from the very beginning of her studies. She is striving to excel academically in her BSc studies so that she can progress and become a post-doc researcher someday, as she is genuinely intrigued by the world of research and she would love to become an active part of it through future contributions.

VIS-specific Background Veronika has a solid understanding of quantitative information and she is also able to make sense of basic data visualizations too. However, she is just guided by her intuitions when she makes such interpretations, as well as she, tries to make use of her prior experiences from her statistics studies. She does not possess any formal VIS background knowledge.

Goals

- Find a full-time job relevant to her studies as soon as possible
- Complete her Bachelor's degree with a grade average below 1.5
- Make the most out of her time by using modern tools for her academic tasks

Motto

> *"Just because we can model every aspect of our lives quantitatively does not necessarily mean that we should."*

Mindset

- Appreciates the power of quantitative sciences but tries to behave as humanly as possible in everyday scenarios
- Does not make unnecessary abstractions & calculations just for fun

- Always tries to be on-point and she adjusts her efforts to be just enough to deliver a high-quality solution for any given task of hers
- Values her time and strives to use established tools for her tasks instead of trying to reinvent the wheel
- She enjoys learning new things on her own, primarily through a ”learning by doing” approach

Typical usage scenario of *VisRecly*

- Being active in scientific studies, she needs to make sense of raw data from time-to-time
- As a person who prefers to complete tasks in a time-efficient way, she will likely prefer looking at data visualizations instead of aggregated tabular data
- Being a BSc student, she lacks the experience to ”predict” what a raw dataset will be good for just by briefly inspecting it & being a VIS novice, she will not be able to consciously define what data tasks would be beneficial for her exact use case, hence she will likely interact with *VisRecly*in an experimental & explorative fashion

Familiar Technology

- Android
- Windows
- Google Chrome
- MATLAB, R

A.2 Albert: Ph.D. Student

Gender	Male
Age	32
Location	Vienna
Career Focus	Just started his PhD studies
Relationship Status	Engaged

Table 5: Albert Demographics

General Background Albert has just finished his Physics master's studies in München and he is a fresh member of the Energy Economics research group in Vienna on TU Wien as a Ph.D. student. He is an active member of various real-life projects in which he supports the faculty both with his theoretical and practical expertise. He has strong technical fundamentals, however, he still needs to adapt to the academic rigor that a Ph.D. study demands. Albert is tremendously grateful for the powerful open source projects he uses to accelerate his everyday work. While he primarily uses programming languages for scientific tasks, not for software engineering ones, he tries to grow and aims to reach a level at which he will be capable of contributing to OSS.

VIS-specific Background Albert is no stranger to creating visualizations himself, he has been using Python's Matplotlib, Plotly and StreamLit libraries in the past for data analysis. This implies that he is familiar with intermediate visual idioms and he is capable of making sense of quantitative details. He is also able to roughly describe the goals he would like to reach when creating visualizations. At the same time, Albert has never been exposed to a formal VIS education, hence he would not be able to consciously come up with rigorous data and task abstractions only by himself.

Goals

- Complete his Ph.D. studies
- Show his mentors how much potential he has
- Get more familiar with rigorous research methodologies
- Discover open source projects useful for his everyday work

Motto

> *"It is better to have a vision and not act on it than acting with no vision at all."*

Mindset

- Strives to maintain a healthy work-life balance
- Always has a vision or a plan first before jumping into a task
- Considers every single project as an opportunity to learn new things
- The success of a research project and his personal growth during that project are of equal importance for Albert

Typical usage scenario of *VisRecly*

- Being a Ph.D. student in a research group that needs to analyze raw data regularly, Albert needs to create visualizations often
- Albert always has at least a vague idea of what sense he wants to make out of a raw dataset, he is aware of potential aspects of interest
- → therefore he will likely interact with *VisRecly*in a directed, objective-oriented way
- Will try to reach his aims in the fewest number of steps

Familiar Technology

- Android
- Linux - Ubuntu
- Firefox
- Octave, Python, R, PostgreSQL, Bash, basics of HTML, JS & CSS

A.3 Reinhard: Professor

Gender	Male
Age	62
Location	Vienna
Career Focus	Professor & senior scientist since decades
Relationship Status	Married

Table 6: Reinhard Demographics

General Background Reinhard is a professor of astronomy, and a well-known & recognized scientist with significant publications and scientific contributions. He spends a notable amount of time behind telescopes, discovering the realm of stars & collecting a tremendous amount of data while doing so. As Reinhard already suffers from mental overhead due to the vast amount of different intellectual activities he is participating in, he considers his time to be his most precious treasure.

VIS-specific Background The data produced by Reinhard via telescope sessions is the very definition of "Big Data". He has an excellent command of interpreting quantitative data, however, when it comes to visualizing the raw data he has at hand, his knowledge is not well-rounded with regard to the visual idioms ideal for his tasks & goals.

Goals

- Make ground-breaking discoveries
- Transfer his knowledge and know-how to younger generations
- Enhance skills in Big Data

Motto

"Publish or perish. I mean it."

Mindset

- Stays humble regardless of his experience and achievements
- He knows that a good tool can save months in a project & yield better results
- Always has a vision or a plan first before jumping into a task

Typical usage scenario of *VisRecly*

- Since Reinhard collects Big Data daily, he needs a way to make sense of them potentially by producing meaningful visualizations
- His experience will allow him to make assumptions about the results of the data analysis or the nature of correlations he is looking for
- → when reaching for *VisRecly*, he will have some sense about tasks & goals
- → his interaction with the dashboard will be semi-explorative

Familiar Technology

- iOS
- Linux - Ubuntu, macOS
- Chromium
- Bash, MATLAB, Python, R, MongoDB, Cassandra

A.4 Judith: Marketing Employee

Gender	Female
Age	41
Location	London
Career Focus	Remote employee at a marketing agency
Relationship Status	It's complicated

Table 7: Judith Demographics

General Background After finishing her high-school studies, Judith was forced to start working out for personal reasons and did not have the chance to pursue a university degree program. She gained experience in various sales positions, then after the social media boom in the 2010s, Judith transitioned her focus to social media and marketing. Her daily tasks include managing Facebook & Instargam sites, creating SEO-optimised content as well as managing web-commerce content on platforms such as WooCommerce and Shopify. Judith is highly result-oriented, therefore she wishes to always know what effects her work & actions have with regards to conversion, visitor-base growth as well as generated revenue.

VIS-specific Background Thanks to the years of experience she amassed in sales, Judith has a reasonable sense of how to interpret quantitative data. She is keen on observing the integrated report dashboards which come with the tools she uses (Google Analytics, Shopify, WooCommerce), therefore she is already familiar with basic chart types. She is in general fairly confident when it comes to analyzing smaller, simpler datasets in her domain. As a result of not being exposed to formal education at all, Judith has no prior VIS knowledge that would put her in the position of being able to formalize data and task abstractions.

Goals

- Succeed at her workplace
- Continue learning new things in an autodidact fashion
- Explore digital experiences to gain inspiration for marketed products

Motto

"Let me Google that for you!"

Mindset

- Makes the most use of online learning materials, to prove that obtaining a university degree is no pre-condition for being competent

- Convinced about being able to master any topic given a good amount of will-power, dedication and time
- Measures her competence and success by the results of the products she manages
- Tries to work smart, not hard: she is a black-belted Google user and is capable of finding an online tool for any problem she has at her hands

Typical usage scenario of *VisRecly*

- As an employee whose most important KPI is growth, conversions, revenue, etc., the data she can export from admin dashboards is valuable and interesting for her
- Exploring & understanding the data means exploring & understanding her performance in private
- With no prior VIS experience, Judith will use *VisRecly*in a "learning by doing" manner, however, presumably with exceptional engagement
- She will likely look for outliers and correlations to understand her performance better

Familiar Technology

- iOS
- macOS
- Safari
- Google Analytics, Adobe Creative Cloud, Shopify, WooCommerce

B Low-fidelity Prototyping

B.1 Pen & Paper Prototypes

The actual environment in which I sketched the figures of this section can be visited here.

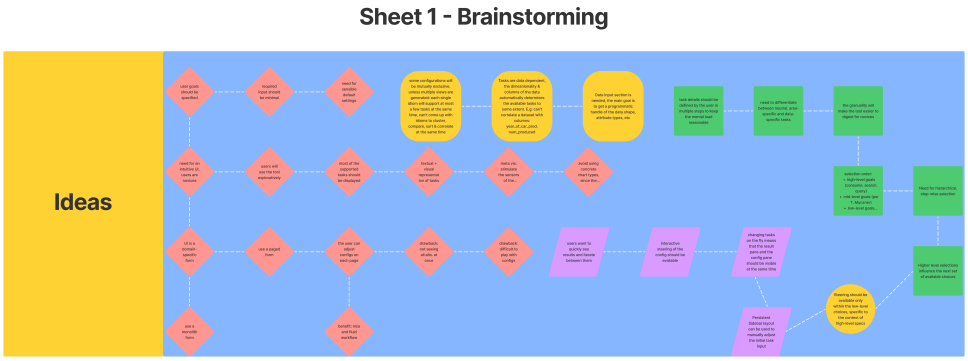

Figure 27: Sheet 1

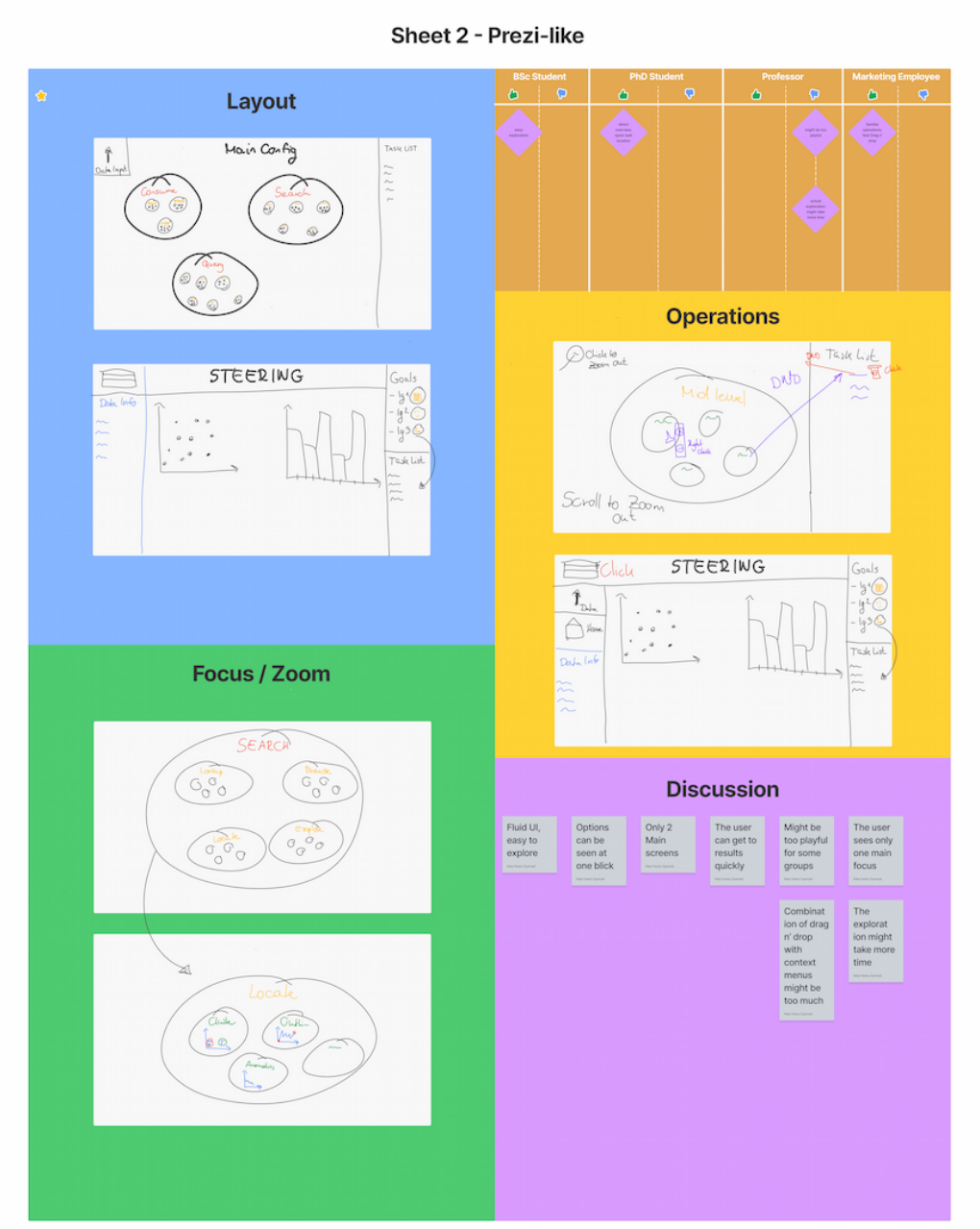

Figure 28: Sheet 2

Figure 29: Sheet 3

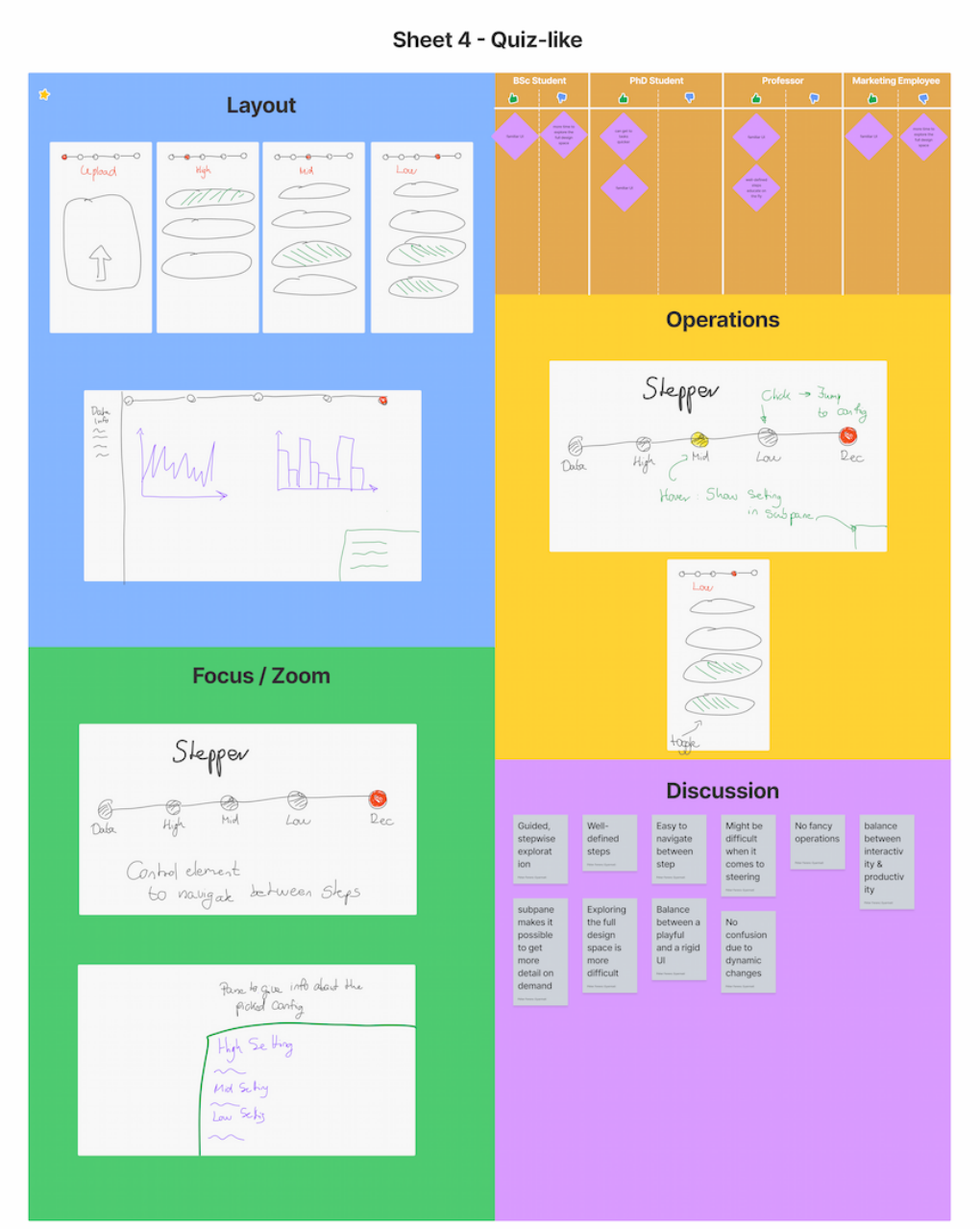

Figure 30: Sheet 4

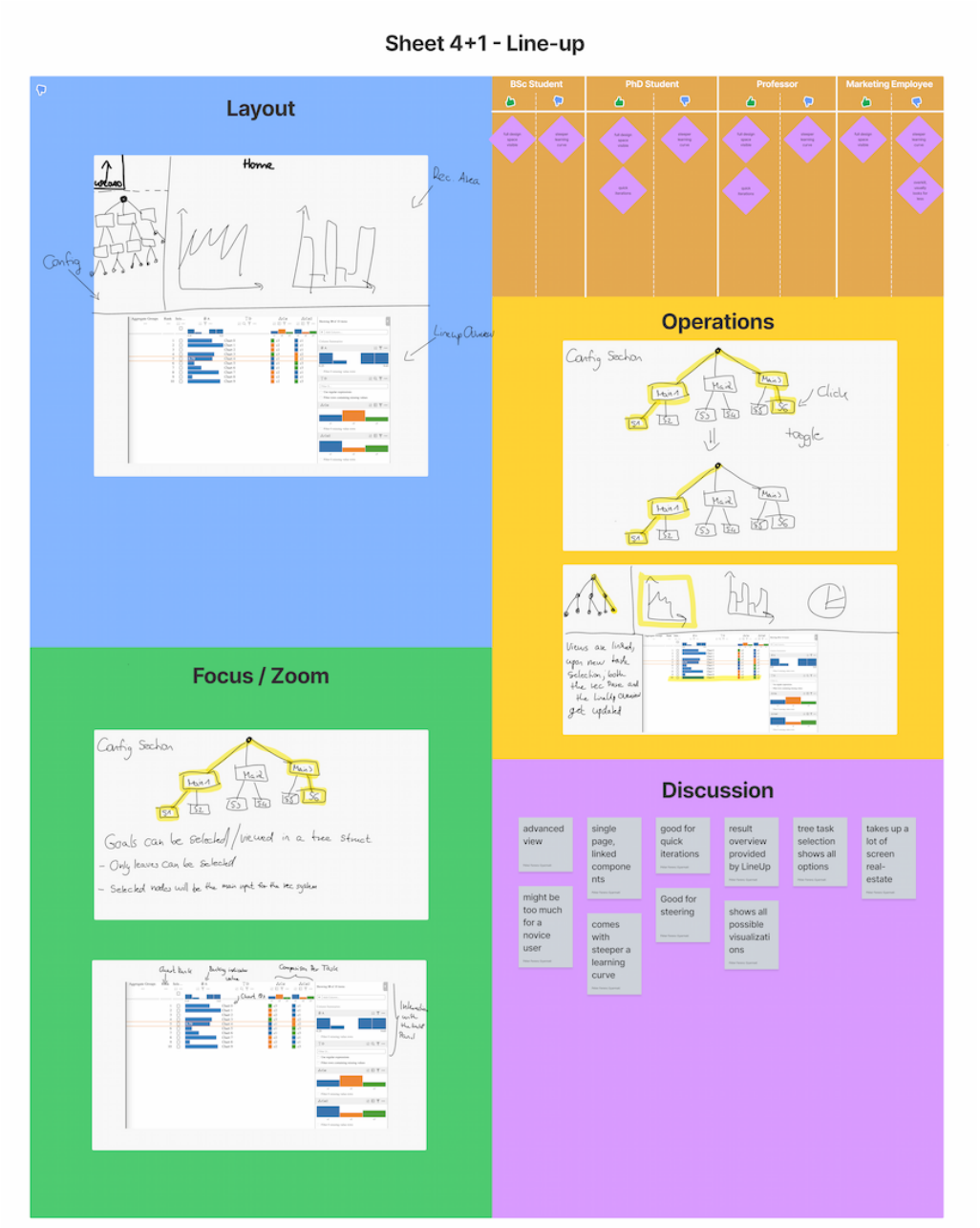

Figure 31: Sheet 4+1

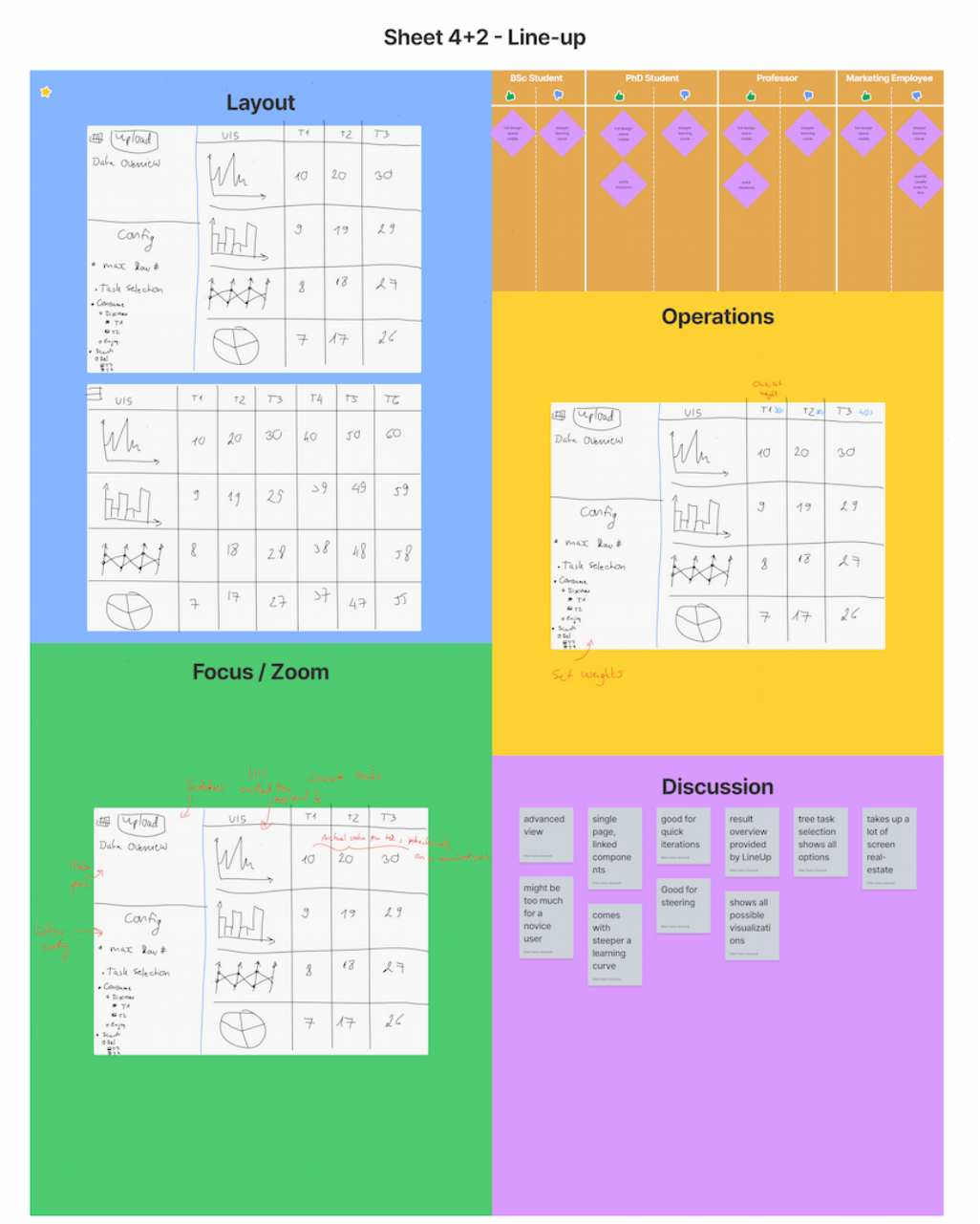

Figure 32: Sheet 4+2

References

[1] Anne Adams, Peter Lunt, and Paul Cairns. *A qualitative approach to HCI research*, pages 138–157. Cambridge University Press, Cambridge, UK, 2008.

[2] Matthew Brehmer and Tamara Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), Dec 2013.

[3] John Brooke. SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, Nov 1995.

[4] John Brooke. SUS: A retrospective. *Journal of Usability Studies*, 8:29–40, Jan 2013.

[5] Sébastien Chopin. The intuitive vue framework. JavaScript Frontend Framework considered for the implementtion of VisRecly.

[6] Victor Dibia and Çağatay Demiralp. Data2Vis: Automatic generation of data visualizations using sequence-to-sequence recurrent neural networks. *IEEE Computer Graphics and Applications*, 39(5):33–46, Sep 2019.

[7] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Multi-shot ASP solving with clingo. (arXiv:1705.09811), Mar 2018. arXiv:1705.09811 [cs].

[8] Sakunthala Gnanamgari. Information presentation through default displays. *Dissertations available from ProQuest*, pages 1–118, Jan 1981.

[9] David Gotz and Zhen Wen. Behavior-driven visualization recommendation. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, IUI '09, pages 315–324, New York, NY, USA, Feb 2009. Association for Computing Machinery.

[10] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Hanspeter Pfister, and Marc Streit. Lineup: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), Dec 2013.

[11] Pat Hanrahan. VizQL: a language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, page 721, New York, NY, USA, Jun 2006. Association for Computing Machinery.

[12] Rich Harris. SvelteKit • the fastest way to build svelte apps. JavaScript Frontend Framework considered for the implementtion of VisRecly.

[13] Progressive web apps (pwas) — mdn. `https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps`. (Accessed on 10/25/2022).

[14] Introduction — Draco. https://dig.cmu.edu/draco2/intro.html. (Accessed on 10/25/2022).

[15] Github - airbnb/visx: visualization components. https://github.com/airbnb/visx, Aug 2022. (Accessed on 10/25/2022).

[16] Github - cmudig/draco2: Modular version of draco visualization recommendation engine. https://github.com/cmudig/draco2, Aug 2022. (Accessed on 10/25/2022).

[17] Github - uwdata/draco-vis: Draco on the web. https://github.com/uwdata/draco-vis, Mar 2021. (Accessed on 10/25/2022).

[18] Github - vega/compassql: Compassql query language for visualization recommendation. https://github.com/vega/compassql, Jul 2022. (Accessed on 10/25/2022).

[19] Github - vega/react-vega: Convert vega spec into react class conveniently. https://github.com/vega/react-vega, Aug 2022. (Accessed on 10/25/2022).

[20] Github - vega/vega-datasets: Common repository for example datasets used by vega-related projects. https://github.com/vega/vega-datasets. (Accessed on 10/25/2022).

[21] Next.js by vercel - the react framework. https://nextjs.org/. (Accessed on 10/25/2022).

[22] Nx: Smart, fast and extensible build system. https://nx.dev/. (Accessed on 10/25/2022).

[23] Prezi: Presentations and videos with engaging visuals for hybrid teams. https://prezi.com/. (Accessed on 10/24/2022).

[24] Discover — tableau public. https://public.tableau.com/app/discover. (Accessed on 10/25/2022).

[25] Develop. preview. ship. for the best frontend teams – vercel. https://vercel.com/. (Accessed on 10/25/2022).

[26] Figma: the collaborative interface design tool. https://www.figma.com/. (Accessed on 10/24/2022).

[27] Grammarly: Free online writing assistant. https://www.grammarly.com/. (Accessed on 10/24/2022).

[28] Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo. VizML: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, Glasgow Scotland UK, May 2019. ACM.

[29] Tobias Isenberg, Petra Isenberg, Jian Chen, Michael Sedlmair, and Torsten Möller. A systematic review on the practice of evaluating visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2818–2827, Dec 2013.

[30] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Profiler: integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 547–554, New York, NY, USA, May 2012. Association for Computing Machinery.

[31] Pawandeep Kaur and Michael Owonibi. A review on visualization recommendation strategies. In *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 266–273, Porto, Portugal, 2017. SCITEPRESS - Science and Technology Publications.

[32] Stephan Kerpedjiev, Giuseppe Carenini, Steven F. Roth, and Johanna D. Moore. Integrating planning and task-based design for multimedia presentation. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, page 145–152, New York, NY, USA, Jan 1997. Association for Computing Machinery.

[33] Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 681–684, New York, NY, USA, May 2012. Association for Computing Machinery.

[34] Doris Jung-Lin Lee. Designing automated assistants for visual data exploration. Jul 2021.

[35] Doris Jung-Lin Lee, Vidya Setlur, Melanie Tory, Karrie Karahalios, and Aditya Parameswaran. Deconstructing categorization in visualization recommendation: A taxonomy and comparative study. *arXiv:2102.07070 [cs]*, Feb 2021. arXiv: 2102.07070.

[36] James R. Lewis. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78, Jan 1995.

[37] Halden Lin, Dominik Moritz, and Jeffrey Heer. *Dziban: Balancing Agency & Automation in Visualization Design via Anchored Recommendations*, pages 1–12. Association for Computing Machinery, 2020.

[38] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. DeepEye: Towards automatic data visualization. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 101–112, Apr 2018.

[39] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, Apr 1986.

[40] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. Show Me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, Nov 2007.

[41] Dominik Moritz. Github - domoritz/wasm-clingo: Clingo compiled for the web. `https://github.com/domoritz/wasm-clingo`, Apr 2021. (Accessed on 10/25/2022).

[42] Dominik Moritz. Github - domoritz/clingo-wasm: Clingo on the web. `https://github.com/domoritz/clingo-wasm`, Jul 2022. (Accessed on 10/25/2022).

[43] Dominik Moritz. Github - uwdata/draco: Visualization constraints and weight learning. `https://github.com/uwdata/draco`, Aug 2022. (Accessed on 10/25/2022).

[44] Dominik Moritz, Chenglong Wang, Greg L. Nelson, Halden Lin, Adam M. Smith, Bill Howe, and Jeffrey Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in Draco. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):438–448, Jan 2019.

[45] Jakob Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, page 152–158, New York, NY, USA, Apr 1994. Association for Computing Machinery.

[46] Alexander Rind, Wolfgang Aigner, Markus Wagner, Silvia Miksch, and Tim Lammarsch. Task cube: A three-dimensional conceptual space of user tasks in visualization design and evaluation. *Information Visualization*, 15(4), Oct 2016.

[47] Jonathan C. Roberts, Christopher J. Headleand, and Panagiotis D. Ritsos. *Overview of the Five Design-Sheets (FdS)*, page 27–41. Springer International Publishing, Cham, 2017.

[48] Steven F. Roth, John Kolojejchick, Joe Mattis, and Jade Goldstein. Interactive graphic design using automatic presentation knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, pages 112–117, New York, NY, USA, Apr 1994. Association for Computing Machinery.

[49] Steven F. Roth and Joe Mattis. Data characterization for intelligent graphics presentation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 193–200, New York, NY, USA, Mar 1990. Association for Computing Machinery.

[50] Bahador Saket, Alex Endert, and Cagatay Demiralp. Task-based effectiveness of basic visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(7), Jul 2019.

[51] Bahador Saket, Dominik Moritz, Halden Lin, Victor Dibia, Cagatay Demiralp, and Jeffrey Heer. Beyond heuristics: Learning visualization design. (arXiv:1807.06641), Aug 2018. arXiv:1807.06641 [cs].

[52] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, Jan 2017.

[53] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), Dec 2012.

[54] Reinhard Sefelin, Manfred Tscheligi, and Verena Giller. Paper prototyping - what is it good for? a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, New York, NY, USA, Apr 2003. Association for Computing Machinery.

[55] Leixian Shen, Enya Shen, Zhiwei Tai, Yiran Song, and Jianmin Wang. TaskVis: Task-oriented visualization recommendation. *EuroVis 2021 - Short Papers*, 2021.

[56] Leixian Shen, Enya Shen, Zhiwei Tai, Yihao Xu, and Jianmin Wang. Visual data analysis with task-based recommendations. (arXiv:2205.03183), May 2022. number: arXiv:2205.03183 arXiv:2205.03183 [cs].

[57] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Pearson, 6th edition, 2016.

[58] Tarique Siddiqui, John Lee, Albert Kim, Edward Xue, Xiaofo Yu, Sean Zou, Lijin Guo, Changfeng Liu, Chaoran Wang, K. Karahalios, and Aditya G. Parameswaran. Fast-forwarding to desired visualizations with Zenvisage. In *Conference on Innovative Data Systems Research*, 2017.

[59] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, Jan 2002.

[60] Melanie Tory and Vidya Setlur. Do what i mean, not what i say! design considerations for supporting intent and context in analytical conversation. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 93–103, Oct 2019.

[61] Thomas Tullis and Jacqueline Stetson. A comparison of questionnaires for assessing website usability. Jun 2006.

[62] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. Towards visualization recommendation systems. *ACM SIGMOD Record*, 45(4):34–39, May 2017.

[63] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. SeeDB: efficient data-driven visualization recommendations to support visual analytics. *Proceedings of the VLDB Endowment*, 8(13):2182–2193, Sep 2015.

[64] Joannès Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proceedings of the 16th European conference on Machine Learning*, ECML'05, pages 437–448, Berlin, Heidelberg, Oct 2005. Springer-Verlag.

[65] Fernanda B. Viegas, Martin Wattenberg, Frank van Ham, Jesse Kriss, and Matt McKeon. ManyEyes: a site for visualization at internet scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1121–1128, Nov 2007.

[66] Larry Wasserman and Larry Alan Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, Sep 2004. Google-Books-ID: th3fbFI1DaMC.

[67] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Towards a general-purpose query language for visualization recommendation. In *ACM SIGMOD Human-in-the-Loop Data Analysis (HILDA)*, 2016.

[68] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, Jan 2016.

[69] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2648–2659, Denver Colorado USA, May 2017. ACM.

[70] Mehmet Adil Yalçın, Niklas Elmqvist, and Benjamin B. Bederson. Keshif: Rapid and expressive tabular data exploration for novices. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2339–2352, Aug 2018.