Milot Gashi, BSc.

Goal oriented Visual Recommendations via User Behavior Analysis

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Sabol, Vedran, Dipl.-Ing. Dr.techn.

Institute of Interactive Systems and Data Science

Co-Advisor

Mutlu, Belgin, Dipl.-Ing. Dr.techn.

Graz, September 2018

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Acknowledgments

The outcome of this thesis could not have been accomplished without the support of many individuals. Besides the honor of collaborating with them, I would like to express my appreciation for their great support and help.

First and foremost, I am very lucky to have Vedran Sabol as my supervisor, and I want to thank him for his time and great ideas while guiding me through this research process. My sincere thanks to my Co-advisor Belgin, whom I am grateful for her support and counsel during this process. Furthermore, I would like to thank Ilija Simic for creating the Visualizer. Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my beloved parents Ajshe and Murtez, my siblings Blerta and Ardian, whose love and support sustained me throughout this journey. I would like to thank all of my colleagues and friends, especially to my cousin Gresa Gashi, who made the great effort to proofread this thesis. Finally, I like to express my very profound gratitude to my lovely and supportive wife Grese, for providing me with unfailing support and continuous encouragement throughout my years of study and for always being there for me.

Abstract

The complexity of using information systems and the amount of data managed by those systems are increasing rapidly, leading to an information overload problem. Information Visualization and Visual Data Analysis are common approaches for dealing with this problem. However, visualization systems require a certain level of knowledge on visual data exploration to be effective. Unfortunately, typical users do not bring adequate experience in visualization, which in turn negatively influences analytical results and the user's satisfaction. This Master's thesis proposes a system which supports the users in visually exploring their data. The system shall first analyze user's current behavior and detect patterns. The recognized patterns are next analyzed to infer user's intended task. Finally, the system recommends the next analysis step, or a sequence of steps, in the form of visual recommendations that guide the user towards successful completion of the intended task. For this purpose, a recommender model based on Markov chains is envisioned, which generates recommendations depending on the behavior of users collected during previously performed data analysis tasks. In contrast to the existing research, we do not only recommend the end result (e.g., appropriate visualization), but the next interaction, or a whole sequence of interactions that lead to a potential insight. Furthermore, this thesis focuses on four well-known analytical tasks, for which the models are trained separately: gain an overview, analyze data anomalies, trends within the data, and data distribution. A user study was performed where the participants was asked to perform selected subtasks for each of the aforementioned task. The collected data was analyzed to recognize meaningful patterns and train the model, and used to perform the evaluation of the algorithms for generating recommendations.

Keywords: Predictive model; Visualization system; Task classification; User interactions; Markov chains; Visualization recommendation; Recommender Systems; Visual Exploration

Contents

List of Figures

List of Figures

1. Introduction

1.1. Motivation

The amount of information produced nowadays is overwhelming, regardless in which discipline the user is working in. In order to gain meaningful insights from raw data, several methods have to be applied. Luckily, humans are strongly visual-driven creatures. On the one hand, not many of us can identify patterns among textual data, on the other hand, we can easily interpret a visualization representation [5, 6, 4] (e.g., Bar chart), and extract meaningful information from visual representations. Therefore, data visualization is a powerful method and is the quickest way to communicate the information to human. However, every data set has its own unique properties. Most importantly, there are different techniques of data exploration which might not be applied to every single dataset or task. Given the variety of techniques (e.g., comparisons, detecting patterns, correlations, etc.) to explore the data and the unique dataset description, different visualization (e.g., Bar chart, Pie chart), and visualization interactions can be applied to deliver various insights. To handle this problem, many visualization systems are implemented with functions that focus on supporting users in exploring their data visually.

In order to use these systems, a significant level of visualization and data analytics skills are required. Unfortunately, average users don't possess these skills which in turn increases the difficulty while using these tools. Even domain experts within specific area mostly have little or no visualization knowledge. Yet, these shortcomings have negative impacts in both business world and research life. Therefore, companies or institutes are forced to employ data analysts with visualization and analysis skills, but little domain knowledge which might increase the complexity and cost of exploring and analyzing data and gaining meaningful insights.

In order to address this issue in this Master's Thesis, we propose a recommendation model based on Markov chain which should guide user in exploring the data. Yet, when it comes to explore data visually, the research distinguishes between 4 analytic tasks [1, 2]: gaining overview, detect anomalies, detect data correlations, and analyze data distribution. Thus, our system intended to infer what analytic task the current user is aiming to perform and guide his/her during this process by recommending the next analysis step (an interaction such as filtering, aggregating, creating views, zooming, etc.) or a sequence of steps.Yet, our technique is integrated within a rule-based visualization recommendation system of the Visualizer [1], that is used to automatically recommend the appropriate visualization. for a given dataset. Our technique uses the generated visualization recommendations and the integrated interactions provided by Visualizer.

The proposed technique relies on two steps (an offline step and an online step) to track the user's behavior and recommend the next interaction, or the whole sequence of interactions within a specific task. The offline step is used to model the user behavior within the visualization system. To achieve this we performed an online evaluation and tracked user's behavior while performing a visual task. Next, the collected information is used to model the user patterns within each task. In the online step, the generated model is used to recommend next step (interaction), or a whole path. The model generates recommendations based on the information acquired through monitoring user's behavior in real time. Moreover, we assume that this tool will affect the time to accomplish the goal and learning rate (e.g., the user learns a new way of exploring data).

Modeling and understanding the sequential user behavior [7] into Visualizer - the visualization systems used to integrate the proposed technique is the core task of the predictive models. Such models help answer questions such as "What kind of interaction does a specific user performs after a particular interaction?". One of the encouraging methods to address the aforementioned needs/challenges is the Markov chain (MC) [8, 9] method, that is used to model the dynamic sequential behavior [7, 10, 11, 12, 13]. Markov chains are especially powerful when it comes to catching short-term dynamics. Recommender systems based on a Markov chain model utilize such sequential data

[1] https://vizrectest.know-center.tugraz.at/

by predicting the users next action based on the last actions.

In nutshell, this thesis makes the following contributions:

- A predictive model based on Markov chain is provided which models the user behavior based on monitoring interactions and recommends the next step (or a path) in order to help users in accomplishing data exploring tasks within a visualization system.
- A search method is used to obtain and recommend a sequence of interactions (path), instead of a single interaction.
- A visual UI in order to guide the users while performing a visual analytic task within a visualization system.
- A framework to classify the tasks which uses the random forest algorithm [14]. The goal thereby is to recognize the task dynamically based on the current user behavior.
- Comparison of various Markov chain orders to model the sequential user behavior.
- Usability Evaluation of the Visualizer for different user groups which is used to collect interaction data and train the models.

After having performed state of the art analysis, to the best of our knowledge, no other system addresses these issue in the way we designed and performed within the course of this thesis.

1.2. Structure of this Thesis

This master's thesis consists of seven chapters. The introduction is followed up by Chapter 2 which provides an overview about the related work in the fields of visual recommender systems, and recommendation approaches based on Markov chains. Chapter 3 introduces Visualizer, the visualization platform with a rule-based recommender system, that is is used to integrate the proposed techniques. Furthermore, this chapter provides an general overview about the visualization systems with a focus on components implemented and required for this work. Chapter 4 describes the methods used to log, process and model the users' behavior. It furher details how the algorithms are trained to learn from user interactions and applied to generate recommendations. A

presentation of the visual UI for presenting the recommended interactions is provided in Chapter 5. A case study in Chapter 5 illustrates the system in action: a step-by-step use case demonstrating how the system is applied for performing analytical tasks. An evaluation of a predictive algorithm, the used methodology for evaluation of the predictive model, a brief description of results of the user experiment, and a statistical analysis is described in Chapter 6. Chapter 7 presents the benefits, limitations, and ideas for further improvements. Finally, Chapter 8 concludes the thesis.

2. Related Work and Background

This section introduces the existing work about the visualization systems, visual recommender systems, and the statistical learning models used to model user behavior within such recommender systems. The main goal of the visual recommender systems is to improve the user satisfaction, reduce the time needed for the user to accomplish a task, reduce the costs while using these systems, etc.

This chapter is divided into five sections. In the first section, the visualization systems are introduced. In the second section, the user modeling regarding the related work is explained in detail. In the third section, various visualization recommendation approaches with emphasis on user behavior are introduced. Whereas, in the fourth section, various visualization recommendation, or adaptive recommendation systems with focus on rule based approaches are introduced. In the last section, the existing work on statistical learning tools, or more exactly on learning predictive models based on Markov chains, is introduced.

2.1. Visualization Systems

This section outlines how interactive visualization systems are constructed, and what their primary purpose is. First, these systems help to detect patterns, trends, correlations and outliers in user's data which might remain hidden by other means of data analysis. To do so, they enable user to understand their data, especially when the data are too large to get a quick overview by viewing them in e.g., tabular form.

The visualization systems differ broadly regarding their capabilities, system purpose, or the end-user support. Regarding to the required skills, we distinguish between scientific- and business-oriented visual analytic tools.

At the same time, these systems vary regarding to the targeted end-user group. It is well known that the end user is the most critical component in a visualization process. Thus, a visualization system should be able to address the needs and preferences of the end user individually considering e.g., their experience. There are mainly three groups of end-users. The expert users who posses an in-depth knowledge regarding the data analytics and visualization systems, the non-expert users who have no experience about data analytics and computers in general, and the third group of users who have a broad knowledge about visualization and computers, but who are not experts on these fields.

Visualization systems mainly consist of three crucial components: data input, presentation in visual context, and interactive data exploration. Additionally, various recommendation tools can be integrated in order to assist users in exploring data visually. In the following, we detail the components mentioned above, including already deployed recommendation tools.

Data Input The main Idea behind visualization systems is to analyze and explore data. One important question in this context thus is: Which format of data is supported by the visualization system. There exist a large number on data sources each structured in a different format and it is very hard for a visualization system to support each of them. Thus, each of these systems is designed to support only a specific data format or a group of data formats. The most popular data formats supported on these systems are Comma-separated data (CSV), JavaScript Object Notation (JSON), or a object-relational database system (e.g., SQL[1], PostgreSQL[2] etc.). In this work, we use a CSV format. Data formats is out of the scope of this work.

Data Visualization Data Visualization is the process of presenting the data in graphical form. Visualization tools usually offer various graphical presenta-

[1]https://docs.microsoft.com/en-us/sql
[2]https://www.postgresql.org

tions like Bar chart, Pie chart, Geographical Maps, Heat Maps, Scatterplots, etc.

Interactive exploration using graphical form The goal of the visualizations systems are not only the graphical representations of the data but also providing interactive tools that aid analysis and exploration of the data. These provide the opportunity to explore the data in a graphical format which, in contrast to textual exploration forms, facilitates the detection of interesting patterns, trends or correlation space.Since the idea is to visually explore the data, it is not enough to just plot the data on the chart. There is a need for various interactions that offer the possibility to explore and analyze the data. Examples of powerful interaction tools are: filtering, zooming, panning, or coordinated linking and brushing.

Recommendation tools Visualization systems are used by users with different background whereby all of them have the same goal: Understanding their data so that whatever visual task they have, they can complete it in the fastest possible way. In order to assist the users, recommendation tools are integrated within visualization systems. The target of the recommendation tools is to help the user to explore the data visually, without a need to be an expert on visual analytics. Additionally, the purpose of these tools is to improve the performance, cost efficiency, user satisfaction and many other aspects regarding analytical processes which are integrated within various software applications. However, users have different visual preferences when exploring their data. This is what a personalized visual recommender has to consider to ensure that a visualization is recommended a visualization that address user's interest and needs in the best possible way.

2.2. Behavioral Approaches

2.2.1. HARVEST

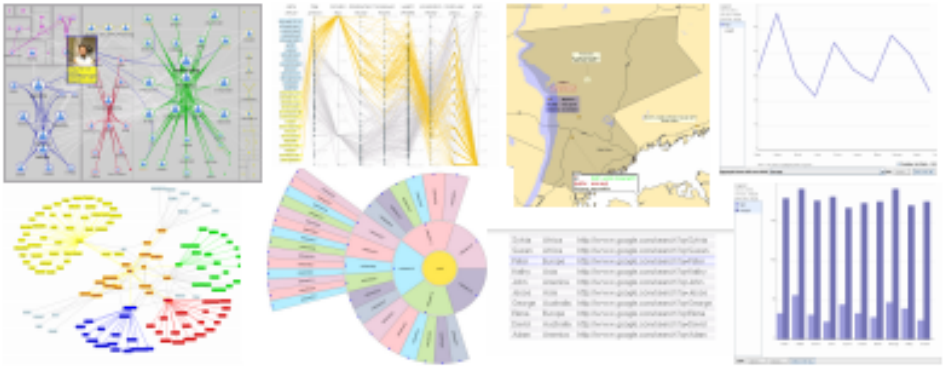


Figure 2.1.: Visualization types supported in HARVEST. Feature an intellegent background. [4]

HARVEST [4, 6] is built based on three critical factors which facilitate complex visual analysis even for users without knowledge in data analysis and data visualization. First, it supports various intelligent visualizations smart visualization widgets that can be easily adapted based on the ongoing state of the continuous visual analytic process. Second, it has the ability to dynamically recommend relevant visualization based on the users' behavior context. Finally, a semantic action tracker is provided which captures the meaningful user behaviors (e.g., patterns). This feature is used only to track previously executed actions (e.g., share the current ongoing task), but it is not considered when recommending visualizations.

The system is created with users in mind who are not experts in the visualization systems, but possess broad knowledge about this field. Gotze et al. [4, 6] tracks the user behavior (interactions within the dashboard) in order to acquire the user intend, respectively to identify the ongoing task by comparing the available behaviors with the pre-defined patterns representing specific tasks. HARVEST supports several visualizations with an intelligent background (can be adapted based on the ongoing context) as shown in Figure 2.1. These

visualizations can be adapted dynamically based on the interaction of users. E.g., during a session, the user performs a follow-up query. In this case, new data will be available, which will be adopted gracefully in the existing visualization: new data are visualized, or extending the current visualization. Also, the visualization widget possesses the capacity to recognize semantic user interaction based on a predefined standard vocabulary, called actions.

The most challenging task for an average user performing a data analytic task in a visualization system is to decide in advance which visualization is most appropriate. To address this challenge, HARVEST has integrated a recommendation algorithm which recommends the most appropriate visualization insights based on current context and ongoing task. The system automatically recommends the top-N best fitting visual metaphors. The recommendation with the best rating will be adapted in the main canvas, and the alternatives are shown in the recommendation sidebar, on the right side of the dashboard (see Figure 2.2).

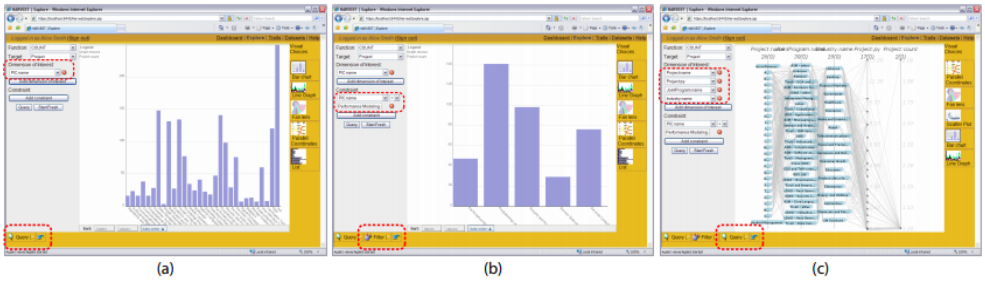


Figure 2.2.: Session example in HARVEST. (a) data are visualized using a Bar chart. (b) User performs some interactions e.g., filtering,. (c) The system generates a recommendation based on the current context. [4]

Another essential future introduced in the HARVEST is the capturing of user interaction with respect to semantic context. The system logs the user interactions and analyses them on the fly building a semantic model of user behavior. If the system detects any meaningful pattern, it reacts with the appropriate recommendations and dashboard adaptation. In this case, the user behaviors are compared against pre-defined patterns (four patterns). Further, the user can review, modify, or analyze the tracked interactions to better understand how it came to a specific context, or even to change the

parameters of early action to re-use the logic in a new context. Furthermore, the user can snapshot a state of the ongoing task using bookmarks which will also include the tracked session. If the user decides to reuse the bookmark, she can see the final view as well as the actions which lead to that view. A use case example is illustrated in the following. The user starts a session when she adopts the parameters on the query section and executes the query. An insight is shown in Figure 2.2(a). In the main window, data is visualized using a bar chart. Next, the user performs some interactions like filtering as shown in Figure 2.2(b). HARVEST utilizes the power of the tracking tool and recommendation engine to gather additional information about the current data context (e.g., filtered data), and recommends parallel coordinates visual metaphor as shown in figure 2.2(c). This approach, which is integrated within the HARVEST, is called behavior driven recommendation system (BDVR).

The approach presented within this thesis differs from the approach introduced within this paper (BDVR) in many aspects. Both approaches analyze user behaviors within the dashboard, but the goal of BDVR approach is to recommend the most appropriate visualization. On the other hand, our goal is to recommend the individual interaction in order to help the user to explore the data morre precisely. Furthermore, the aim of this thesis is to analyze user behaviors within four specific analytical tasks (gain an overview, detect anomalies, detect data correlations, and analyze data distribution.), which is not the case regarding the BDVR approach. Additionally, D. Gotz et al. define the patterns in advance by considering the experts knowledge. In our case, the patterns are modeled based on real data collected during usage, and considering only the user behavior within a specific task. Additionally, we consider not recommending only one interaction at the time, or multiple individual interactions, but also a whole path (sequence of interactions) leading to an insight. Furthermore, BDVR differs between basic interactions, our approach goes beyond a basic interaction by considering more complex interactions (e.g., zoomin, filtering, changing mappingm etc.)

2.2.2. Adaptive VIBE

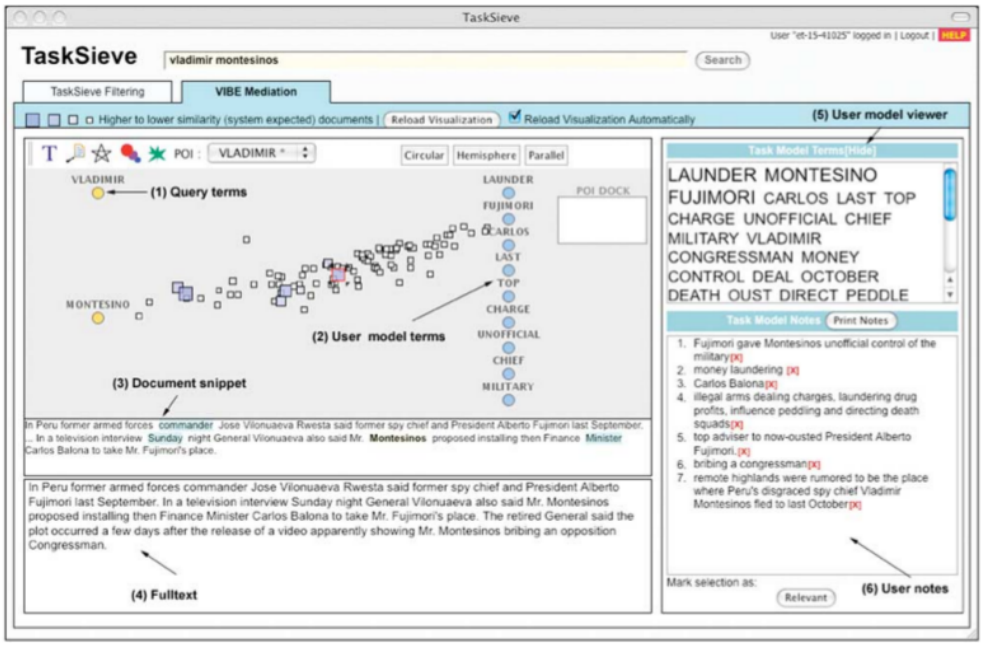


Figure 2.3.: Figure 2.2: TaskSieve adaptive news filtering system with integrated adaptive VIBE. (1) query terms (keywords), (2) User model keywords (POIs), (3) Document snippet, (4) full text, (5) user notes, and (6) User model viewer. [.

Adaptive VIBE [15] is a personalized search system that is built based on two critical components: the queries and user models. The user can pose a query using the search box and obtains a bunch of documents as a result. Next, the search results are shown visually (see Figure 2.3) and adapted based on the user preferences extracted from the available built user model. The user models are extracted analyzing user behaviors, such as obtaining most essential keywords from the notes saved in the ”Task model notes” area, issued search queries, selected documents from the search results, etc. This information is essential on interpreting user’s interests. Task model notes are collected based on the user’s interest which is defined explicitly by the user. For instance, the user can analyze the description from selected document, and insert it to the relevant list shown in the ”Task model notes” section.

In contrast, we introduced a default user model which is built implicitly. In this case, the user behaviors are extracted using an tracker feature which collects are the interactions between the user and the dashboard interface. This information are then used to build the user Model.

Adaptive VIBE supports document analysis based on the point of Interest (POI). POI is represented in the visual context as movable objects inside the presentation canvas. For instance, the user searches for a specific keyword, and the results which might be a budge of documents are not just listed in a textual context but are visualized as POI. Besides the documents, also user model keywords are shown as POIs. In order to distinguish between a document POI and user model POI different forms of representations are used. For example, the document POIs are square formed, and the user model POIs are drawn using the circles.

An important feature regarding adaptive VIBE is the support of visualization approach based on a relevance method. That means the POIs are linked to each other based on how relevant they are to each other. As an example, if the user moves a POI object inside the canvas, the relevant POIs to the current one will be affected too, and the position will change together with the current one.

Ahn and Brusilovsky [15] presented an adaptive visualization approach which represents the query terms and user model terms in two separate groups and visualizes them using point of references (POIs). In contrast, we do not adopt the visualization interface based on the user model information, but we guide the user toward completing the visual analytic task by recommending the next step, or a sequence of interactions based on the current user behaviors. Moreover, the retrieved documents will be adaptively arranged according to their relevance to each group. Similar to this feature, the recommended interactions or sequence of interactions will be adapted dynamically based on the current user behaviors. Additionally, similar to this approach we introduced a default user model in order to provide the appropriate recommendation. In contrast to the user model presented in this paper, we learn the user model implicitly by analyzing the user behaviors. Finally, similar to the feature introduced in this paper which presented three POI layouts, we consider various information visualizations. In our case, we do not consider only three possibilities of visualizations, but twelve.

2.2.3. Adaptive Semantic Visualization for Bibliographic Entries

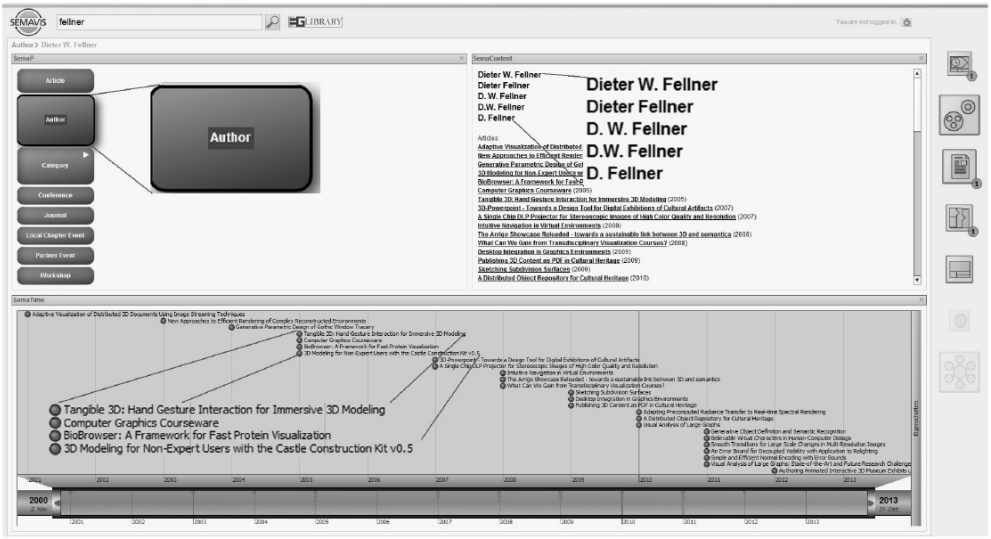


Figure 2.4.: An example using the term "Fellner" as a search keyword. Different visualization used to present the informations. The list of recommendations visualization shown on the section placed on the right side. [3]

K. Nazemi et al. [3] provides an adaptive semantic visualization system based on user behaviors. The target is to help users to explore the data visually by adapting the visualization based on the user behavior and content data. This approach, adapts the results visually based on a canonical as well as a personalized used model. The user interactions with the dashboard (e.g., visualization placed on the screen, or drags and drops interactions) are used to derive a canonical/personalized user model. On the one hand, to enable the personalized user model (see Figure 2.5), the user has to login intro the system. On the other hand, if the users are note logged in, then the canonical user model is used to adapt the interface. We follow a similar approach to build a default user model by collecting the user behaviors implicitly. Yet, we collect these information by tracking a larger number of interactions (e.g., filtering, zooming, brushing, etc.), and not only the basic interactions (e.g., selecting, drags and drops).

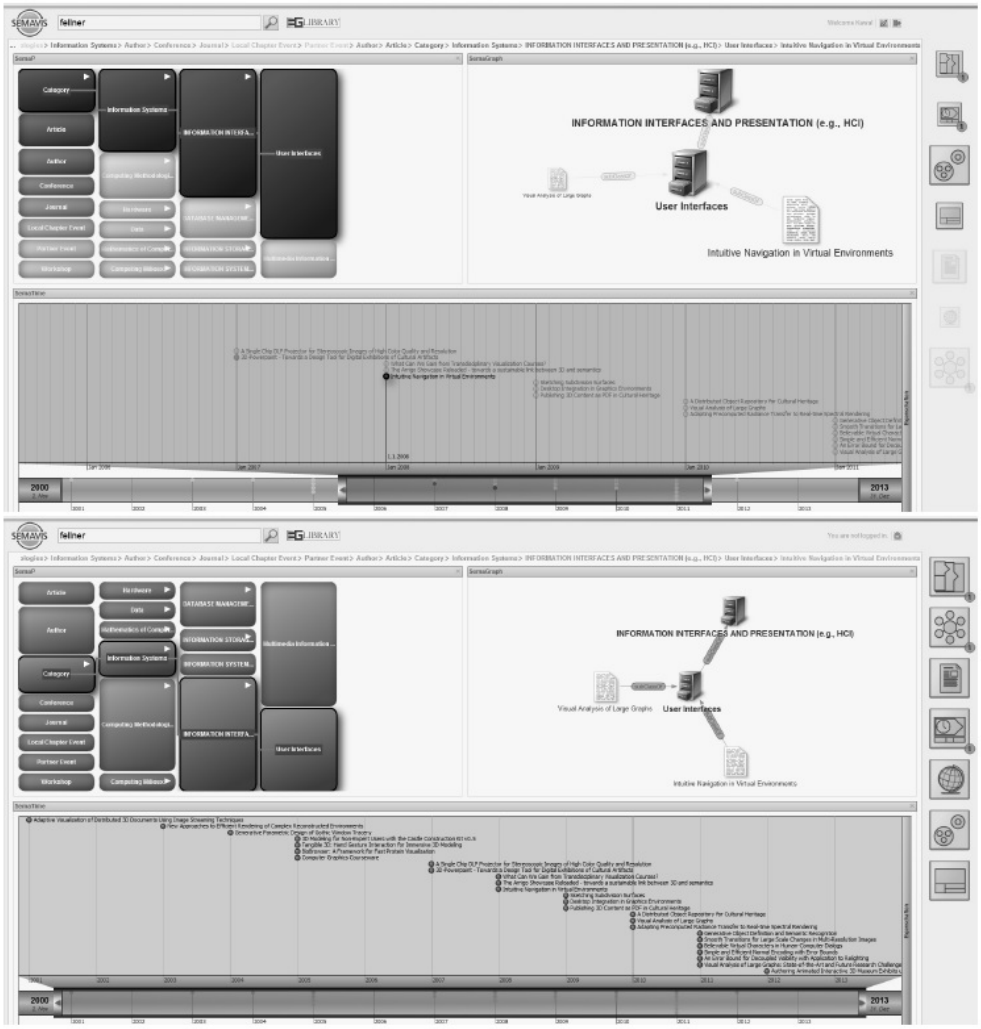


Figure 2.5.: Presentation Adaptation: the visualization on the top represents the visual adaptation based on the behavior of an individual user; the visualization on the bottom shows an example using the canonic user model.[3]

This approach is included in the visualization system called "SemaVis" (Figure 2.4). Furthermore, the user has the ability to interact with the system just by clicking or dragging and dropping the elements within the cockpit. Addition-

ally, SemaVis supports seven different visualization which are responsible for the placement and arrangement of the objects in the screen. Moreover, in order to provide more efficient adaptability, the visual layer is separated in spatial arrangement (Layout) and visual presentation (Presentation). The layout is responsible for placing and arranging different visualizations on the screen. Moreover, all visual variables are regarded as a presentation, e.g., color, shape, texture, size, etc. of edges, nodes, objects, etc.

Nazemi et al. introduced an adaptive approach which adapts the visualizations without the need for an expert to design the visual layer. In contrast to this approach, our system is concentrated on guiding the user toward solving the tasks by recommending the next interaction or a sequence of interactions instead of adapting the visualization automatically. Furthermore, we mainly focus on analyzing user behavior during data exploring process.

2.2.4. SeeDB

SeeDB [16] is a visualization system for digital libraries which considers user behaviors in order to recommend the most appropriate visualizations. The SeeDB is designed as a layer on top of a traditional relational database system (DBMS), and is comprised of two parts: frontend and backend. The frontend is responsible to issue queries and display the visualizations, whereby the backend executes the queries and retrieve the data from DBMS. Furthermore it performs all computations required to generate and select the most appropriate visualizations.

A typical use case within a SeeDB is shown in the following. First, the user issues a query on the frontend. Next, the backend performs all the required computations based on the specified query, and selects the most appropriate visualizations. Metric used to measure the utility of the visualizations, whereas the top-k visualization with the largest deviation from a reference dataset are chosen to be displayed. The deviation of a visualization is defined as the distance between the probability distribution of the target visualization and the probability of the equivalent visualization derived from the entire database.

In contrast, we introduce an approach which is based on user behavior during the whole analytic process. Furthermore, we provide a step by step analytic solution instead of showing only one most appropriate visualizations. Most importantly we consider the user preferences by providing a user model.

2.3. Rule Based Approaches

In the following, different visual analytics application are introduced with a focus on rule based approaches.

2.3.1. CODE - Visualization Wizard

Belgin et al. [17] proposed a visualization system which automatically recommends the most appropriate visualizations based on two crucial aspects:

- The content and structure of the data
- The semantics of the visualizations

The system called VizRec is integrated in a platform that is used to extract, organize and release research data originally extracted from scientific publications. The purpose of VizRec thereby is to visualize the extracted data to aid the intended data exploration. Note that the extracted data is structured following the specification of RDF Data Cube Vocabulary [3].

This section details the data analytics process and describes the strategy used to recommend visualizations.

Visualization Recommendation technique Belgin et al. method differs from many other solutions which recommends the visualizations considering the properties of the data being visualized (see Figure 2.6).

[3] https://www.w3.org/TR/vocab-data-cube/

Figure 2.6.: Example of automatically generated visualization data in CUBE[17]

In order to recommend a visualization, a description about the content of the data, and the visualizations are needed a priori. Therefore, Belgin et al. used a vocabulary, the Visual Analytics (VA) Vocabulary, that provides a common semantic descriptions of the visualizations using the Web Ontology Language. The vocabulary describes the visualizations with their visual channels (e.g., x-axis, y-axis, color, size, etc.), data-type they support (e.g., string, integer, date), occurrence (single or multiple) and persistence (mandatory, optimal). Further, a mapping technique is defined which defines a possible relation between dimensions (identify the observation) and measures (are related to concrete values) of the RDF data cube, and the corresponding visual channel of the visualization. In order to do this, the algorithm analyzes the structural- and data-type compatibility. For instance, the bar chart has two axes: x-axis and y-axis. Regarding the structural compatibility, the bar chart can only

visualize RDF data cubes with one dimension and one measure. With respect to the data-type compatibility, x-axis supports only categorical data whereby the quantitative data can be mapped onto the y-axis.

Finally, the recommended visualizations will be presented to the user as illustrated in Figure 2.6 (left): highlighting the icons of the recommended visualizations. When the user wants to create one of the visualizations, she just clicks on the highlighted button and the visualizations will be created automatically. In other words, the presented approach visualizes data automatically without requiring any manual specification from the user.

2.3.2. Visualizer

Visualizer is a web-based application with a thick client architecture which supports visual data manipulation and interactive data exploration. The Visualizer has a non-personalized rule based visual recommender which is always available. It provides two user modes: personalized, and non-personalized. The difference between using the personalized mode versus non-personalized one is that some features are only available in the personalized modus e.g., the personalized visual recommender). The target, is to recommend visualization to the user based on the user preferences.

Visualizer was improved during the work of this thesis, and used to integrate the developed recommender. A detailed description about the tool, and the improved aspect can be find in the chapter 3.

2.3.3. ShowMe

Mackinelay et al. [18] introduced the ShowMe system which integrates a set of user interface commands in order to generate visualization automatically for Tableau.[4]

ShowMe provides the user with the appropriate graphical presentations that may address his/her task. In order to select the appropriate visualizations, the data properties, such as data type (text, date, time, numeric, Boolean), data

[4]https://www.tableau.com/

role (measure or dimension), and data interpretation (discrete or continuous) are considered. Furthermore, ShowMe considers both new and experienced users. On the one hand, to help the new users, an intuitive and preditable presentation functionality that is provided automatically is needed. On the other hand, experienced user require a system that they will fit into their workflow (flow of visual analysis).

2.4. Predictive Models

In this section, we will present the existing theoretical works about recommendation models. The learning models presented in this section, are based on Markov chains.

Each approach is introduced including the used algorithm, and concluded by showing the differences, respectively similarities compared to our approach and the visualization system used to present our work.

2.4.1. An MDP-Based Recommender System

In this work, Shani et al. [13] presents a specific predictive model based on Markov chain and its improvements. Moreover, it details the evaluation of the model, and the evaluation methodology of the predictive Model. The performance of this model is evaluated on a commercial site. In this section, we will explain the predictive approach, the definition of the main Markov chain components and the evaluation metrics briefly, which represent the crucial aspects of this thesis.

The Markov Chain Model

In order to define a Markov chain model, developers need to specify three main components adequately. The first component is the space of states where each state represents a particular position in time. The second component is the transition function which represents the probability distribution of moving from current state toward all the states. Finally, the initial vector as one of

the essential components, represents the starting state. In this approach, a state represents the choices made by users within the system. For example, a specific order of items (market products) with a size k is a state, e.g. (x_1, x_2, x_3). Personal information about the user like age, gender, or profession is ignored. Further, when the data is collected from a commercial system which represents a complex set of sequences, and this leads to an unmanageable state space, with an high risk for problems such as data sparsity. Therefore, only sequences of at most k items are considered. The sequences are represented as a vector of size k, e.g. $(x_1, ..., x_k)$. Since the system collects the items ratings explicitly, the initial vector is a sequence of states in which the ratings are collected explicitly.

Suppose that a user has already selected items $(x_1, ..., x_k)$. The transition function calculates the probability that a user who already has selected k items $(x_1, ..., x_k)$ will choose the item $x_k + 1$. The transition function is defined using user data, and a maximum likelihood approach as shown in equation 2.1.

$$tr_M C(< x_1, x_2, x_3 >, < x_2, x_3, x_4 >) = \frac{count(< x_1, x_2, x_3, x_4 >)}{count(< x_1, x_2, x_3 >)} \tag{2.1}$$

Whereby count(x_1, x_2, x_3, x_4) represents the number of appearances of this sequence in observed data set.

Shani et al. considers higher order Markov chain models in order to improve the recommendation quality. Higher order Markov chain represent a Markov chain with a order higher than the first order e.g., second or third order Markov chain, etc. Futhermore, in order to estimate the quality of the predictive model, Shani et al. use two different evaluation metrics: recommendation score, which represents numerically if next item is among the top m recommended items, and the exponential decay score which does rate the recommendation not only based on the content of the list but also considering the order of items in it.

Similar to the abovementioned approach, we do not consider user information like age or experience although this could contribute significantly on improving the accuracy of the predictive model. Furthermore, we also consider higher order Markov chain models. In contrast, our tool is deployed in order to improve the user performance when performing visual analytics which is not the case regarding the Shani et al. paper (user satisfaction when buying

items on commercial site). Furthermore, the technique introduced by Shani et al. considers only the most frequent sequences with items size of only five. In contrast, we consider a higher length of sequences. Another shortcoming of this approach is that they consider accessing the items in sequential order. In this case, they consider only one direction when they model the dependency between items e.g., accessing item X leads to the item Y. Not that accessing item Y leads to the item X. In our approach, we consider the sequential order in both directions, because we believe that the order of interactions may be interchanged in many case while still preserving the same outcome.

2.4.2. Markov Chain Recommendation System (MCRS)

Ahmed and Naomie [12] introduce a novel recommendation system based on Markov chain model. In contrast to the previous approach [13] , they consider the time factor (considers the period when the same items were accessed from different users) for the recommendations. Furthermore, the model is deployed with web applications and is used to recommend to the user the relevant items she can be interested on. In the following, the model is introduced in more details, especially the definition of Markov chain's components regarding the domain (movies).

The first crucial component of the Markov chain model is the space of states. In this tool, the individual items are seen as states, whereby the whole set of items represents the space of states.

$$I = \frac{user\ vector}{(the\ number\ of\ times\ of\ accessing\ all\ items\ by\ active\ user)} \tag{2.2}$$

The initial vector (see equation 2.2) represents the so called active user's vector divided by the number of times the active user accessed the items. The active user's vector represents a vector of all items. Where each item is denoted using the numbers one or zero. The number one is assigned to the item that is currently accessed, otherwise it has the number zero. The last component is the transition matrix (see equation 2.3) which contains the probabilites of moving from one item to another. $row_{(i,j)}$ is the row of $item_i$ where i represents the row, and the j the column (item).

$$T_{(i,j)} = \frac{\sum_{i=1}^{n}(row_{(i,j)} \text{where the column of } item_{(i)} = 1)}{\sum_{j=1}^{n}\sum_{i=1}^{n}(row_{(i,j)} \text{where the column of } item_{(i)} = 1)} \tag{2.3}$$

This approach handles the sparsity problem by considering all sessions of all users. In other words, this technique ensures that the probability of accessing all items with any items is achieved.

In contrast, to our approach, only the first-order Markov chain is considered. Furthermore, the focus of this approach lies on recommending items in web applications (movieLens dataset is used to evaluate the model). However, we recommend user's next analysis step within a visualization system.

3. Visualizer and Visualization Systems

3.1. Visualizer

In this section, an overview of the Visualizer functionalities will be provided with a focus on components which have been defined within the scope of this thesis. This section is splitted into two parts: In the first subsection the architecture, used tools, and the functionality of the tool will be outlined. Since the visual interactions are one of the main focuses of this thesis, the second subsection will provide a detailed description of all the interactions. Here, we will explain what the purpose of each interaction is, and how it is implemented. Further, we will emphasize the interactions which we integrated within the scope of this thesis.

3.1.1. Visualizer Architecture and Functionalities

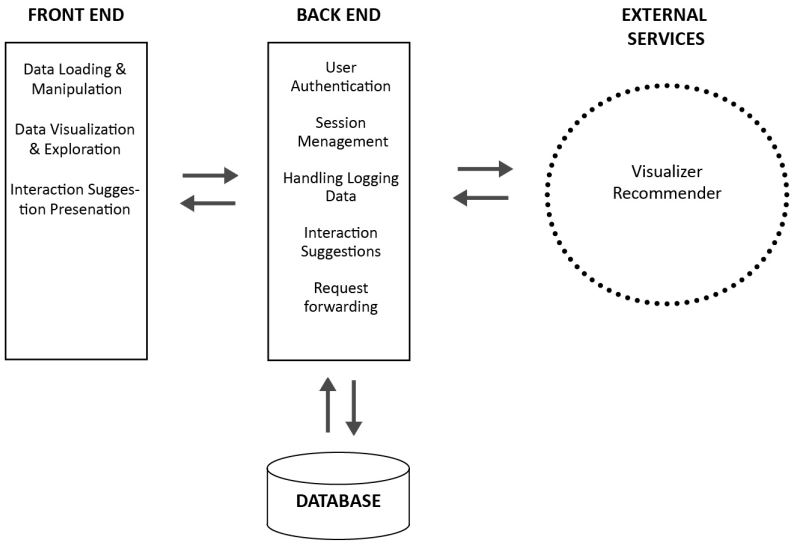


Figure 3.1.: Visualizer architecture components.

Visualizer is a web-based application with a thick client architecture which supports visual data manipulation and interactive data exploration. Visualizer takes into account both users with experiences, and without experience in information visualization. Further, it uses an external server to personalize the recommendations.

The architecture of Visualizer is shown in Figure 3.1. In a nutshell, Visualizer is divided in the front end and in a back end. It uses a database for user management, external services, and web applications.

Backend Visualizer mainly has a client-based architecture. However, some of the functionalities are moved on the back end because of the performance

and scalability issues. The backend uses Apache[1] web server which supports CGI scripts used to interact with the front end Python[2] scripts and SQLite 3[3] are used to collect user authentication data.

The backend is responsible for handling user session management, such as:

- User creation
- Authentication and personalized presentation
- Session management
- Managing user style sheets
- Managing user-uploaded visualizations
- Fetching datasets from online sources
- Handling the communication with the external services (e.g., recommender visualizer)
- Handling logging of user interactions
- Handling live tracking of user interactions

Handling logging of the user interactions and live tracking of user interactions will be explained here briefly (for a detailed explanation, please see Chapter 4 and 5). Logging of user interactions functionality is implemented to collect user interaction data and save it on the server.

For each session, the back end defines a new logger identification number (ID) which is saved and managed on the server side. Given that the logging data is periodically updated during a session, the ID makes it easier to identify the current logger file. Furthermore, logger handler receives these data in the JSON[4] format and saves these as JSON files into user activities container (folder). In addition to the interactions, these files also contains some additional data such as client id, timestamp, etc. that are needed for the behavior pattern definition.

Live tracking of user interaction is also responsible for user interaction monitoring, but in contrast to logging of user interactions functionality, it provides just information needed for the tracker analyzer (e.g., sessions of states).Yet, these data are only used by the interaction tracker to recommend (if available)

[1] https://www.apache.org
[2] https://www.python.org
[3] https://www.sqlite.org
[4] https://www.json.org

the next step, or a full path. The live tracking data is saved on the client side and are deleted after a session is finished.

External services A visualization recommendation is used as external service which defines personalized visualization. This is only possible when the user has an account and is logged in. Visualization recommender delivers to the frontend a ranked list of visualization based on the user preferences. In order to recommend the visualization, the recommender needs information about the user a priori, and a description about the current context. To do so, Visualizer sends a list with possible visualizations to the visual recommender as well as the current tags (tags are added to the visualization manually). Using these tags and, if available user's ratings, the system ranks the visualizations and presents them to the user as personalized visual recommendations [20].

Frontend The frontend of the visualizer is divided into three main interfaces: interface for loading the data, the dataset Table where the user can obtain an overview about the loaded data and perform data transformations and cleaning, and the visualization dashboard where the user can explore and analyze the data visually. In the following, we will focus on the visualization dashboard, as it is used as the main tool for this thesis to add the algorithms for tracking user interactions and learning from them to recommend the next interaction.

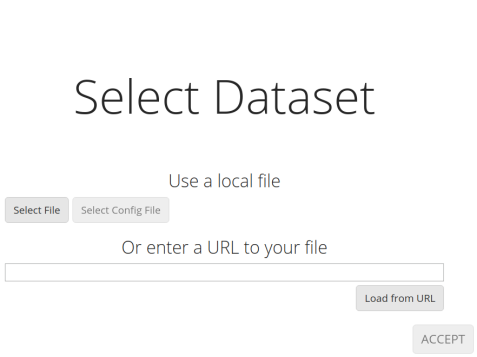


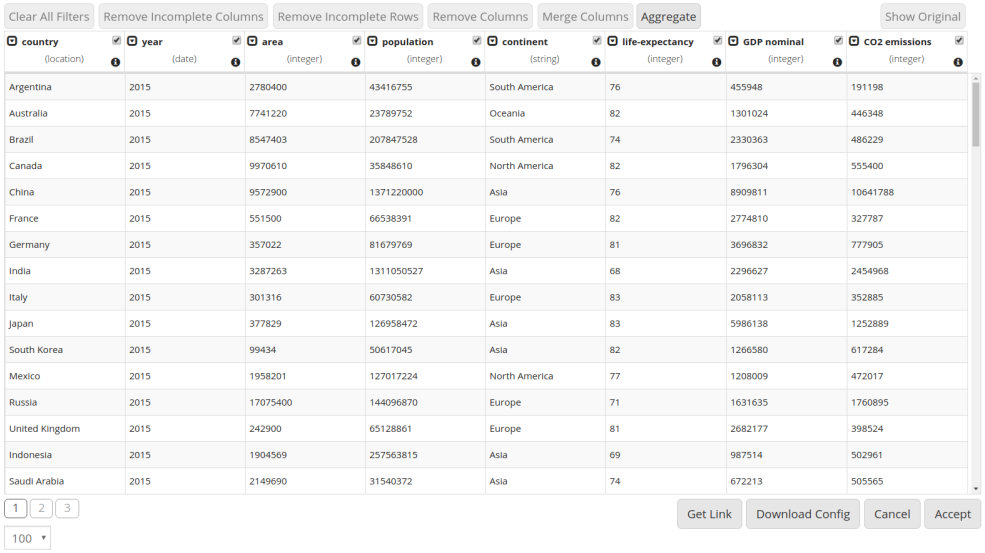
Figure 3.2.: Visualizer' user interface to load the dataset. User can load a dataset by selecting a local file, or using an external data source.

In order to demonstrate different user interactions, a dataset has been prepared which illustrates the CO_2 emissions of G20 countries.[5] This dataset also provides some additional information for each country, such as

- the continent where the country is located
- area of the country
- life-expectancy
- gross domestic product (GDP)
- year when this value has been measured

Assume, you have some data in the plain text, and you want to load this data in Visualizer in order to explore and analyze it. There are two opportunities how to load the dataset, as illustrated in Figure 3.2. The user can either load the data locally using the "Select File" button or from an external source using the URL to the source. After the loading, the data will be shown in a table (Dataset Table) as illustrated in Figure 3.3. Now, the user can modify the data using the data transformation operations integrated into the table.

[5] http:/dfat.gov.au/international-relations/international-organisations/g20/pages/the-g20.aspx



country (location)	year (date)	area (integer)	population (integer)	continent (string)	life-expectancy (integer)	GDP nominal (integer)	CO2 emissions (integer)
Argentina	2015	2780400	43416755	South America	76	455948	191198
Australia	2015	7741220	23789752	Oceania	82	1301024	446348
Brazil	2015	8547403	207847528	South America	74	2330363	486229
Canada	2015	9970610	35848610	North America	82	1796304	555400
China	2015	9572900	1371220000	Asia	76	8909811	10641788
France	2015	551500	66538391	Europe	82	2774810	327787
Germany	2015	357022	81679769	Europe	81	3696832	777905
India	2015	3287263	1311050527	Asia	68	2296627	2454968
Italy	2015	301316	60730582	Europe	83	2058113	352885
Japan	2015	377829	126958472	Asia	83	5986138	1252889
South Korea	2015	99434	50617045	Asia	82	1266580	617284
Mexico	2015	1958201	127017224	North America	77	1208009	472017
Russia	2015	17075400	144096870	Europe	71	1631635	1760895
United Kingdom	2015	242900	65128861	Europe	81	2682177	398524
Indonesia	2015	1904569	257563815	Asia	69	987514	502961
Saudi Arabia	2015	2149690	31540372	Asia	74	672213	505565

Figure 3.3.: Dataset Table Interface. Data can be viewed or modified using the integrated data transformation functions.

The data can be views (pagging), sorted, filtered and aggregated. Further, columns can entirely be removed, or the incomplete data completed. After clicking on the "Accept" button, the data will be passed to the Dashboard. Since the dataset Table is out of the scope of this thesis, we will not go into the details. For more details, we refer to [19].

The Dashboard is divided into four sections (see Figure 3.4):

- Toolbar (shown in Section C): It provides different possibilities such as user login, bookmarking, navigation (e.g., switching to the data table view), view (responsible to control the dashboard interface e.g., hide VisPicker section, etc.).
- Visualization Box (shown in Section A): It contains already generated visualizations.
- VisPicker(shown in Section B): This section offers the opportunity to select recommended Visualizations.
- Dimensions Section (shown in Section D): It contains all the dimensions (categorical data) and measures (numerical data) of the data.

The dashboard is designed in such a way that each user, regardless of being an expert or not expert in information visualization, can readily become familiar with the system.

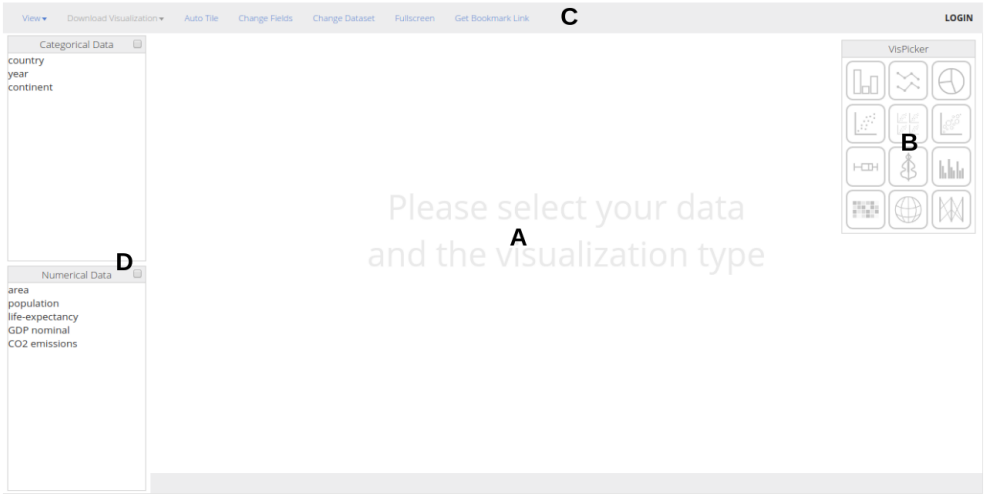


Figure 3.4.: Visualization Dashboard (Visualizer): (A) shows the created visualizations, (B) shows available visualizations, and if a visualization is recommended its icon will be highlighted, (C) Navigation tool intended to aid users in controlling the dashboard view and accessing information, (D) Dimensions section shows all the possible data fields (columns).

The process of data visualization in Visualizer is straightforward. In the first step, the user selects the dimensions which he/she wants to explore which are shown on the left side of the dashboard in ”Categorical data” area. The measures, however, are shown in ”Numerical data” area. Each time, the user selects a dimension or a measure, the icons of the possible visualizations will be highlighted in ”VisPicker” area. Yet, some visualizations (e.g., bar chart, pie chart) may need an aggregation of the numerical values, per default. To give user the opportunity to decide which aggregation (average, max, min) has to be performed, each visualization has been endowed with a ”aggregation dialogue”.

The Visualizer provides two user modes: personalized, and non-personalized mode. The difference between using the personalized mode versus the non-

personalized one is that there are some features only available for the personalized modus (e.g., personalized visual recommender). Only the users that are logged-in have the opportunity to use the personalized features. This is because, for the personalized visualization, the system applies user's preferences to recommend only the visualizations that are closer to what he/she is interested in. Further, users can upload and integrate their own visualizations, while the system is running. Uploading a new visualization requires from the user to implement a simple interface which is responsible for loading the data set.

The next section details the integrated interactions. We will also take a closer look at the components which have been defined or extended within the scope of this thesis.

3.1.2. Visualizer Interactions

In the previous section, an overview of the Visualizer has been provided, showing the most critical components of system's architecture and of the dashboard itself. This section provides more details about each component and demonstrates the integrated interactions in appropriate use cases.

Once the user passes the data to the dashboard, he/she will see the dimensions and measures of the dataset. In order to obtain visualization recommendations, user has to select at least one dimensions and one measure. In the following, we detail the selection of data fields, the visualization recommendations and, finally, the interactions.

Selecting Data Fields Data fields of the data set is divided into two categories.

- Categorical data: represents data that can be divided into categories such as gender, genre, age etc.
- Numerical data: represent data that can be measured such, as time, amount, height etc.

All the fields listed in the sections categorical data and numerical data are clickable. When the user wants to visualize the values of a field, she just selects the corresponding field. Figure 3.5(*a*) illustrates how the fields are subdivided into categorical and numerical data and are ready to be selected by the user. Figure 3.5(*b*), however, illustrates how the user selects data fields of interest.

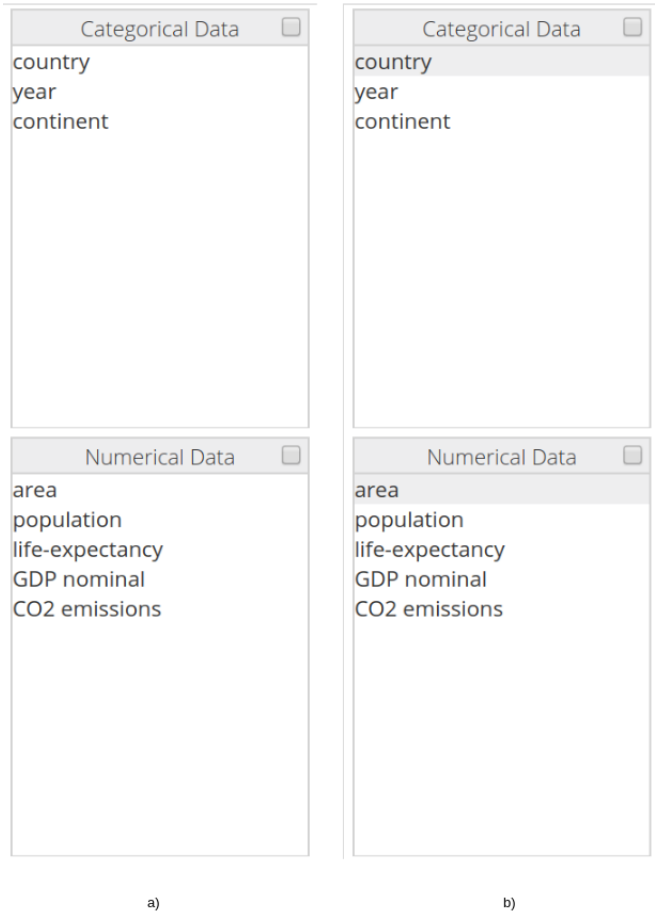


Figure 3.5.: Selecting data fields. a) Subdividing data fields into two groups: categorical and numerical. The user has to click on a field in order to visualize its values. b) The user has already selected data field *country* and *area*.

Note that choosing a field to explore, or removing already selected dimensions are interactions which are tracked during the interaction analysis.

VisPicker Assume the user has already chosen some fields to explore. Subsequently, the system will automatically recommend appropriate Visualizations which are highlighted on the VisPicker. By the time of writing this Thesis, twelve visualizations were included into the dashboard (see Figure 3.6). The visualizations are mainly implemented using D3[6] (Data driven Documents) javascript library. The integrated visualizations are:

- Bar chart is a graph which represents categorical data in the x-axis associated with numerical data in the y-axis.
- Pie chart is a graph divided into slices. The bigger slices represent the dominant ones regarding the numerical distribution.
- Line chart is a view, which represents the data using series of data points connected with a straight line.
- Scatterplot is a diagram based on cartesian coordinates which are used to display the correlation between two variables within the data set.
- Scatterplot-Matrix contains all the pairwise scatterplots of the available variables.
- Bubble chart is a chart that displays the correlation between three variables, where the third one is represented by item size.
- Boxplot is a one-dimensional graph of numerical data based on the five number summary, which includes the minimum value, the 25th percentile (known as Q1), the median, the 75th percentile (Q3), and the maximum value.
- Violinplot is a way of representing the distribution of the data and its probability density.
- Grouped Barchart is an extension of bar chart including the third variable which clusters the data in distinct groups.
- Heatmap is a visualization of data representing the distribution as colors
- Map is a chart representing data often in a geographical context.
- Parallel Coordinates represents a way of visualizing high-dimensional data.

[6] https://d3js.org/

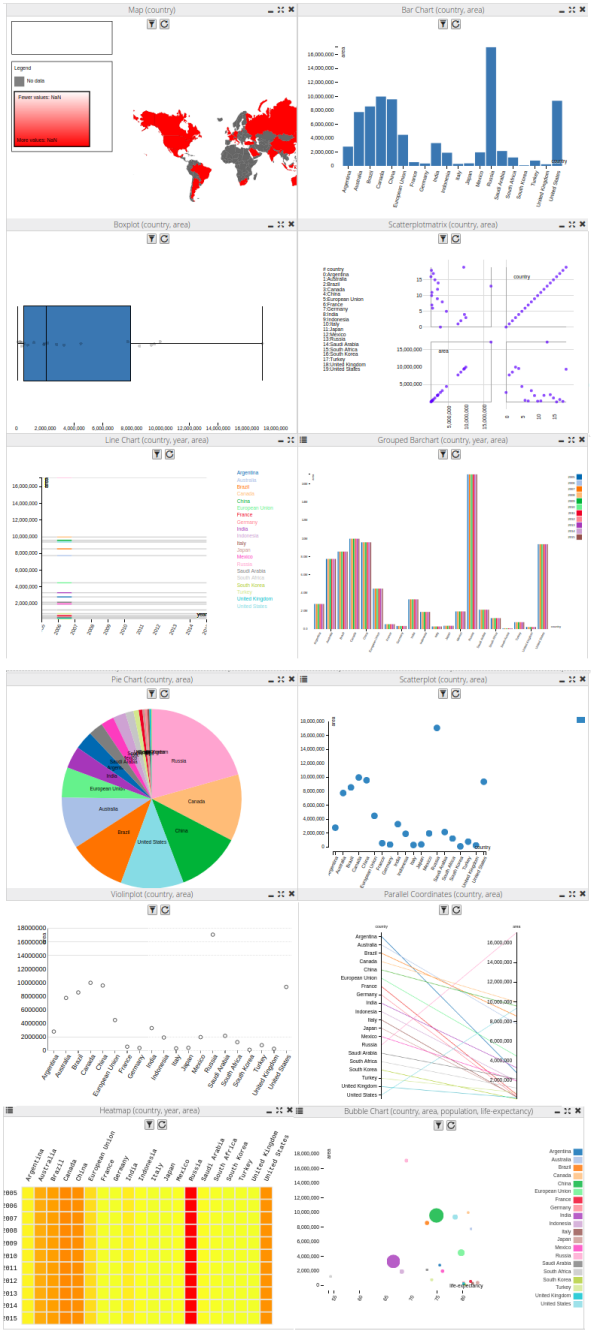


Figure 3.6.: Visualizations supported by Visualizer.

Every time, the user selects or deselects a field, the visualization recommendation on the VisPicker will be adapted as illustrated in Figure 3.7(b).

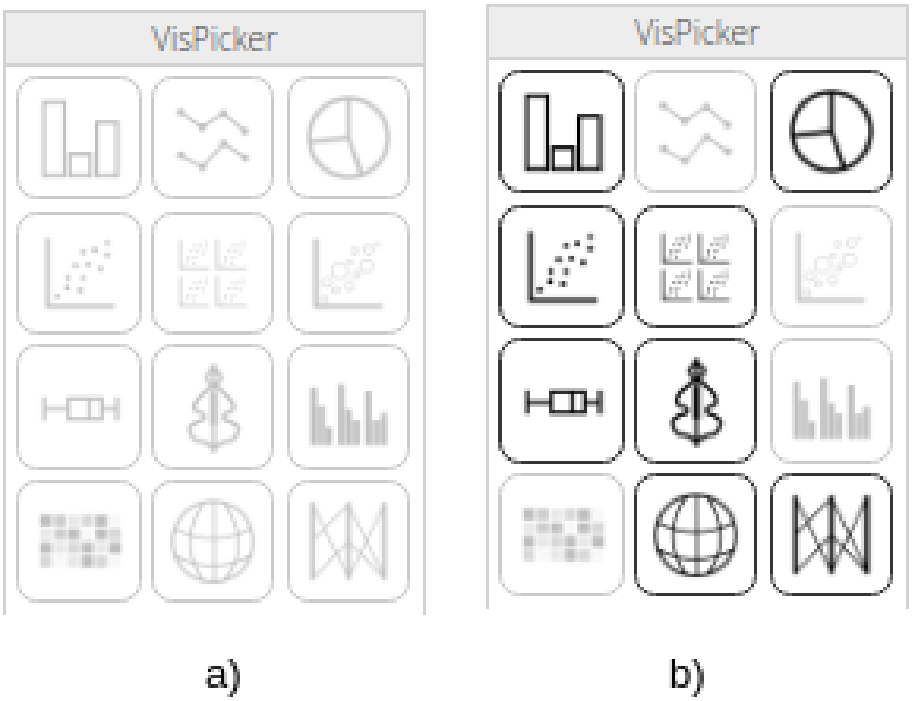


Figure 3.7.: VisPicker examples: a) since there are no fields chosen to explore, all the visualizations are disabled. b) Some fields to explore are already selected, and corresponding charts are highlighted.

Choosing a visualization within the VisPicker is also an interaction which will be tracked during the interaction analysis.

Thus, assume that the user has decided to visualize the selected fields on a bar chart. To do this, she clicks on the bar chart icon on the VisPicer. Now, the system requires from the user what the height of the bars should represent: average value, the maximum or minimum of the values or the sum (see Figure 3.8). Please note, for some visualization the aggregation is optional.

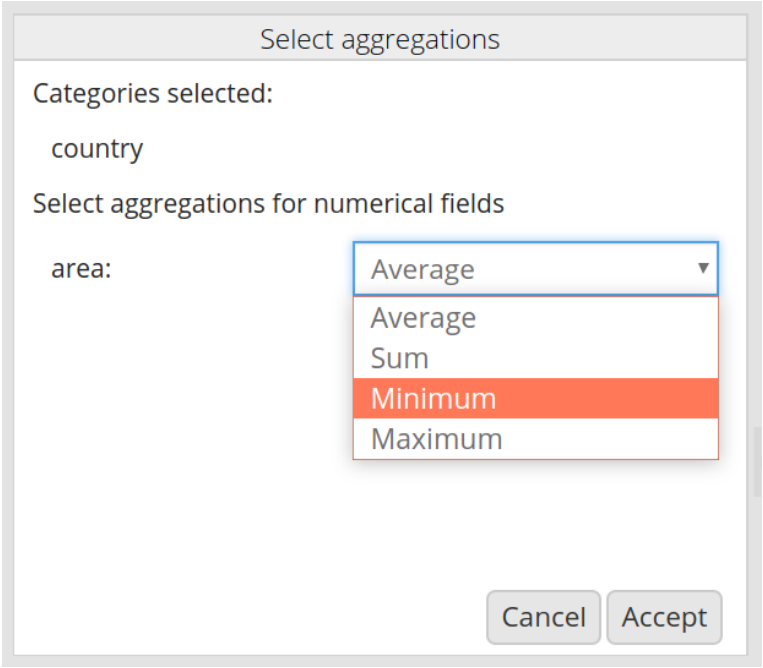


Figure 3.8.: Aggregation of field area using Minimum.

After the user has decided how the data should be aggregated, the system automatically generates and displays the visualization (see Figure 3.9).

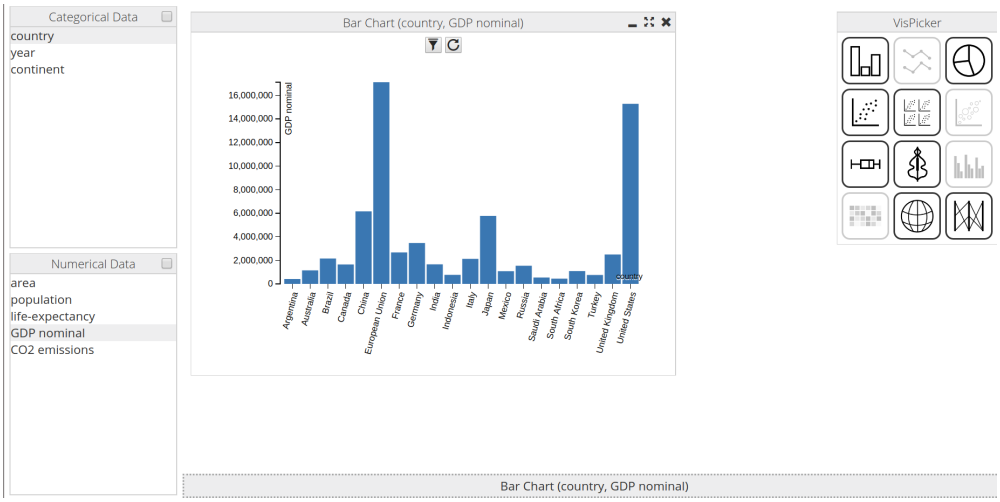


Figure 3.9.: Dashboard example with a generated Bar chart, showing nominal GDP per country

Visualization Box The Visualizer's visualization Box is designed to contain multiple visualizations, each of them shown in a separated window (I-Frame). The visualizations (windows) inside the visualization Box have some common features. For Instance, they can be moved freely, resized, minimized, or closed again. Moreover, each visualization has a variety of features included to support the user when exploring the data:

- Zooming and panning
- Filter
- Revert Filter
- Linking and Brushing
- Channel remapping

Zoom data inside the visualization Zooming the data inside the visualization increases the quality of the data exploration and analysis since the user can focus on a more detailed view of a certain part of his/her data. Figure 3.10 illustrates how zooming works. The bar chart on the left side shows the entire dataset, the same chart with a zooming effect is presented on the right side which shows only a certain part of the user's data.

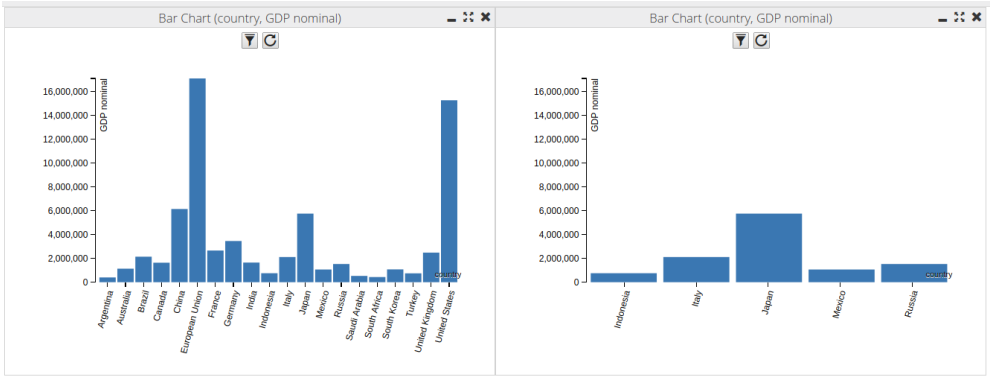


Figure 3.10.: Zoom applied inside the visualizaiton. a) The visualization without aplying zoom. b) A zoomed Bar chart

The zooming behavior is implemented as a distinct method inside every single chart. Within the visualization, the zoom event is typically activated using functionality from D3 library and embedded to the deployed zooming method.

Listing 3.1: Enabling zoom event.

```
var svg = d3.select(targetSelector)
        .append("svg:svg")
        .call(d3.behavior.zoom()
        .scaleExtent([1, 10])
        .on("zoom", zoom))
        .on('mousedown.zoom', null)

function zoom() {
//  zooming functionality implemented here
}
```

Filter Two Buttons placed inside the visualization window are used to filter the data, or if already filtered, to revert the chart to its initial state (having all data shown). Clicking the button on the left side will open a filter dialog as shown on the left chart in Figure 3.11. The categorical and numerical

data types are handled in different ways when filtering data. Therefore, two different kinds of filters are introduced: numerical (range) and categorical (check boxes) ones. The Figure 3.11 shows the same bar chart after filtering the data.

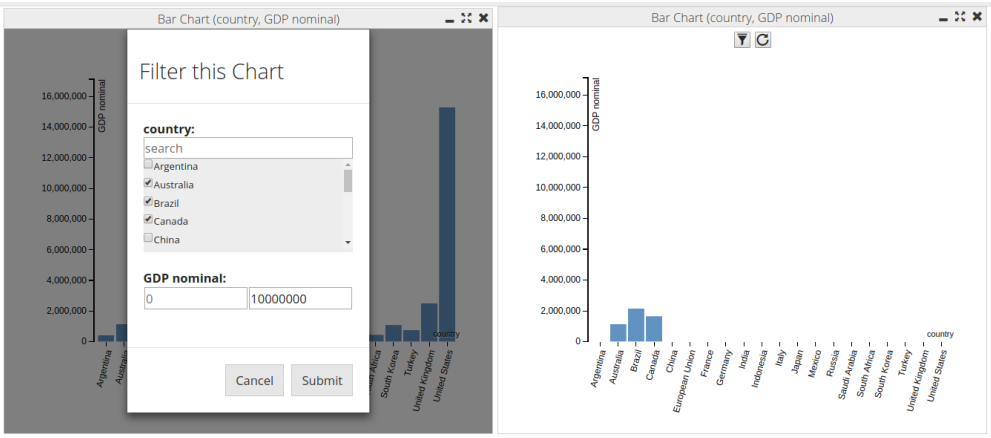


Figure 3.11.: Filter example applied inside the visualization. a) The visualization with opened filter dialog. b) A Bar chart showing only filtered data

The integration of the filtering required some changes in the main javascript file. For example, to integrate the Filtering functionality in the Bar chart following changes has to be made.

Listing 3.2: Integrating filter functionality.

```
filterObj = new filterChart (svg, channelMappings,
        data, color, x, y, chartName, chartRowIndex)
filterObj.createFilter()
filterObj.createDialog()
```

First, a new filter object is created (shown in listing 3.2). Since inside the filter class, there are some methods implemented to modify the SVG[7] (Scalable Vector Graphics) object, the SVG object is passed as a parameter also passed is the information about the channel mappings (mapping of the x-Axis/y-Axis and the corresponding channel), all the channels and the data. The

[7]https://www.w3.org/Graphics/SVG

last two parameters are there for naming and referencing the objects. The chartRowIndex is the index of the current visualization since the visualization Box can contain more than one visualization at the same time.

Moreover, the handling of the filter button is required to call the method openDialog(). This method is implemented in the filter file, and it creates the filter dialog with the appropriate data.

Listing 3.3: Triggering filter dialog.

```
$("#filterBtn").on("click", function (e)
{
   filterObj.openDialog()
});
```

Revert Filter The filtered data can be reverted to the original view using the button "revert filter" placed alongside the filter button. The process of bringing the user into the initial version is shown in Figure 3.12. The chart on the left shows the filtered data whereas the chart on the right side shows the original data, after the button "revert filter" was clicked.

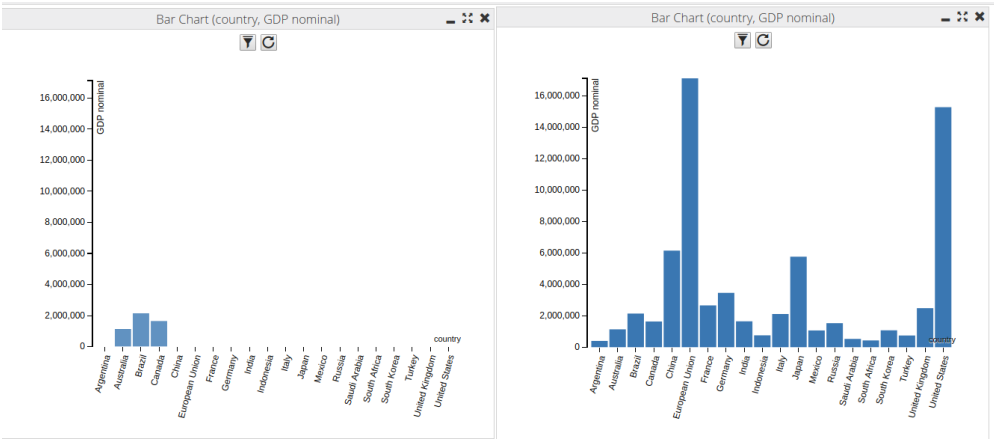


Figure 3.12.: Revert to original view after a filter is applied. a) A Bar chart showing filtered data. b) A Bar chart after revert filter is applied

Linking and Brushing Linking and Brushing is one crucial interaction supported by each visualization. The main idea behind linking and brushing is to combine different visualizations to overcome the shortcomings of single techniques. Interactive changes made in one visualization are automatically reflected in the other ones. Visually, the data values selected by the brush retain their original color, while data elements not selected by the brush are shown in gray (see Figure 3.13).

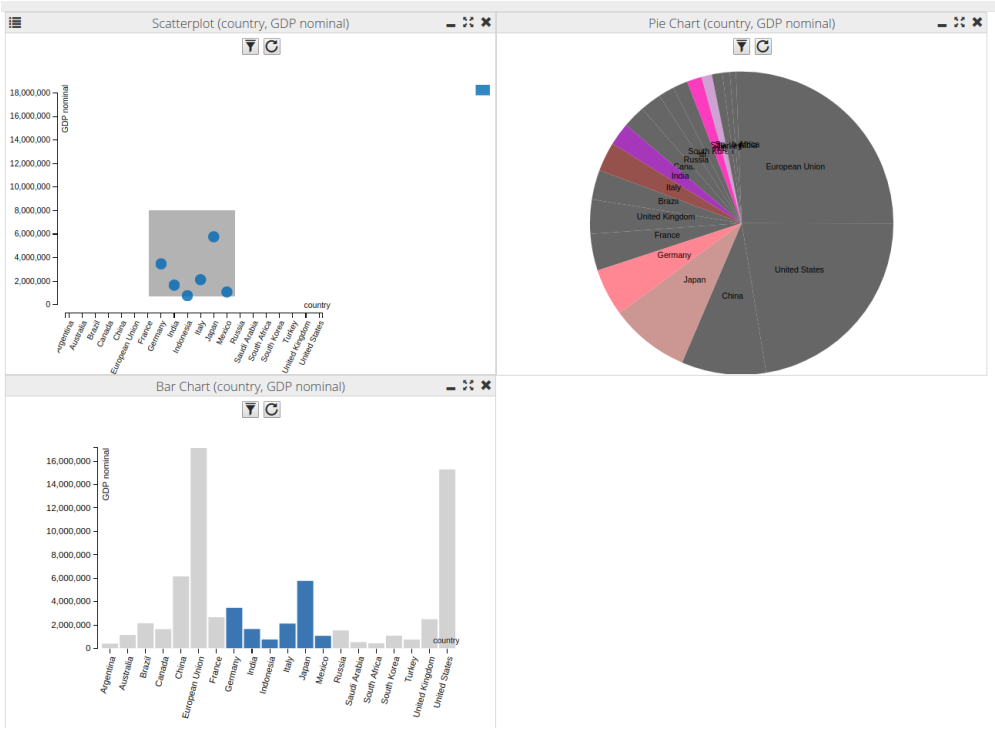


Figure 3.13.: Linking and brushing example showing related data between three visualizations. The brush, visible as a semi-transparent rectangle, is applied in the scatterplot.

The goal of the brushing and linking feature is to interconnect multiple visualization, respectively the information shown within these visualizations. Therefore, the brushing class in the implementation of this feature acts as an observer class, which listens for new information from all registered instances (already created visualizations). Every visualization, whenever it is

created, will be registered to the brushingObserver (the class that controls and is responsible for transferring the new information between already created visualizations). The BrushingObserver will have control over all the visualizations, and will wait for the notification from any of them. If a notification comes, the "BrushingObserver" will handle the related data separately for each of the visualization and will send a notification to all of them. Besides registering itself to the brushingObserver, there are two crucial tasks to be fulfilled by every single chart. First, for any brushing event within the chart, the current visualization will notify the observer about the changes. Second, a brushing handler is implemented inside the chart which is responsible for handling any notification coming from the brushingObserver. Suppose that there are already two charts in the visualization Box. In the visualization 1, some data is brushed. Next, the visualization 1 notifies the brushingObserver and sends the brushed data. The brushingObserver takes this data and notifies all other registered charts. The visualization 2 highlights the related data, it has received from the brushingObserver.

Channel remap The option of channel remapping is possible only on the visualizations that support multiple mappings of fields (a mapping actually represents one possible configuration for a particular chart), such as scatterplot which possesses three fields. The button on the left side in the visualization toolbar is for channel remapping. Clicking the button a dialog will open which offers the possibility to change the mapping of the fields to visual channels. In Figure3.14 the remapping dialog is shown for a scatterplot chart.

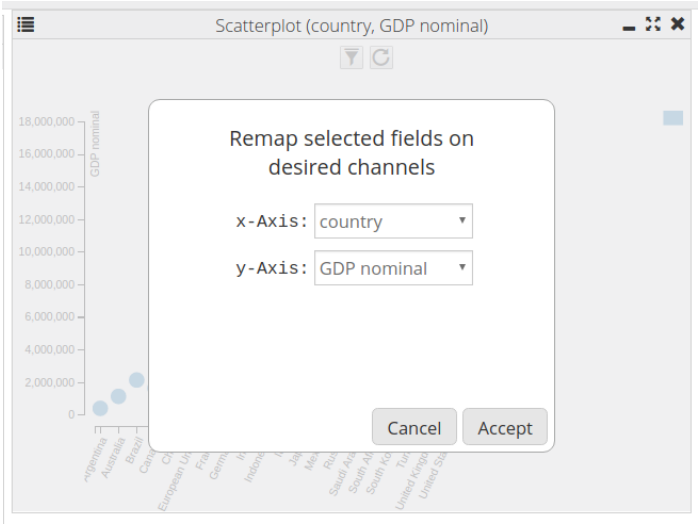


Figure 3.14.: Remap channels dialog. Connect data fields with visual channels of a chart.

Since these interactions play a crucial role in exploring and analyzing the data in visual context, all the interactions described in this section are logged and included in interaction tracking analyzer.

Toolbar The toolbar is the fourth component of the dashboard which contains some important features. Figure 3.15 shows the toolbar including all the elements. Some of the features included in the toolbar are within the scope of this thesis. Hence we will explain all the features briefly, and we go into more details for the ones that are relevant for the interaction tracking analyzer.

Figure 3.15.: Toolbar including all elements.

The button view placed on the left side of the toolbar provides the possibility to personalize the dashboard interface. Clicking the view button a drop-down list will be shown (Figure 3.16) where the user can click over the elements of the list to activate or deactivate the selected section (e.g., VisPicker). The

button "Download Visualization displays a list of all currently generated visualizations as shown in Figure 3.17. The user can download a screenshot of any of the visualizations using this feature. AutoTile is another essential feature of the dashboard and is responsible for automatic arranging of all visualizations inside the visualization Box. The AutoTile interaction is included in the user interaction tracking analyzer.

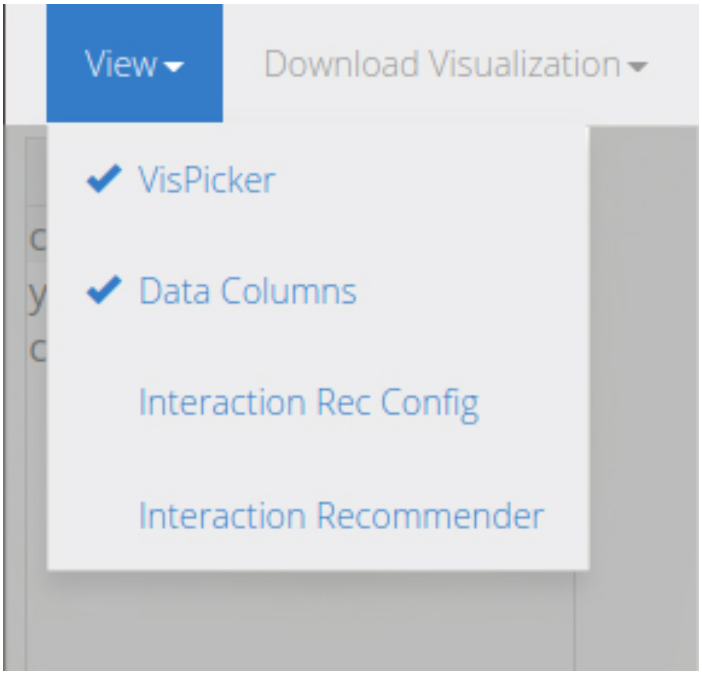


Figure 3.16.: View dropdown menu.

The next two buttons "Change Fields" and "Change Dataset" are for navigating to the Dataset Table described in Section section 3.1.1. Clicking "Change Fields" Button the system will navigate the user to the dataset Table interface, where the user can modify the dataset using the table form. Clicking the Button "Change Dataset" will navigate the user to the Loading data interface where the user can load a different dataset. The next button called "Fullscreen" is responsible for changing the view of the dashboard from the default to full-screen mode.

Another important feature within the toolbar is the "Get Bookmark Link" which will generate a link that captures the state of the dashboard including all created visualizations, their positions, size, the data mappings, all performed data transformations (aggregations, filters etc.), and the current brush. When opening the bookmark link in the browser, another (or the same) user can instantly recreate the previous state of the dashboard. This interaction is also included in the user interaction tracking analyzer.

The last button, placed on the right side of the toolbar, is there for the users that want to use a personalized version of Visualizer, where the user has to be logged in. A personalized session provides some extra features which are not offered to the guest users. For more details about the personalized visual recommender we refer to [20].

Figure 3.17.: Download visualization list.

4. Learning and Recommending Interaction Sequences

This Chapter outlines techniques for collecting, preprocessing, learning, and recommending interaction sequences. Additionally, the implementation of these techniques is described.

Our primary intention is to develop a predictive model that can recommend the next user interaction or a whole path (sequence of interactions) within an interactive visualization system. The fundamental idea behind our work is to support users in exploring the data which should lead to a increased level of user satisfaction and a higher success rate in completing the analytical tasks. In order to realize this idea, analysis of user behavior during different stages (gaining overview, identify trends, find anomalies, and analyze data distribution) of data exploration is required. Concretely, we focus on how users interact with the system when exploring the data. First, the user interactions are collected. Next, the relevant ones are appropriately preprocessed and structured, and finally, the collected data is used to build a statistical learning model which is then applied to generate recommendations.

This Chapter is split into three sections. First, the technique of collecting data is introduced. In the second section, data preprocessing is explained in detail. This stage is responsible for cleaning and customizing the collected interaction data so that they can fit into the model. Finally, the predictive model is explained into detail. It demonstrates the learning algorithm, the techniques used to build it, the process of task recognition and how the data is used for training to generate the model for a specific task. Given a particular context (e.g., selection of data fields), the model is then used to generate recommendations.

4.1. Collecting Data

Collecting quality user interaction data is one of the most important steps during the whole process which starts with user exploring a dataset and ends with the definition of the predictive model.

The entire process starts with collecting raw interaction data. Thereby we focus on collecting as much data as possible to increase the likelihood of having collected all the information required to support the later exploration purposes.

The goal of this step is to track and collect all user interactions performed by interacting with Visualizer. Collecting data within a visualization system is quite challenging: one has to define a full session (star and end of session), handle parallel sessions (distinguish between all active sessions), and handle datasets and tasks.

This section is divided into two subsections. In the first section, the challenges and the techniques used to handle the logging process are described in details. The subsequent section describes the implementation of the logging process.

4.1.1. Logging Methodology and Challenges

Various tasks regarding the functionality of data logging are listed below:

- Logging Initialization
- Interaction Handling
- Session Handling
- Data Format Storage
- User Session Handling
- Dataset Handling
- Task Handling

In the following, we will detail each of these problems separately.

Given that every interaction event has its own signature description (various parameters used to describe the event), the section ”Logging Process” describes every single event and its signature.

Logging Initialization When a session starts, the logger process will be initialized. It means the dataset name, logger-ID (current logger session ID) and current Task (used during the data collection process to label the data with the correct task) are extracted and stored in local storage on the client side.

Interaction Handling In the dashboard, the user can perform various interactions (e.g., field selection, create a visualization, zooming, filtering, etc.) which vary both in functionalities and in parameter description. Each interaction has its own features, and therefore, it is essential to handle each interactions separately. Filter and zooming, for instance, are two different interactions, and thus require a different type of information. On the one hand, to describe a zoom event the scale value is needed. On the other hand, a filter event is defined using the selected data/numerical range to be filtered. Thus, every interaction event has its own signature description.

Session Handling One of the main concerns with regard to collecting data within a session is to be aware of the start and the end of a session. A session in Visualizer begins when the Visualizer dashboard is loaded into the browser and ends when the dashboard tab is closed. Another issue is that each session is unique, so, it is crucial in managing it accordingly.

In order to address the concerns mentioned above, we follow a specific method which we will describe in the following. When the Visualizer is loaded in the browser, a session request is issued from the client side, and the server generates and returns a unique session ID. As a result, we now have two crucial pieces of information on the client side: the session has started, and a unique ID to this session is assigned. Next, all the user interactions within this session will be tracked and saved on the server side in a specific file containing a clear description of the session, session start timestamp, and session ID. The user interactions are collected in real-time, meaning that each interaction of the user triggers the server to update these data.

Data Format and Storage When collecting data, a significant aspect is the supported format of the current dataset and how the data is stored. Yet, with regard to the data we collected about the user, we focused on two issues:

data transmission, and data format. Our goal, on the one hand, was to have a high-speed data transmission and on the other hand, a flexible data format that is easy to import and export. Considering that, JSON seemed to be the most proper format to work with: it is flexible, compact and comfortable to use.

Every interaction event is collected and structured using JSON format on the client side. All the interaction events during a session are pushed in a list of JSON objects and transferred directly to the server where they are saved in JSON files. As already mentioned, all these JSON objects have a different signature description, and therefore, the JSON list will contain objects with various signatures description. The events within a JSON object are formated as follows: $'order'\_'eventName'$. Further, within a JSON object (event) we can collect even lists of data (e.g., list of brushed data) which in our case is crucial to define recommendations.

Users Session Handling So far we have described how a session is defined and how the data is collected in a specific session. Another important aspect is to identify sessions performed by a particular user. On the client side, we generate a particular ID of the device with a length of thirty two digits. The device ID is saved in DOM storage which is accessed using Web Storage API[1] and is sent together with the session data to the server. The device ID helps us to identify which session is collected with a particular device.

Dataset Handling The Visualizer is a system designed to visualize and explore any kind of user dataset provided in CSV format. Therefore, the data set loaded during a session is identified and stored on the client side. Further, every time the session data is logged, the data set name is transmitted together with the session data to the server side. On the server side, the data set name is saved together with the session data, in order to facilitate the identification of the used data sets during that particular session.

[1] https://developer.mozilla.org/en-US/docs/Web/API/Storage

Task Handling The goal of this thesis is to guide the user in performing visual tasks (gaining overview about the data, detecting trends/outliers etc.). Therefore, the identification of user's current task is one of the challenges we have to address. To achieve this, we extended the visualizer path with a new parameter named "task". This parameter is initialized with the corresponding task number representing the task ID. The task ID is generated on the client side, and together with the session information created during the session initialization process, is sent to the server side.

4.1.2. Logging Process

In the previous section, all the essential logging components are introduced. In this section, we detail the logging process.

The logging process starts when a new session (Visualizer dashboard is loaded) has been initialized and can be split into two essential phases: logging initialization and data collection.

Logging Initialization When a session starts, the logging process will be initialized. It means that the dataset name, logger ID and current task are generated and sent to the server side. On the server side, the logger handler takes over the task of handling the received data to identify the user, the dataset, the task and the session. Note that these parameters are summed up in one term: initial parameters.

The collected data are stored as JSON file which are placed in the buffer "user-activities". The extracted information is used to name the file, that looks like this:

$[Task]\_[Dataset]\_[DeviceID]\_[SessionID].json$
e.g.
$1\_G20 - Countries\_1294d7a8 - c884 - 4fb3 - 9a2b - 54feb22da585\_44.json$

Data collection In this phase, every new interaction will be registered in the logger local storage, and at the same time will be encoded as a JSON object including the initial parameters. This will then be sent to the server where the already generated JSON file for this session will be updated, whenever new data is received from the client side.

The Table 4.1 describes the JSON object structure and all its parameters which are sent to the server.

JSON Object logger	
Parameters	**Description**
Task	The task name
Dataset	The dataset name
DeviceID	The device ID
SessionID	The current session ID
Data	The lists of JSON objects (interaction events)

Table 4.1.: JSON object parameters

In the following we introduce the most important events and their parameters description (see Tables 4.2 to 4.9 below):

Filter event parameters	
Parameters	**Description**
ChartName:	Name of a chart used for the interaction
Timestamp:	The timestamp of the event occurrence
Channels Mappings:	Information about current channel mappings regarding the current visualization
Selected Categorical Data:	The lists of selected filtering data
Selected Numerical Data:	The lists of selected data ranges

Table 4.2.: Filter event parameters

[H]

Zooming event parameters	
Parameters	**Description**
ChartName:	The name of the visualization the user interacted with
Timestamp:	The timestamp of the interaction occurrence
Pan in X-Axes:	Information about the changed position over the x-axis
Pan in Y-Axes:	Information about the changed position over the y-axis
Scale:	Information about the data scale status

Table 4.3.: Zooming event parameters

Brushing and Linking event parameters	
Parameters	**Description**
ChartName	The name of the visualization the user interacted with
Timestamp:	The timestamp of the interaction occurrence
selectedData	The data selected (=brushed) on the current visualization

Table 4.4.: Brushing and Linking event parameters

Dimensions selelecting event parameters	
Parameters	**Description**
Timestamp:	The timestamp of the interaction occurrence
Types	The type of data field (categorical, numerical)
Labels	The selected data fields

Table 4.5.: Dimensions selelecting event parameters

Chart selelecting event parameters	
Parameters	**Description**
Timestamp:	The timestamp of the interaction occurrence
Chart	The selected visualization

Table 4.6.: Chart selelecting event parameters

Download visualization event parameters	
Parameters	**Description**
Timestamp:	The timestamp of the interaction occurrence
Filename	Downloaded filename
ID	Visualization ID

Table 4.7.: Download visualization event parameters

Bookmarking event parameters	
Parameters	**Description**
Timestamp:	The timestamp of the interaction occurrence
Link	The Bookmark Link

Table 4.8.: Bookmarking event parameters

New Mapping event parameters	
Parameters	**Description**
Timestamp:	The timestamp of the interaction occurrence
newMapping	The new mapping

Table 4.9.: New mapping event parameters

4.2. Preprocessing Data

In this phase, we clean and structure the collected data in such a way that at the end of the process, it can be readily analyzed by the algorithms. Since we are modeling sequential data using a learning model based on Markov chain, we parsed the data in a specific format (see section 4.1). These data are ready to be loaded and to fit into the learning model. The cleaned data is processed and structured in the CSV format where each line represents a session sequence, and each word within a sequence represents a specific interaction. For example:

start, select_dimension_country, select_dimension_GDP_nominal, Finish
start, select_dimension_country, select_Map_Chart, Finish

4.3. Predictive Model

This section introduces a predictive model based on discrete statistical sequences [8, 12] which models the user behavior and generates recommendations within a specific task. Moreover, the theory behind task recognition is introduced. Furthermore, the process of integrating the tool into an existing visualization system, Visualizer, is described in detail.

In other words, the fundamental idea of this work, is to recognize the ongoing task and show to the user step-by-step instructions on how to succeed in performing the task. In this case, the whole process is split into two main parts. In the first step, we attempt to recognize the task. In the second one, we will concentrate on recommending a step-by-step solution to the user within previously identified task.

4.3.1. Task Recognition

The goal of this phase is to identify the different tasks based on the m first interactions using a classification model. We ran a random forest [14] technique over the data collected during the visualizer evaluation process (that has been performed to collected user data by performing visual analysis tasks). More details about the evaluation can be found in the Chapter 6

Random forest is a machine learning strategy which is able to handle supervised regression and classification problems [21]. Further, it can be helpful on the classification of important features within the data[21]. The main idea behind this algorithm is to apply the ”divide and conquer” principle. The data is sampled randomly in small parts then used to grow a randomized tree predictor within each small data subset. The idea behind generating multiple random prediction trees helps to increase the diversity in forests, which leads to more robust overall predictions [22].

The random forest provides a high quality accuracy, is simple, and has the ability to deal with the small data problem. It performs well when small data set is provided, because its fundamental idea is based on the wisdom of the random and diverse crowds.

To train the algorithm for the task recognition, we only use sequential data with a specific length. The focus is to evaluate the performance of model, regarding only the first x (e.g., three) steps. The position of the events within the sequences define a specific feature. For instance, all the first events regarding all sequences within the training data build a specific feature. In this work, we used random forest implementation from Scikit-learn[2] which is implemented in python and supports various data exploration and machine learning techniques.

4.3.2. Modeling user behavior

This work aims to explore and analyze sequences of events issued by users working in a dashboard. The user is required to perform a chain of interactions (e.g., select data attributes, zoom data, etc.) to accomplish a task with the dashboard. Having collected interaction data, we will build a model that captures these sequences and learns from them in order to recommend visual interactions (analysis steps) that should assist the user in completing his/her visual task.

In our work, we use a simple Markov chain model to capture this information and build our predictive model. Markov chain is a process invented by Andrei A. Markov, assuming that a given trial could influence the result of the next trial [23, 8].

Let's assume that our system Visualizer is trivial and provides only two visualizations: bar chart and line chart. When the user decides to explore fields "country" and "area", he/she has three possibilities to interact with the system. First, the user can stay in the same state (it means doing nothing). Second, the user can select the Bar chart to explore the data, which means changing from current state to "Bar chart state". Third, the user can select the Line chart to explore the data which means the change from current state to "Line chart state". An Important and interesting question for the user might be now: Which is the best next state to explore the data quickly and gain a better overview of the data? An essential feature of this behavior is that the

[2] http://scikit-learn.org/stable/

next state depends only on the current state. This feature corresponds to the essential property of Markov chains.

The next section introduces the Markov chain including the basic concepts and notations. Furthermore, the appropriate definition of these components that are used to integrate a Markov model into the architecture of Visualizer is shown. Finally, The implementation of Markov chain model is presented.

Markov Chains

Throughout this section, we will present the fundamental properties of Markov Model, and their interpretation within the Visualizer. Moreover, we introduce Markov chain models based on different orders.

Markov model is part of a stochastic model family which can model sequential data or capture the sequential patterns. In other words, it presents a way to model the dependencies [8, 12, 11, 10, 13, 7] of current information with the previous data. Markov Chain is a method used to capture the short-term dynamic data. Using Markov Chains (first order Markov chains), we can learn statistics of sequential data, make predictions, or recognize patterns. The Markov property is described mathematically as follows:

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = P(X_{n+1} = x | X_n = x_n)[8,\ 10] \tag{4.1}$$

which denotes that the current state only depends on the previous state.

Properties Markov chains have three fundamental properties: the state space, the transition matrix, and the initial vector. First, the state space 4.2 has to be defined appropriately. Second, the transition matrix needs to be calculated to determine the probabilities of moving from one state to another. Finally, the initial vector has te be defined which represents the start state.

In our particular case, the set of all states represents the set of interactions defined within the Visualizer. For instance, in Visualizer the interactions, such as *selecting a specific dimension, brushing, filtering* are interpreted as states, and all together define the state space. We denote the state space with the

capitalized letter S, and each element (state) is represented as small s followed by a state index. e.g.

$$S = \{s_1, s_2, s_3, .., s_n\} \tag{4.2}$$

whereby n denotes the number of different states within the state space.

The transition Matrix or the transition function describes the probability that a user whose current state is s_i will select the next state s'. The transition Matrix is denoted as P. An example for the transition matrix is provided below. In a nutshell, this table shows three states and their transition probabilities distribution. For example, the first row presents the state *Brush* together with its transition probability distribution. Based on this distribution, if a user brushes over a chart he/she is more likely to zoom as next.

$$\mathbf{P} = \begin{array}{c|ccc} & \textit{Brush} & \textit{Filter} & \textit{Zoom} \\ \hline \textit{Brush} & 0.2 & 0.1 & 0.7 \\ \textit{Filter} & 0.3 & 0.5 & 0.2 \\ \textit{Zoom} & 0.6 & 0.4 & 0 \end{array}$$

Additionally, each element in the transition matrix is indexed as P_{ij} where i represents the row and j the column.

$$P_{ij} >= 0 \qquad \sum_i P_{ij} = 1, \qquad j = 0, 1, ..., n \tag{4.3}$$

First Order Markov Chain The first order Markov chain or a so-called one-step Matrix defines the transition distribution based only on the current state. An example considering p steps is shown in Diagram 4.1.

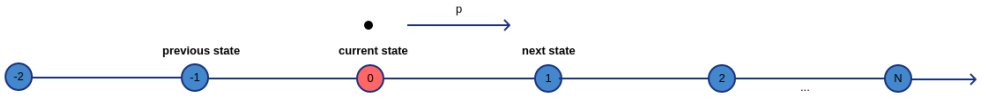


Figure 4.1.: First order Markov chain.

Usually, this model performs with lower accuracy, but on the other hand, it requires smaller amount of memory for the model compared to higher order Markov chains.

Second Order Markov Chain The second order Markov chain or a so-called two-step Matrix goes one step back in the history and defines the transition probability based on current and previous states. Essentially, it is an extension of first-order Markov chain and only changes the dependency property whereby in this model the next state depends on the two previously completed states. Diagram 4.2 shows how the model works considering p steps.

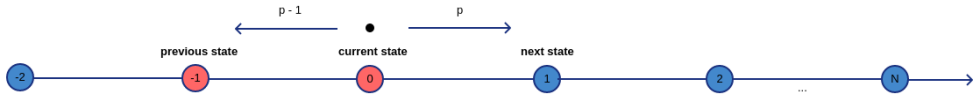


Figure 4.2.: Second order Markov chain.

Third Order Markov Chain The third Markov order model is also an extension of the second-order Markov chain which goes a step further on looking back in the history of completed steps. In this model, the next step depends on the three previously performed steps. In the Diagram 4.3 a general overview about the third order Markov chain is shown.

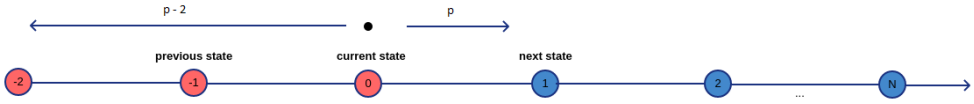


Figure 4.3.: Third order Markov chain.

Taking a higher order of Markov chain could improve the model accuracy (if a huge amount of data is available), but we have to mention that after some point increasing the order of Markov chain will not have a positive effect on certainty. A disadvantage of using a higher order of Markov chain is that the number of states will increase in the quadratic way which simultaneously will

affect the memory performance. Therefore, selecting the appropriate Markov order (considering the computation and memory performance) in one side, and the model with high accuracy in the other hand, is challenging. Usually, the third order Markov chain is preferable [13]. Regarding the advantages and disadvantages of the various order Markov chains, we consider all the three models, which are previously explained in detail.

4.3.3. Learning Implementation

This section outlines the methodology used to learn the Markov chain models. Additionally, in this section, we will show how the essential properties of Markov chain are implemented.

In the previous sections, we showed the whole process of collecting data, preprocessing them, and presented the Markov model and its properties. In order to prepare the data collected (JSON format) during the Visualizer evaluation for the learning process, we converted this data in CSV and stored it in files. The data contains sequences of states (interactions), where each sequence defines the path used to accomplish the goal. These data are used to train the Markov chain models. In contrast to many other learning models that require high processing performance and time to converge, Markov chains are fast learners, or one-step learners. In the following, we describe how the model learns and also how the initial vector is determined in our use case.

The transition matrix is calculated using the Maximum likelihood estimation (MLE)[24, 10, 13]. For each state $s_i \in S$, let n_i be the number of times that state s_i is observed. And let n_{ij} be the number of observed consecutive transitions from state s_i to state s_j in $s_1, s_2, ..., s_n$. The analytic way of MLE used to calculate the probability of transitions from s_i to s_j is given by:

$$\hat{p}(j \in S_n | i \in S_{n-1}) = \frac{\hat{p}(j \in S_n \wedge i \in S_{n-1})}{\hat{p}(i \in S_{n-i})} = \frac{|\{(S_n, S_{n-1}) : j \in S_n \wedge i \in S_{n-i}\}|}{|\{(S_n, S_{n-1}) : i \in S_{n-1}\}|} [10] \quad (4.4)$$

Markov chain implementation[3] is extended by the author using the programming language Python . Additionally, for data handling and statistical

[3] *https : //github.com/superbly/markov_chain*

computations, we used an external library: Numpy is used to structure data in an appropriate way and allows us to compute different kind of statistics.

In the following, we detail how the Markov chain model is implemented, and how our data saved in CSV File is loaded, prepared and fit into the Model. The Markov chain is defined as a python object. In the Table 4.10, the most important parameters and functions of Markov chain object (implementation) are presented.

MC Parameters	
Parameters	**Description**
n_order	Specifies the Markov chain order
p	The transition matrix
recommender_score	The scoring parameter used to measure the quality of the model (see Section 6.3.3)
states_labels	The list of all states used to encode and decode the unprocessed data (non-numerical values)
MC Functions	
Fit()	This function is responsible to fit the model using the sequential data
Predict()	This function is responsible for predicting the next state given current user behavior
PredictPath()	This function is responsible for predicting the whole path using a forward search algorithm (see Section 4.4)

Table 4.10.: Markov chain object parameters

So far, we introduced the most important features of the Markov chain object. Now, we will see the process of loading and fitting the data into the model 4.1. First, to load the data, we have implemented function called $readFile('fileName')$ who reads the CSV data that contains the training data and saves it in a python array where each line represents a sample. Second, the model needs to know a priori how many states our model has. This can be set by hand through initializing the value n_state manually, or use the method $getStatesSize('dataset')$ which scans all previously loaded data and counts the distinct states. Third, we create the Markov chain model calling

the constructor *MarkovChain*('*number_of_states*'), and the only parameter we have to pass is the number of states which we extracted in the previous step. Finally, we are ready to fit the data, and in upcoming step to make prediction using the functions *mc.fit*('*train_data*'), and *mc.predict*('*test_data*').

Listing 4.1: Integrating Predictive Model

```
'''
A code session performing loading, fiting and
testing of Markov chain Model
'''
dataset = utils.readFile ('dataset.txt')
n_states = utils.getStates(dataset)
mc = MarkovChain(n_states)
mc.fit(dataset)
mc.predict(test_data)
```

To fit the data within a Markov model means to calculate the transition matrix using the MLE. Furthermore, The "fit" function supports learning of Markov chain models using various order.

4.4. Recommendation of Interaction Sequences

In this section, we will introduce the methodology used to recommend next interaction based on the Markov chain model. Additionally, we will explain the method used to recommend a list of sequences.

Using our model is pretty straightforward to predict the next most probable state. The idea is, being in a state s_i to recommend the most probable states (interaction) that the user can go next. We have come up with two methods of calculating the recommendations. In the first method, we recommend a sequence of interactions. In this case, the user can choose between various recommendations, where each recommendation represents one single interaction. Further, we do not want only to recommend a list of possibilities for the next probable interaction, but we go further and try to recommend a list of sequence of interactions. In this case, the result is a two-dimensional, where each row represents a sequence of interactions. Each sequence of recommended

interactions could lead to a potential insight. In the following, we will describe these methods in details and show the implementation of these methods.

In the prediction method (shown in listing 4.2), we pass a sequence of interactions (representing current user behavior) as a parameter. The method does the following: it determines the current state regarding the selected order Markov chain algorithm and acquire the list of recommendation (if available) from the transition matrix. The method returns then the list of recommendations representing only interactions which have probability higher than zero. The recommended list represents the next possible interactions.

Listing 4.2: Prediction Method

```
"""
Prediction method
"""
def predict (self, user_behavior):

  prediction = {}
  columnsLength = self.p.shape[1]

  for state in range (0, columnsLength):

    prob = self.mean_over_all_transitions
                          _probabilities (state, sequence)
    prediction[state] = prob

  # filter the recommendation list
  prediction_threshold= dict([item for item in
                  prediction.iteritems() if item[1] !=0])

  return prediction_threshold
```

Additionally, we extended the current recommendation method, such as not to recommend only a list of next possible interactions, but to recommend a list of next possible sequence of interactions. In this method which we called *predictPath*('*user_behaviours*'), we pass as parameter the sequence of interactions which specifies the current user behavior.

This approach applies a forward search technique where in each step the recommend method shown previously is called. We named it "forward search" since in each upcoming step the recommended data are passed to the recommend method. To clarify this, please consider the following example: Suppose, we have current user behavior shown below, and we are using the first order Markov chain model.

$[a]$

First, the function performs the prediction function over current behavior and the function returns following recommendations including their probabilities: $[c : 0.6, d : 0.5, e : 0.2]$

Next, we build separate lists each containing current user's behavior and one of the recommended interactions from the recommendation list. Each element in the recommendation list is appended to the current user behavior separately building a list for every recommendation. This is illustrated in the following:

$[a, c], [a, d], [a, e]$

In the next step for each list, the prediction function is called, and the recommended element with the highest probability is added to the corresponding list. This process is iterated until no recommendation is returned. In the end, we will have a two-dimensional list (matrix) containing exactly k sub-lists where each sub-list can have m elements or less. In order to have only recommendations with high confidence, a threshold is provided, which can be adjust from the client itself on the client-side. For each sequence of interactions (sub-list) from the two-dimensional recommendation list, the confidence value is estimated by multiplying the probability of every single interaction in the corresponding sequence of recommendation, and this is compared against the threshold. The outcome (matrix) is presented to the user using the implemented visual guidance interface (see Chapter 5 for details)

4.5. Integration of Predictive Model into Visualizer

This section introduce how the interaction process of Visualizer looks like. Since the predictive model is an external tool, only small changes are neces-

sary to any visualization system in order to integrate it. The recommended model will remain on the server side and will listen to the client, waiting to receive user's current behavior state and use this information to generate recommendations which are send to the client. The server, implements the process of listening for the user behavior information and passing it to the learned model. On the client side, the user behavior has to be tracked in real time and sent to the server.

In the next subsection, we will show how the Markov chain Model is integrated into the Visualizer. We will look into the details on both server, and client side.

4.5.1. Integration on the Server side

The whole server-side process can be split in four steps. In the first step, listen for new user behavior information. In the second step, extract the important information (current sequence of steps performed on the client side) required by the Markov chain model in order to generate recommendations. In the third step, pass these information into the Markov chain model. Finally, the model returns a list of recommendation which is sent back on the client side.

4.5.2. Integration on the Client side

On the Client side, the user activities (interactions) are registered in the real time, and after every new interaction, the list of logged user interactions is sent to the server. Next, the recommendations are delivered back to the client. Further, on the client side, a recommendation-interpreter is implemented, which is responsible for handling the incoming recommendation and updating the recommendation box within the UI. In the next Chapter, we will demonstrate in detail the process of visualizing recommendation in the UI recommendation Box.

5. Visual Guidance Interface

In the previous section, we have described a method step-by-step fashion, which analyses the user behavior within a visualization dashboard and recommends next interactions or paths (sequences of interactions). For an easier communication, we will refer the resulting tool as User Tracker Analyzer (UTA). The next section details the user interface of the UTA tool. The main feature of UTA is it to be inituitive and easy to use.

This Chapter is divided into two central sections. In the first section, the user interface, used to represent the interaction recommendations, is introduced. Whereas, in the second part, a case study is performed in order to show the advantages of using the UTA tool in the context of visualizer dashboard.

5.1. User Interface

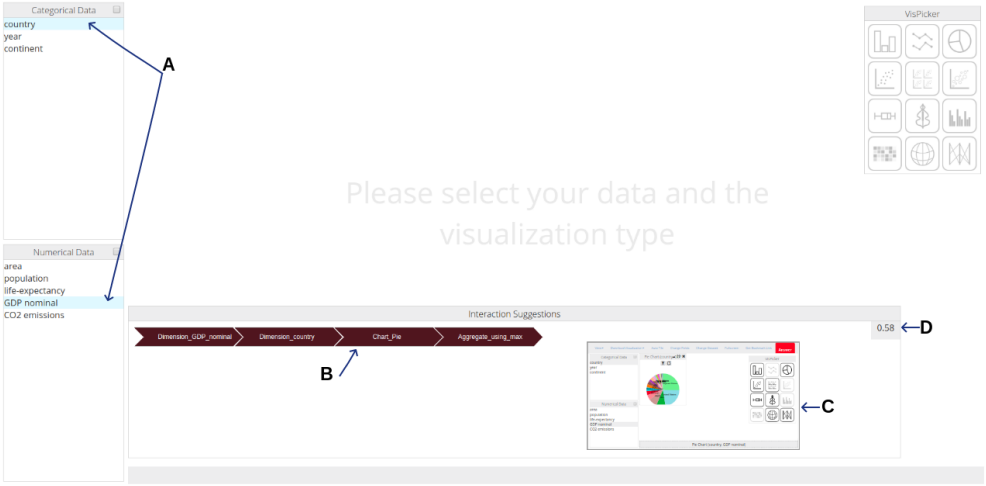


Figure 5.1.: Visualizer with the integrated User Tracker Analyzer (UTA). User guidance UI includes: (A) The recommended fields. (B) The recommended interactions. (C) Result preview of the current recommendation. (D) Probability (confidence) of the recommendation.

Lindon Leader quoted "I strive for two things in design: simplicity and clarity. Great design is born of those two things." [25]. Motivated from Leader's quote the main idea behind designing UTA's user interface was to keep it simple and clear.

Now, we introduce the information we want to present to the user with the UTA interface. Previously, we showed how the user behavior is tracked and delivered to the UTA algorithms. Basically, we now show the following information on the UI.

The information to be shown in the interaction recommendation UI is clustered into four groups:

- the recommended fields (dimensions of the data)
- the recommended interactions

- the probability of the recommendations (represents the recommendation accuracy)
- Recommendation preview (shows the end result of the recommendation path)

Figure 5.1 introduces all the components of visual user guidance interface shown in the lower part of the Visualizer UI. The interaction sequence recommendations is placed on the lower-left side of the Visualization box (Figure 5.1, section *B*). In section C, a preview resulting from the recommendation path is shown. In the preview section, the user can see the result (e.g., a Pie chart showing aggregated data) before accepting the recommendation. Further, in section D, the probability of the recommended path is shown. The probability is calculated by multiplying the probability values of each interaction with each other. For instance, if the system recommends a path consisting of two interactions (e.g., data aggregation and visualization) where the probability of each interactions is described by a real number value (representing how probably that interaction will be performed next), the probability value shown in the visualization box is calculated by multiplying these two values with each other. Moreover, this value represents the estimated quality of the recommended path based on the deployed recommended model. Finally, in section A which is outside the visualization box, the recommended fields are highlighted.

Next, we will describe in detail each component introduced above. Further, we will have a closer look over their functionality and purpose.

Interaction Recommendations The whole idea behind our tool is to recommend the next interaction or sequence of interactions based on the current user behavior. Therefore, the interaction recommendation is the central component of the visual guidance interface (UTA). UTA is modeled in such a way that it can recommend more than one interaction (if available), or a whole path (see Figure 5.2). Consequently, path recommendations are listed vertically, where each row determines a new recommended path. The lines are colored based on the probability of each path recommendation : the darker the color the higher the probability.

Figure 5.2.: Multiple recommendation of paths of interactions. On the left side, we see a list of recommended paths. The active path is highlighted and the end result is shown on the preview window.

Further, each recommended path consists of one or more interactions. The interactions are ordered from left to right which means the steps will be executed sequentially starting from the left side and moving to the right side executing one step at a time. Figure 5.1 shows an example where a path is consisting of four interactions is recommended. The sequence is shown using a typical process representation: an arrow-formed chain, where each arrow represents an interaction, and the direction of the arrows describes the order of steps, making it clear how the sequence will be executed. For example, the first interaction in the path is to select the data field GDP_nominal which is a numerical field, the second interaction recommends to select the field country, etc.

When the user is clicking in the dashboard the interaction recommendations are updated in real time. If the current user behavior changes at a time, then the recommendation paths will be updated accordingly including paths and interactions within a recommended path. Figure 5.3(*a*) shows an example where the recommended path consists of four interactions. In (b), the recommendation path is updated dynamically after the user has selected the field GDP_nominal, which was the first recommended interaction.

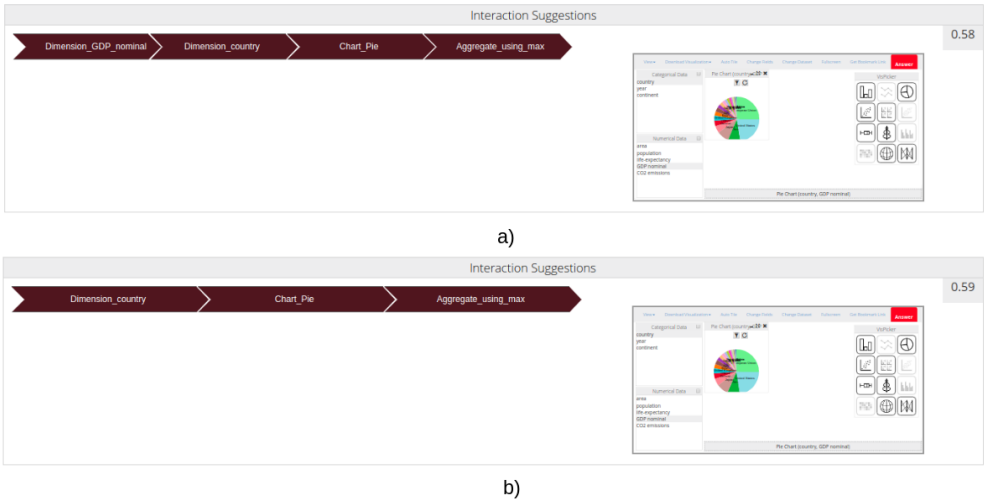


Figure 5.3.: Recommendation Interaction example. a) The recommendation path based on current user behavior. b) The recommendation path after user performed the first recommended interaction, which is selecting *"dimension_GDP_nominal"*

Field Recommendations Second component of a UTA is the field recommendation. A field represent an attribute or label within a dataset. When within a recommended path data field selection is recommended, this information is additionally represented in the field box (lists numerical and categorical fields). In this case, the recommended fields will be highlighted within the correspondin group (categorical section) as shown in Figure 5.1(*a*).

Recommendation Probability Next component is the recommendation probability as shown on the top-right side of visualization recommendation box (see Figure 5.1(*d*)). The recommended path has its probability calculated as product of the probability of each interaction within a path. If multiple paths are recommended then the probability value shown in the small box on the top-right side represents the probability of the currently selected recommended path. Consequently, if only one path is recommended then the path is selected by default and its probability value is shown (see Figure 5.2).

Recommendation Preview Finally, we present the last component which is the recommendation preview (see Figure 5.1(*c*)). The recommendation preview shows the final outcome of the recommended path. In other words, this component offers the opportunity to see in advance the result of the recommendation, before accepting it. This component is placed on the right side of the interaction recommendation component, and shows the result will look like when the user accepts all recommended interactions in a path. The advantage of the preview is that the user can decide in advance if he/she wants to accept, or ignore the recommendation. Another function of recommendation preview is, that instead of performing multiple interactions manually in order te explore the data, the user can do it by a single click on the preview. Moreover, the user can explore other recommendation (if multiple recommendation at disposal) by clicking on them over the recommended list, and the result will be presented automatically in the preview window.

5.1.1. Accept recommendations

When a recommendation is defined, there are two opportunities the current user have: either the user decides to execute each step manually where he/she can observe how the systems reacts to each action, or he/she can accept the recommendation using the preview component end obtain the final result directly. There are some interactions (e.g., selecting the aggregation method) which still can not be performed automatically without the help of the user. In this case, the user has to step up and decide how he/she wants to perform the step in that case. For instance, a sequence recommends to select some fields and generate a Bar Chart. Accepting this recommendation, will force the system automatically to select the recommended fields, and start the process of generating the Bar Chart, but instead of creating the Bar Chart immediately, the system show the Aggregation dialog where the user has to select one of the aggregation methods in order to generate the Bar chart. In the future version we might try an recommend a particular aggregation too. Now suppose that there is a recommendation to brush the Bar chart, which is another activity where the user has to step up and do the job. If the user decides to accept the recommendation, then he/she has to perform this step manually. Nevertheless, we believe that a recommendation to apply a brush in a chart may already represent a valuable hint.

5.2. Case Study

In this section, we will show a step-by-step execution of a analytical task within the Visualizer. On the one hand, we will illustrate the task without using the UTA. On the other hand, we will perform the same task when UTA is activated. In this case, we will follow UTA recommendations and compare the results. Further, we will discuss the possible advantages and disadvantages when using the UTA tool.

We used an dataset which shows the CO2 emissions/per ton of the G20 countries[1] between the years 2005-2015. Furthermore, this dataset has 221 entries and provides the information about the area, life expectancy, population, GDP nominal, continent name, and country name of each country (see Figure **??**).

Task: How many countries have GDP greater than 4 000 000 US$MM?

Figure 5.4.: Dashboard

[1] http://dfat.gov.au/trade/organisations/g20/Pages/g20.aspx

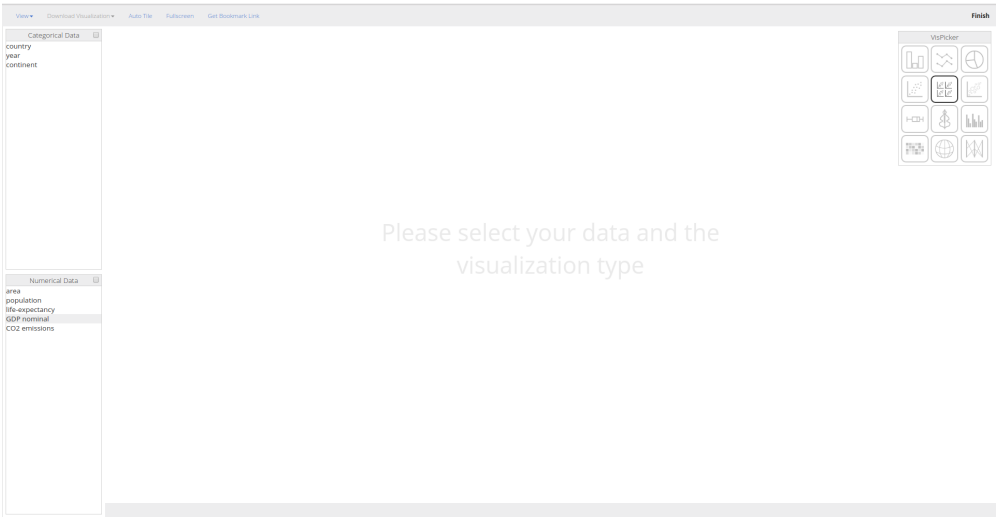


Figure 5.5.: Step 1 performed without using the User Tracker Analyzer.

Figure 5.6.: Step 2 performed without using the User Tracker Analyzer.

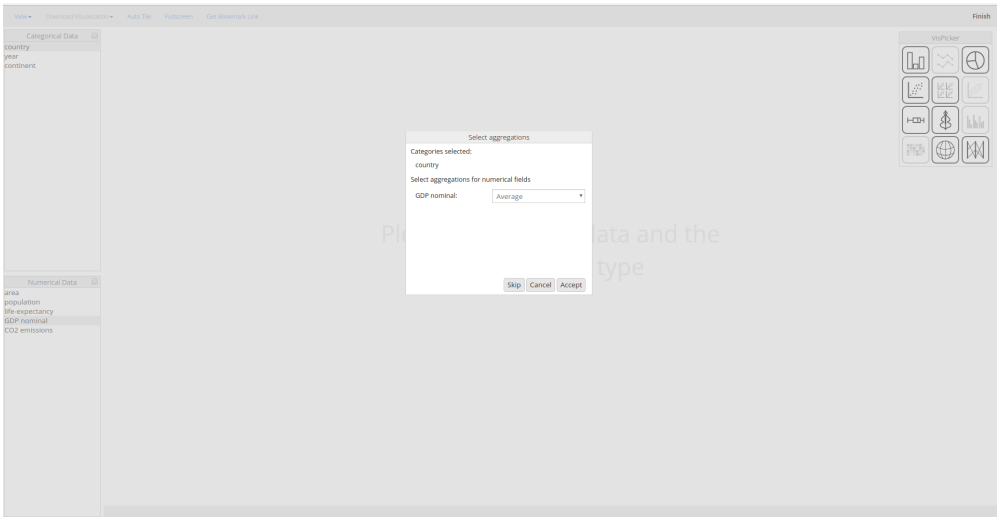


Figure 5.7.: Step 3 performed without using the User Tracker Analyzer.

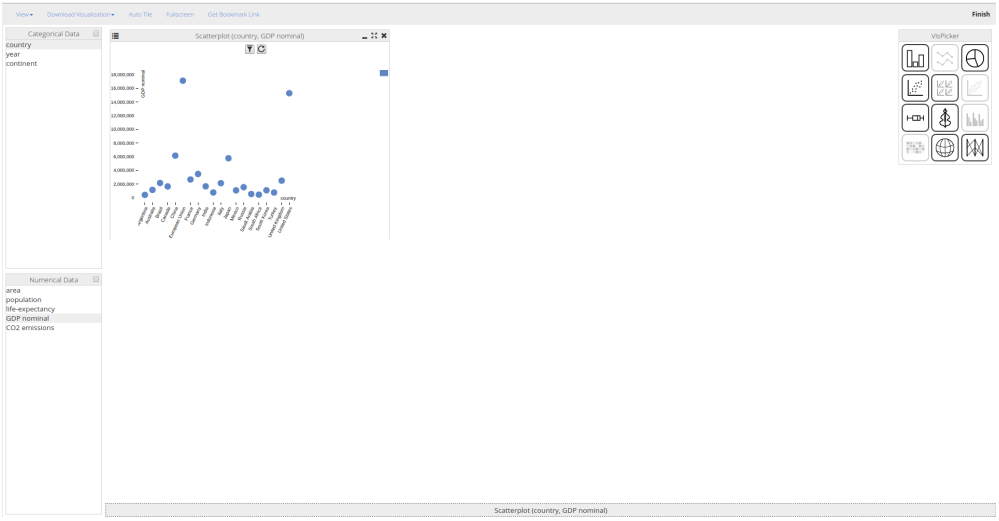


Figure 5.8.: Step 4 performed without using the User Tracker Analyzer.

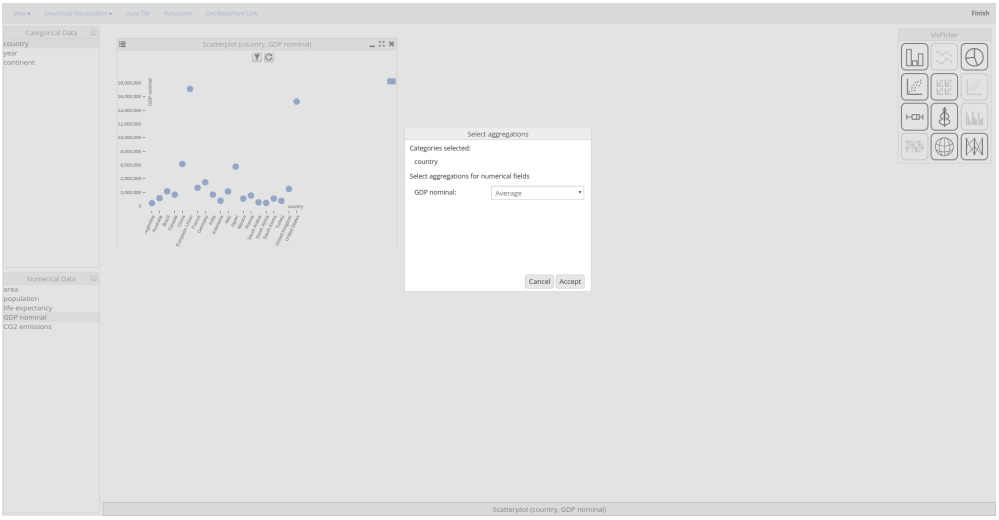


Figure 5.9.: Step 5 performed without using the User Tracker Analyzer.

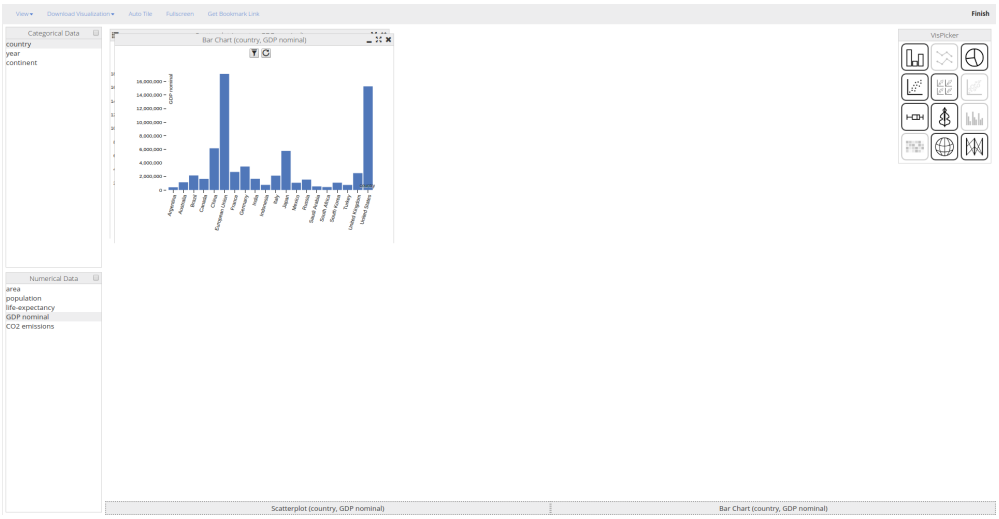


Figure 5.10.: Step 6 performed without using the User Tracker Analyzer.

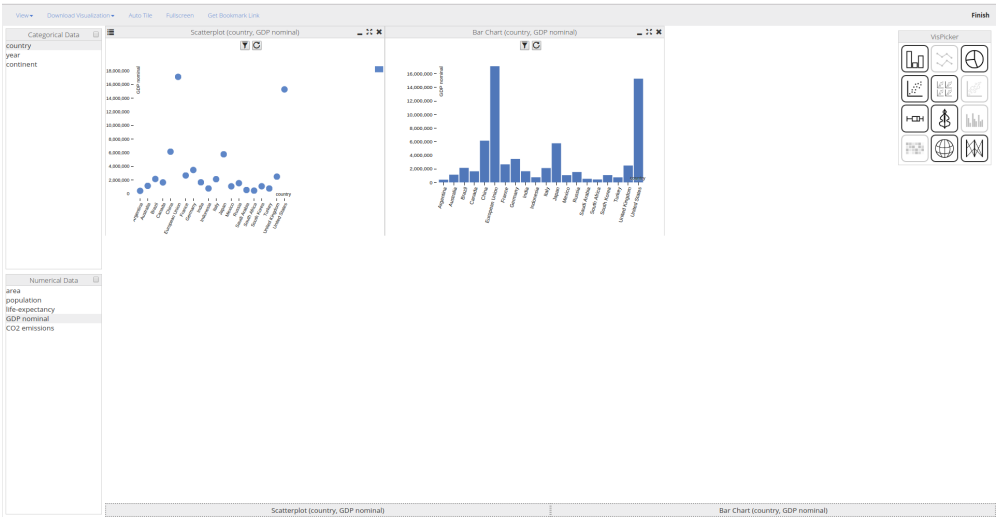


Figure 5.11.: Step 7 performed without using the User Tracker Analyzer.

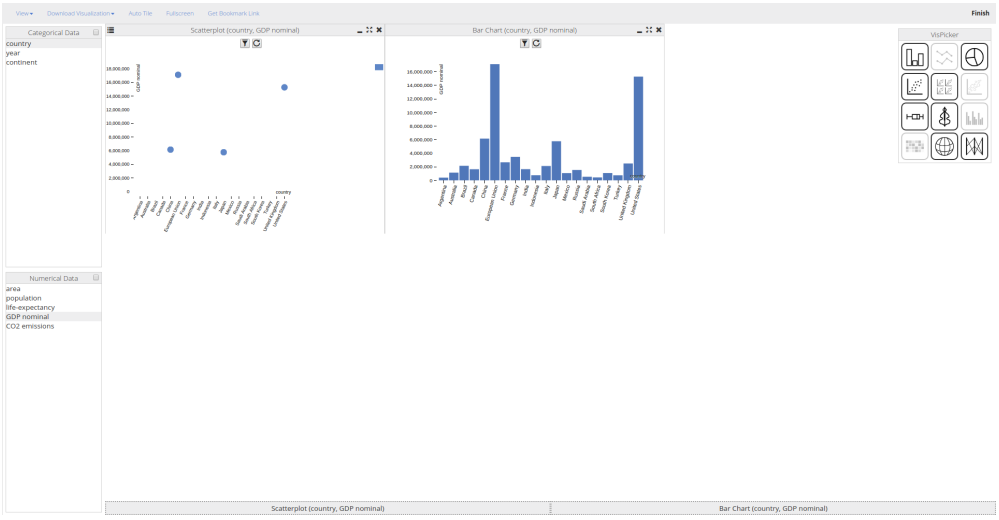


Figure 5.12.: Step 8 performed without using the User Tracker Analyzer.

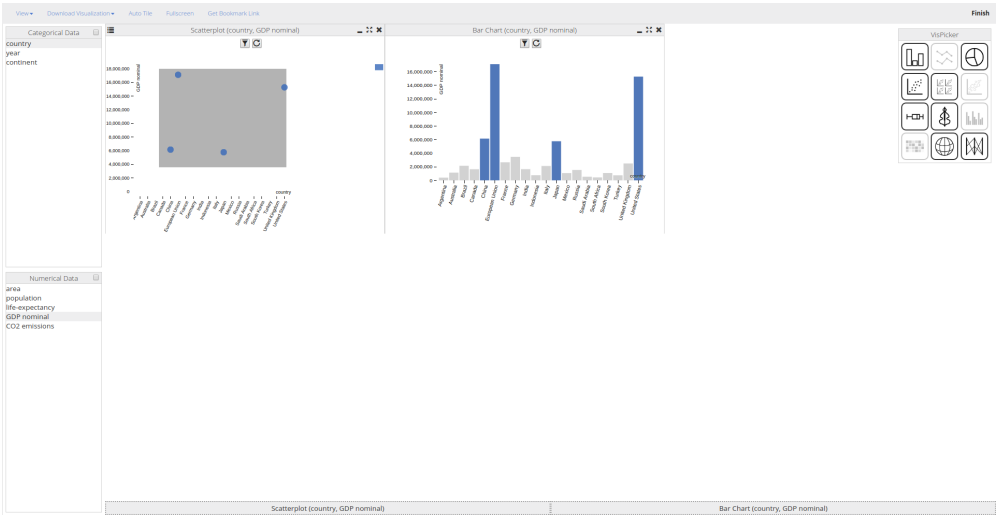


Figure 5.13.: Step 9 performed without using the User Tracker Analyzer.

Task solution without using the User tracker Analyzer (UTA) Each step performed to accomplish the goal is shown in Figures 5.4 to 5.13. In the first step, the user selects the "GDP nominal" field. Second, the user selects the "country" field. In the third and fourth step, a scatter plot is generated and the data are aggregated by average value (numerical values by time series data have to be aggregated in order to be visualized). In the fifth and sixth step, the user creates a Bar Chart and aggregates the data by average value. In the seventh step, the charts are arranged within the visualization box using the AutoTile feature, which fits the sizes and positions of visualizations in such way that none of them overlap each other. In step eight, the user filters the data in the Scatter-plot using the filter button placed inside the visualization. The data are filtered applying the value 4 000 000 US$MM over field "GDP nominal", and step nine shows the selected data highlighted in the bar chart. Therefore, to successfully reach the goal, the user needed to perform nine steps.

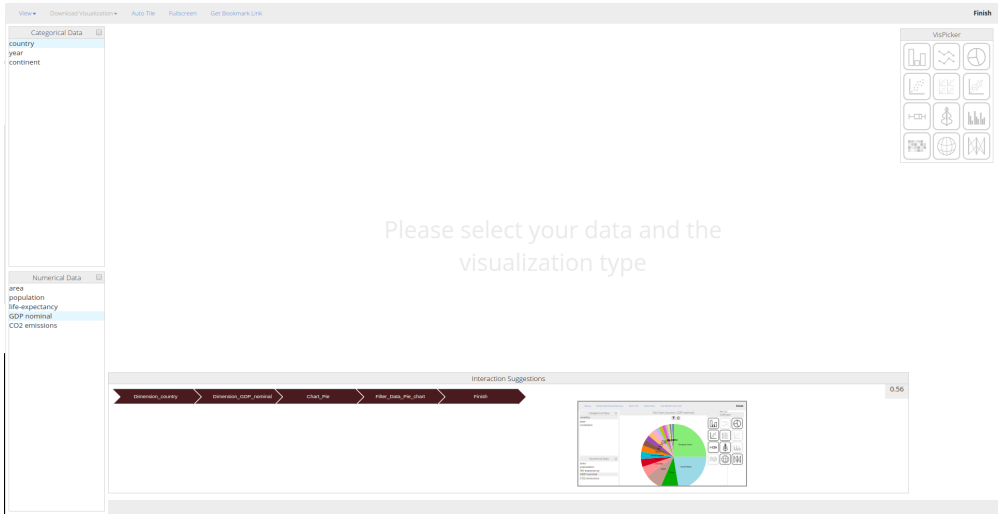


Figure 5.14.: Dashboard including the User Tracker Analyzer

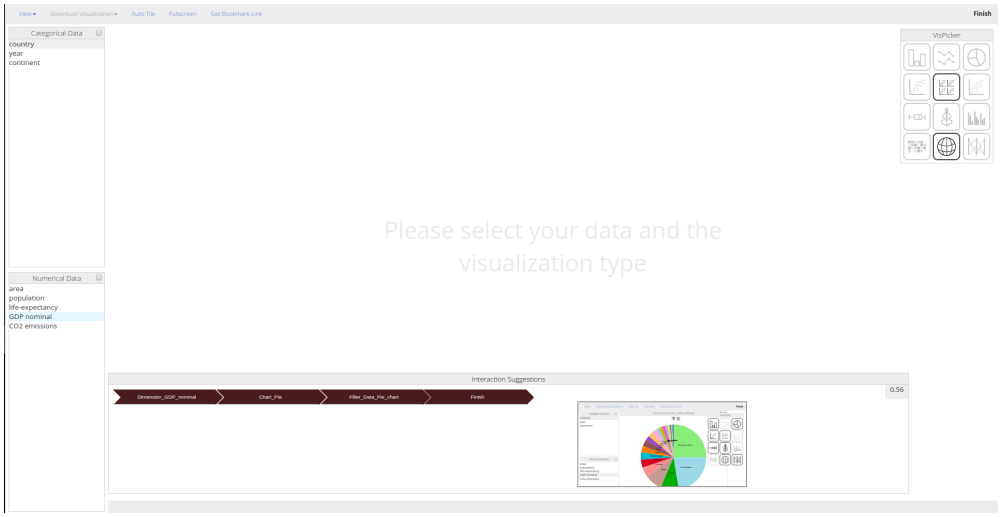


Figure 5.15.: Step 1 performed using the User Tracker Analyzer.

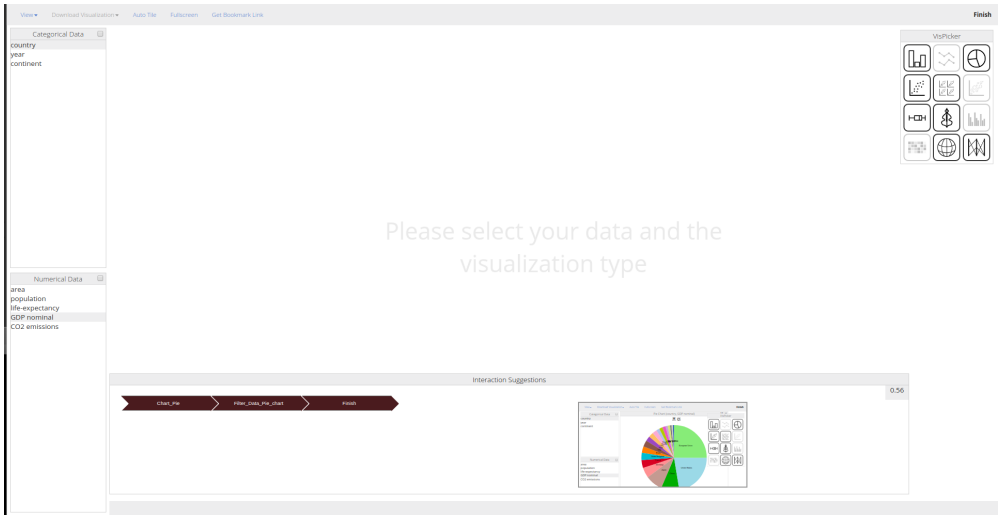


Figure 5.16.: Step 2 performed using the User Tracker Analyzer.

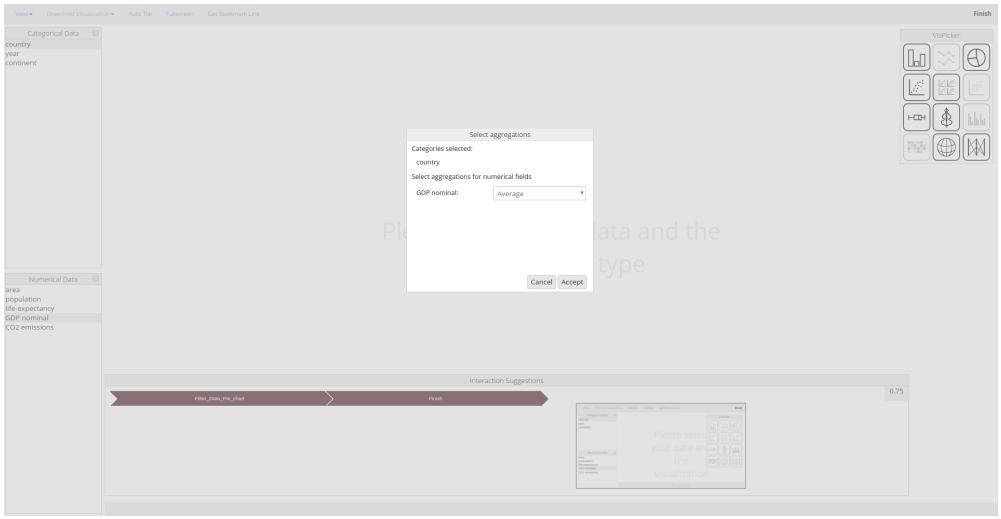


Figure 5.17.: Step 3 performed using the User Tracker Analyzer.

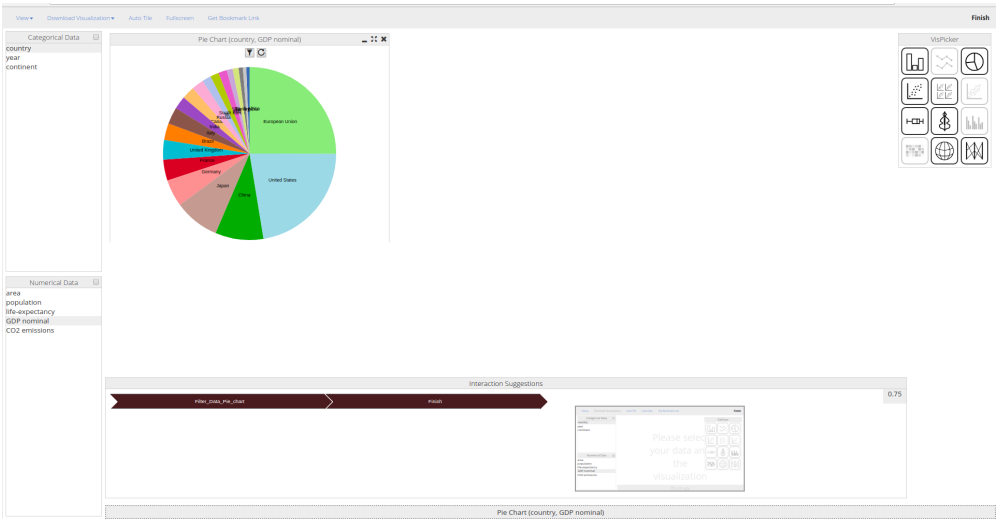


Figure 5.18.: Step 4 performed using the User Tracker Analyzer.

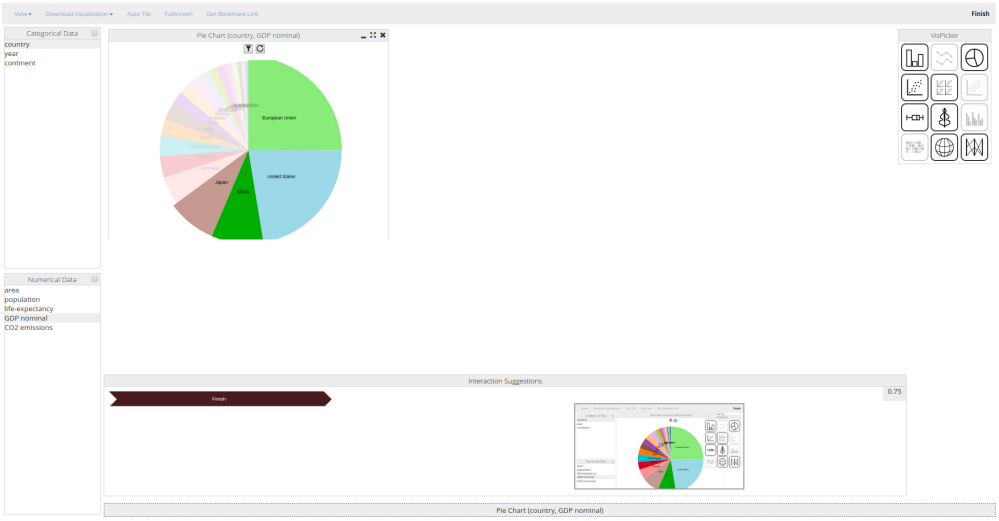


Figure 5.19.: Step 5 performed using the User Tracker Analyzer.

Task solution using the User Tracker Analyzer (UTA) In this scenario UTA is activated and tries to infer what the task of the user is. At the beginning of the scenario, user clicks through the fields (GDP) and tries to select the right chart in the VisPicker. Based on the behavior patterns it learnt previously, the system recognizes that the user is trying to get overview about the data and recommends a path the user can take in order to get the desired overview. Figures 5.14 to 5.19 show the recommendations that should help the user to perform this task successfully. The recommended solution consist of five steps. The user agrees with the recommendations, and performs every step manually to follow the unfolding of the analytical process, which requires five steps, as visible in the Recommendation (see Figure 5.14). First, the user selects the field "country". Second, selects the field "GDP nominal". In the third and fourth step, the Pie Chart is created, and the data are aggregated by average value. Finally, the data are filtered over the field "GDP nominal". Therefore, accomplishing the task with UTA activated, reduces the number of steps required to perform from nine to five. We conclude that UTA can help the user to find new efficient ways in order to solve the task easier and faster.

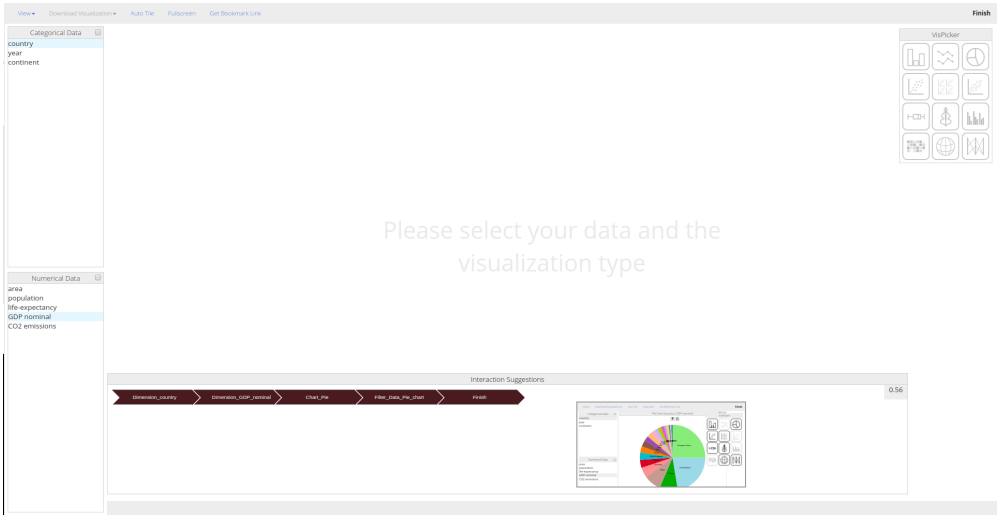


Figure 5.20.: Dashboard including the User Tracker Analyzer: The system UI initial state.

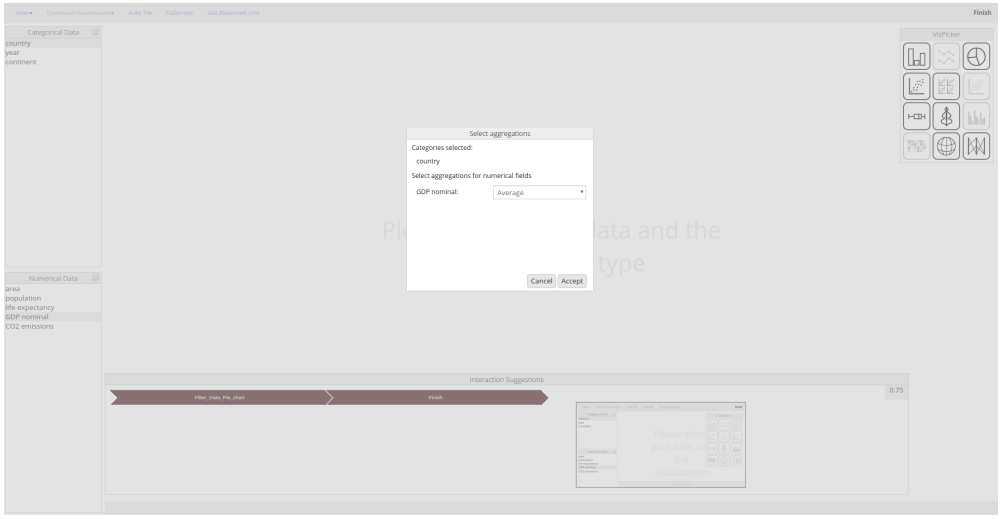


Figure 5.21.: Step 1 performed using the User Tracker Analyzer. In this step the preview accept step is performed.

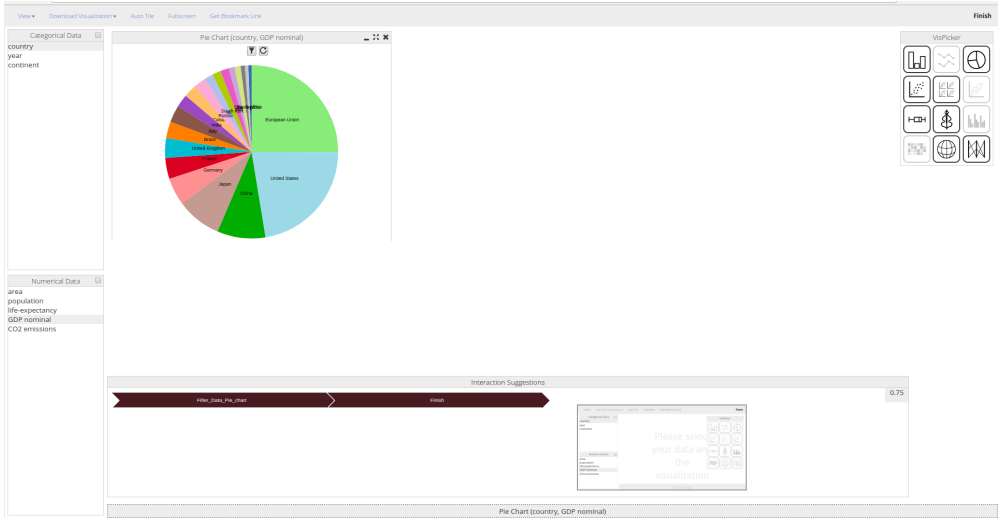


Figure 5.22.: Step 2 performed using the User Tracker Analyzer.

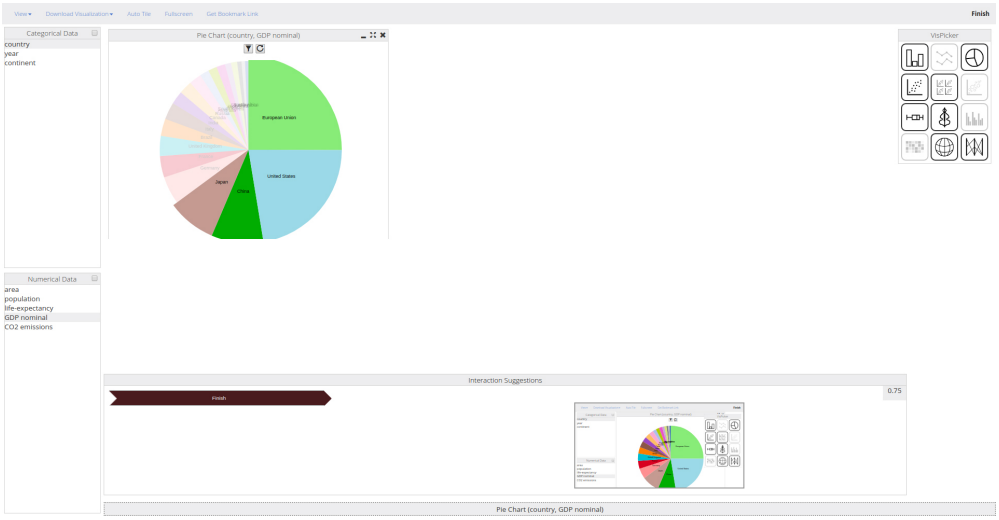


Figure 5.23.: Step 3 performed using the User Tracker Analyzer.

Further, the preview functionality can further help to reduce the number of steps required to finish the task. The user can click on the preview and execute the first three steps using only one mouse click as shown in Figures 5.20 to 5.23.

On the one hand, using the preview window helps the user to a priori see the end result before accepting the recommendation. On the other hand, it can reduce the exploring time significantly by reducing number of interactions with the system in order to achieve the goal. First, user clicks on the preview to accept the recommended path, then, in step 1, user is offered to aggregate the data in order to generate the appropriate chart. In step 2, we see the result which shows the generated Pie chart. In the last step, the user is recommended to filter the data which is shown in step 3 (see Figure 5.23)

5.2.1. Discussion

On the one hand, to accomplish the goal with UTA deactivated there are nine steps required to perform to fulfill the goal. On the other hand, performing the task with UTA enabled, may offers rotundly faster way to reach the goal.

The number of steps needed to accomplish the task using the UTA can be strongly reduced. Provided the system can detect user's task (objective) and provide suitable recommendations. Moreover, using the preview component from Visualizer recommendation box, the user can inspect the end result, and if it is suitable, automatically execute multiple steps. In the presented example, the goal was reached by performing only three steps instead of nine steps performed without using UTA.

Some studies have revealed that the user behavior are "sticky" [6]. This means that if a user accomplishes a goal in one specific way, which not necessarily is the best one, most probably he/she will stick with that way and every time he/she wants to execute a similar task, the user will follow the same path. Showing a better solution e.g., shorter route, or even better visualization using the UTA provided the learning model is trained using enough suitable data collected from other users, our user might be informed that there are better ways to accomplish the task. This hypothesis will remain open for future work where an evaluation for interaction recommendation is required in order to prove or deny it.

5.3. Summary

In this Chapter, we introduced the UTA. Moreover, a case study is presented where a step-by-step use case is performed to demonstrate how the system is applied to perform a analytical task(s). An use case is performed with UTA enabled, and is compared against the system where UTA was not integrated. In this case, the advantages and the usefulness of my work is highlighted at every step of the work-flow.

6. Evaluation

The evaluation is split into two parts: First, a user study is performed in order to (i) collect data for our tool, (ii) to evaluate the usability of the tool, and (iii) to evaluate the subjective workload. Second, the evaluation of the Markov chain model. In this case, the data collected during the user study are used to train and validate the model.

6.1. Procedure of the online evaluation

This evaluation is about a visual dashboard (Visualizer) which can automatically recommend appropriate visualizations for a given dataset. The dashboard integrates 12 visualizations with different functions. In this study, participants had to accomplish four tasks with Visualizer. Each task focuses on different scope within the same dataset and contained three subtask. After finishing a task, the participants had to answer some questions about the workload and difficulty level of the task. After finishing the tasks, they were asked to fulfill a questionnaire to provide feedback about the system usability. Further, every interaction within the Visualizer was logged and saved on the server side. The data within a subtask was logged and saved in a distinct file. This data are then used to learn the predictive model and classify the tasks. The whole evaluation process was designed in a way, that it could be accomplished without help from the supervisor.

6.1.1. Evaluation Preparation

In order to complete the evaluation process, several tasks have been prepared and implemented a prior, which are listed in the following:

- **Evaluation and system introduction:** an introduction in written form is prepared in order not to affect the evaluation results providing the same information for every participant.
- **Create the guideline and the questionnaires:** a google form was created that guides the participant through the evaluation process. This form contained all the information (questionnaires, tasks, training video, etc.).
- **Training video about the Visualizer:** preparing a video explaining all the interactions within the Visualizer.
- **Navigation path:** preparing the link that directs to Visualizer and identifies the current task automatically.

6.1.2. Tasks

Each participant had to accomplish four tasks, whereby each task focused on different scopes. Meanwhile, each task contained three subtasks.

Task 1 Task one was designed in such a way, that the user was supposed to gain an overview of the dataset: The participants had to identify the countries represented within the dataset, or recognize any pattern within the data, etc.

Task 2 In this task, the goal was to detect any outliers within the data. In contrast to the task one, in this task, the questions were designed in such a way, that the user was exploring only particular fields. In other words, the participants were asked to deliver a detailed answer instead of a general one.

Task 3 The goal of this task was to analyze the trends or correlation within the dataset. The questions were formulated in such a way, that the user will explore specified fields in detail.

Task 4 During this task, users were asked to analyze the data distribution. The questions were formulated purposefully in a more generalized way. In this case, the participants were interpreting the tasks differently based on their previous knowledge. The real purpose of the task was to explore the data distribution.

6.1.3. Questionnaires

After completing all four tasks, the participants were asked to give feedback about the Visualizer and to answer some questions about the usability of the dashboard and the expertises the participants made during performing the tasks.

The evaluation too comprised pre-questionaries that aimed to collect statistics about participants gender, age, skills, expertise, and education. We also wanted to know if the people already know Visualizer. This data are planned to be used for user modeling for an extended version of our tool in the future. The post-questionary served in the first place to analyze the usability of the Visualizer, and the expertise user made using it for this study.

6.1.4. Data sets

We prepared an example dataset which showed the CO2 emissions/per ton of the G20 countries[1] between the years 2005-2015. Furthermore, this dataset has 221 entries and provides the information about the area, life expectancy, population, GDP nominal, continent name, and country name of each country.

6.1.5. Participants

The evaluation process was accomplished by twenty-two participants. At the beginning, we performed a pilot-test where two of the participants attended

[1] http://dfat.gov.au/trade/organisations/g20/Pages/g20.aspx

the pilot test. The remaining 20 users participated in the main evaluation process.

6.2. Procedure

We used our own device, and the evaluation was performed using Google Chrome [2] as a browser. The procedure was divided into four phases: introduction, training, performing the tasks and feedback.

6.2.1. Introduction

At the beginning, the supervisor provided a short overview of the evaluation procedures and the whole workflow. Further, the participant was asked to read the participant content and sign it, if agreed to participate in the evaluation process. Finally, the participant had to answer some statistical questions regarding the country of origin, education, gender, age, user interests, etc. After answering the statistical questions, the participant moved to the training phase.

6.2.2. Training

The training phase was split into three steps: First, the participant was asked to read the instructions provided in written form. In the second step, the participant was asked to watch a video which shows the Visualizer including all the supported interactions. Finally, the user was asked to play with the system in order to get used to it. The participant had the opportunity to play with the tool for 5 minutes. Most of them felt confident with the system and moved on to the next step without accomplishing the full five minutes during this phase.

[2] https://www.google.com/chrome/

6.2.3. Tasks Performing

As we stated already, the participants had to accomplish four tasks where each of them focuses on a different scope. Moreover, each task contained three subtasks. After finishing a task, participants had to answer some questions about the workload and difficulty level of the task (NASA TLX[3]).

6.2.4. User Feedback

In the last phase, the participants were asked to provide a feedback about their experience and impressions about the Visualizer. The feedback was gathered using the questionnaire prepared beforehand.

[3]ttps://humansystems.arc.nasa.gov/groups/tlx/

6.3. Results

6.3.1. Participants description

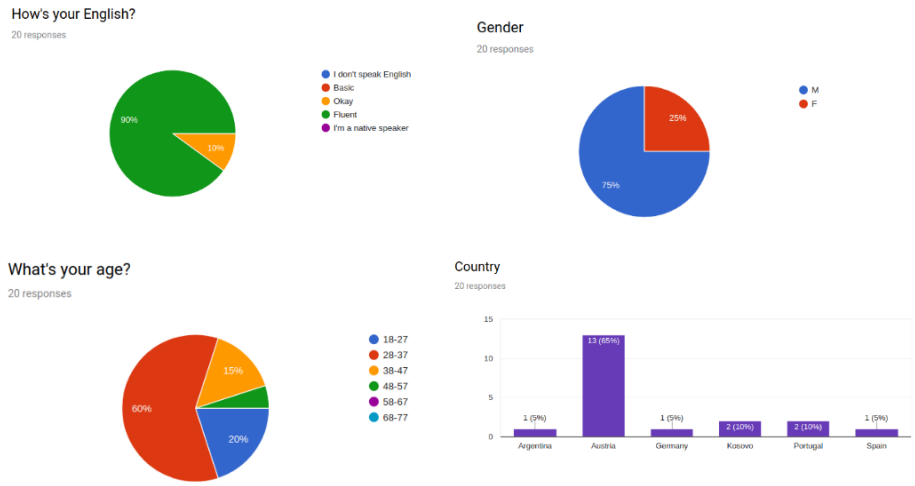


Figure 6.1.: Background information about the participant during the evaluation process. Participants from different age groups, genders, and countries attended the evaluation.

In the following, we will provide some information about the participant's background. During the evaluation process, participants from different age groups, country, or gender attended the evaluation (see Figure 6.1). Further the figure 6.2 shows that 80% of the participants had not seen the Visualizer before. Additionally, the participants had experiences from different levels with the visualization systems and analysis tools. Most of the participants were already familiar with visualization tools like Excel, PowerBI, Matplotlib, etc. Furthermore, most of the users had experiences with data analysis tools. For example, some of the tools that were already known for the participants are: R, Python, Matlab, etc.

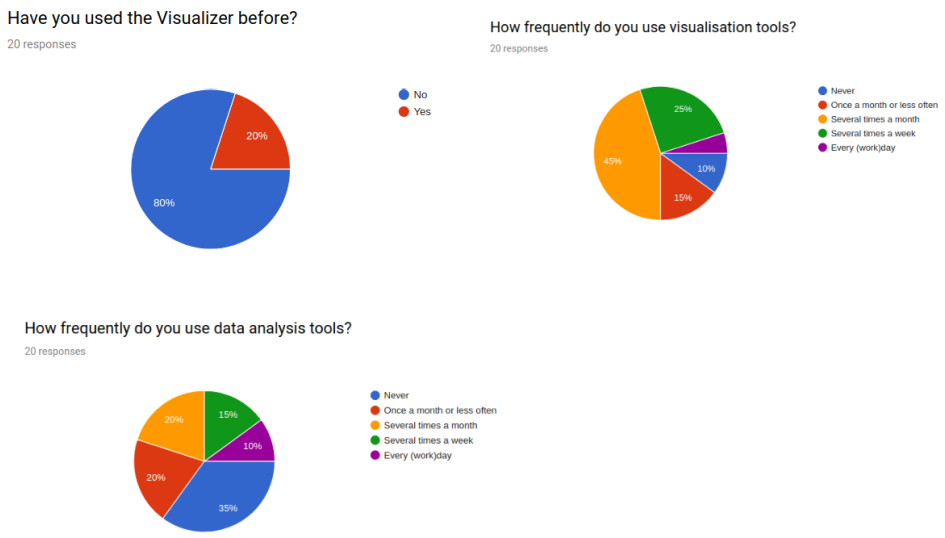


Figure 6.2.: Participants' knowledge about visualization systems and data analysis tools. Most of the participants did not know about the Visualizer before participating on the evaluation. The participants had different interests, and background with regard to the preferred visualization- and data analysis tools.

Figure 6.3 introduces the participant's education level and their interests in different fields. The participants had a different education level where 60% of them had a master degree, 20% had a bachelor degree, 10% had a doctorate or higher degree, etc. Furthemore, their preferences vary highly. For instance, 2 participants have no knowledge regarding the visualization, and only three of them are feeling as experts in this field, etc.

Figure 6.3.: Participants education level. Participants with different background and level of knowledge about data science tools attended the evaluation.

6.3.2. Usability Evaluation

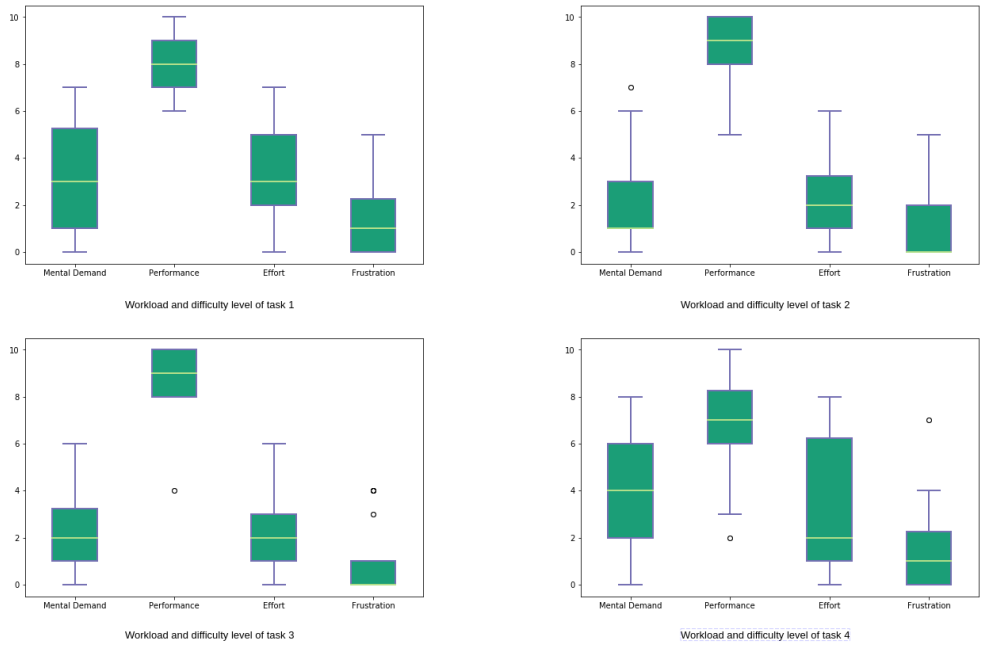


Figure 6.4.: Workload and difficulty level of all four tasks. The level of difficulty and frustration was higher in the Task 1 and 4. Moreover, the users were mainly satisfied with their performance.

As already announced, the participants had to accomplish four tasks. After finishing each task, they had to answer some questions about the workload and difficulty level of each task (see Figure 6.4). The participants found out that the task one and four were more mental demanding compared to task two and three. Moreover, when performing task one and four participants were more frustrated compared to the tasks two and three. This could have several reasons. First and foremost, tasks one and four were formulated more generally, meaning that the questions were not clearly specified. This was the reason that various participants interpreted the tasks differently.

In the last phase of the evaluation process, the participants provided us with feedback about the usability of Visualizer. In the following, we will present

the results gathered during the evaluation.

Figure 6.5 shows that the participants were satisfied with the Visualizer. They found the tool easy to use, simple and easy to interact with. Further, they felt pretty confident that they can learn to use it without the help of an expert. Additionally, they found the various functions in Visualizer were well integrated, and the tool was pretty consistent.

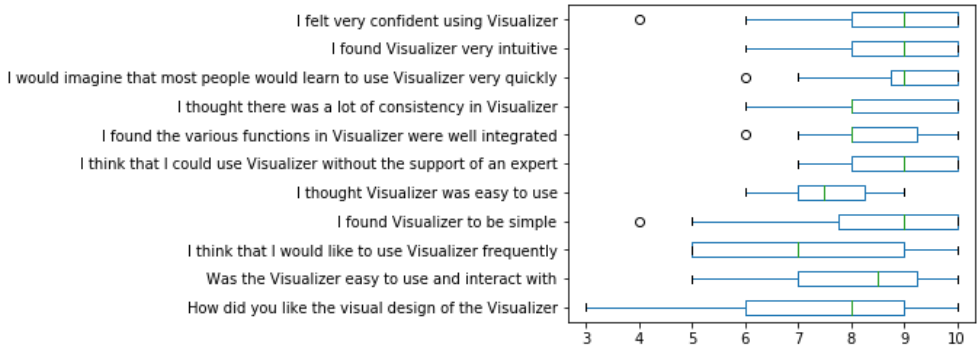


Figure 6.5.: User feedback about the Visualizer.

Nevertheless, the participants also made some suggestions about how we can increase the usability of the dashboard, they found out that more colors to the design could help to make the interface more user-friendly. Further, zooming feature over the y-axis on the the line chart could help to read the data in line chart easier, especially when too many lines are focused on the same space. Finally, a support of grouped boxplots could be also an improvement of the tool. Currently, the Visualizer supports only a single Boxplot. Multiple boxplots within a graph representing different features can be considered.

Based on the feedback from participants, we can conclude that most of the users found the Visualizer pretty simple to use and work with it.

6.3.3. Markov chain Model Evaluation

Validation Methodology

This part emphasizes the methodology used to validate the Markov chain Model. First, the metrics used to validate the model accuracy is introduced. Finally, we introduce a method to test if the model can be generalized.

A proper way of measuring the quality of the model is quantifying it. To quantify the quality of a model there are plenty of methods out there, and all of them use a different way of measuring it. We use two metric scores which we changed slightly to fit into our sequential domain and will best quantify the quality of our Markov chain model. The scoring methods we chose are: Recommendation Score [13], and Mean Prediction Rank (MPR)[7].

Recommendation Score This scoring method (see Listing 6.1) is defined as the percentage of the cases where the recommendation is successful. A recommendation is classified as successful, if the observed event is part of the top m recommendations. The score is normalized, and a score of one means that the model predicted with hundred percent accuracy, and a score of zero means that the model was not successful.

$$RS = \frac{1}{|o|} \sum_{i=1}^{|o|} in(o_i, pred, m) \tag{6.1}$$

The function *in* (o_i, *pred*, *m*) (shown in equation 6.1) gives one, if the observed event occurs inside the top m recommendations, zero otherwise. The o parameter represents the current observed sequence that is being tested. Whereby o_i represents the current observed subsequence. Whereby *pred* contains the list of recommendations, and the parameter m defines the successful range. For instance, if the next step is present in the first two top recommendations within the list of recommendations, than the recommendation is classified as successful.

Listing 6.1: Recommendation Score

```
"""
Recommendation Score
"""
def recommenderScore (self, observedSeq):

  self.recommender_score = 0;

  for index in range (self.n_order, len(observedSeq)):

    prediction = self.predict (observedSeq[:index])
    if observedSeq[index] in prediction.keys()[:self.m]:

      self.recommender_score += 1

  return float (self.recommender_score)/len (observedSeq)
```

Mean Prediction Rank The second scoring method used to measure the accuracy of Markov chain model is the MPR (see Listing 6.2) and is defined as the mean of recommendation based on the ranking of the observer inside the top m recommendations.

$$MPR = \frac{1}{|o|} \sum_{i=1}^{|o|} rank(o_i, o_{i:i-1}) \tag{6.2}$$

The function rank (see formula 6.2) indicates the position of the observed interaction o_i inside the sorted recommendation list. The MPR metrics used in this work [7], was defined to deliver the rank based on the recomendation success. For instance, if the observed interaction corresponds to the most possible one, the occurrence has a rank of one, the second one has the rank of two, etc. In order to compare the results conducted from both metrics (MPR and Recommendation Score), the MPR results are changed slightly. In this case, the results are not the ranks, but the rankes are represented by using a perceptual form. For example, the function delivers 1 if the observed interactions are top-ranked (represent the 100% accuracy), 0.66 if ranked on position two, 0.33 if ranked on position three and 0 if the interaction is not present within the first three recommendations.

Listing 6.2: Mean prediction Rank

```
"""
MPR - Mean Prediction Rank
"""
def meanprank (self, observedSeq):

  self.mpr = 0;

  for index in range (self.n_order, len(observedSeq)):

    prediction = self.predict(observedSeq[:index])
    self.mpr = self.rank (observedSeq[index], prediction)

  return float (self.mpr)/len (observedSeq)



'''
Get Rank (Normalized against the m_predictions)
'''
def rank(self, currentEvent, predictions, m_predictions):

  sorted_predictions = sorted(predictions.iteritems(),
                  key=operator.itemgetter(1), reverse=True)

  sorted_predictions_list =  [x[0] for x in
                                        sorted_predictions]

  if currentEvent in sorted_predictions_list
                                        [:m_predictions]:

    position = sorted_predictions_list[:m_predictions]
                                    .index(currentEvent)
    return (m_predictions - position)/float (m_predictions)

  else:

    return 0
```

Next, we will introduce the evaluation methodology (see Figure 6.6) we followed during this study to evaluate the Markov chain model. The validation process is split into two main steps. In the first step, the goal is to select the best model that will perform within the collected data. Moreover, in the second step, the aim is to validate the model which performed better in the first step. Next, we are going to explain in more detail the whole work-flow followed within this methodology.

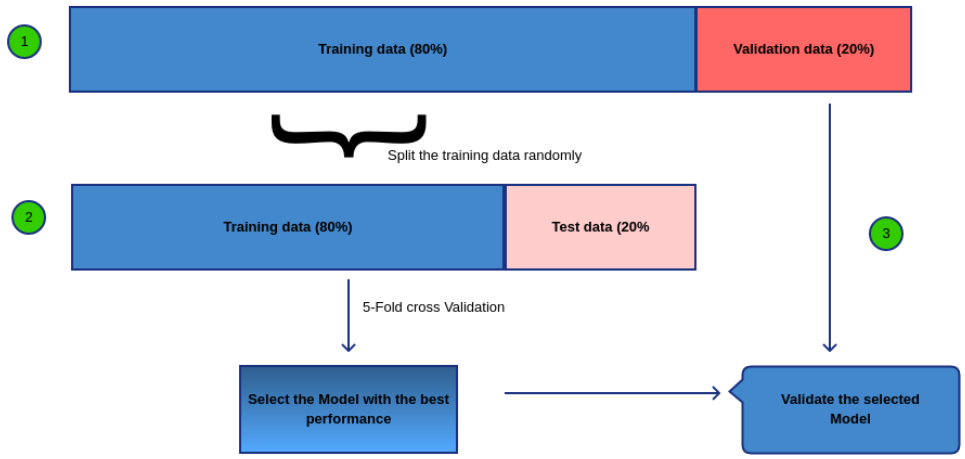


Figure 6.6.: Markov chain model validation Methodology.

This process is applied to the data collected within each task separately. For each task, the data is split into training and validation data randomly as shown in Figure (6.6, first step). In the second step, the 5-fold cross validation [26] technique is applied over the training data (shown in the Figure 6.7). In this step, the training data from the step one, is randomly split between the training and test data randomly in order to learn the best Markov chain model (best order). Cross-validation techniques split data into five folds and iterate over the data 5 times. In each iteration, one fold is chosen as test data and the others as training data. The models are trained using the current train data, and the quality is measured using the test data. Further, the accuracy results are saved for each model separately within every iteration. At the end of this step, the model with the best average result over all four tasks is ranged as the best model.

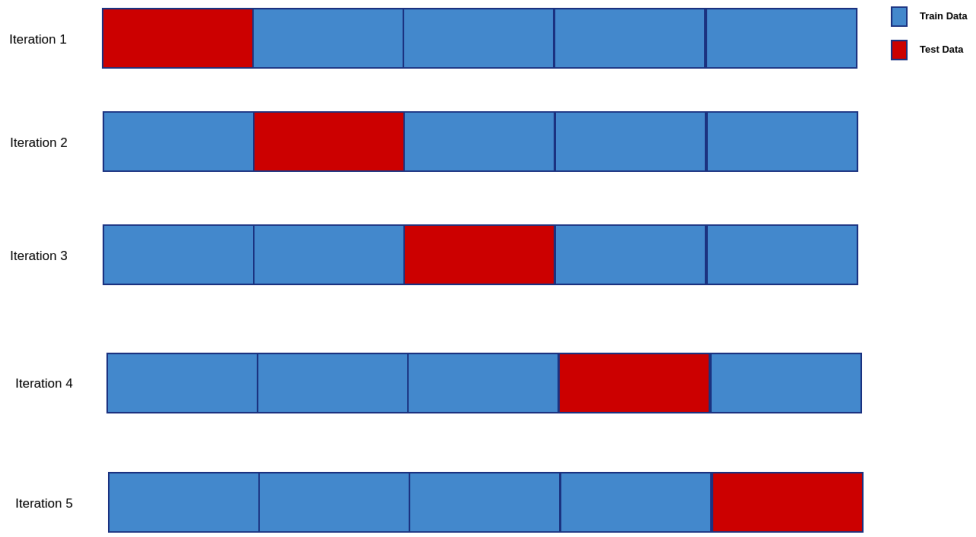


Figure 6.7.: Cross Validation Diagram. The 5-Fold chunking concept.

Finally, we select the best-performed model from the step two (see Figure 6.6) and validate against unseen data. In this case, we use the data (training and validation data) from step one (see Figure 6.6) to train and quantify the quality of the selected model.

Results

In the following, we present the results we performed during the model evaluation. First, we will present the results acquired by evaluating the models using the two different metrics (MPR and Recommender Score) explained in the previous section 6.3.3. Next, we generated two different datasets (tracked user behaviors) to evaluate the model. The datasets differs from each other regarding the number of states (individual interaction within the dashboard). In the first dataset, the process of generating a visualization (chart) is regarded as a single interaction (generate chart). For instance, if the user generated consecutively two charts (Bar chart and Bubble chart), both these interactions are

regarded as a specific state named "create a chart". Based on this assumption we generated a dataset which contains approximately ten different states. In the second dataset, we regard each specific chart (e.g, Bar chart) generation as a distinct state. For example, if the user generated one after another two charts (Bar chart and Bubble chart), than this interactions represent two distinct states within the second dataset. In this case, the second dataset contains approximately twenty different states.

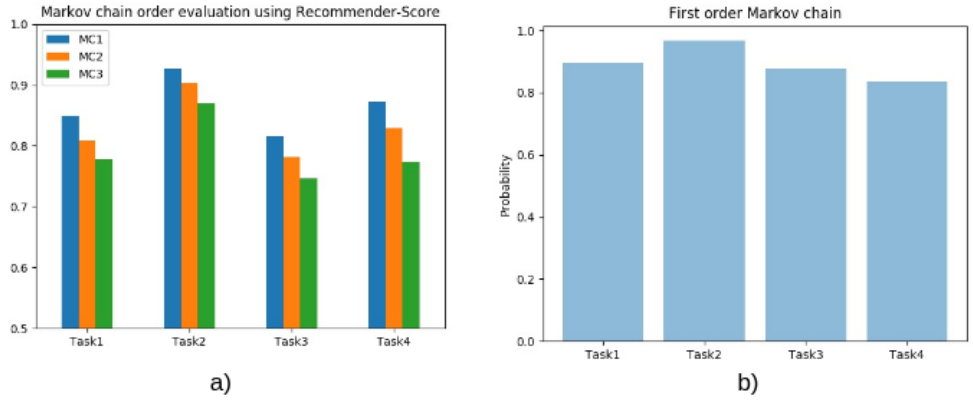


Figure 6.8.: Predictive model evaluation performed over data (ten states). Accuracy Metrics used in these evaluation is Recommendation score. Based on the accuracy metrics, the first order Markov chain model (MC1) outperforms the second order Markoc chain model (MC2) and third order Markov chain model (MC3).

Figure 6.8 shows the accuracy results computed by using the dataset which contains approximately ten states and the metrics used to evaluate the quality of the models is Recommender Score. The Figure 6.8 (*a*) compares first, second and third order Markov chains models over the data from the four various tasks. Based on the results, we can conclude that the first order Markov chain performed slightly better over the four tasks with accuracy over 80%. In the Figure 6.8 (*b*), first-order Markov chain is trained and evaluated against validation data over the four tasks. The results show that the first Markov chain order can predict the next state with accuracy over 90% when applied over unseen data (validation data, see Section 6.3.3).

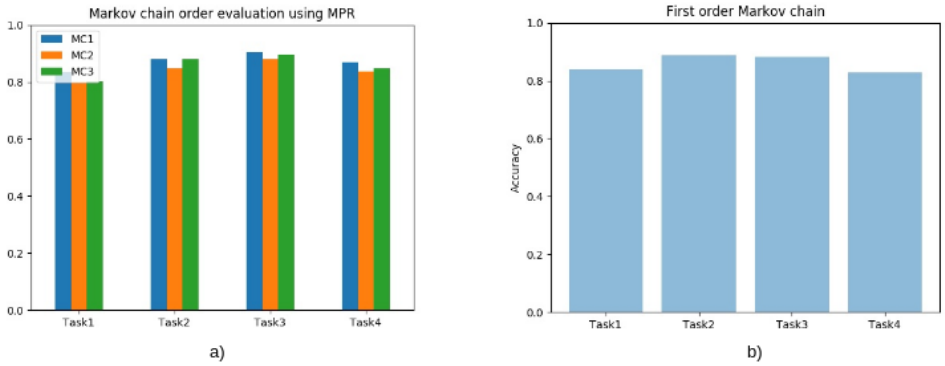


Figure 6.9.: Predictive model evaluation performed over data (ten states). Accuracy Metrics used in these evaluation is MPR score. a) Shows the result of evaluation of models based on the MPR score, the first order Markov chain model (MC1) outperforms the second order Markoc chain model (MC2) and third order Markov chain model (MC3).

Next, we present the results (shown in Figure 6.9) using the dataset which contains approximately ten states, but this time we evaluate the quality of the models by applying a different scoring metrics, the MPR, respectively. In the Figure 6.9 (*a*) the first, second and third Markov chain model are compared using the MPR score metrics. The figure shows that the first-order Markov chain performs slightly better compared to the other two. Whereas, in the next figure (see 6.9 (*b*)), the chosen model (first-order Markov chain) is trained over the data, and evaluated against validation data. Based on the evaluation results, we can conclude that the first order Markov chains perform better on these data.

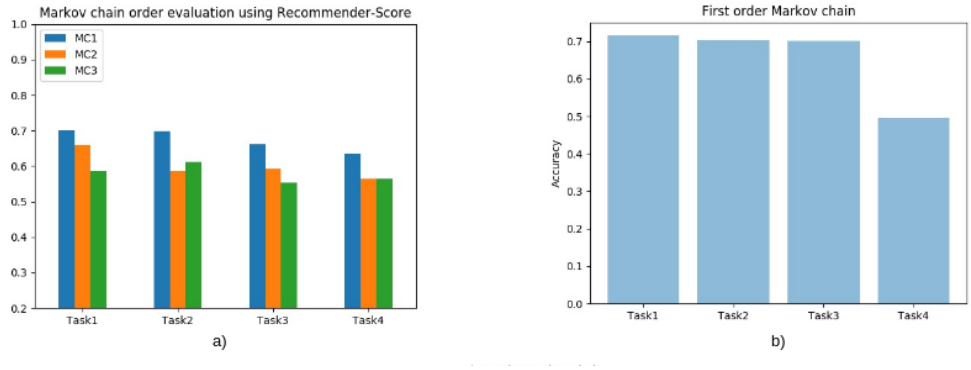


Figure 6.10.: Predictive model evaluation performed over data (twenty states). Accuracy Metrics used in these evaluation is Recommendation score. Based on the accuracy metrics, the first (MC1) Markov chain order outperforms the second (MC2) and third (MC3) Markov chain order.

Furthermore, we performed the evaluation workflow with the data where approximately twenty states are identified. The goal was to compare the performance of the recommendation model over the data consisting from various number of states (especially when the number of states is increasing). In the first phase, we compared the various Markov chain orders (first, second and third order) using the recommender score metrics. Based on the results we obtained during the evaluation (see figure 6.10 (*a*)) the first order Markov chain performs better over the data with accuracy about 70%. Next, we trained the first-order Markov chain model for each task separately and we tested the accuracy of the model using the validation data (unseen data). The figure 6.10 (*b*) shows that the model performs with accuracy of approximately 65%.

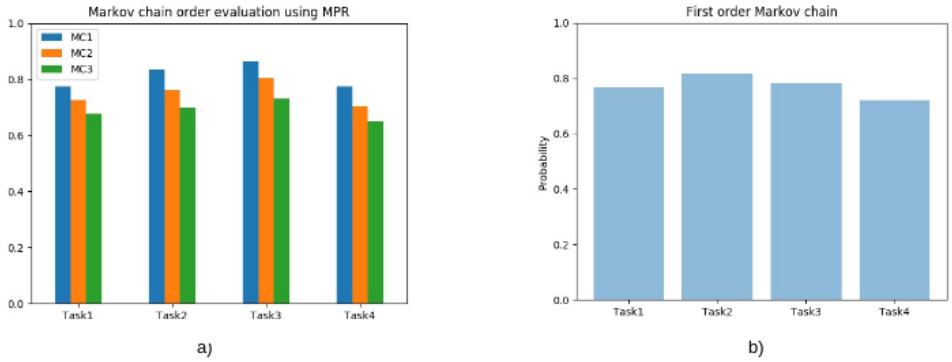


Figure 6.11.: Predictive model evaluation performed over data (twenty states). Accuracy Metrics used in these evaluation is MPR score. Based on the MPR score, the first (MC1) Markov chain order outperforms the second (MC2) and third (MC3) Markov chain order

In the last phase, we evaluated the recommendation Model with the same dataset, where approximately, twenty states were identified. In this phase, the MPR score metrics was used to quantify the model quality. Based on the results shown on the figure 6.11 (*a*), the first order Markov model performs better compared to the second and third ones. Further, the first order Markov model was trained and evaluated against validation data (see figure 6.11 (*b*)).

Summarized, during this evaluation we found out that the first order Markov chain performs better over our data gathered during the usability evaluation. Further, the model was trained using the dataset where a variable number of states were identified, respectively ten and twenty states. The results show that the predictive model performed slightly better over the dataset including ten states. Additionally, we can conclude that the performance of the predictive model decreases slightly by increasing the number of states (redouble it). The data used to train and evaluate the predictive model were gathered during the usability evaluation where twenty participants attended. It means the size of data was not sufficient which affected the performance of the model significantly.

6.3.4. Task Identification

Classification Algorithm And Validation Methodology

This subsection emphasizes classification algorithm used to identify tasks and the methodology used to validate the results. The goal of this phase was to identify the different tasks based on the m first interactions using a classification model. We performed a random forest [14] over the data collected during the evaluation process.

In order to represent the classification outcomes we used the confusion matrix [27] which is commonly used to represent the accuracy results. Furthermore, it shows a clear image of the class overlaps when making predictions. The classifier analyses using this tool helps to identify the combinations of classes that performed worse when applying the classification model.

Results

The performance of the random forest classification algorithm depends strongly on the amount of data. In this Master thesis, we gathered data from the 20 participants who took part in the online user study. For each task, three subtasks were defined, which means that for every task we gathered 66 entries.

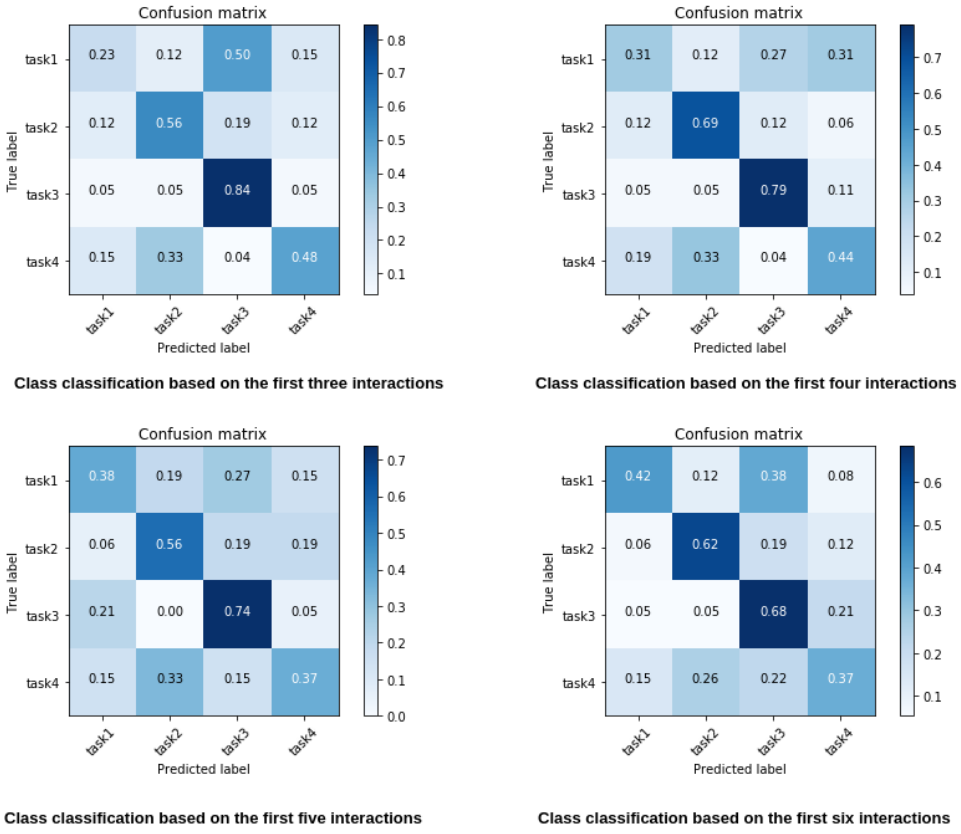


Figure 6.12.: Task classificaiton: Confusion matrix. The performance of the Random Forest over the Task two and three is sufficient compared to the task one and four.

In the following, we will discuss the outcomes of task identifications process. Figure 6.12 represents the classification results regarding different numbers of interactions. The random forest algorithm (see section 4.3.1) was applied over the dataset regarding the various number of interactions. For instance, we compare the performance of the task classification model by regarding the first three interactions.

The confusion matrix on the top-left side of the Figure 6.12 represents the classification results regarding only the first three interactions.

Moreover, the confusion matrix on the top-right provides the results based on the first four interactions. Furthermore, the confusion matrix placed on the bottom-left shows the performance of random forest over the data focusing only on the first five interactions. Lastly, the confusion matrix on the bottom-right introduces the results based on the first six interactions. The results discussed above confirm that the performance of the classification model improves while the number of regarded classes (interactions) increases. Besides, the model performs better over the task two and three. In contrast, the model is being confused when performing over task one and four.

Based on our data, we can recognize two aspects that were strongly influencing the classification results. On the one hand, the task definition (questions) when gathering data influences the outcomes. Task two and three were more clearly formulated, and the users during the evaluation interpreted them correctly. Task one and four were too general formulated which lead to the various interpretation from the participants; therefore, the variability of the data was higher concerning these tasks.

Based on the classification results we can assume, that a precise formulation of the tasks when gathering data, will provide a higher quality of data. Moreover, the amount of data when learning the model is crucial. We collected with our user study data from only 20 people which is rather small for training the model. In order to improve the accuracy of the classification model, we can consider gathering more data while simultaneously redesign the tasks more clearly.

6.4. Summary

In this Chapter, we introduced the online evaluation we performed to collect data for our tool that can recommend visual interactions based on current user's behavior and task. Futhermore, we performed a usability analysis of the used visualization recommender. Further, the evaluation of Markov chain model using the data gathered during the usability evaluation was presented. Finally, a task classification model regarding the various length of sequences was performed over the data, and the results are provided.

Summarized, the evaluation has revealed that the first order Markov chain performs better over our data gathered during the usability evaluation. Further, the model was trained using the dataset where a variable number of states were identified, respectively ten and twenty states. The results have shown that the predictive model performed slightly better over the dataset including ten states. Additionally, we can conclude that the performance of the predictive model decreases slightly by increasing the number of states (redouble it).

The data used to train and evaluate the predictive model were gathered during the usability evaluation where twenty participants attended. It means the size of data was not sufficient which then influences the performance of the model strongly.

Further, the classification model labeled correctly the task two and three with the accuracy of approximately 60%. Based on the classification results we can assume, that a precise formulation of the tasks when gathering data, will provide a higher quality of data.

7. Conclusion and Future Work

In this Master Thesis, a goal and behavior based visual recommender is provided, which is integrated into a rule-based visual recommender called Visualizer. Although Visualizer automates the whole visualization process, it still requires a certain level of user experience when it comes to perform visual tasks (gain overview, detect anomalies, detect data correlations, and analyze data distribution). To tackle this issue, we present a system that observes user's current behavior in order to infer user's current visual task and recommends the next analysis step. In doing so, the system aims to assist the user in completing the task successfully. Additionally, the recommendation model is extended in such a way, that it can recommend multiple sequences of interactions. Furthermore, a classification algorithm based on Random Forest method is applied on user behavior the data to identify user's current task. Moreover, a guidance user interface is introduced in order to present the recommendation to user. Finally, an online evaluation of the platform for different user groups was performed to collect user behavior data, which was used to train the recommender model. The collected data will also be used for the evaluation of the Visualizer user interface, however that work is not part of this Master's Thesis.

7.1. Future work

There remains still considerable work to be done in order to improve, on the one hand, the recommendation model, and on the other, the guidance user interface.

- **Collection of more data:** The data used within the scope of this thesis have been gathered during the usability evaluation where only twenty people attended. However, to improve the accuracy of recommendation

model, it is crucial to collect more data. In the future, a new online study could be performed in order to gather more data, which will be used to improve the accuracy of Markov chain and classification models.

- **Recommendation model improvement:** Currently, we deployed a recommendation model based on the Markov chains. Furthermore, we compared various Markov chains over currently gathered data. Nevertheless, the recommendation performance leaves space for improvements. Therefore, in the future, it would be interesting to have a look at various aspects in order to improve the recommendation performance. For instance, looking for other solutions outside the Markovian world (e.g., neural networks), and compare with each other could help in this aspect.
- **Task identification:** currently, the Random Forest classification algorithm has been applied in order to classify the tasks. However, there are still opportunities for further enhancements. One possibility could be a definition of tasks more different tasks and collecting behavior data for them in the planned user study. Furthermore, it would make sense to compare the performance of various classification methods on user behavior data.
- **Personalized recommender model:** In the currently deployed recommender model, the focus has been on using a non-personalized recommender model. The deployed recommender model needs to be changed only slightly in order to support the personalized recommendation. In this case, a server side component would have to support logging-in with user credentials to differentiate between users.
- **Identify a successful completion of an analytical task:** An open issue is how to reliably identify a successful completion of an analytical task. Acquiring this information could help to improve the quality of the predictive model which then will extend the current predictive model to a self-learning model. A possible way, to determine that the user completed the task could be by using the information which we gain when the user tags and rates a visualization.
- **Guidance user interface evaluation:** a usability evaluation of the currently deployed guidance user interface for different user groups could be performed. This evaluation could support further improvements, or help to recognize possible shortcomings within the current guidance user interface.

- **Active and passive recommendation study:** providing the user with a recommendation in an active or passive way still remains challenging. For example, an active recommendation could interrupt the working process of the user which could easily reduce the user satisfaction with the system. Whereas, a passive recommendation could be easily overseen. Therefore, finding a balance between these two aspects is crucial when implementing a guidance user interface.
- **Tracking the user behavior:** Presenting the observed behavior of the current user within the dashboard could help him/her learn and perform better. The user can explore the history of interactions within the dashboard, and later on reapply it whenever she/he wants to. The user can also solve a task and forward it (e.g., using a link) to other users to illustrate the task's solution step by step.
- **The explanation about the recommended interactions:** The attempt to explain why the interactions are recommended, could help to convince the users that the recommendation is relevant, or even recognize easily that a recommendation is not appropriate in his/her case. In this case, we can consider adding user comments to the individual recommendations.

Appendix

Appendix A.

Usability evaluation questionnaires

Statistics

1. How's your English?
2. Have you used the Visualizer before?
3. How frequently do you use visualization tools?
4. If you use visualization tools: which tools you have experience with and what do you use them for?
5. How frequently do you use data analysis tools?
6. If you use data analysis tools: which tools you have experience with and what do you use them for?
7. How frequently do you look up information on the Internet?
8. What's your age?
9. Gender?
10. Country?
11. What's your Primary Job Function
12. Education?
13. Primary interests?
14. Please provide your level of expertise in each of the following keywords: range 0 (No knowledge) – 5 (Expert)

Effort and demand questions The following assessment is used to measure user's personal opinion on how much workload was required of the user during the task she just completed.

1. Task [X]: Mental Demand

   a) How mentally demanding was the task?
2. Task [X]: Physical Demand
   a) How mentally demanding was the task?
3. Task [X]: Physical Demand
   a) How physically demanding was the task?
4. Task [X]: Temporal Demand
   a) How hurried or rushed was the pace of the task?
5. Task [X]: Performance
   a) How successful were you in accomplishing what you were asked to do?
6. Task [X]: Effort
   a) How hard did you have to work to accomplish your level of performance?
7. Task [X]: Frustration
   a) How insecure, discouraged, irritated, stressed, and annoyed were you?
8. Task [X]: Any comments? What was good / bad / unexpected / difficult?

Task questions

1. Task 1: Gain Overview in the Visualizer
   a) Which countries are represented in this dataset? Can you recognize a pattern?
   b) Which countries have the highest life expectancy (between 2005-2008)?
   c) How many countries have a GDP greater than 4 000 000 USD?
2. Task 2: Analyze outliers (Find Anomalies) in Visualizer
   a) When you look at the life expectancy for each country, can you detect any outliers?
   b) When you look at the population for each continent, can you detect any outliers?

   c) When you look at the CO2 Ton/Person for each country, can you detect any outliers?

3. Task 3: Analyze trends (Correlation)
   a) Is there a trend of increasing/decreasing CO2 Tons/Person rate in a certain country over 10 years? If yes, name these countries?
   b) Is there a trend of increasing/decreasing population rate in a certain country from 2005-2015? If yes, name these countries?
   c) Is there a trend of increasing/decreasing GDP value in a certain country from 2005-2015? If yes, name these countries?
4. Task 4: Compare Variable Distribution (Distribution)
   a) What is the distribution of life expectancy per country between 2005 and 2015?
   b) What is the distribution of the GDP per country between 2005 and 2015?
   c) What is the distribution of population per country from 2010 to 2015?

Final questions I

1. What did you like about the Visualizer?
2. What did you dislike about the Visualizer?
3. How did you like the visual design of the Visualizer?
4. Do you have suggestions for improving the design of the Visualizer?
5. Was the Visualizer easy to use and interact with?
6. Do you have suggestions for improving the usability of the Visualizer?
7. For which tasks would you personally use the Visualizer?
8. Comment shortly the Visualizer. What did you like or dislike, what would you use it for?
9. If you could have solved any of the tasks with other tools of your choice, which ones would you have used?

Final questions II

1. I think that I would like to use Visualizer frequently: range 0 (Strongly disagree) – 5 (Strongly agree)

2. I found Visualizer to be simple: range 0 (Strongly disagree) – 5 (Strongly agree)
3. I thought Visualizer was easy to use: range 0 (Strongly disagree) – 5 (Strongly agree)
4. I think that I could use Visualizer without the support of an expert: range 0 (Strongly disagree) – 5 (Strongly agree)
5. I found the various functions in Visualizer were well integrated: range 0 (Strongly disagree) – 5 (Strongly agree)
6. I thought there was a lot of consistency in Visualizer: range 0 (Strongly disagree) – 5 (Strongly agree)
7. I would imagine that most people would learn to use Visualizer very quickly: range 0 (Strongly disagree) – 5 (Strongly agree)
8. I found Visualizer very intuitive: range 0 (Strongly disagree) – 5 (Strongly agree)
9. I felt very confident using Visualizer: range 0 (Strongly disagree) – 5 (Strongly agree)
10. I could use Visualizer without having to learn anything new: range 0 (Strongly disagree) – 5 (Strongly agree)
11. I felt very confident using Visualizer: range 0 (Strongly disagree) – 5 (Strongly agree)
12. I felt very confident using Visualizer: range 0 (Strongly disagree) – 5 (Strongly agree)

Bibliography

[1] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pp. 111–117, IEEE, 2005.

[2] H.-J. Schulz, T. Nocke, M. Heitzler, and H. Schumann, "A design space of visualization tasks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2366–2375, 2013.

[3] K. Nazemi, R. Retz, J. Bernard, J. Kohlhammer, and D. Fellner, "Adaptive semantic visualization for bibliographic entries," in *International Symposium on Visual Computing*, pp. 13–24, Springer, 2013.

[4] D. Gotz, J. Lu, P. Kissa, N. Cao, W. H. Qian, S. X. Liu, and M. X. Zhou, "Harvest: an intelligent visual analytic tool for the masses," in *Proceedings of the first international workshop on Intelligent visual interfaces for text analysis*, pp. 1–4, ACM, 2010.

[5] D. A. Keim, "Visual exploration of large data sets," *Communications of the ACM*, vol. 44, no. 8, pp. 38–44, 2001.

[6] D. Gotz and Z. Wen, "Behavior-driven visualization recommendation," in *Proceedings of the 14th international conference on Intelligent user interfaces*, pp. 315–324, ACM, 2009.

[7] K. Nazemi, C. Stab, and D. W. Fellner, "Interaction analysis: An algorithm for interaction prediction and activity recognition in adaptive systems," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 1, pp. 607–612, IEEE, 2010.

[8] W.-K. Ching and M. K. Ng, "Markov chains," *Models, algorithms and applications*, 2006.

[9] A. TOLVER, "An introduction to markov chains,"

[10] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 811–820, ACM, 2010.

[11] V. V. Mayil, "Web navigation path pattern prediction using first order markov model and depth first evaluation," *International Journal of Computer Applications (0975-8887)*, vol. 45, no. 16, 2012.

[12] A. A. Ahmed and N. Salim, "Markov chain recommendation system (mcrs)," *International Journal of Novel Research in Computer Science and Software Engineering*, vol. 3, pp. 11–26, 2016.

[13] G. Shani, D. Heckerman, and R. I. Brafman, "An mdp-based recommender system," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1265–1295, 2005.

[14] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[15] J.-w. Ahn and P. Brusilovsky, "Adaptive visualization of search results: Bringing user models to visual analytics," *Information Visualization*, vol. 8, no. 3, pp. 167–179, 2009.

[16] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis, "Seedb: automatically generating query visualizations," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1581–1584, 2014.

[17] B. Mutlu, P. Hoefler, V. Sabol, G. Tschinkel, and M. Granitzer, "Automated visualization support for linked research data.," *I-SEMANTICS (Posters & Demos)*, vol. 1026, pp. 40–44, 2013.

[18] J. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," *IEEE transactions on visualization and computer graphics*, vol. 13, no. 6, 2007.

[19] I. Simic, "Visualizer an extensible dashboard for personalized visual data exploration," *Graz University of Technology*, 2018.

[20] B. Mutlu, E. Veas, and C. Trattner, "Vizrec: Recommending personalized visualizations," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 6, no. 4, p. 31, 2016.

[21] Yhat, "Random forests in python." http://blog.yhat.com/posts/random-forests-in-python.html, 2013. Accessed: 2018-05-06.

[22] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[23] G. P. Basharin, A. N. Langville, and V. A. Naumov, "The life and work of aa markov," *Linear algebra and its applications*, vol. 386, pp. 3–26, 2004.

[24] I. Teodorescu, "Maximum likelihood estimation for markov chains," *arXiv preprint arXiv:0905.4131*, 2009.

[25] Lindon, "Lindon leader quote." https://www.adrianeleighvisuals.com/projects/5952196, 2013. Accessed: 2018-05-15.

[26] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, pp. 1137–1145, Montreal, Canada, 1995.

[27] R. Susmaga, "Confusion matrix visualization," in *Intelligent Information Processing and Web Mining*, pp. 107–116, Springer, 2004.