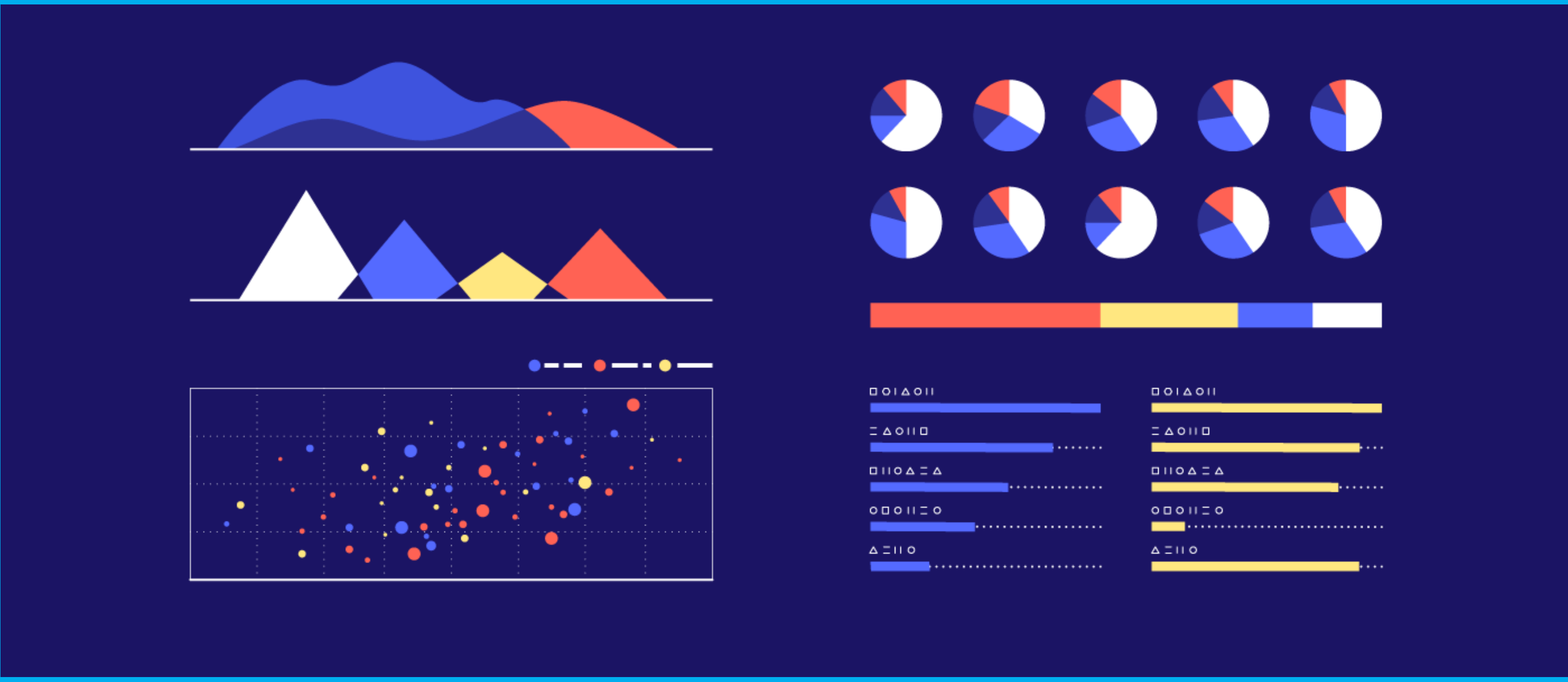
Title

Interactive Data Visualization

Glenn van den Belt
Shaan Hossain
Joost Jansen
Adrian Kuiper
Philip Tempelman



Title
Interactive Data Visualization

by

Glenn van den Belt
Shaan Hossain
Joost Jansen
Adrian Kuiper
Philip Tempelman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 1, 2013 at 10:00 AM.

Student number:	1234567	
Project duration:	March 1, 2012 – January 1, 2013	
Thesis committee:	Prof. dr. ir. J. Doe,	TU Delft, supervisor
	Dr. E. L. Brown,	TU Delft
	Ir. A. Aaronson,	Acme Corporation

Preface

Preface…

Glenn van den Belt
Shaan Hossain
Joost Jansen
Adrian Kuiper
Philip Tempelman
Delft, may 2021

Summary

Contents

1 Introduction

Looking at a visualized form of data, such as a graph, and finding correlations is a trivial task for researchers. Plotting the data onto such a visualizer is not as trivial as one might think as problems, such as slow response, arise when working with large data. A good visualization removes the noise from data and highlights the useful information. Unfortunately, There are little to none tools as such available online. And if there are, they require a background in programming to be able to use them. Having such a tool will help researchers in their findings when working with large amounts of collected data. For example, a researcher studying the cause of cardiac failure after collecting data of physical activities can easily find correlations with a visualization tool. Aiding in this task the TU Delft faculty of Pattern Recognition & Bioinformatics has provided several data sets and an pre-alpha build of an visualization tool.

This report documents the research, design and implementation done in the project. For the design, the following question was asked: *How can we create a data visualizer with the current tools and software available online?* To answer this question multiple libraries were thoroughly researched and tested. For each library, their functionalities and extensibilities were compared with each other. To find the most compatible library for this project the requirements and requests given by the client were considered. The aim of this project is to create a Graphical User Interface (GUI) application in which it makes it easy for the user to upload their data and visualize it.

This report is structured as follows. Chapter 2 provides a problem analysis consisting of the problems that arise when working with large datasets. Followed by solutions on how they can be tackled and how some solutions are created with already existing software. Chapter 3 discusses the different ways on how the data can be visualized and what features are present that allow the user to modify the data to aid them in their visualization. In chapter 4 the final product will be presented. All its features are discussed in the subsections along with the reason for their presence. Chapter 5 contains a discussion about the final product. It discusses the realisation in terms of requirements and the comparison between the clients expectations and the actual result. Chapter 6 mentions how the project can be possibly expanded with different features, a reflection on how the team has worked together on the project and a final conclusion on whether the team has reached the goals they aimed to achieve with this project. Finally, Appendix A contains a list of all requirements listed according to MoSCoW-method.

2 Problem Analysis

2.1. Main Goal

As data sets grow larger and larger, performing computations with them becomes cumbersome and slow. In addition to this, deriving valuable information from these immense data sets is not always trivial. Our goal is to create an efficient and extensible application that visualizes data into useful graphs. This visualizer should be interactive and when using it the user should be able to derive valuable information from the data, for instance by recognizing possible correlations between different data features. Furthermore, the user should be able to use the features to filter out irrelevant or unwanted data. Finally, the client has emphasized that building such an interactive visualizer should not be done from scratch unless doing so would otherwise be infeasible.

2.2. Usage of Related Libraries

Picking a suitable library that meets our needs is a primitive task. After discussing, we have concluded that to meet our client needs, we need an extensive, high-level library and fully use its functionalities or use a low-level library and build all the functionalities from scratch. Most high-level libraries are not extendable in terms of features and are thus limited to what we can do with them. The other possibility is to use low-level libraries and build all the functionalities from scratch. Low-level libraries give us more freedom but will be more complicated since we will have to create everything from the ground up and combine multiple different libraries.

2.3. Processed Type of Data

The data will always be formatted in such a way that it fits the application best. This data will have a time series format and might sometimes contain continuous variables. In case of continuous variable the visualizer must change into a heat map instead of a scatter-plot. Furthermore, each data points must each carry its own meta-data and must be visualized when hovered over them. This metadata can be either delivered in a table separate from the actual data and linked with relational properties.

2.4. Nested Filtering on Data

All data points in a scatter plot are independent, and each carries its own meta-data. According to the user needs, additional filters could be applied when viewing a subset of data points. This phenomenon is called nested filtering, as multiple filters may be applied to a set of data points. When using multiple filters, we must avoid parsing the whole dataset again due to efficiency and memory reasons. We must find a convenient way to store intermediate results while minimizing memory use, such as caching.

2.5. Apply Alternative Functions and Algorithms on Data

After the dataset has been plotted the user will need to be able select a set of data points. The user can give implement their own function or algorithm (t-SNE, PCA, etc.) and this will be applied over the selected data points. The result of this will be outputted and plotted. This function or algorithm will be written by the user themselves in the back-end. We will have to ensure that the plotter will work with any given well-defined function.

2.6. Computation Intensity and Thumbnails

As the client provides us with a large amount of data (in the order of Gigabytes), it might be cumbersome to load all the data in at once. The client has pointed out that a feature should be created, which allows the user to give an order of which part of the data is loaded first. We should keep this feature in mind when expanding the project to a web application, since the browser may be limited in memory and could raise some problems due to efficiency reasons. When for example, a filter box is ticked, the application will query the whole database, and therefore the loading time of the application will take a lot longer. Additionally, to be able to filter on specific points, the application also needs to create a small thumbnail that visualizes a plot of that specific point when hovered over it.

2.7. Possibility of Extendibility

The client has indicated that they should be able to extend the visualiser with additional charts, graphs, filters and/or similar things. Therefore we have to create some sort of framework which makes it straightforward for our client to extend the visualiser with additional features in the future.

2.8. Library Study

3 Visualising Data

3.1. Usage of the Dash library

”aaa”[**Kivy**]

3.1.1. Nested Data Filtering

3.1.2. Data Selection in the GUI

3.1.3. Re-Plotting Selected Data

3.1.4. GUI Layout

3.1.5. User-defined functions

3.2. Usage of the Pandas library

4
Results

4.1. Final Product

4.2. Efficiency

4.2.1. Lazy Loading of Data

4.2.2. Caching Plotted Graphs

4.3. User-defined Functions and Algorithms

[1]

5
Discussion

6
Product Recommendations and Conclusions

6.1. Next Steps

6.1.1. Web-app

6.2. Reflection on Teamwork

6.3. Conclusion

References

[1] A. Einstein. “Eine neue Bestimmung der Moleküldimensionen”. In: *Annalen der Physik* 324.2 (1906), pp. 289–306. DOI: `10.1002/andp.19063240204`.

A Appendix

This appendix includes a list of the requirements ordered using the MoSCoW model.

A.1. Non-functional Requirements

- The system will be able to select, filter and visualize data within 2 minutes.
- The system should be user-friendly to the extend that people from all machine learning backgrounds are capable of understanding how to use it within 15 minutes.
- The system must be able to handle dataset sizes of up to at least 5 gigabytes.

A.2. MoSCoW Method

A.2.1. Must Have

- The system must be built in Python 3.
- The system must be able to run in Linux.
- The system must have an easy to use graphical user interface.
- The system must be able to select data using an selection method such as ’rectangle selection’ or ’lasso selection’ or something similar.
- The system must be able to plot data points with a scatter plot.
- The system must be able to use multiple machine learning tools written by the user on the loaded data.
- The system must be able to show the meta-data of a specific point in a thumbnail, which for example pops up when hovering over that specific data point.
- The system must be able to visualise data which has three or less dimensions.
- The system must be able to run locally.
- The system must be able to run on data written in a csv file.
- The user must be able to filter on specific points.

A.2.2. Should Have

- The system should be extendable so that the client can implement additional data visualization and filtering methods.
- The system should have a short description in which the current tool of visualizing the data is explained.
- The system should be able to plot continuous data with a heat map.
- The system should support to inject any functions before plotting. An example would be a function to compute a density map before plotting or plotting the trajectory of data over time by connecting data points with lines and arrows.
- The user should be able to give an order of loading when a dataset is loaded.
- The user should be able to export and download figures made by the system.
- The system should avoid parsing the whole dataset again when applying multiple filters.
- The system should support nested filtering.
- The system should be able to create multiple subplots from one large plot and view them in on single time series

A.2.3. Could Have

- The system could run as a web app.
- The user could save presets of settings, in the form of ticked boxes etc.
- The user could be able to load in a csv file and select, filter and visualize the data.
- The system could be able to load in a csv file with meta-data within the desired table or with meta-data in a separate table.
- The user could use a Drag Drop to upload the data.
- Selection of specific colors for specific datasets.

A.2.4. Won't Have

- Ability of users to have an account.
- Converter of data, since the client will deliver data in our demanded format.
- There won't be any support for any local/online databases, since all data will be loaded in
- There won't be support for mobile and tablet versions.