Visual Analytics for Investigative Analysis

Thomas Absenger, Mohammad Chegini, Thorsten Ruprechter, Helmut Zöhrer

Graz University of Technology
A-8010 Graz, Austria

3 July 2017

Abstract

Parallel coordinates plot is a way to represent multivariate datasets, in which, each record in the dataset is shown as a polyline that starts from the first axis of dimensions and ends at the last axis. Since currently there is no open source solution to represent multivariate datasets in Java, based on JavaFX Chart, we implemented an open source library for Parallel coordinates plot. The library consists of two parts. One part can be used standalone, for developers that want to use parallel coordinates in their application. The second part is a showcase that uses the core classes and has multivariate data import and some additional features to parallel coordinates. The implemented parallel coordinates plot have features like filtering, inverting the axes, selection, brushing and axes reordering.

Contents

Credits

This survey was created for the master's course "Information Visualization" at Graz University of Technology and is based on a skeleton provided by courtesy of Andrews [2012].

List of Figures

Chapter 1

Introduction

Due to the recent advances in data gathering, scientists in various domains are confronted with large amounts of data and collections of datasets. Such datasets usually have many records, which hold many attributes (dimensions). Visualizing and interacting with these high-dimensional datasets to explore data and find interesting patterns are a continuing challenge. In a paper used to illuminate the path for visual analytics researchers Keim [2002] argues that although there are a handful of techniques to represent data sets in traditional manner (e.g. scatter plots), usually it is not adequate to apply these techniques to high-dimensional data sets. Therefore, novel techniques like Scatter Plot Matrices (SPLOMs) and Parallel Coordinates Plots are suggested by researchers.

On the other hand, Java is a popular language for developers. JavaFX is a new graphics library included in the core of JavaSE. This new framework also contains the possibility to visualize charts. The JavaFX Chart classes support traditional charts like pie chart, scatter plot, and line chart but not multivariate dataset visualization. Since currently there is no other multivariate dataset visualization option available for Java, we decided to develop a small library for high-dimensional dataset representation and started with parallel coordinates plot.

1.1 Multivariate Data

Multivariate datasets usually have more than three dimensions. Finding the relation of the records in these datasets and visualizing them is not always straightforward. Researchers in fields like biology, medical science, economics, and social science are confronted with such datasets frequently. Information visualization tech-

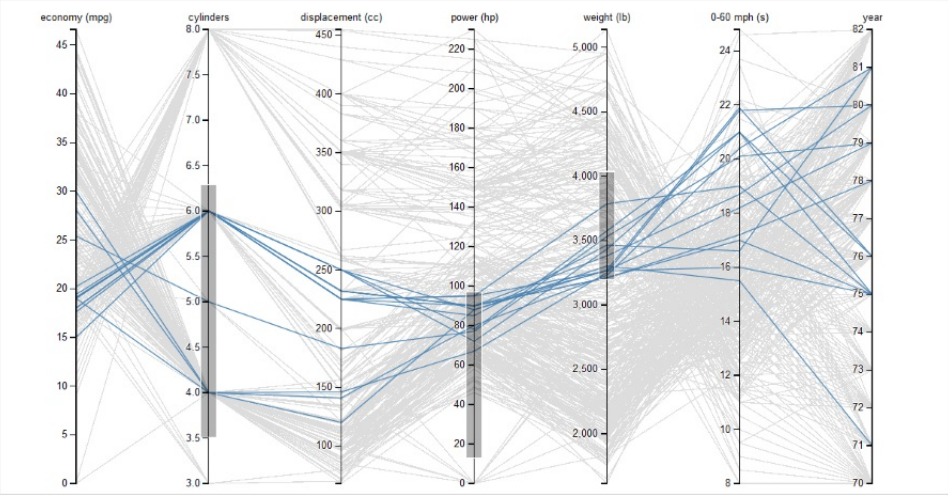

Figure 1.1: Screenshot of parallel coordinates plot of classic car dataset, draw by D3.js library.

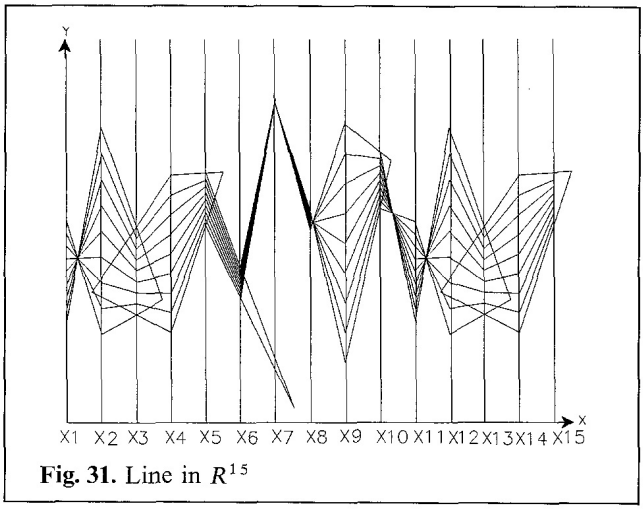

Figure 1.2: A parallel coordinates plot including fifteen dimensions. [Image extracted from Inselberg and Dimsdale [1990]. Used under the terms of Austrian copyright law: §42]

niques to represent multivariate datasets help scientists to have a better understanding of the data. SPLOMs and parallel coordinates plots are popular for this situation. An SPLOMs is a table that consists of scatter plots of all pair-wised dimensions of the dataset, therefore each record is represented by many points in the plot. In contrast, parallel coordinates plots contain axes, and each axis represents a dimension. A single record is represented by one polyline.

1.2 Parallel Coordinates Plot

Using a parallel coordinates plot for visualisation was first suggested by Philbert Maurice d'Ocagne in 1885. Later, Inselberg popularized it in 1959. Figure 1.2 shows a parallel coordinates plot from Inselberg [1985]. As displayed in figure 1.1, a parallel coordinates plot consists of a number of axes, which represent dimensions, and polylines that represent records. Parallel coordinates are a popular way to represent multivariate datasets.

As mentioned before, in contrast to SPLOMs each record is shown by one polyline. In a SPLOM, linking and brushing techniques should be used to display the connection of records. Moreover, similar to an a SPLOM, the user is able to view all the dataset in just one glance. Although the curse of dimensionality is sometimes a problem to visualize the whole dataset, with proper interactions and dimension reduction, it is possible to focus on the interesting parts of the dataset.

Despite the advantages of a parallel coordinates plot, it also has some drawbacks. Firstly, it is not very easy for a non-experienced user to understand. For example, a parallel coordinates plot may not be convenient to show in a board meeting. Although a parallel coordinates plot is not suitable for ordinary users, it is a strong way of multivariate data representation for pattern visualization and high-dimensional data interaction Few, 2006. For example, each cluster can be visualized by different colors and the user is able to see the similarity of records in each cluster by glancing at the chart. One of the suitable analysis on a parallel coordinates plot is regression analysis Li et al., 2010. On the one hand, if the lines across two neighbor axes are parallel, the regression between the variables is positive. On the other hand, if the lines crossing each other the variables have negative relation. Finally, if there is no pattern, no correlation between them can be detected.

Another problem with parallel coordinates is a scalability issue. By changing the axes of the plot, everything has to be redrawn again. Also, for a high number of dimensions, it is not convenient to show everything on one screen - the user needs panning and zooming to grasp the whole dataset.

Chapter 2

Motivation

Java is one of the most popular programming languages. However, the graphical side of JavaSE is not well developed and usually not the first choice for GUI development. In 2008, a new graphic library for Java which is called JavaFX was developed. JavaFX intends to replace traditional Java Swing library for Java Standard Edition. Support for CSS styling, better performance, and more stylish GUI makes JavaFX an interesting library to develop desktop applications. [Oracle, 2014]

Currently, there are two major ways to represent data using JavaFX or Swing: JFreeChart and JavaFX chart. None of these solutions support high-dimensional dataset visualization. JFreeChart is a free library written in Swing [**j**]. As mentioned before, Swing is not well maintained anymore and was now replaced by JavaFX. Furthermore, this library is abandoned for more than two years. At the right side of figure 2.1 it can be seen that this library supports various charts like bar chart, pie chart, scatter plot, and line chart. The following list can provide an overview of the advantages and disadvantages of using JFreeChart as a base for the project:

+ Customizable: developers can create various two-dimensional charts based on the library.

+ Open Source, therefore easy to modify.

– No high-dimensional dataset support.

– Written in Swing, which is now being replaced by JavaFX.

– No update in the past 2 years (from 2015).

– Third-party-library and not part of JavaSE.

On the other hand, JavaFX Chart is part of JavaSE and supported by Oracle. This library contains six different traditional charts:

- BarChart: Represents data using rectangles.
- LineChart: Displays data as a series of points and lines connect the points.
- AreaChart: Similar to LineChart, but the areas below the lines are filled with colors.
- ScatterChart: Uses Cartesian coordinates to represent two-dimensional datasets.
- BubbleChart: Similar to a ScatterChart, but each glyph represents three-dimensional data.
- PieChart: Divides each area of a circle according to the value of a record.

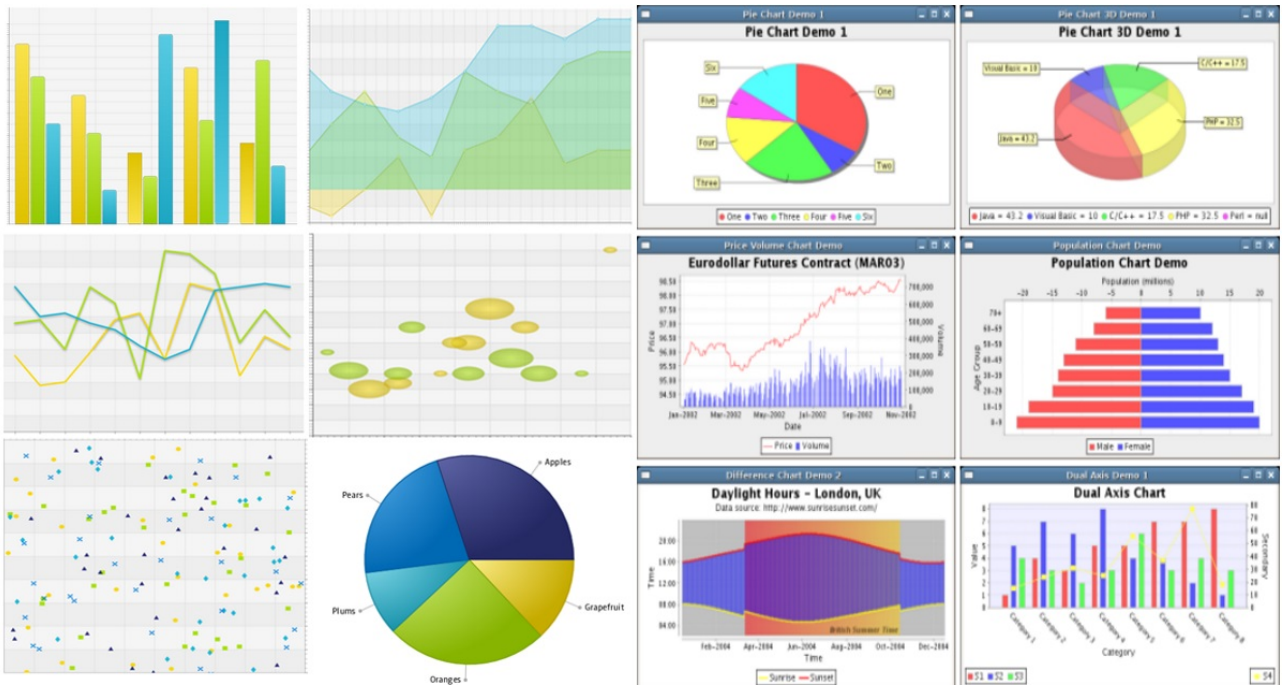

Figure 2.1: (left) Examples of JFreeChart including bar chart, 3D bar chart, line chart and area chart. (right) Chart supported by JavaFX including bar chart, area chart, line chart, bubble chart, scatter plot and pie chart.

Some of the aforementioned charts that contain two axes are derived from a class called XYChart. XYChart holds XYChart.Data as an inner class which represents one record in both the dataset and the XYChart. XYChart.Series represents a set of records (e.g. clusters). Since XYChart only supports two-dimensional datasets, we had to develop our own abstract class derived from the the JavaFX superclass Chart and called it HighDimensionalChart. Later, based on this abstract base class, ParallelCoordinatesChart was created. The high-dimensional chart class also contains Series and Record to be used by developers to load their custom datasets. We mainly focused our development on the ParallelCoordinatesChart class for now.

Chapter 3

Features

As mentioned before, we propose a library for developers to implement their custom multivariate data visualization plots using JavaFX. However, our main goal was to implement a parallel coordinates plot. The plot is can be easily used as a component in a JavaFX application. Figure 3.1 roughly visualises the package structure of the project. Red boxes represent classes that were already part of the standard JavaFX Chart library. Yellow boxes describe classes which are part of a third party library [Giles, 2017]. The blue boxes show classes which we implemented ourselves due to realize this project. Chart is a Java class belonging to JavaFX. One of the extensions of this class is XYChart. Based on this concept, we created an abstract HighDimensionalChart class which contains Series, Record, and MinMaxPair. Based on the concept of XYChart.Data and XYChart.Series, we added the concept of Series and Record to the class. Record simply represents one record in the dataset which is drawn as a polyline in the ParallelCoordinatesChart. Series is a set of records (e.g. clusters). One has to add Records to a Series to display them in the chart.

ParallelCoordinatesChart, which is the main contribution of the project, is implemented by extending HighDimensionalChart. Almost all functionalities of ParallelCoordinatesChart are implemented in this class. In addition to that, ParallelCoordinatesAxis, a class that represents an axis in the ParallelCoordinatesChart and holds references to all UI elements and information belonging to this axis, was created. We used the RangeSlider class published by Giles [2017] to implement some interaction techniques. Moreover, this class contains use NumberAxis, Label, and Button.

Based on the aforementioned structure, we implemented four different categories of features: basics, graphics, interactions and CSV import. Basics are related to the representation of the parallel coordinates plot like title, legend, axes and drawing polylines. Graphics are related to more details like opacity, color, and size of polylines. Interactions are set of techniques to manipulate parallel coordinates plot, including filtering, reordering axes, inverting, brushing and selection. At the end, a simple CSV importer is written for a showcase of the library.

In addition to those features mentioned above, a tool was developed which can be used to test and try out the given functionalities. This tool was published under a MIT license [Massachusetts Institute of Technology, 1988]. The source code for In addition to that, a tool was developed to demonstrate the basic functionalities of the graph and also enables import of simple csv-data. Source code for both tool and parallel coordinate implementation can be found on Github [Group 5, 2017]. The setup of ParallelCoordinatesChart and the tool will be further explained under 4.

3.1 Basics

The two main parts of the chart implementation are the basic parallel coordinates plot representation and basic dataset storing. Figure 3.2 shows a basic parallel coordinates plot of a dataset representing all car records before any interaction. By using the basic NumberAxis of JavaFX, seven parallel axes are drawn together to shape the basics of a parallel coordinates plot. At the bottom of the axes, the label of the given attribute (dimension) is shown. Each record is represented by a polyline and a color. The color distinguishes which Series the

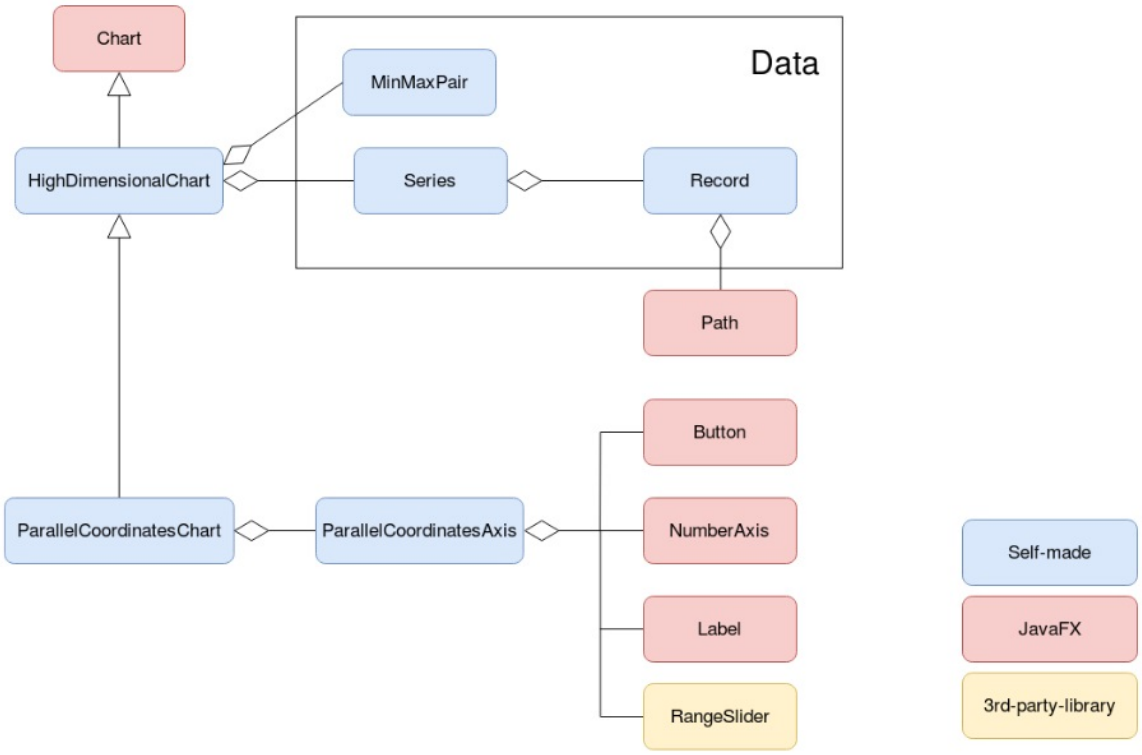

Figure 3.1: An overview of the rough package structure of the parallel coordinate chart implementation.

record belongs to. Above the plot, the title of the graph is shown. A legend can be optionally drawn below the main chart. The buttons on the top are used for axis interactions (e.g. inverting and moving). Since our class is derived from the main JavaFX Chart superclass, most of the basic Chart attributes are customizable using FXML. ParallelCoordinatesChart can be included in any FXML file as following:

```
<ParallelCoordinatesChart fx:id="parcoordChart" title="ParCoord Test"
    BorderPane.alignment="Center">
</ParallelCoordinatesChart>
```

3.2 Data

As mentioned before, single data records are stored as objects of class Record. A Record represents an array of numbers corresponding to dimensions. The simplest way to create a new Record is as follows:

```
Record record = new Record(index, attributes);
```

A set of records is called Series. For example, a series can represent a cluster. In the abovementioned classic car dataset, Japan, U.S., and Europe are three series. It is possible to control color, opacity, and name of a series. A parallel coordinates chart has a collection of Series. Figure 3.3 demonstrates a simple data hierarchy. To create and add a Series, one could use this code as an example:

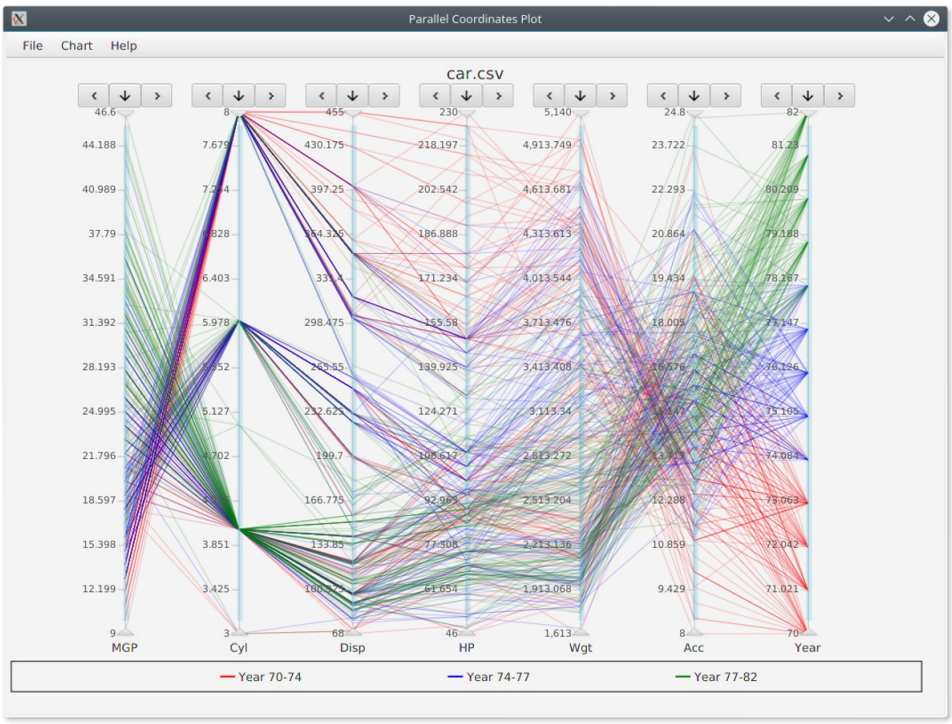

Figure 3.2: A basic parallel coordinates plot without any interaction. It shows the classic car dataset.

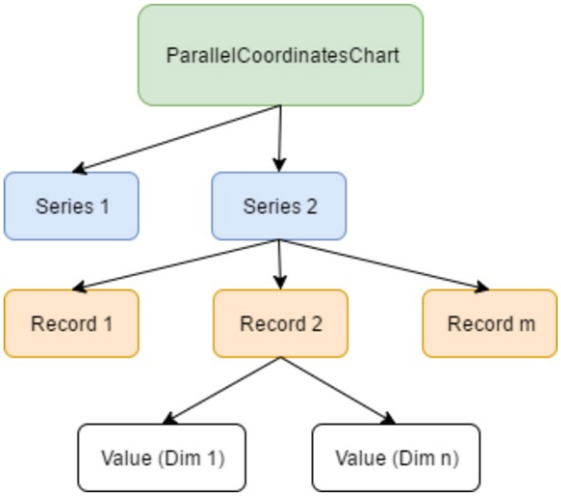

Figure 3.3: A simple data structure for parallel coordinates chart.

```
Series s = new Series("Series 1", listOfRecords, Color.Black, 0.2);
parcoordChart.addSeries(s);
```

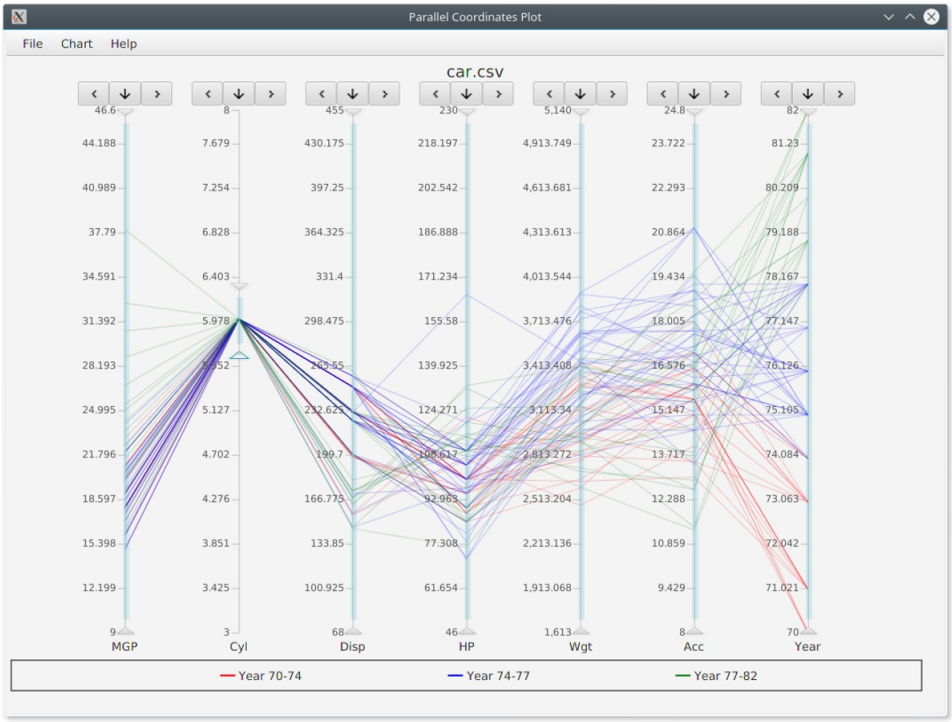

Figure 3.4: Example for using the filter sliders in the ParallelCoordinatesChart.

3.3 Interactions

Information visualisation systems consist of two main components, namely representation and interaction. Data representation concerns the way that mapping happens from data to display Yi et al., 2007. On the other hand, interaction starts with an intent followed by an action of user and reaction of the system and finally feedback given to the user.

To better interact with the parallel coordinates plot, we implemented a set of interaction techniques, including filtering, inverting, selection, brushing, and axes reordering.

Filtering is an interaction technique in which a user can filter which range of values of a specific axis should be visualised. We support this interaction by providing an draggable arrow on top and bottom of each axis as shown in Figure 3.4. Filtering can be done per axis. It is also possible to turn off this feature and set the opacity of the lines which are not currently displayed:

```
parcoordChart.setUserAxisFilters(true);
parcoordChart.setFilteredOutOpacity(0.1);
```

Selection or **Highlighting** is a task in which a user selects one or multiple polylines in a parallel coordinates chart for further inspection. Figure 3.5 shows a user selecting multiple polylines on the left. On the right side of the figure, a user simply hovered a polyline. Hovering lines also leads to highlighting them.

There are four main commands to control the selection and highlighting of the ParallelCoordinatesChart:

```
parcoordChart.setUseHighlighting(true);
parcoordChart.setHighlightColor(Color.Red);
parcoordChart.setHighlightOpacity(1.0);
```

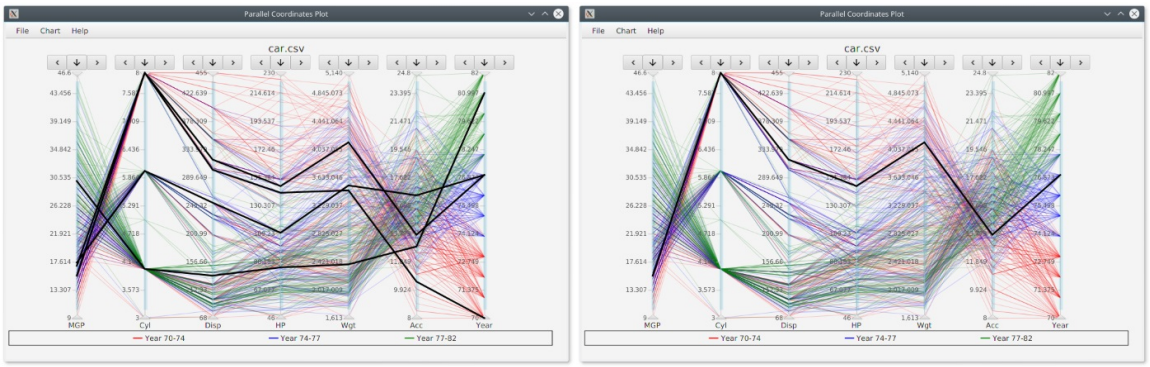

Figure 3.5: Highlighting (selection) of records in the ParallelCoordinatesChart.

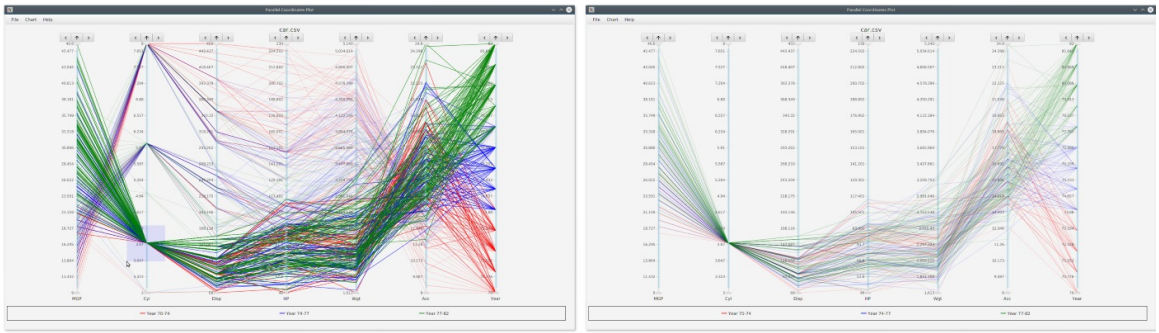

Figure 3.6: (left) User selects a rectangle. (right) Brushing of the chart.

```
parcoordChart.setHighlightStrokeWidth(2.0);
```

Brushing can be done by drawing a rectangle in the chart. By drawing such a rectangle, all the polylines that are inside of this rectangle will be selected. Figure 3.6 demonstrates brushing and how currently hovered lines are highlighted using higher opacity. It is possible to enable/disable brushing by:

```
parcoordChart.enableBrushing();
```

Inverting is another interaction in which the user is able to invert an axis in the ParallelCoordinatesChart. It can be done by clicking on the invert button on top of each axis.

One of the most important interactions in parallel coordinates plot is **Moving** the axis and changing their order. We provide three different interactions for this purpose. First, the arrows above the axes can be used to move them to the left or right. Second, drag and drop functionalities were provided. A user can directly drag an axis onto another axis, which leads to two axes being swapped. Additionally, a user can drop an axis on the space between two other axes. This inserts the dragged axis at the chosen area and moves all other axes accordingly. Figure 3.8 left shows both of these drag and drop interactions.

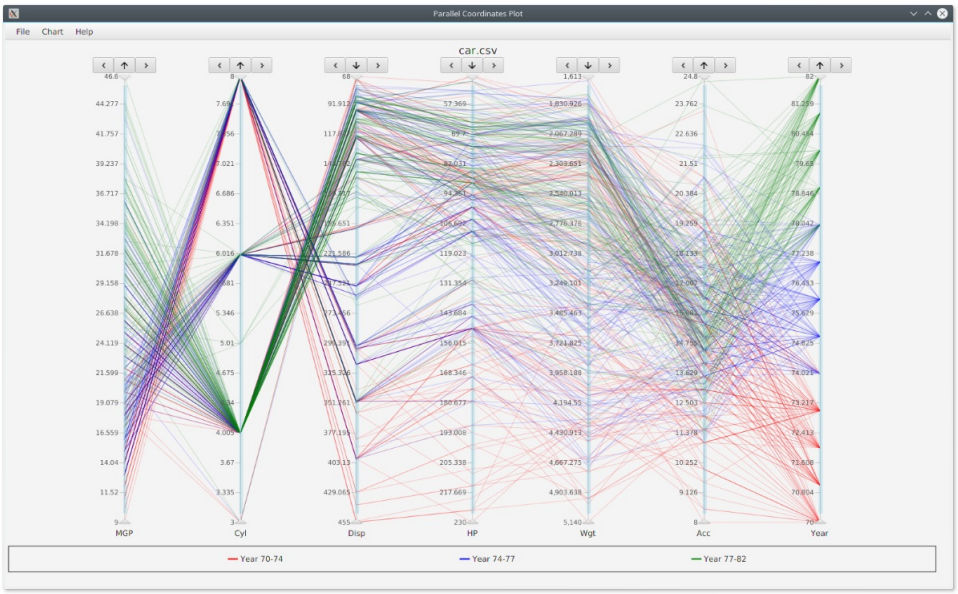

Figure 3.7: Inverting parallel coordinates chart. The three axes in the middle were inverted.

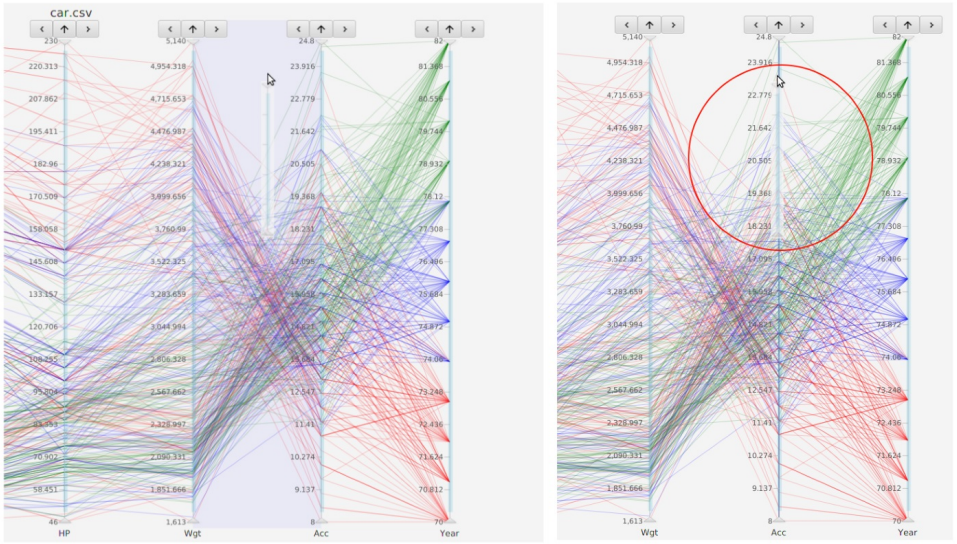

Figure 3.8: (left) Drag and drop of an axis onto space between two other axes. (right) Drag and drop of one axis directly on the other axis.

Chapter 4

Setup

TODO

Chapter 5

Performance

The details behind an efficient JavaFx Chart implementation are often hard to detect and grasp. As an example, even the standard JavaFX LineChart seems to not be very performant when dealing with a huge numer of records (and therefore lines to draw). However, we were able to come up with an approach which is sufficient to provide good performance for roughly 6000 data records. As mentioned in 6, it may still be possible to further increase the performance, especially regarding user interactions.

To elaborate further, it will be shortly explained how the currently existing performance for our ParallelCoordinatesChart is achieved. The main idea behind drawing and detecting interactions with specific lines is a 3-layer approach using the JavaFX Path and Canvas classes. Firstly, each record holds a Path which defines the exact polyline that will be visible on the screen. However, this Path has its opacity set to zero. This stems from the fact that the Path class itself leads to very bad performance, if a line for it has to be drawn on screen. When set invisible by setting its opacity to zero, no performance trade-offs are recognizable. Although this seems like strange behaviour, it is still possible to make use of that. In combination with the Record class this Path holds all the information about the given polyline. Such information is for example the given state of the line (if it is brushed, filtered, or visible as normal) as well as the color (defined by the series it belongs to). It is also used for detecting interactions with the user. In addition to this "interaction layer" defined by Paths, a JavaFX Canvas is used to draw the visual representation of all records. Such a Canvas seems to not influence the performance in negative ways like drawing visible Paths would. It even allows to completely redraw the whole screen in real time without the user noticing (although it might be noticeable with a huge amount of lines and in full screen). The last component of the three-layer approach is another canvas, which is used separately to highlight lines which are currently brushed. For this highlighting, the opacity of the brushed lines is set to a maximum. This three-layer-approach seemed to be the best way to provide good performance.

Although performance was already improved heavily in comparison to early stages of development, the chart could still be made more efficient. Currently, each user interaction leads to a redrawal of the whole chart. Multiple Canvas objects could solve this problem by only visualising specific areas of the chart. For example, if a user currently inverts the last axis, the whole chart will be redrawn. This could be solved by using multiple canvas objects which only redraw the affected areas. For now, it has to be said that the remaining time was not sufficient to implement this feature.

Chapter 6

Future Work and Limitation

One of the limitations of the project is definitely how much automation the library possesses and how much possibilities are left to the users. There will always be a tradeoff between these two prospects. More automation of features limits the user (or programmer) to customize the code. For example, currently axes are handled by the chart. This leads to less control for the programmer when including the ParallelCoordinatesChart in one of his projects. Also, currently our tool used for demonstration still misses some features like supporting non-numeric data and a proper CSV-file importer. Usability of the project is another issue. Currently, the implementation may not be polished completely for efficient usage. It still takes time for the user to understand the mechanism of some advanced features.

To summarize, the following features could still be implemented in the future:

- Categorial data support. Currently, only numeric data are supported.
- Dimension selection.
- More control for the user (or developer) over the library.
- Improve performance for interactions (as described in 5).
- Add a more advanced CSV import module for the demonstration tool.

Chapter 7

Conclusion

In this report, we discuss the importance of using a parallel coordinates plot to represent multivariate datasets. A parallel coordinates plot contains parallel axes for each dimension and polylines that start from the first axes and ends at the last axes for records. Since currently there is no suitable library to support drawing multivariate datasets in Java, we developed a high-dimensional chart library and a parallel coordinates chart. This library is based on current JavaFX Chart classes and contains visualization of parallel coordinates and interactions. Our ParallelCoordinatesChart contains functionalities like inverting, swapping and moving the axes, filtering, as well as brushing. In addition to that, a tool was provided which can be used to demonstrate the basic functionalities of the ParallelCoordinatesChart. All of the source code was published at GitHub under an MIT license.

Bibliography

Andrews, Keith [2012]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. 22nd Oct 2012. http://ftp.iicm.edu/pub/keith/thesis/ (cited on page iii).

Few, Stephen [2006]. "Multivariate analysis using parallel coordinates". *Perceptual edge* [2006], pages 1–9 (cited on page 2).

Giles, Jonathan [2017]. *ControlsFX*. fx experience. 7th Jul 2017. http://fxexperience.com/controlsfx/ (cited on page 5).

Group 5 [2017]. *parcoord-fx*. GitHub. 7th Jul 2017. github.com/ruptho/parcoord-fx (cited on page 5).

Inselberg, Alfred [1985]. "The plane with parallel coordinates". *The visual computer* 1.2 [1985], pages 69–91 (cited on page 2).

Inselberg, Alfred and Bernard Dimsdale [1990]. "Parallel coordinates: A Tool for Visualizing Multidimensional Geometry". In: *IEEE Visualization*. IEEE Comp.Soc. 1990, pages 361–76 (cited on page 2).

Keim, Daniel A. [2002]. "Information visualization and visual data mining". *IEEE Transactions on Visualization and Computer Graphics* 8.1 [2002], pages 1–8. ISSN 1077-2626. doi:10.1109/2945.981847 (cited on page 1).

Li, Jing, Jean-Bernard Martens and Jarke J Van Wijk [2010]. "Judging correlation from scatterplots and parallel coordinate plots". *Information Visualization* 9.1 [2010], pages 13–30 (cited on page 2).

Massachusetts Institute of Technology [1988]. *MIT license*. Open Source Initiative. 1988. opensource.org/licenses/MIT (cited on page 5).

Oracle [2014]. *JavaFX Overview*. Java documentation. 2014. http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm (cited on page 3).

Yi, Ji Soo, Youn Ah Kang, John Stasko and Julie Jacko [2007]. "Toward a deeper understanding of the role of interaction in information visualization." *IEEE transactions on visualization and computer graphics* 13.6 [2007], pages 1224–31. doi:10.1109/TVCG.2007.70515 (cited on page 9).