# Project Report:
# Responisve Visualization

Jannik Hildebrandt, Michelle Perkonigg, Patrick Steyer

Institute of Interactive Systems and Data Science (ISDS),
Graz University of Technology
A-8010 Graz, Austria

27 Jan 2019

**Abstract**

This report focuses on responsive visualizations which are a part of responsive web design. In the following sections you will get an insight in how the project team was able to extend a given framework with much new functionality like hovering, flipping and tooltips. Additionally ideas for future work are given and the hardest problems which occurred explained. Since the hardest problem was time constraint most of those problems can be implemented with enough dedication. Another problem which occurred during the project was that each chart type is handled as a single project and each of those implements the same functions and has the same variable names. This causes that trying to make one big project out of those single projects was not possible. During the work many useful functions of the used D3 library could be found. For example the call functionality was used repeatedly to update axes of the charts while the window got more and more narrow. During this work many improvements to this field could be achieved. The responsive data visualization project is lead by Keith Andrews and you can see the work on the website `https://projects.isds.tugraz.at/respvis/`. Besides scaling visualizations freely, responsive visualization as part of responsive web design has the goal to display data according to the users device and adapt the interaction possibilities to the characteristics of this device.

# Contents

# List of Figures

# List of Tables

# List of Listings

# Chapter 1

# Introduction

The purpose of this report is to give an understanding about responsive visualization and responsive web design. Furthermore it is also meant as a project documentation about the work which has been done in this specific field. In the end recommendations for future work are given.

Responsive Visualization, which is also the focus of this work, is a subtask of responsive web design and has the goal to display data according to the current screen size. [*Responsive Data Visualisation* 2019]

As a starting point a framework for line chart, bar chart, parallel coordinates and scatterplot was given and had to be extended in order to become more responsive. The charts were implemented in JavaScript by using the D3 library and except of updating the library no major changes were made to the framework. Except for the scatterplot all charts could be extended by functionalities like hovering, tapping and tooltips.

The hardest problem which occurred in the project was that all the charts have a similar structure and therefore no feasible way for automating a the build process with gulp was found. Second, implementing a select function for the parallel coordinates chart could not be achieved in the time period of the project.

# Chapter 2

# Theory: Responsive Visualization

This chapter explains what Responsive Web Design is and how Responsive Visualization is related to it. Moreover it gives a short introduction to the JavaScript library D3, which is used for this project.

## 2.1 Responsive Web Design

"Responsive web design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones)" [*HTML Responsive Web Design* 2019]. The first thing worth noting in this definition of responsive web design is the word automatic. First the screen size and screen ratio of the current display has to be detected and then the according actions can be performed. Apart from resizing, hiding, shrinking and enlarging of elements on a website, it is also about rearranging the element in a new layout, which better fits the current screen. Another thing worth noting in the definition is the fact that responsive web design builds up on HTML and CSS. It can also use JavaScript, but important is, that the actions are performed on client-side.

An important guideline for creating responsive websites is to never use pixel values. With these the display strongly depends on the device which is used to view the website. It is better to use 'em' or 'rem' instead, which are units that depend on the current font size.

## 2.2 Responsive Visualization

Responsive visualization is a subtask of responsive web design, which focuses on finding the perfect way to display data with respect to the current screen size [*Responsive Data Visualisation* 2019]. In the case of this project, this is achieved by using D3v5. In the JavaScript code the current screen size and ratio is detected and the layout of the different graphs is then adapted accordingly.

## 2.3 D3.js

D3.js is a JavaScript library which was created to create visualizations of data [*Data-Driven Documents* 2019]. It uses HTML, CSS and SVG. It manipulates the Document Object Model (DOM) of the browser. D3 can be used to create interactive SVGs with smooth transitions.

D3 works with selections. Elements can be selected by tag, class or id. On these selections the actions are performed. An action can for example be to add a style tag or an attribute to each item in the selection.

# Chapter 3

# Project: Responsive Visualization

Within this chapter the goal of the project will be explained. Since the goal of the project is to make an already existing project more responsive, the following section includes the starting basis for this project. Afterwards all applied changes will be listed and explained in more detail.

## 3.1 Starting Basis of Project

The base project consists of a responsive line graph, a responsive bar chart, a responsive parallel coordinates chart and also a responsive scatterplot [*Responsive Data Visualisation* 2019]. The charts were already responsive, but there was still room for improvement. Table 3.1 shows briefly which features were already supported at the beginning of this project. Since the project mainly focuses on the line chart, bar chart and parallel coordinates only these three are listed.

| | narrow mode | interactivity | axis customization |
|---|---|---|---|
| Line chart | switches to sparkline | none | none |
| Bar chart | flips axes | none | none |
| Parallel coordinates | not supported | none | select axes to be shown |

**Table 3.1:** The overview of all features which were already supported in the base project.

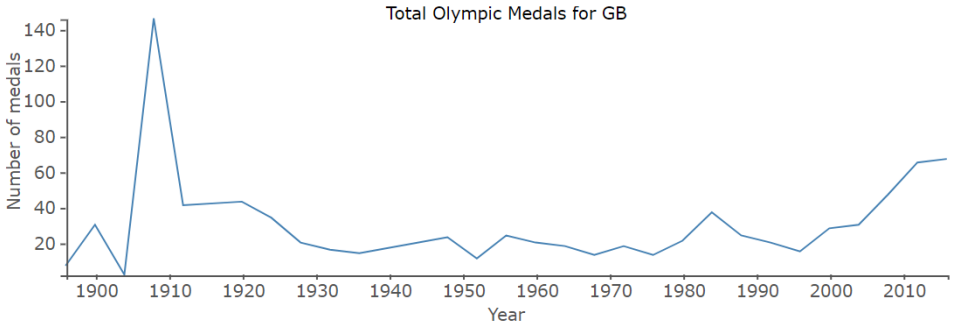
In the following subsections, it is described in more detail which functionalities the charts already had, before the work on the project was started.

### 3.1.1 Line Chart

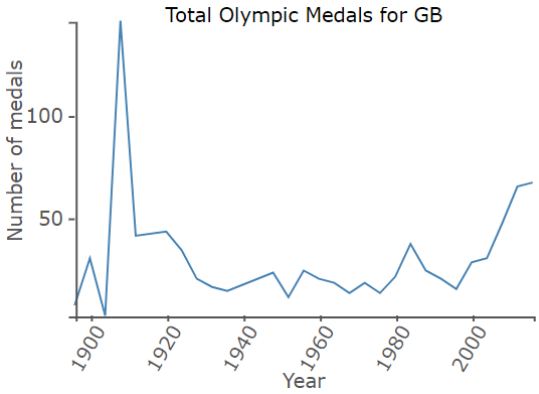
The line chart shows the number of Olympic medals won by Great-Britain over the years. When reducing the screen size, the labels of the x-axis rotate, to prevent cluttering. On a very narrow screen width the axes disappear completely and the graph changes to a sparkline.

### 3.1.2 Bar Chart

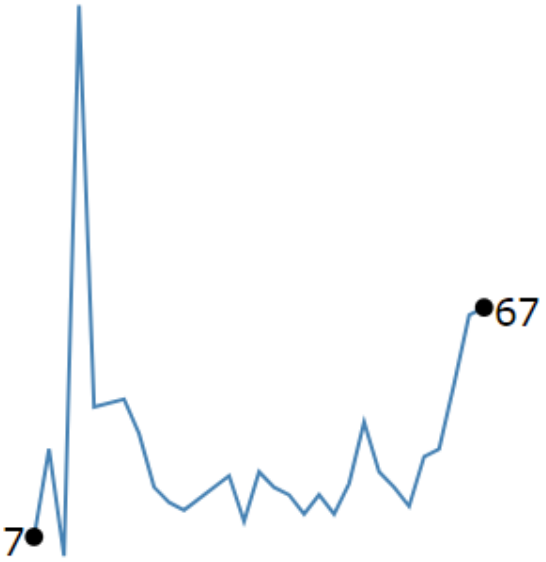
The bar chart has names on the x-axis and undefined values on the y-axis. As in the line chart, the labels of the x-axis rotate, when the screen width is reduced. When the screen width deceeds a certain value, the bar chart is flipped, so instead of vertical bars it now has horizontal bars, which is a much better use of the available space.




**(a)** Line chart form and layout if window size is high.




**(b)** Line chart form and layout if window size is medium sized. The axis label are tilted to adapt to the available space.




**(c)** Line chart form and layout if window size is very narrow changes to a sparkline.

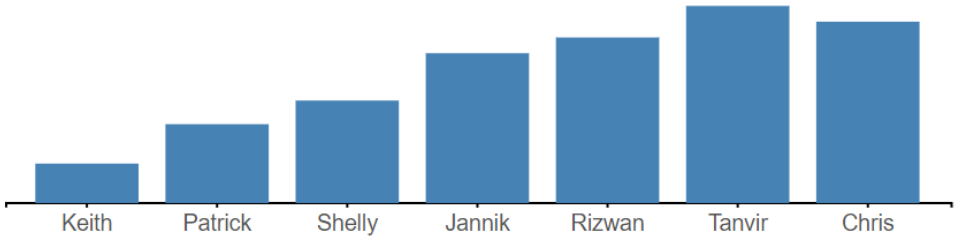
**Figure 3.1:** The three main form and layout types of the line chart according to window size.

### 3.1.3 Parallel Coordinates

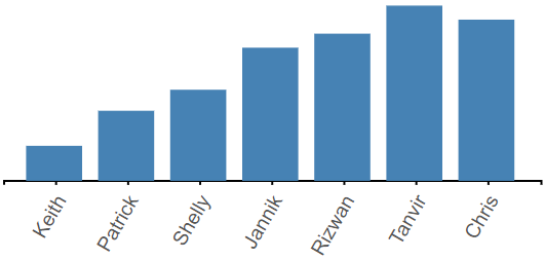
The parallel coordinates chart compares different car models in different categories (y-axes). When the screen width is reduced, the number of displayed axes is automatically reduced, until there are only two axes left in the most narrow view. There is a 'Dimensions' button, which lets the user choose the axes that should be displayed. Once the user has manually selected the categories, he has taken control, and therefore the number of axis is not adapted when changing the window size.

## 3.2 Applied Changes During Project

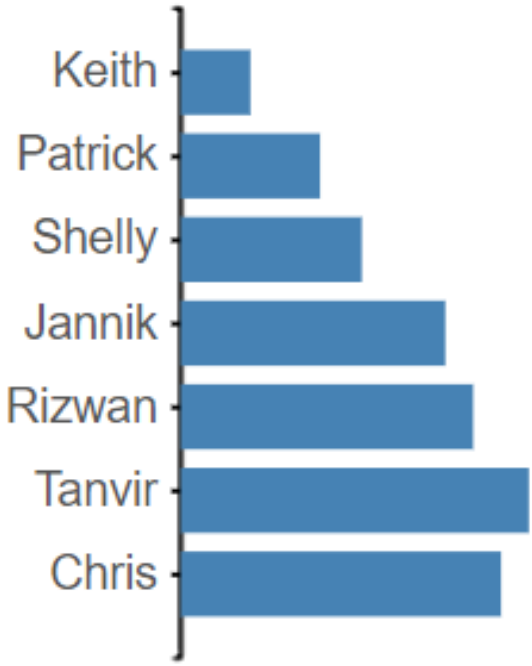
This section is focused on how the existing project was changed, such that it becomes even more responsive. These changes go from the update of the library version over additional interactivity to new views when the window size is very narrow.




**(a)** Bar chart form and layout if window size is high.




**(b)** Bar chart form and layout if window size is medium sized. The axis label are tilted to adapt to the available space.




**(c)** Bar chart form and layout if window size is very narrow flips the axis.

**Figure 3.2:** The three main form and layout types of the bar chart according to window size.

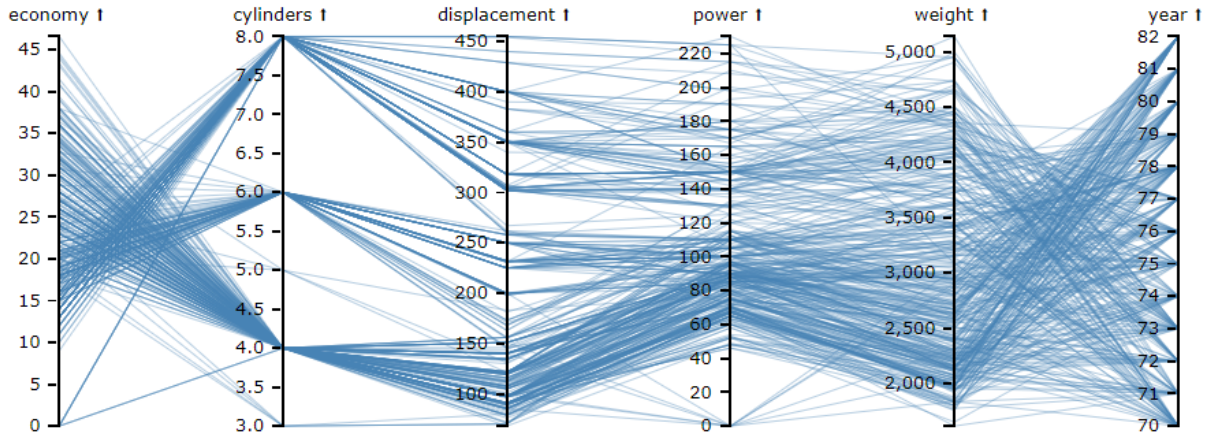



**Figure 3.3:** Parallel Coordinates form and layout if window size is high.

```
d3.select("#chart").selectAll("rect")
    .on('mouseover', function (d, i) {
        //executed when hovering over a bar
    })
    .on('mouseout', function (d, i) {
        //executed when the mouse leaves the bar
    })
    .on('click', function (d, i) {
        //executed when the bar is clicked
    });
```

**Listing 3.1:** The syntax, how the mouseover, mouseout and click functionality is added to the bars of the bar chart.

```
document.getElementById('chart').addEventListener('mousedown', mouseDown, true);
function mouseDown(){
    rgroup.selectAll("rect").transition().duration(100)
        .attr('opacity', '1');
    labels.style("display", "none");
}
```

**Listing 3.2:** The syntax, how an onclick-listener is added to the chart and an example how the selection is cleared in case of the bar chart.

### 3.2.1 Migrate from D3v4 to D3v5

The first step was to update the version of the D3.js library from version 4 to version 5. The only change encountered when migrating to version 5 was the change of how the data is loaded. Now D3v5 uses the Fetch API and promises, while in version 4 XMLHttpRequests and asynchronous callbacks were used [*Changes in D3 5.0* 2019]. These changes were only applied to the responsive scatterplot since only the responsive scatterplot loads the data from a .csv file.

### 3.2.2 Add Interactivity

In order to make the charts more appealing, interactivity was added to all of the charts. This was implemented by including events which are triggered by hovering over an element or by clicking at elements. In D3 hovering has to be split into two parts: mouseover and mouseout. The mouseover function is triggered, when the mouse touches the element, which the function is assigned to. The mouseout function is triggered, when the mouse leaves this element. The functions can be assigned to a D3 selection with a very easy syntax, as can be seen in Listing 3.1.

In addition to this D3 function, an onlick-listener was added to the svg element of the html file. This listener calls a function, if you click anywhere in the image. In this function all selections are cleared. In Listing 3.2 the sample code for adding such an onclick-listener is shown by the example of the bar chart.

In the following subsections it is explained in detail, what has been added to which chart in terms of interaction.

#### 3.2.2.1 Line Chart

In the line chart, circles have been added at the data points in the graph. These circles appear, when the mouse hovers over them. The size of the hover target has been increased using CSS, so that it is easier to hit the circle on a touch device. In addition to the circle, a tooltip is displayed on hover, which displays the data of this data point. When a circle is clicked, this tooltip becomes sticky and stays on the screen.

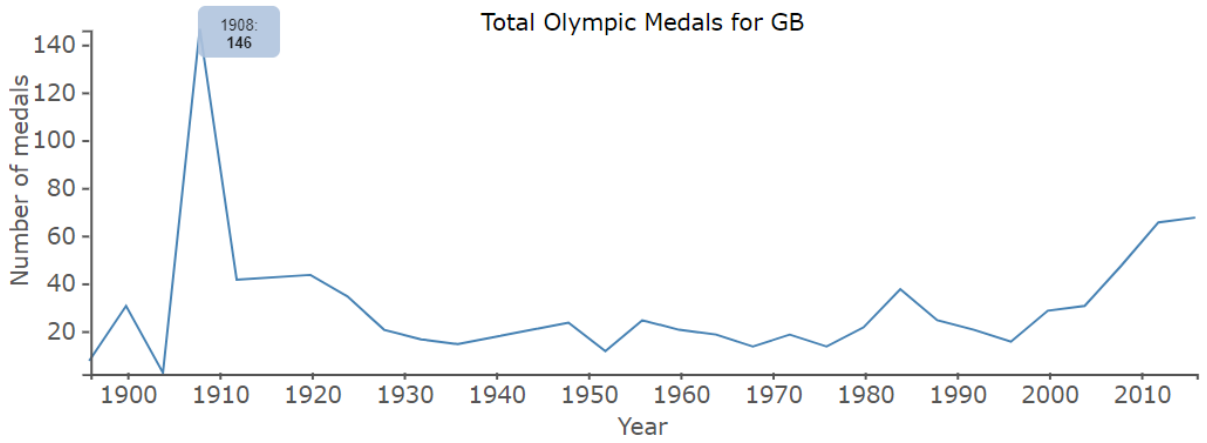



**Figure 3.4:** Line chart with sticky tooltip at datapoint caused by clicking on a datapoint.

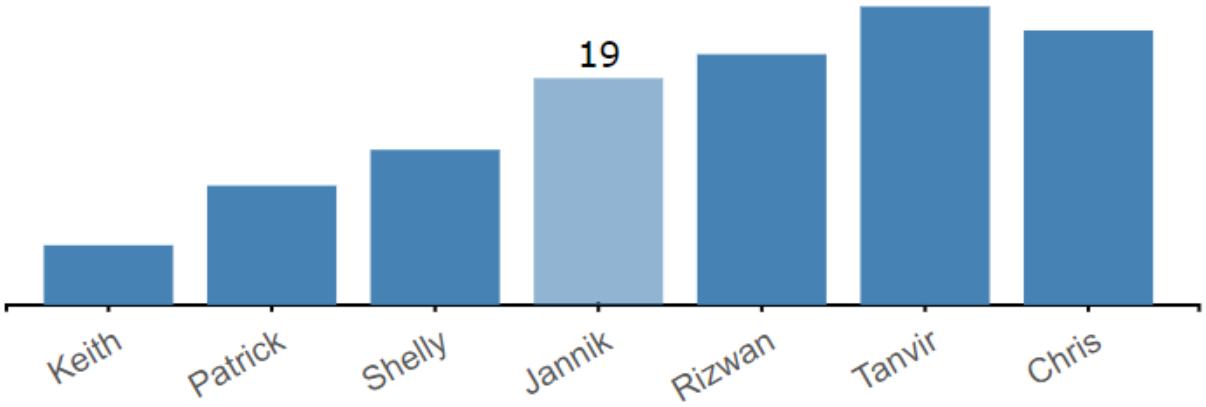



**Figure 3.5:** Bar chart with tooltip. Clicking on a bar creates a sticky tooltip while hovering over a bar does not.

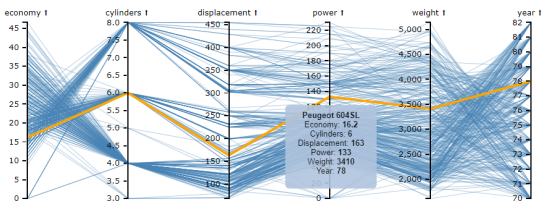
While the tooltip is sticky, the user can still hover over other circles, but no tooltip is shown. The selection can be cleared by clicking anywhere in the chart.

#### 3.2.2.2 Bar Chart

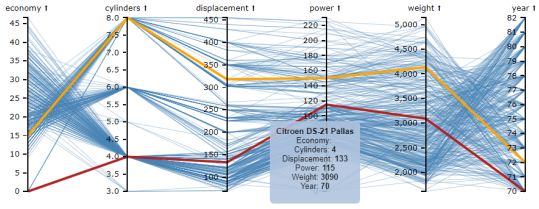
The interactivity of the bar chart is very similar to the one of the line chart. On hover, the opacity of the bar changes and the data value is displayed above the bar (unlike the previous version, where all values were always displayed). when clicking on a bar, this selection becomes sticky, but it is still possible to hover over other bars. The selection is cleared by clicking anywhere in the chart.

#### 3.2.2.3 Parallel Coordinates

For the parallel coordinates chart another tooltip, containing the data of the cars, was added. When hovering over a line, the line-width of the whole line is increased, the colour of the line is changed to orange and the tooltip of the line is displayed. When clicking on the line, the colour changes to red and the tooltip becomes sticky. The orange line still appears on hover for other lines, but no tooltip is displayed. Once again, the selection can be cleared by clicking anywhere in the chart.




**(a)** Parallel coordinates with orange line and tooltip caused by hovering over a line.




**(b)** Parallel coordinates with red line and sticky tooltip caused by click on line. Hovering still colors line to orange but no tooltip is shown while there is a sticky one.

**Figure 3.6:** Behaviour of the parallel coordinates if hovering or clicking on a line.

```
d3.select("#chart")
  .select('.x.axis')
  .transition()
  .duration(400)
  .style("opacity", "0");
```

**Listing 3.3:** An animation for fading out the x-axis of a chart, using functions of D3.

### 3.2.3 Animate Transitions

Animated transitions have been added to the chart, because it feels very satisfying to change the window size and have smooth transitions of changes. These animations will never be seen by the majority of users, because you have to resize the window manually in order to see it. However, they are really cool to those people which are interested in responsive design and that take the time to play around with the window size. As the project is an example project for responsive visualization, the target audience is usually of the latter kind, therefore it was decided to add the animations to the charts.

In general there are two ways, how animations can be added to the charts: by D3 or by CSS. For this project the approach of using built-in D3 functions was chosen, because the effort for including them was small and the functionality was working well. The function '.transition()' can be called before changing an attribute or a style and D3 will interpolate the positions in-between and animate the change of style/ attribute. With the function '.duration()', called directly after the transition, the speed of the transition can be influenced. In Listing 3.3 an example code for smoothly fading out the x-axis can be seen.

The other method for adding animations, which was not chosen for this project, is by CSS [*Animated Media Queries* 2019]. For any CSS element, the property 'transition' can be added. This property takes two inputs, the name of the property which should be transitioned and a time value in seconds. If the selected property changes, the change is animated by CSS automatically. An example code can be seen in Listing 3.4.

The following changes have been animated in our charts:

- Line chart
  - Rotation of the x-axis labels
  - Fade-in/out for tooltip and data point circles
  - Fade-in/-out of axes and labels when changing from/to narrow mode
- Bar chart

```
.thing-that-moves {
  position: absolute;
  width: 10em;
  height: 10em;
  background: red;
  top: 0;
  left: 0;
  transition: left 0.5s;
}
```

**Listing 3.4:** In this case, a change of the property 'left' would be animated with a duration of 0.5 seconds.

  - Rotation of the x-axis labels
  - Rotation of bars and axes when changing to narrow mode
- Parallel Coordinates
  - Rotation of the x-axis labels
  - Fade-in/out for tooltip
  - Rotation of axes and lines when changing to narrow mode

### 3.2.4 Drag and Drop

In order to rearrange the axis, the drag and drop functionality was introduced for the responsive parallel coordinates. At the beginning of the project it was only possible to order the axis as the user wishes by using the "Dimensions"-button and enabling the axis in the desired order. To simplify this and to make the chart more user friendly the drag and drop functionality was introduced. Moving the axis can be done by dragging either the axis itself or the title of the axis to the desired position. When dropping, the axis is automatically rearranged such that the space between all displayed axes is constant.

Listing 3.5 shows how to add the drag and drop functionality to the dimensions and furthermore how to handle the three phases of drag and drop which were

1. Start
2. Drag
3. End

In the start-phase the initial position of the axis to be dragged is saved. In the drag-phase the chart is updated such that it can be seen were the axis is dragged to. To mention a flag here, dragOnlyAxis, was introduced to enable or disable whether only the axis or also the lines will be updated while dragging. Finally in the end-phase the new position of the axis will be calculated. The drag and drop functionality works at every window size.

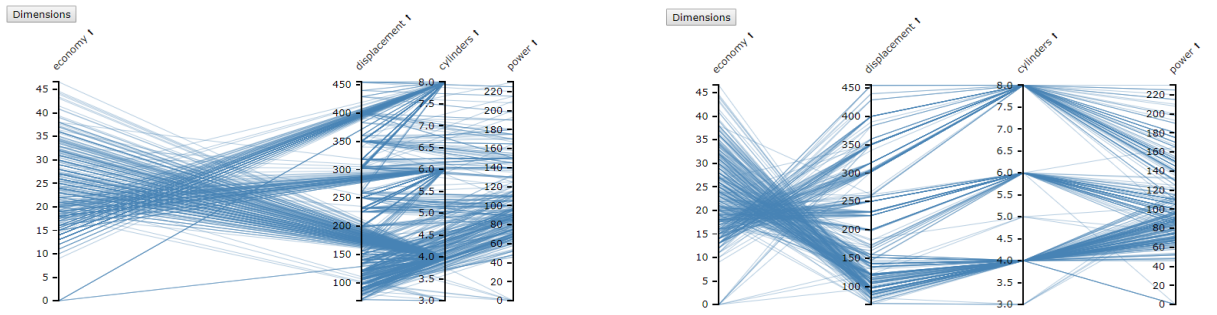
Figure 3.7 shows the process of dragging the axis with the title "cylinders" between the axis "displacement" and "power". While Figure 3.7a demonstrates how it looks like during dragging, Figure 3.7b illustrates the parallel coordinates after the position switch triggered by the drag and drop action.

### 3.2.5 Invert Axes

This section will give a short overview how the inverting of the axis in the parallel coordinates chart was implemented.

```
svg.selectAll(".dimension")
    .each(function(d) {
        d3.select(this).call(d3.drag()
            .on("start", function() {
              //executed first time the axis got selected (start of dragging)
            })
            .on("drag", function(d) {
              //executed during dragging the axis
            })
            .on("end", function(d) {
              //executed when dropping the axis
            })
        )
    });
```

**Listing 3.5:** The syntax, how the drag and drop functionality is added to the axis (including the title) and how to handle the three phases of drag and drop




**(a)** Behaviour when dragging an axis to another poistion.

**(b)** Behaviour when dropping an axis at a specific poistion reorders the axis accordingly.

**Figure 3.7:** Behaviour of the parallel coordinates if reordering axis via drag and drop is executed.

First a new list was introduced which is filled with the ids of the dimensions which should be displayed inverted. The id is saved into this list when the title or the invert icon is clicked and also the id is removed if the same axis is inverted again. To access the click event, the onclick function of the title element was extended accordingly.

Then only the updating of the view needed to be done, which is easily achievable by calling the D3 "call" function, which can be seen in Listing 3.6.

### 3.2.6 Flip Axes

For really narrow screen width, the old version of the project just displayed two axes and moved them closer together. As there is more height than width available, it made sense to flip the graph by 90 degrees in narrow mode. In order to implement this, the x-axis has to be scaled according to the height of the window and all y-axis have to be scaled according to the screen width. This also leads to the nice effect, that the width of the chart in narrow mode scales with the width of the window. The axis labels have been moved to the right side of the axes with a 45 degree rotation, because this saves width but still prevents cluttering. On the left side,the problem occurred, that long axis names were hidden behind the dimensions-button, but this problem disappeared when moving the labels to the right. Cluttering was also reduced by reducing the number of ticks in the narrow view.

The hardest part of the parallel coordinates chart was to make all the features work together. It has to

```
svg.selectAll(".axis")
        .each(function(d) {
            d3.select(this).call(axis.scale(y[d]).ticks().tickSize("1.5").
                tickPadding("1.5"));
    })
```

**Listing 3.6:** Updating the axes with the D3 "call" function.

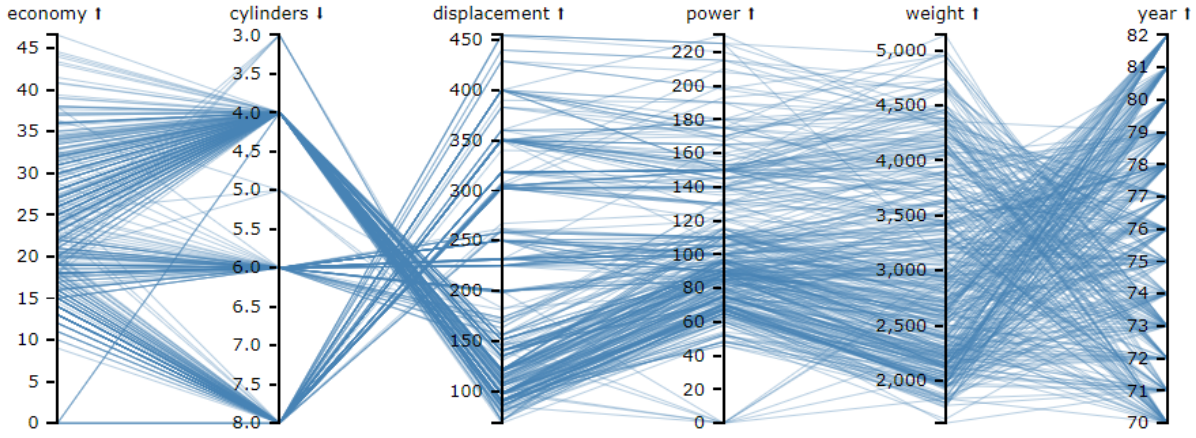
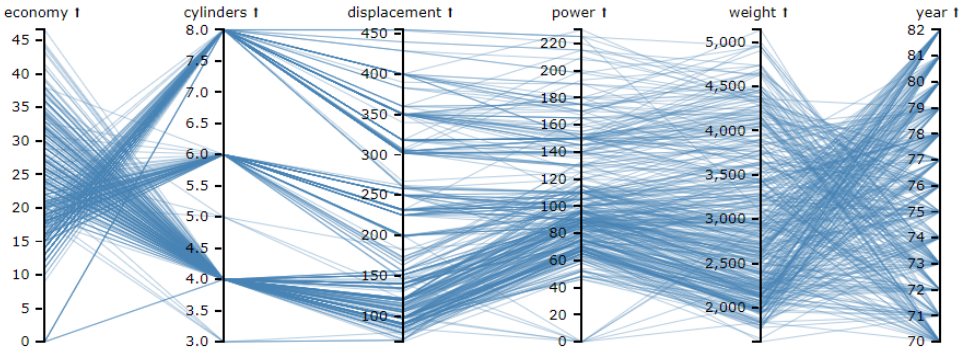
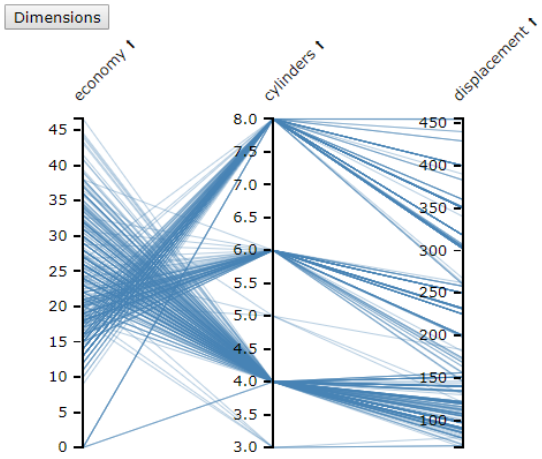



**Figure 3.8:** Inverting the axis changes also the direction of the arrow right beside the axis title.

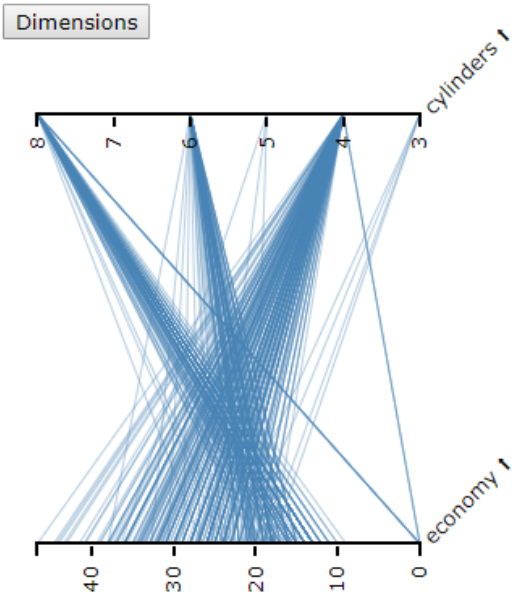
be possible to drag and drop axes in narrow mode, or to drag an inverted axis, or to make selections in narrow mode after axes have been added, inverted and moved. All combinations have to work together, have to be implemented accordingly and have to be tested thoroughly. In the final version all these things work together perfectly.




**(a)** Parallel Coordinates form and layout if window size is high.




**(b)** Parallel Coordinates form and layout if window size is medium sized. The axis label are tilted to adapt to the available space and also the number of dimensions is reduced if not selected manually.




**(c)** Parallel Coordinates form and layout if window size is very narrow flips the axis.

**Figure 3.9:** The three main form and layout types of the parallel coordinates according to window size.

# Chapter 4

# Future Work

The first things which should be done in the future are to migrate the whole project to typescript and to refactor the code heavily in this progress. Currently the hardest problem is that all the different charts are copy pasted and then extended. By refactoring this code into one large project it would be possible to get rid of duplicate code. Probably a base class and then inheriting all the variables and base functions would be a good idea.

Things which need to be taken into account are definitely that all the charts need the resize event of the browser window and if you insert more than one chart into a web page only one resize function will get triggered, therefore a workaround for this needs to be found.

Functionalities like zooming and panning would improve the user experience and interaction possibilities.

Furthermore some functionalities for especially the responsive parallel coordinates could be very interesting to implement like a filter and a select function for the axis. Besides this, the data should be removed from the .js file and inserted into a separate file (e.g. a .csv file) and then loaded accordingly.

## 4.1 Approaches and Findings

During the project different additional extensions for the project were tried out, but due to a lack of time or also the high number of problems caused by this the extensions were currently not supported. This section includes a documentation of the tried approaches and findings during this approaches.

### 4.1.1 Gulp

Gulp is software which purpose it is to automate processes especially in the web development.

It is easily possible to generate single CSS, HTML or JS files from multiple sources. If you want your web page to have an as small size as possible you can just use packages like 'gulp-htmlmin' and 'gulp-uglify'.

**Problems:**

The hardest problem which occurred and why it was decided to not pursue this matter any more, was that while creating the single JS file the problem of namespaces occurred. Since all the files have the same variables and functions just concatenating the files was not doing any good. Therefore gulp should be used to insert namespaces to the content of the files. Sadly no suitable package was found which could do this reliably.

A possible solution, which a colleague mentioned, after the project was concluded, was to just insert curly braces at the start and the end of each file such that all those variables and functions which are named the same are in separate scopes. Theoretically this approach could do the trick, but without testing

```
svg.selectAll(".axis")
  .call(d3.brushY()
      .extent([[-10, margin.bottom], [10, height ]])
      .on("end", function(a) {
          //executed when drawing box over axis to select range is finished
      })
  );
```

**Listing 4.1:** The syntax, how the select functionality is added to the axis and how to handle the phases of select

this thoroughly it is not sure if it would not break some of the code parts where the responsiveness is achieved.

### 4.1.2 Select

Select was meant to be a part of parallel coordinates. It describes the possibility to select a sub range of the axis to be shown by for example drawing a box over the axis. D3 offer the brush, brushX and brushY function, where a box can be drawn in a specified area either in all directions, or horizontally with a predefined height or vertically with a predefined width. For this approach the d3.brushY() function was used because the box should be drawn along the axis with a predefined width. Listing 4.1 shows the general syntax on how to call the brushY function on the axis. The .extend part describes the limitations of the box to be drawn (maximum and minimum of height and size of width).

**Problems:** The problems encountered were that this functionality disabled all the other already working functionalities for the axis like invert of the axis and drag and drop. Furthermore the problem to get the right range for the selected axis encountered. It also must be thought about the combination of drag and drop and select since both of them work on the exact same area of the chart.

# Bibliography

*Animated Media Queries* [2019]. 17 Jun 2019. css-tricks.com/animated-media-queries/ (cited on page 10).

*Changes in D3 5.0* [2019]. 17 Jun 2019. https://github.com/d3/d3/blob/master/CHANGES.md (cited on page 8).

*Data-Driven Documents* [2019]. 23 Jun 2019. d3js.org (cited on page 3).

*HTML Responsive Web Design* [2019]. 29 Jun 2019. w3schools.com/html/html_responsive.asp (cited on page 3).

*Responsive Data Visualisation* [2019]. 26 Jun 2019. https://projects.isds.tugraz.at/respvis/ (cited on pages 1, 3, 5).