Responsive Parallel Coordinates

Peter Grassberger, Lukas Burgstaller, Hussain Hussain, Nikita Lvov

30 June 2018

Abstract

We upgraded an existing parallel coordinates visualization library based on D3 to improve responsiveness and add gestures for devices with touchscreens.

Contents

Contents **i**

List of Figures **iii**

List of Tables **v**

List of Listings **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Parallel Coordinates 1

1.3 Inspiration 2

2 Developer Guide **5**

2.1 Overview 5

2.2 Options 6

2.3 Filtering specification 6

2.4 Axes Specification 7

2.5 Usage Example 8

2.6 Device support and issues on iOS 8

3 User Interactions **11**

3.1 Inverting an Axis 12

3.2 Reordering Axes 12

3.3 Filtering on Axes 13

3.4 Angular Filtering 15

3.5 Selecting Displayed Axes 15

3.6 Bounding Box 16

4 Challenges and Possible Improvements **19**

Bibliography **21**

List of Figures

1.1 Example of Parallel Coordinates plot. 2
1.2 Single axis brushing and angular brushing. 3
1.3 Two axis brushing. 3

2.1 The flow of the web page (interactions and plotting). 5

3.1 Inverting an axis 12
3.2 Reordering axes by drag-drop operations. 13
3.3 Filtering on one axis 14
3.4 Filtering on two axes 14
3.5 Angular filtering 15
3.6 Selecting displayed axes 16
3.7 Bounding Box 17

4.1 The flow of the web page (interactions and plotting). 19

List of Tables

3.1 Gestures with number of touch points and touch types. 11

List of Listings

2.1 Default values of the options object 6
2.2 Filtering specification object 7
2.3 Axes specification object 7
2.4 Example of plugging data in and controlling some options 8

Chapter 1

Introduction

1.1 Motivation

In the first research survey (DataVis on Mobile) we took a look on the responsive design. We figured out that each platform have its own peculiar properties that have to be accounted for during the development process. Therefore this survey is a logical continuation of the first paper.

1.2 Parallel Coordinates

Nowadays the amount of processed data is massive. Users have to look for the new instruments and tools to deal with it. However not all tools are comfortable to use. With the growing rate of the mobile devices, the adaptive of tools to be used on multiple platforms, have to be placed on the top of the problems stack.

One of the methods to visualize the comparison of multiple sources of the data is the Parallel Coordinates. In a Parallel Coordinates Plot, each variable is given its own axis and all the axes are placed in parallel to each other and are equidistant. Each axis can have a different scale, as each variable works off a different unit of measurement, or all the axes can be normalized to keep the scales uniform. Values are plotted as a series of lines that connected across all the axes. This means that each line is a collection of points placed on each axis, that have all been connected together. Objects which are very similar will have polylines which follow each other. Andrews 2012

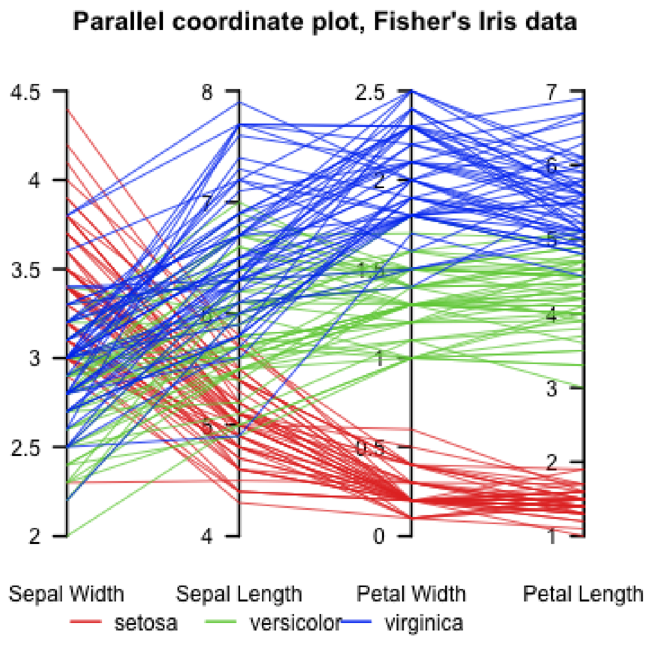
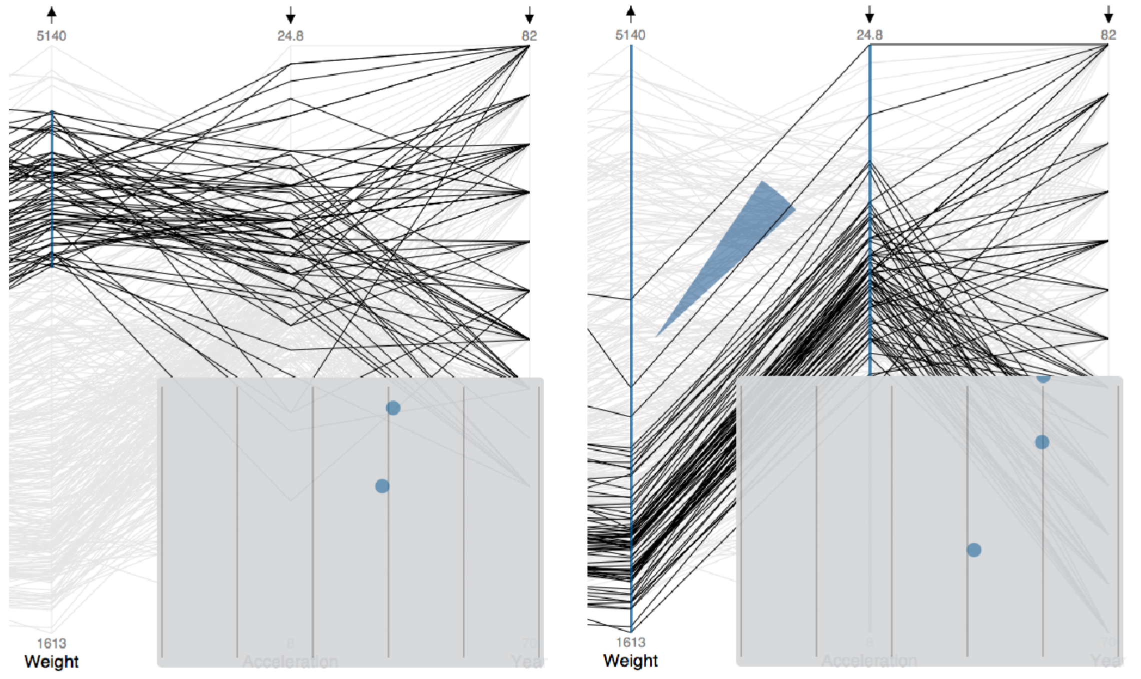


Figure 1.1: Example of Parallel Coordinates plot.
"Parallel coordinate plot of Fisher Iris data" 2011

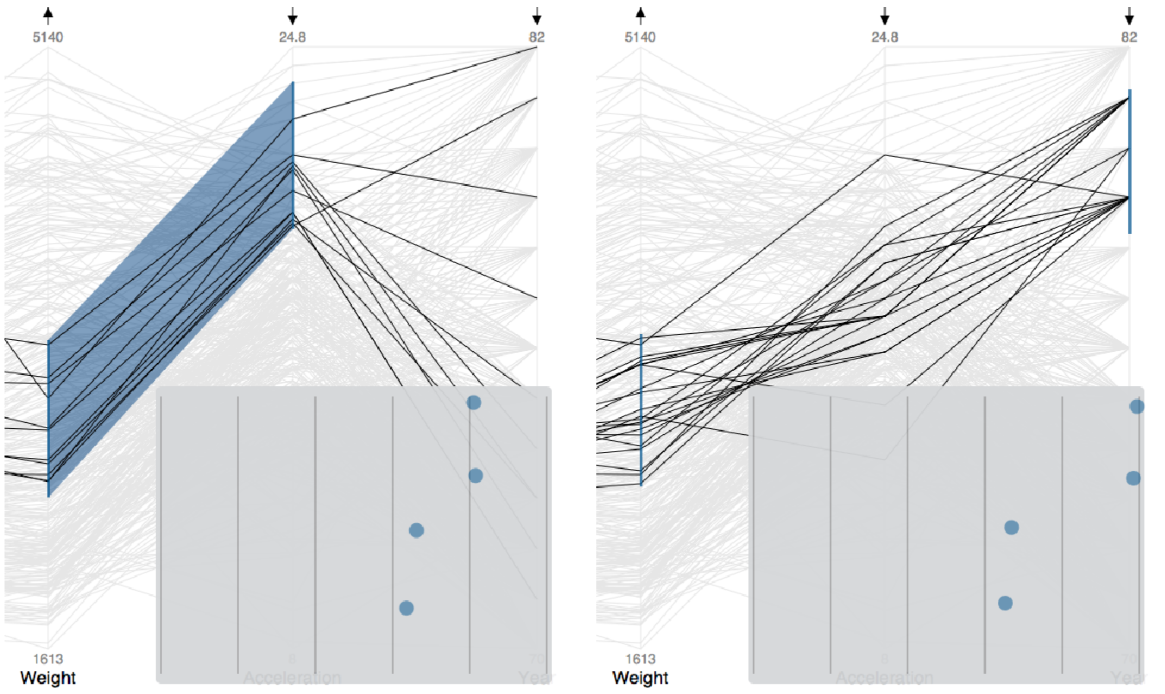
1.3 Inspiration

In this work, we implement a parallel coordinates visualization with the ability to plug in data and edit appearance options. The main multi-touch interactions are inspired by Kosara 2011, namely the filtering on axes (Figure 1.3) and the angular brushing (Figure 1.2).



(a) Two fingers brushing on a single axis. (b) Angular brushing on adjacent axes using three fingers.

Figure 1.2: Single axis brushing and angular brushing.
Kosara 2011



(a) Brushing adjacent axes selects lines in the trapezoid. (b) Four fingers independently brush two non-adjacent axes.

Figure 1.3: Two axis brushing.
Kosara 2011

Chapter 2

Developer Guide

2.1 Overview

The main idea of the project is to plug in your data (JSON or CSV format), to view it in Parallel Coordinates visualization with the ability to interact with it on touch devices as shown in 3. Development takes place on GitHub. Grassberger et al. 2018a There is also Vega-like options to control some parameters of the visualization such as the distance between adjacent axes, the stroke of the axes, etc. When not setting any options the default parameters apply which are set by us.

Figure 2.1 shows a flow chart of how the project interacts and re-plots data. Plotting again causes a performance overhead on some devices. This is only necessary because adding new objects to the chart may cover the labels. A solution for this problem will improve performance and therefore usability a lot.

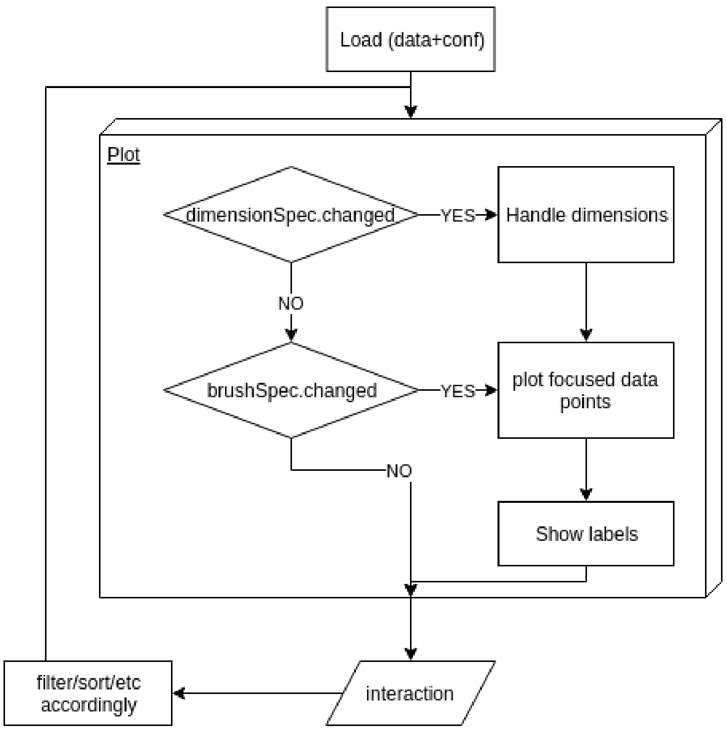


Figure 2.1: The flow of the web page (interactions and plotting).

2.2 Options

Vega-like options included in this project as a JSON object provides the ability of specifying some parameters of the chart, adjusting it to the data or to the target devices. The set parameters in the calling snippet will override some predefined default values, and the other values will stay the same.

Listing 2.1 shows the default values and also explains them.

```
const optionsDefault = {
  svgSelector: '#chart',
  minSegmentSize: 50,
  breakpoint1: '35em',
  breakpoint2: '50em',
  margin: {
    top: 20,
    right: 30,
    bottom: 5,
    left: 2
  },
  dy: 2, // displacement for top margin
  dW: 0.5, // stroke width and displacement
  titley: 0,  //displacement for axes titles
  titleStep: -1.5,
  angStep: -45,  // rotation step
  ignoreDimensions: []
};
```

Listing 2.1: Default values of the options object

2.3 Filtering specification

The chart is intended to support two types of filtering on data (or none at all):

- Filter with ranges on axes

  For each dimension there is two values (`from` and `to`). The filtering is done by selecting data points which satisfy all these ranges for all specified dimensions. This is encoded as shown in listing 2.2 in the array `ranges` (it is a map from the dimension name to the range which is an array of two values i.e. `from` and `to`).

  The interactions for such filtering are included in chapter 3. This fitlering is activated when `filterSpec.type == FILTER_ON_AXES`.

- Filtering by angular brushing

  In angular brushing, two dimensions are specified (in order) and also two slopes. The filtering is done by choosing the data points that have a slope, when considering their value changes from the first dimension to the second one, falling into the range of the two selected slopes. This type of brushing is done with three fingers as also is described in 3. This fitlering is activated when `filterSpec.type == ANGULAR`.

Listing 2.2 shows the fields for the filtering specifications. In order for the change of this object to be reflected on the chart, the field `filterSpec.changed` should be set to true and then the method `plot()` should be called. This is how selected or not selected lines are only re-plotted when the filterSpec changes, in an effort to improved performance. As a future improvement the `filterSpec` object could be exported as a specification of what criteria are selected at the moment as it does encode that; the specifications for the angular brushing may be not very convenient though.

```
var filterSpec = {
type: NONE,
range: [],
angular: {
  dim1: undefined,
  dim2: undefined,
  frm: undefined,
  to: undefined,
  ref: undefined
},
changed : true
};
```

Listing 2.2: Filtering specification object

2.4 Axes Specification

The `dimensionSpec` object in listing 2.3 defines which axes are shown, which are inverted, and whether the number of dimensions should be responsive to the changes in screen size. On changing screen sizes dimensions are hidden automatically, these dimensions can be re-enabled manually, when doing so automatic hiding and showing of dimensions is disabled with `dimensionSpec.hard`.

```
var dimensionSpec = {
/* Nonresponsive */
hard : false,
/* set to true after the object is changed to reflect it on the chart
   */
changed : true,
/* a set of the names of shown dimensions */
selectedDimensions : [],
/* a set of the names of inverted dimensions */
inverted : []
};
```

Listing 2.3: Axes specification object

2.5 Usage Example

In listing 2.4, we show an example of loading a cars dataset, ignoring the dimension 'economy'. Data can be loaded from different file formats that are supported by D3, loading from CSV or JSON files can be seen in the example. The example shows D3 using Promises with the `await` operator which are newly supported in D3 version 5. Promises and D3 version 5 are not required, version 4 previous methods of data loading are also supported.

```
// const csvDataPath = 'data/cars.csv';
const jsonDataPath = 'data/cars.json';

const options = { // set the options object
  // none, means that defaults are used.
  ignoreDimensions: [
    'economy'
  ]
};

async function init() {
  //const data = await d3.csv(csvDataPath);
  const data = await d3.json(jsonDataPath); // load data
  respParcoords(data, options); // call main method
}

// wait until website is loaded
window.addEventListener('load', function() {
  init();
});
```

Listing 2.4: Example of plugging data in and controlling some options

2.6 Device support and issues on iOS

We implemented a prototype for testing multi-touch on a range of different devices. Concluding that multi-touch gestures are not usable in browsers on iOS devices.

Four and five finger gestures are used by the operating system to switch between apps. This is not overridable in Javascript. Touching with four and five fingers works, but as soon as the user moves the fingers the gesture (in Javascript) is canceled and the operating system takes the gesture over.

Mobile Safari and the native web view do not support preventing scrolling via Javascript since iOS 10. This is explicitly mentioned as removed in the iOS changelog. It is still possible to disable scrolling on the underlying scrollview, but this is only accessible if the webview is embedded in your own native application (and scrolling has to be disabled from within the native code, not javascript).

Mobile Safari also uses the zoom gesture to switch between browser tabs (but only on the iPad, not on the iPhone), Google Chrome uses scrolling down as a gesture to create new tabs (only on the iPhone, not on the iPad), iCab Mobile uses left / right scrolling as a way to switch between tabs. All of these gestures happen on browser-level, not page-level and are therefore not overrideable by Javascript.

Support on other devices like desktops PCs, Andriod tablets and phones is give as far as we tested. We didn't test newer state of the art versions of Andriod devices as we didn't have access to these devices. Some

CSS level disabling of scrolling like `touch-action: none;` was necessary to make the visualisation gesture interaction usable.

Chapter 3

User Interactions

In this chapter we will explain how an end user would use the library and what options are available. It also gives an overview of features that are currently available, like the gestures shown in table 3.1. An example can be viewed on the GitHub Pages Site. Grassberger et al. 2018b

Gesture	Touch Points	Touch Types
Inverting an Axis	1	Click
Reordering Axes	1	Drag
Filtering on Axes	2 or 4	Click, Drag
Angular Filtering	3	Click, Drag
Selecting Displayed Axes	1	Click
Bounding Box	1	Drag

Table 3.1: Gestures with number of touch points and touch types.

3.1 Inverting an Axis

Tapping one of the up or down arrows above one of the axes inverts the axis. If the axis went from 0 to 100 before, it'll now go from 100 to 0. This can be helpful when filtering on multiple axes.

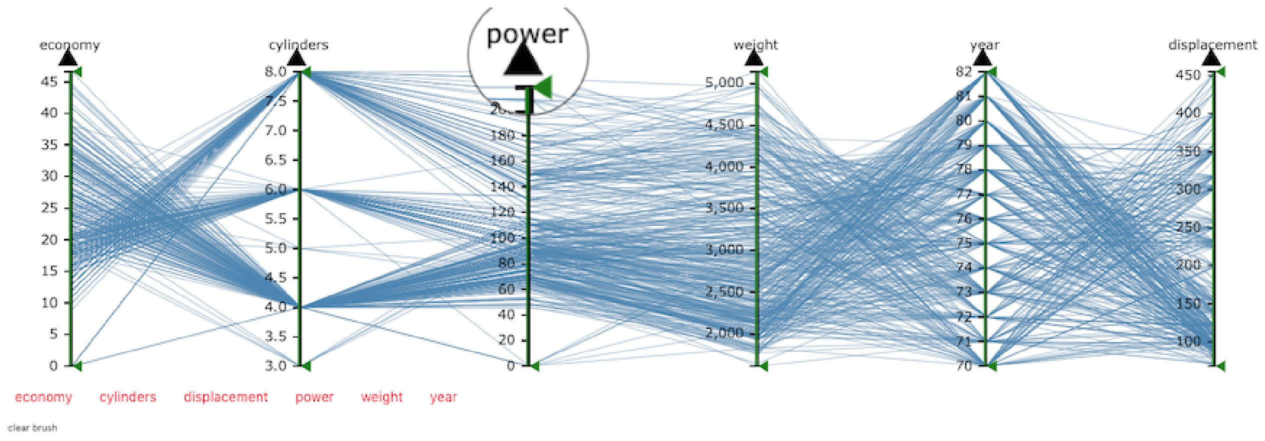


Figure 3.1: Tapping the up/down arrow inverts the axis.

3.2 Reordering Axes

It is possible to reorder the displayed axes via drag and drop by dragging the axis title to the desired position. After such interaction, the specified selection of the data points is still active as shown in figure 3.2

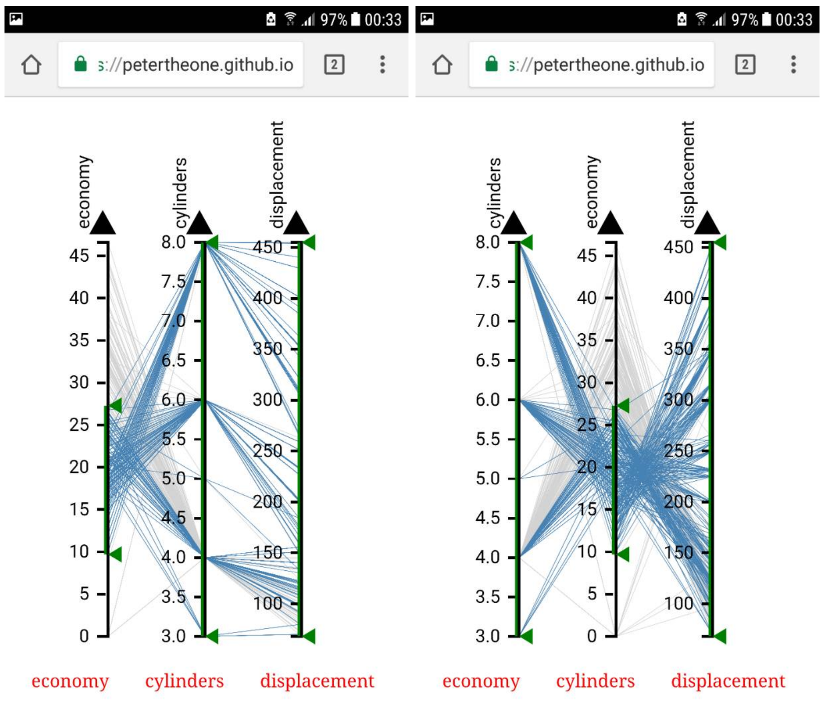


Figure 3.2: Reordering axes by drag-drop operations.

3.3 Filtering on Axes

When using two fingers on one axis, the user can select a range. Only elements passing through this range on this axis will be highlighted (figure 3.3). Green arrows (triangles) indicate the start and end points of the selected range. A green line between the arrows visually reinforces what range is selected. Without selection these arrows are at the top and lower most position on the axes, indicating that the full range is selected.

It is also possible to use four fingers on two axes to select two ranges, only elements passing through the selected range on both axis will be highlighted (figure 3.4). In future work this could be expanded to allow for selection on an arbitrary number axes.

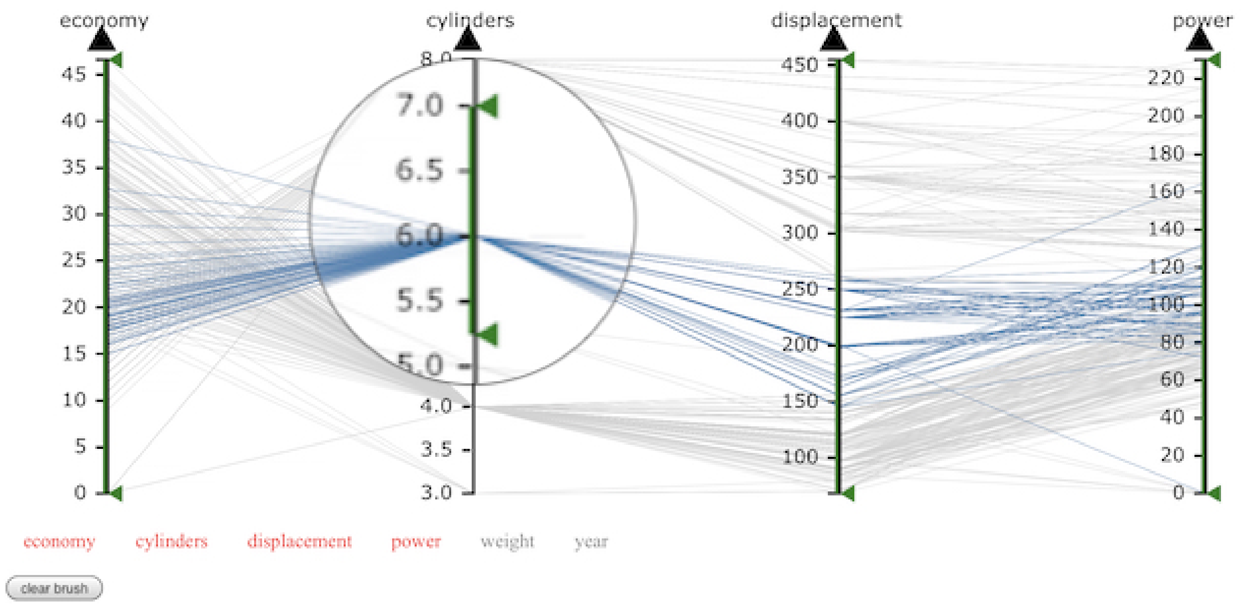


Figure 3.3: Using two fingers you can select a range on one axis

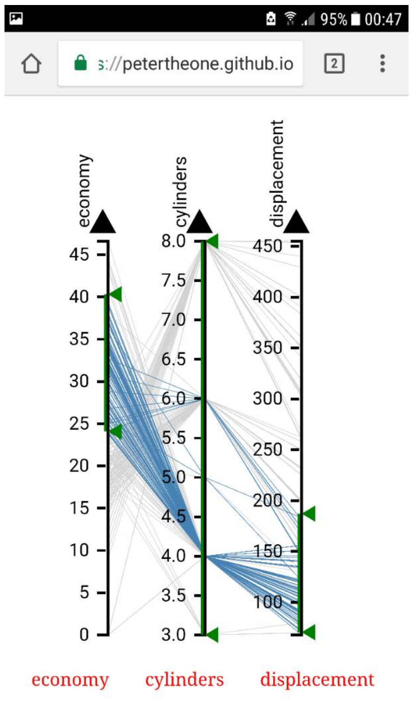


Figure 3.4: Using four fingers you can select two ranges on two axes

3.4 Angular Filtering

Angular Filtering (or Brushing) was first described in Hauser et al. 2002. When using three fingers the user can filter for element within a certain angular range. All elements whose angle between the axes is within the selected range are highlighted, regardless of their vertical position. The current version doesn't allow for axes filtering and angular filtering at the same time.

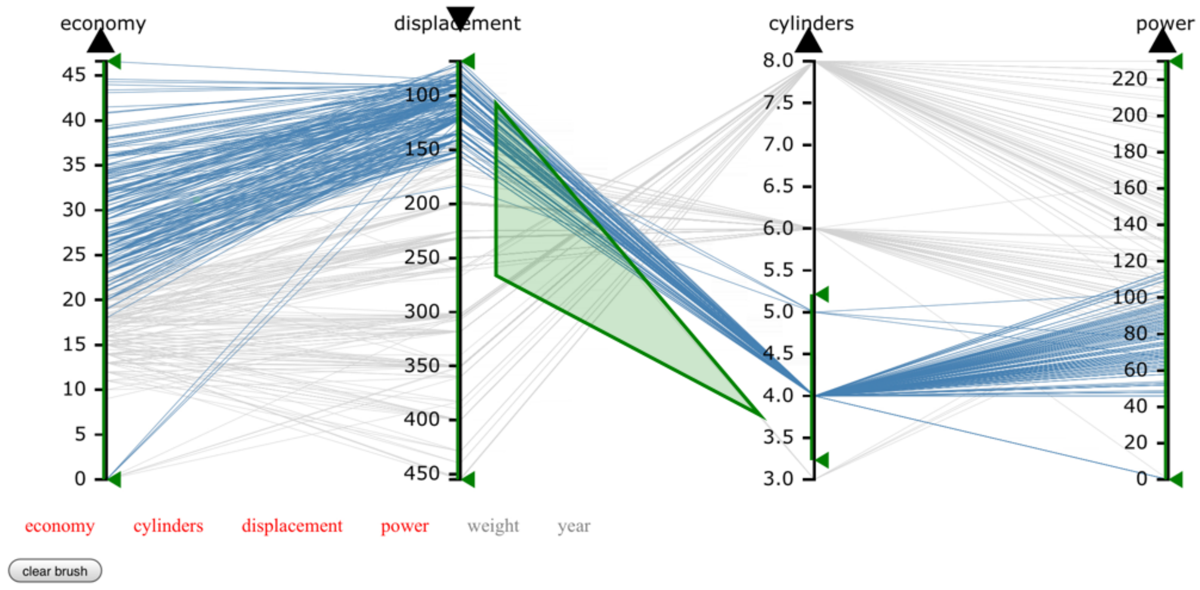


Figure 3.5: Elements with an angle within the selected range are highlighted.

3.5 Selecting Displayed Axes

Axes are automatically turned on and off based on the screen size, but the user can also manually toggle individual axes on and off by tapping the buttons below the graph. When doing so automatic showing and hiding of axis on screen re-sizing is disabled, to respect the intentions of the user.

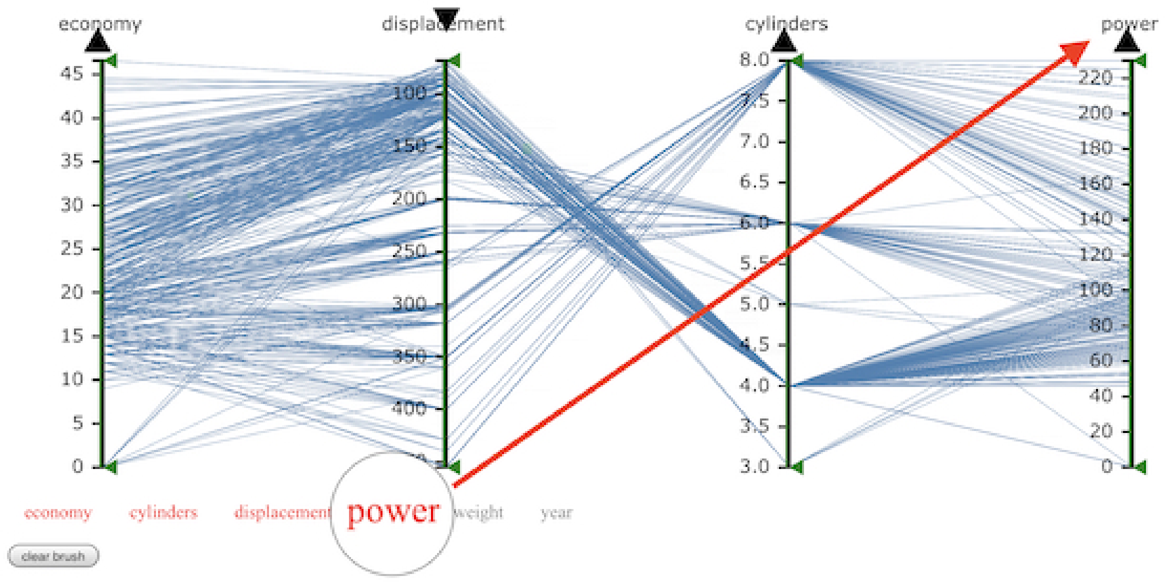


Figure 3.6: The buttons at the bottom toggle axes on and off.

3.6 Bounding Box

Swiping with one finger diagonally downward and left creates a bounding box that selects multiple ranges across multiple axes. The selected data points are the ones that have for each specified dimension, the value within the corresponding range.

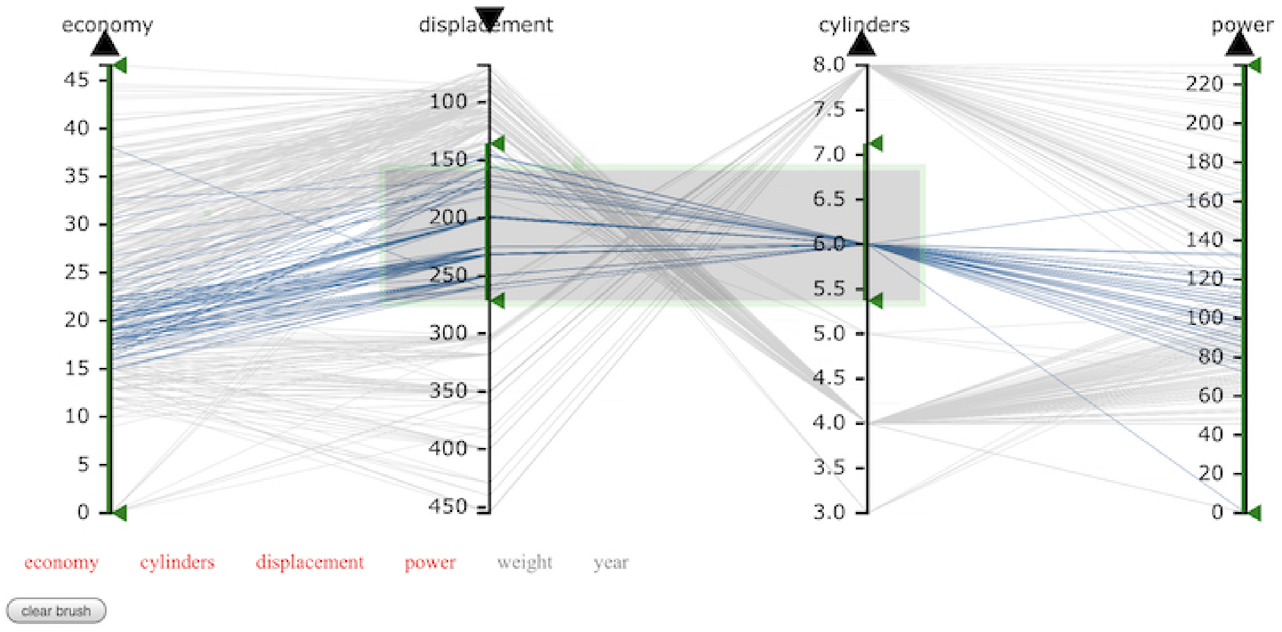


Figure 3.7: Selection via a bounding box.

Chapter 4

Challenges and Possible Improvements

A major issue is that some gestures are not available on iOS browsers as discussed in 2.6. Unfortunately, this cannot be overcome on iOS browsers, so an alternative would be to make a native app with a webView. Since that was not within the scope of the project, it is left for further improvements.

Another big problem is the bad efficiency while interacting on some small touch devices. It comes from regenerating the chart every time an interaction is made. This causes an annoying delay as well as leaving the trace of an old bounding box behind as shown in figure 4.1.

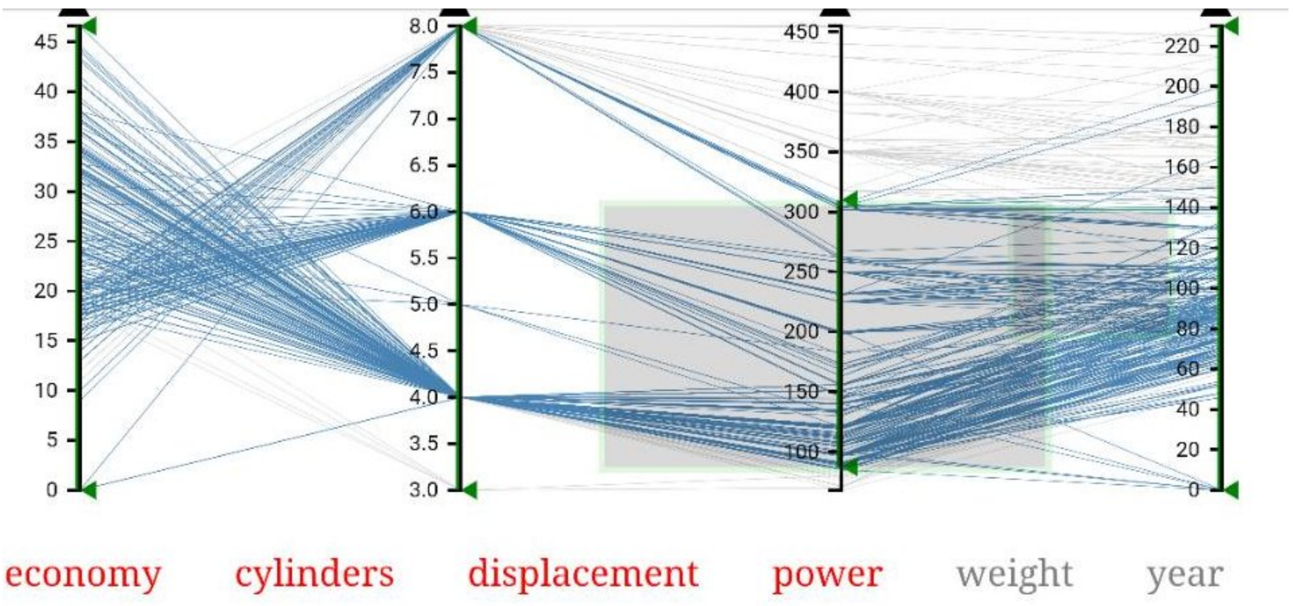


Figure 4.1: The flow of the web page (interactions and plotting).

A very nice improvement would be to add slight animations (less than 1 second) when interacting with the axes. For example when inverting an axis, or swapping two axes. This idea provides better understandability for the user of the connection between the shape before and after the interaction Robertson et al. 1991.

Finally there is still no functionality to retrieve selected data from our resp-parcoords library itself. Sites including our library could show the selected that in the form of a list or table to their users or could use it in another processing step. Currently the selected data is only preserved as SVG path objects in the `focused` variable, but it could be easily be changed to save the raw selected data and making that data available via a JavaScript function.

Bibliography

Andrews, Keith [2012]. *Information Visualisation: Lecture Notes*. 2012. http://courses.iicm.tugraz.at/ivis/ivis.pdf (cited on page 1).

Grassberger, Burgstaller, Hussain and Lvov [2018a]. *resp-parcoord*. 2018. https://github.com/PeterTheOne/resp-parcoord (cited on page 5).

Grassberger, Burgstaller, Hussain and Lvov [2018b]. *resp-parcoord-example*. 2018. https://petertheone.github.io/resp-parcoord/example.html (cited on page 11).

Hauser, Helwig, Florian Ledermann and Helmut Doleisch [2002]. "Angular Brushing of Extended Parallel Coordinates". In: *Proc. IEEE Symposium on Information Visualization (InfoVis 2002)*. (Boston, Massachusetts, USA). IEEE Computer Society. Oct 2002, pages 127–130. doi:10.1109/INFVIS.2002.1173157 (cited on page 15).

Kosara, Robert [2011]. "Indirect multi-touch interaction for brushing in parallel coordinates". In: *Visualization and Data Analysis 2011*. Volume 7868. International Society for Optics and Photonics. 2011, page 786809 (cited on pages 2–3).

"Parallel coordinate plot of Fisher Iris data" [2011]. In: 27th Aug 2011. https://en.wikipedia.org/wiki/File:ParCorFisherIris.png (cited on page 2).

Robertson, George G, Jock D Mackinlay and Stuart K Card [1991]. "Cone trees: animated 3D visualizations of hierarchical information". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1991, pages 189–194 (cited on page 19).