Steerable Parallel Coordinates in D3 (SPCD3)

Project Report

Romana Gruber

706.414 Seminar/Project Interactive and Visual Information Systems 4SP WS 2023/2024
Graz University of Technology

30 Oct 2024

Abstract

Parallel coordinates are a visualistion technique for multidimensional datasets. They are widely used to analyse larger datasets, often in combination with other visualisations like scatterplots and similarity maps. Interactivity is essential to support an analyst: showing and hiding dimensions, adjusting a dimension's range, moving and inverting dimensions, brushing and linking records, and filtering records.

The JavaScript library Steerable Parallel Coordinates in D3 (SPCD3) implements a parallel coordinates component which has both built-in interactivity, as well as being steerable programmatically from the outside via API method calls. The library is written in TypeScript using D3v7. Ultimately, the SPCD3 library will be used to build an explorable explainer for parallel coordinates.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Multidimensional datasets are typically stored in tabular form, like a spreadsheet, with columns representing dimensions (variables) and rows representing records (data points). A header row often gives the names of the dimensions. Large multidimensional datasets can have hundreds or thousands of dimensions and thousands or tens of thousands of records.

Parallel coordinates (PC) is a visualisation technique for multidimensional data, which uses parallel axes to represent dimensions and polylines to represent records, as shown in Figure 1.1, with three dimensions and two records. Dimensions are typically drawn as vertical parallel lines from left to right, but are sometimes also drawn horizontally from top to bottom. A dimension can be either numerical or categorical (strings). Each dimension has a range and scale, and dimensions can also be normalised. Each record has a data value on each dimension. Parallel coordinates are primarily used for analysis to help identify patterns, trends, correlations, and outliers.

The number of dimensions is limited only by the amount of horizontal space, which can be extended by using horizontal scrolling. Inselberg [2009, Chapter 10] worked, for example, with datasets of around 800 dimensions and 10,000 records. Of course, the plot can become very dense and challenging to interpret. However, techniques have been developed to help deal with highly cluttered plots, including axis reordering, using curves, edge-bundling, and sampling [Lu et al. 2016; Graham and Kennedy 2003; Palmas et al. 2014; Ellis and Dix 2006].

An early form of parallel coordinates were developed in 1883 by Henry Gannett, as part of the Statistical Atlas of the United States [Gannett and Hewes 1883]. An example can be seen in Figure 1.2, showing the ranks of the 47 states along 10 dimensions. In 1885, Maurice d'Ocagne published a mathematical theory of parallel coordinates [d'Ocagne 1885; Nature 1885]. In the computer age, parallel coordinates were popularised by Alfred Inselberg [Inselberg 1985; Inselberg 2009].

In 2023, a group of four students implemented a prototype steerable parallel coordinates component called SPC [Drescher et al. 2023b; Drescher et al. 2023c]. Steerable, in this context, means that the visualisation is steerable from outside by calling well-defined methods from an API, as well as directly using interactive operations. The prototype supports interactivity such as showing and hiding dimensions, moving dimensions, inverting dimensions, and hovering over records. However, the SPC component was also made steerable, by providing a selection of methods which can be called on the component, such as `show(dimension)` and `invert(dimension)`. A small example application (demo) was built to illustrate this functionality, which can be seen in Figure 1.3. Controls were added to the example application beneath the parallel coordinates visualisation to select (show and hide) and invert dimensions. These controls in turn call methods from the SPC's API.

In this project, the SPC prototype was taken and expanded in terms of its appearance and functionality, and was renamed Steerable Parallel Coordinates in D3 (SPCD3) [Gruber 2024b]. Important new features include adjusting a dimension's range, selecting records, and filtering records. Moreover, the API was extended with a large number of externally callable methods to steer the SPCD3 component.

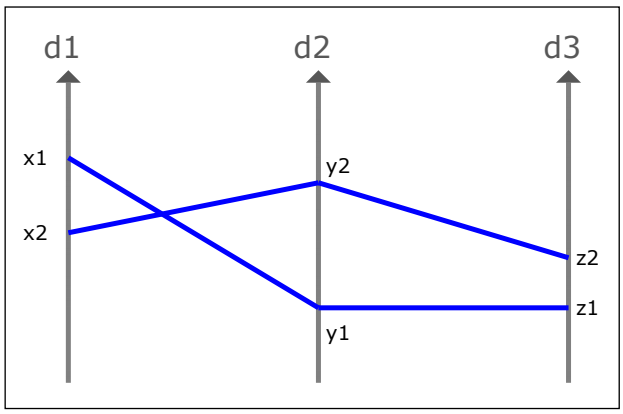

Figure 1.1: A simple parallel coordinates plot with three dimensions (vertical axes) and two records (polylines). Each record touches each axis once at the point corresponding to its data value. [Image drawn by the author of the report.]

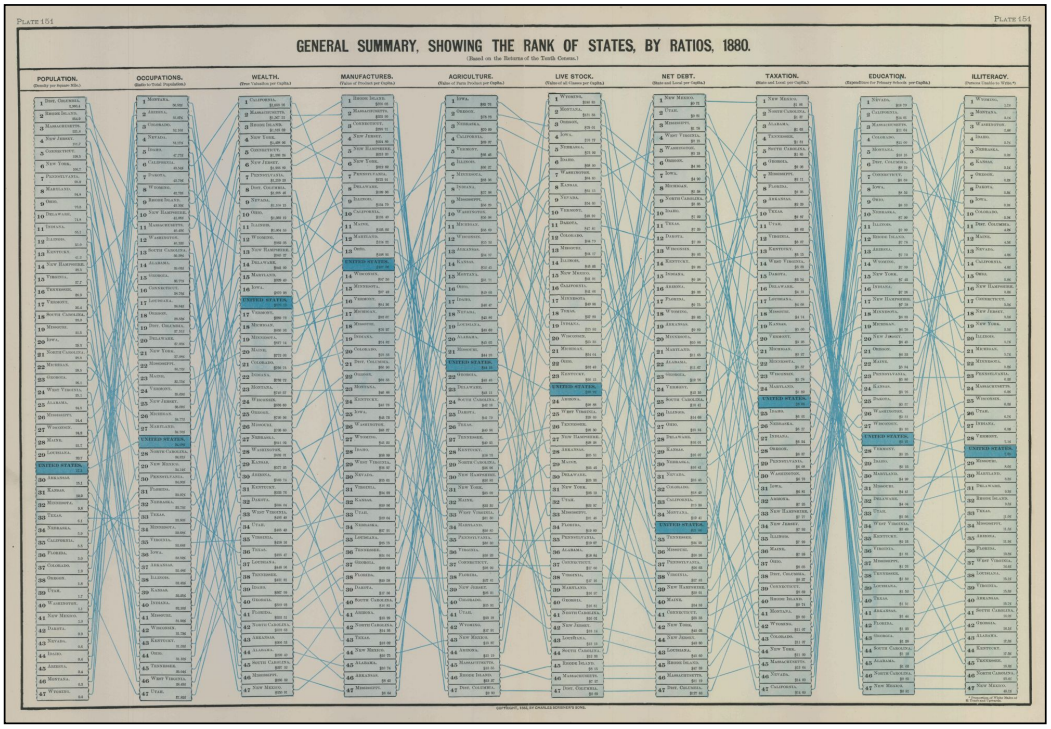

Figure 1.2: Henry Gannett developed an early form of parallel coordinates chart to show the ranks of 47 states along 10 dimensions. The darker blue items indicate the average across all states. A polyline connects the instance of each state on each dimension. [Image kindly provided by the Library of Congress, Geography and Map Division.]

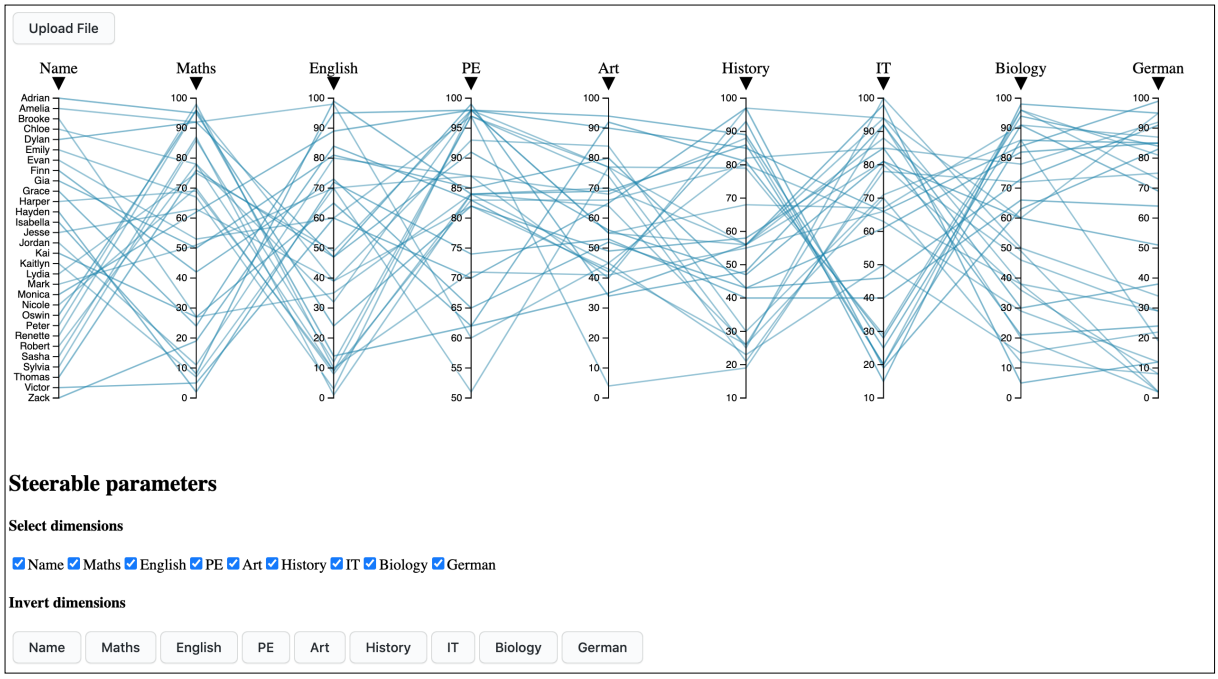

Figure 1.3: A small example application was built around the SPC library to illustrate its use. Here, a dataset of student marks is being visualised. The user can interactively move and invert dimensions and hover over records directly within the plot. However, it is also possible to control the plot from the outside using the controls beneath the plot. [Screenshot made by the author of the report from the example application of Drescher et al. [2023c].]

The ultimate goal is to use the SPCD3 library to build an explorable explainer for parallel coordinates. An explorable explainer guides a reader through an interactive tutorial about a particular topic [Victor 2011; Drescher et al. 2023a]. The illustrations are typically explorable, i.e. can be manipulated by the reader. An explorable explainer about the parallel coordinates visualisation technique would need a parallel coordinates component which can be steered from the narrative, to illustrate particular points along the way.

Chapter 2

SPCD3

Steerable Parallel Coordinates in D3 (SPCD3) is a JavaScript library written in TypeScript, which implements a parallel coordinates (PC) visualisation. The parallel coordinates visualisation can be manipulated in two ways: a) by the end user through mouse and keyboard interactions, and b) programmatically through an API. The library is open source and is available on GitHub [Gruber 2024b].

2.1 Dependencies

The SPCD3 library is written in Typescript [Microsoft 2024] and uses D3 version 7 [Bostock 2024]. D3 is modular and it is not necessary to include the whole of D3. The following D3 modules are used:

- `d3-dsv`: To parse a CSV file containing data.
- `d3-selection`: To get a selection and to set attributes, styles, properties, and more.
- `d3-drag`: To drag and drop the dimensions along the x-axis and for filtering the records.
- `d3-shape`: To draw the polylines of the parallel coordinates.
- `d3-axis`: To create the x-axis and y-axes of the parallel coordinates.
- `d3-scale`: To set the x-scale and y-scale.
- `d3-transition`: To transform changes smoothly rather than instantaneously.

In addition to D3, the following JavaScript libraries are used within SPCD3:

- `mini-svg-data-uri`: To convert SVGs into data URIs [Hunt 2022].
- `xml-formatter`: To prettify the SVG file of the parallel coordinate plot for download [Bottin 2024].

2.2 Build Tools

The task runner Gulp used to automate repeatable tasks [Gulp 2024]. There are four public Gulp tasks:

- `build`: Creates a new build of the library in three formats (CJS, ESM, and IIFE) and stores the generated library packages into the `dist/library/` folder. Additionally, the example folder is copied to `dist/example/`.
- `serve`: Executes the build task, then additionally executes a private task called `watcher`, which is used to initialise a live server for the `dist/example/` folder.
- `clean`: Removes the existing `dist/` directory in order to enable a clean rebuild of the project.

```
src/
├── example/
│   ├── index.html
│   ├── data/
│   │   ├── ...
│   ├── main.js
│   ├── styles.css
└── lib/
    ├── icons/
    │   ├── icons.ts
    │   ├── iconsbase64.ts
    ├── index.ts
    └── scripts/
        ├── brush.ts
        ├── helper.ts
        ├── io.ts
        └── parallelcoordinates.ts
```

Listing 2.1: The main folders and files in the SPCD3 project.

- `cleanAll`: Restores the project folder to its virgin state, as if it were freshly cloned, by deleting the existing `dist/` and `node_modules/` directories and the `package-lock.json` file.

The module bundler Rollup [Rollup 2024] is used to bundle and build the library. The SPCD3 library is packaged into three different formats: CommonJS (CJS), Immediately Invoked Function Expression (IIFE), and ECMAScript Modules (ESM), in both unminified and minified versions, and with a corresponding `map` file for better debugging.

2.3 Software Architecture

The code is separated into two main folders. The `example/` folder contains the example application code, consisting of a `data/` folder with a few example datasets, an HTML file `index.html` with a corresponding CSS file `styles.css`, and a JavaScript file `main.js`. The second folder `lib/` contains the library itself, consisting of an `icons/` folder and a `scripts/` folder. The folder structure is illustrated in Listing 2.1, and the dependencies of the library's files are shown in Figure 2.1.

2.4 Built-In Interactivity

There are several built-in functions to manipulate the visualisation. The user can invert dimensions, move dimensions, hide dimensions, adjust dimension ranges, filter records, hover over records, and select records.

Inverting Dimensions

By default, a dimension's axis places higher values at the top and lower values at the bottom. A dimension can be inverted either by clicking the arrow above the dimension axis or by right-clicking the dimension name to activate the context menu and selecting Invert. Inverting a dimension can be helpful to corroborate a suspected correlation between adjacent dimensions.

Moving Dimensions

When looking for correlations between dimensions, meaningful relationships are only revealed between adjacent dimensions. Hence, it is important to be able to move one dimension next to another dimension of interest. In SPCD3, dimensions can be dragged and dropped at the desired position.

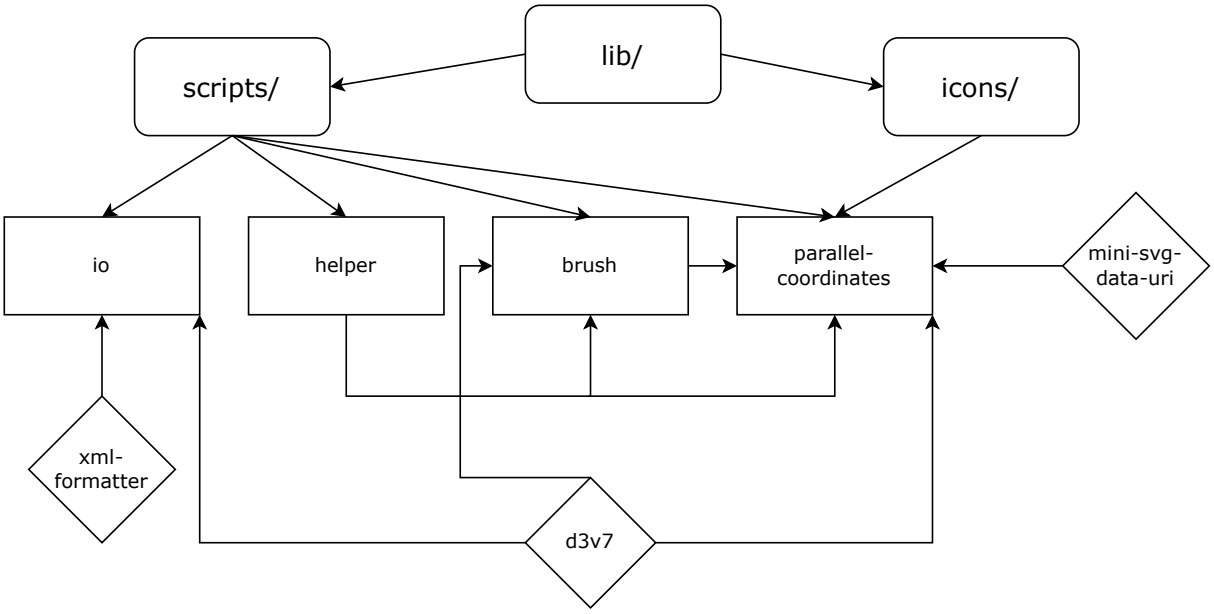

Figure 2.1: The dependencies of the SPCD3 library. In the `lib/` folder, there are two further folders, one with all the scripts and one with the icons. Scripts are indicated with rectangles, modules with diamonds. The arrows indicate dependencies. [Diagram created by the author of the report using draw.io [2024].]

Hiding Dimensions

Displaying too many dimensions can be overwhelming, so it is important to be able to hide individual dimensions. A dimension can hidden by right-clicking the dimension name to activate the context menu and selecting Hide.

Adjusting a Dimension's Range

By default, the range of the dimension's axis is set to the rounded down minimum and rounded up maximum values present in the current dataset. The user can adjust the range of a dimension from its context menu, where two options are available: Set Range to set the range to specific values and Reset Range to reset the dimension to the original range.

Filter Records

Records can be activated and deactivated by setting filters on one or more dimensions. A double-edged range slider can be manipulated on every dimension to filter out records by values on that dimension. Records outside the range are automatically deactivated and greyed out. A further option is to set and reset filters from a dimension's context menu: Set Filter sets a filter to specific values and Reset Filter reset the filter to include all records.

Hovering over Records

If there are a large number of records, a parallel coordinates visualisation contains a large number of potentially crossing and overlapping polylines, so it is challenging to recognise which polyline belongs to which record. Hovering over one or more polylines highlights the polyline(s) beneath the mouse pointer in red and displays a tooltip with the labels of the corresponding records. By default, the label is taken from the first column of the dataset.

Selecting Records

It is helpful to be able to select one or more records in the dataset and to highlight them accordingly. Left-clicking one or more polylines selects the corresponding record(s) and highlights them in orange. Shift-left-clicking adds one or more records to the current selection. Control-left-clicking toggles the selection status of the corresponding records. Left-clicking in empty space clears the current selection.

2.5 Steerable API

SPCD3 provides a set of methods which can be used to steer the visualisation from outside (in essence, an API). The methods are organised in eight groups, as described below.

I/O Functions

- `function loadCSV(csv: string): []`
  Loads a dataset from a CSV file.
- `function drawChart(data: []): void`
  Creates a parallel coordinates chart from the given dataset, using D3 to dynamically create SVG elements in the DOM. This chart is considered to be the current chart.
- `function deleteChart(): void`
  Deletes the current parallel coordinates chart.
- `function saveAsSvg(): void`
  Saves the current parallel coordinates chart as an SVG file with a default name of `parcoords.svg`.

Show and Hide Functions

- `function show(dimensionName: string): void`
  Makes a hidden dimension visible. The dimension is assigned the status shown.
- `function hide(dimensionName: string): void`
  Hides the given dimension. The dimension is assigned the status hidden.
- `function getHiddenStatus(dimensionName: string): string`
  Returns the visibility status of the dimension, which can be either shown or hidden.

Invert Functions

- `function invert(dimensionName: string): void`
  Inverts the given dimension.
- `function getInversionStatus(dimensionName: string): string`
  Returns the inversion status of a dimension, which can be either ascending or descending.
- `function setInversionStatus(dimensionName: string, status: string): void`
  Sets the inversion status of the given dimension to one of ascending and descending.

Move Functions

- `function move(dimensionNameA: string, toRightOf: boolean, dimensionNameB: string): void`
  Moves dimension A either to the left side of dimension B or to the right side of dimension B.

- `function moveByOne(dimensionName: string, direction: string): void`
  Moves a dimension one position to the left or right, independent of other dimensions.
- `function swap(dimensionNameA: string, dimensionNameB: string): void`
  Swaps the positions of the given dimensions.
- `function getDimensionPosition(dimensionName: string): number`
  Returns the position of the given dimension (0...n).
- `function setDimensionPosition(dimensionName: string, position: number): void`
  Sets the position of the given dimension (0...n).

Range Functions

- `function getDimensionRange(dimensionName: string): [min, max]`
  Returns the given dimension's current range (min, max).
- `function setDimensionRange(dimensionName: string, min: number, max: number): void`
  Sets the range of the given dimension to specific values (min, max).
- `function setDimensionRangeRounded(dimensionName: string, min: number, max: number): void`
  Sets the range of the given dimension to rounded specific values (min, max).
- `function getMinValue(dimensionName: string): number`
  Returns the minimum data value of a dimension.
- `function getMaxValue(dimensionName: string): number`
  Returns the maximum data value of a dimension.
- `function getCurrentMinRange(dimensionName: string): number`
  Returns the current minimum value of a dimension's range (in data coordinates).
- `function getCurrentMaxRange(dimensionName: string): number`
  Returns the current maximum value of a dimension's range (in data coordinates).

Filter Functions

- `function getFilter(dimensionName: string): [min, max]`
  Returns the minimum and maximum values of the filter of a dimension.
- `function setFilter(dimensionName: string, min: number, max: number): void`
  Sets the filter for a dimension by specifying minimum and maximum values. If the minimum value lies below the current range, the filter minimum is set to the current range minimum. If the minimum value exceeds the current range, the filter maximum is set to the current range maximum.

Selection Functions

- `function getSelected(): []`
  Returns all selected records in the chart as an array. Each record is identified with its label, taken by default from the first column of the dataset.
- `function setSelection(records: []): void`

Selects one or more records by handing over an array of labels.

- `function toggleSelection(record: string): void`
  Toggles the selection of a given record by specifying its label.
- `function isSelected(record: string): boolean`
  Returns the selection status of a given record by specifying its label.
- `function setSelected(record: string): void`
  Selects a given record by specifying its label.
- `function setUnselected(record: string): void`
  Deselects a given record by specifying its label.

Selection Functions with ID

- `function setSelectionWithId(recordIds: []): void`
  Selects one or more records by handing over an array of IDs.
- `function toggleSelectionWithId(recordId: number): void`
  Toggles the selection of a given record by specifying its ID.
- `function isSelectedWithId(recordId: number): boolean`
  Returns the selection status of a given record by specifying its ID.
- `function setSelectedWithId(recordId: number): void`
  Selects a given record by specifying its ID.
- `function setUnselectedWithId(recordId: number): void`
  Deselects a given record by specifying its ID.

Helper Functions

- `function getAllRecords(): []`
  Returns all records as an array.
- `function getAllDimensionNames(): []`
  Returns an array of all dimensions names in order.
- `function getNumberOfDimensions(): number`
  Returns the number of dimensions.
- `function getDimensionPosition(dimensionName: string): number`
  Returns the position of a dimension $(0..m-1)$.
- `function isDimensionCategorical(dimensionName: string): boolean`
  Returns true if a dimension is categorial and false if not (i.e. it is numerical).
- `function setDimensionForHovering(dimension: string): void`
  Sets the label for hovering by specifiying its ID.
- `function getRecordWithId(recordId: number): string`
  Returns the record by specifying its ID.

Chapter 3

Example Application

An example application was built to demonstrate the functionality of the SPCD3 library. It loads a sample dataset, creates a parallel coordinates chart with SPCD3's built-in interactivity, and adds UI controls to showcase the steerable API. The initial screen is shown in Figure 3.1.

3.1 Dataset

A fictitious dataset of student marks was created by Drescher et al. [2023b]. It consists of a header row, 30 rows of data (records), and 9 columns (dimensions), including the name of the student. Each row represents one student and their marks between 0 and 100 in 8 subjects. Each dimension, apart from the first, represents one subject. The first 11 rows of the dataset in CSV form are shown in Listing 3.1. The full dataset is shown in tabular form in Table 3.1. The student marks dataset was deliberately curated to contain some interesting outliers and correlations. For example, students who are good in English are usually also good in German.

3.2 Built-In Interactivity

Some interactivity is built into the parallel coordinates visualisation: the user can invert dimensions, move dimensions, hide dimensions, adjust dimensions ranges, filter records, hover over records, and select records.

A dimension can be inverted by clicking the arrow above its axis. This can be seen in Figure 3.2, where the dimension Maths is inverted. Clicking the arrow again returns the dimension to its non-inverted state. A further option is to invert the dimension via the context menu, which opens with a right-mouse click on the dimension name.

Moving dimensions is important in finding correlations, because only adjacent dimensions can be compared. Within the chart, a dimension can be dragged and dropped to the desired position. In Figure 3.3, English was moved next to German to check the theory that students who are good at German are also good at English.

Each dimension has a context menu, accessed by right-clicking the dimension name, as shown in Figure 3.4. Through the context menu, users can hide or invert the dimension, set or reset the range, and set or reset the filter. This flexibility allows users to adapt the visualisation to their specific needs and preferences. Dimensions can only be hidden in the context menu. Once a dimension is hidden, it can only be shown again by selecting the Show All item in the context menu, which makes all dimensions in the dataset visible.

By default, the range of the dimension's axis is set to the rounded down minimum and rounded up maximum values which are present in that column of the current dataset. A user can adjust the

```
Name,Maths,English,PE,Art,History,IT,Biology,German
Adrian,95,24,82,49,58,85,21,24
Amelia,92,98,60,45,82,85,78,92
Brooke,27,35,84,45,23,50,15,22
Chloe,78,9,83,66,80,63,29,12
Dylan,92,47,91,56,47,81,60,51
Emily,67,3,98,77,25,100,50,34
Evan,53,60,97,74,21,78,72,75
Finn,42,73,65,52,43,61,82,85
Gia,50,81,85,80,43,46,73,91
Grace,24,95,98,94,89,25,91,69
...
```

Listing 3.1: The first 11 rows of the student marks dataset in CSV form.

Name	Maths	English	PE	Art	History	IT	Biology	German
Adrian	95	24	82	49	58	85	21	24
Amelia	92	98	60	45	82	85	78	92
Brooke	27	35	84	45	23	50	15	22
Chloe	78	9	83	66	80	63	29	12
Dylan	92	47	91	56	47	81	60	51
Emily	67	3	98	77	25	100	50	34
Evan	53	60	97	74	21	78	72	75
Finn	42	73	65	52	43	61	82	85
Gia	50	81	85	80	43	46	73	91
Grace	24	95	98	94	89	25	91	69
Harper	69	9	97	77	56	94	38	2
Hayden	2	72	74	53	40	40	66	64
Isabella	8	99	84	69	86	20	86	85
Jesse	63	39	93	84	30	71	86	19
Jordan	11	80	87	68	88	20	96	81
Kai	27	65	62	92	81	28	94	84
Kaitlyn	7	70	51	77	79	29	96	73
Lydia	75	49	98	55	68	67	91	87
Mark	51	70	87	40	97	94	60	95
Monica	62	89	98	90	85	66	84	99
Nicole	70	8	84	64	26	70	12	8
Oswin	96	14	62	35	56	98	5	12
Peter	98	10	71	41	55	66	38	29
Renette	96	39	82	43	26	92	20	2
Robert	78	32	98	55	56	81	46	29
Sasha	87	1	84	70	56	88	49	2
Sylvia	86	12	97	4	19	80	36	8
Thomas	76	47	99	34	48	92	30	38
Victor	5	60	70	65	97	19	63	83
Zack	19	84	83	42	93	15	98	95

Table 3.1: A fictitious dataset of student marks between 0 and 100 for 30 students in 8 subjects, originally created by Drescher et al. [2023b]. The table has a header row, 30 rows of data (records), and 9 columns (dimensions) including the name of the student.

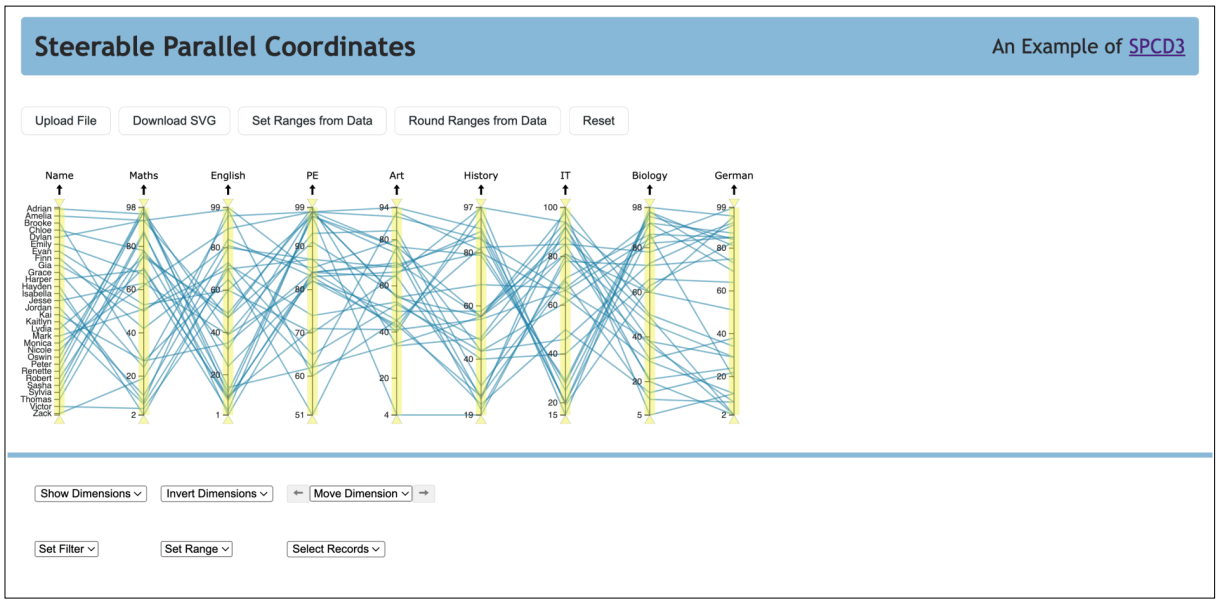

Figure 3.1: The initial screen of the example application. The student marks dataset has been loaded, the parallel coordinates chart has been drawn, and additional UI controls beneath the chart demonstrate the use of the steerable API. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

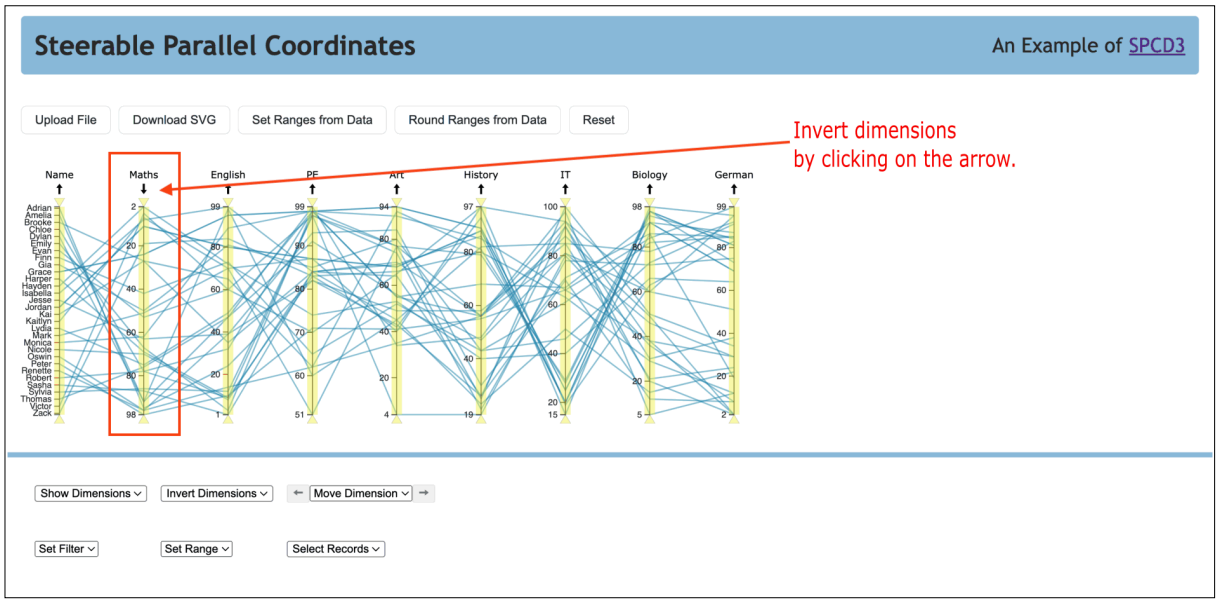

Figure 3.2: Clicking the arrow above a dimension inverts the dimension. Here, the dimension Maths has been inverted. Clicking the arrow again returns it to the non-inverted state. [Screenshot by the author of the report using the Demo of Gruber [2024a].]

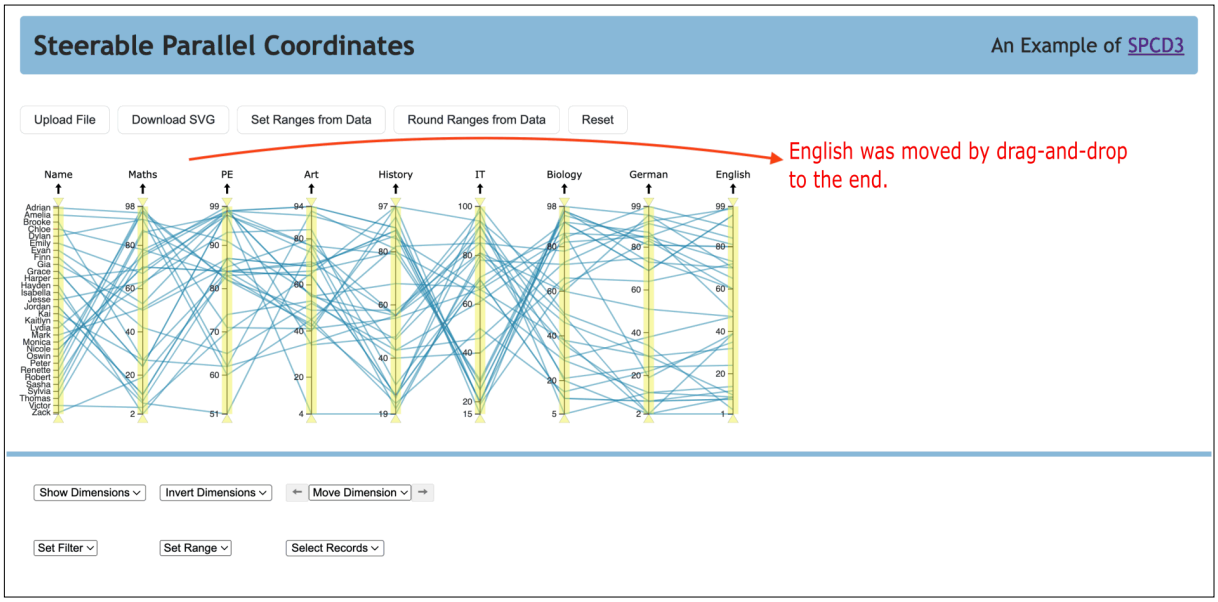

Figure 3.3: A dimension can be repositioned by drag-and-drop. Here, English has been moved to the final position after German. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

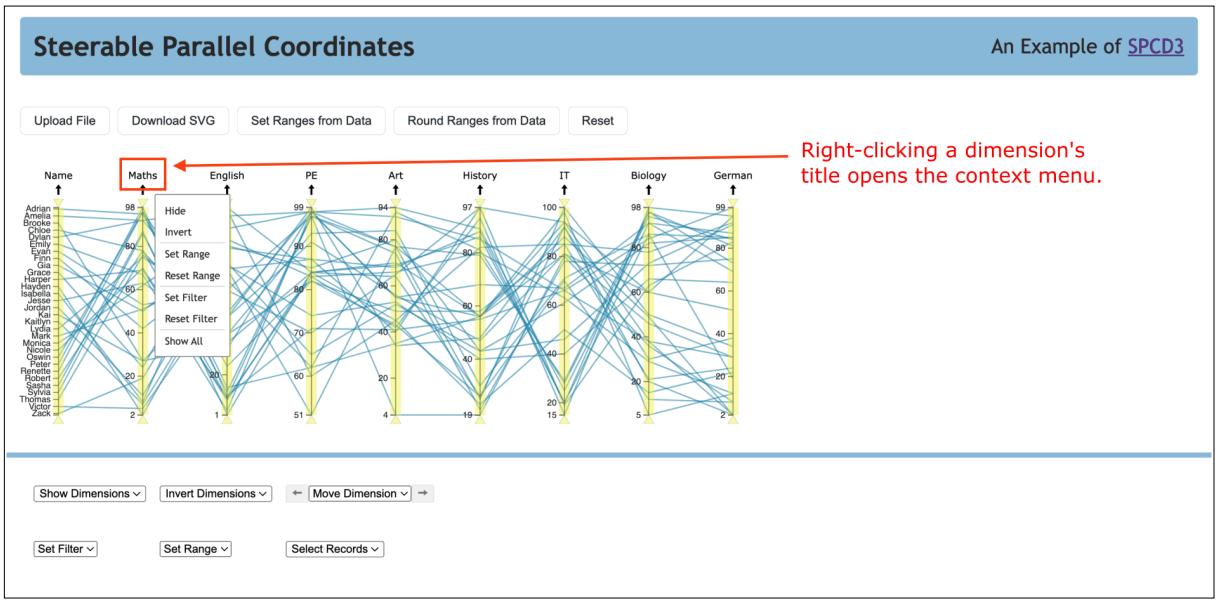

Figure 3.4: Right-clicking a dimension's name opens the context menu. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

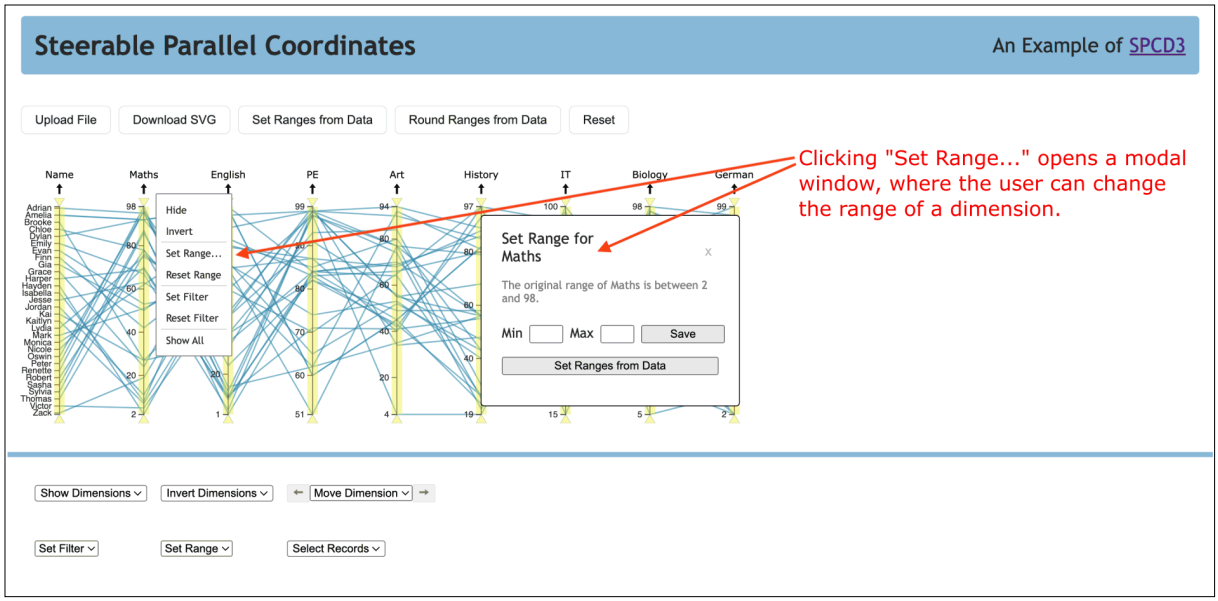

Figure 3.5: Manually setting a range for a dimension. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

dimension's range by selecting Set Range in the context menu and completing the dialogue in the resulting popup window, as shown in Figure 3.5. It is also possible to reset the dimension to its original range. For example, if student marks in Maths are between 0 and 100, but the worst performing student achieved 2 out of 100, and the best student achieved 98 out of 100, then the axis range would show 2 to 98 by default. In such a dataset, it might make sense to adjust the range of every axis to show 0 to 100 to better reflect the domain.

Filtering records can be done for each dimension by moving the upper or lower filter control of the double-edged range slider up and down. All records outside of the filtered area become inactive and are greyed out. The effect is reflected immediately in the visualisation. It is also possible to move the whole range slider up or down, once it is no longer at its maximum size. In Figure 3.6, a filter was set on the dimension Maths, where the upper filter is 80 and the lower filter is 20. All records outside of the filter become inactive and greyed out. A filter can be precisely set for a dimension by selecting Set Filter in the context menu. A popup window opens, where a new minimum and maximum value for the filter can be entered.

When hovering over polylines in the chart, the corresponding records are highlighted in red and their names are shown in a tooltip to identify them. The records of the student dataset, for example, are identified by the Name dimension, as shown in Figure 3.7.

Furthermore, it is possible to select one or more records in the chart by left-clicking, shift-left-clicking, and control-left-clicking to select, extend the selection, and toggle membership in the selection respectively. Selected records are highlighted in orange, as shown in Figure 3.8.

3.3 Additional UI Controls using Steerable API

An essential part of the SPCD3 library is the steerability of the parallel coordinates visualisation component from outside, via the API functions described in Section 2.5. In order to illustrate the use of the API functions, the example application implements a number of features in its UI with the help of underlying API functions. The following features, shown in Figure 3.9, are integrated via buttons and dropdown menus:

- Upload File: Upload a CSV file containing a dataset.

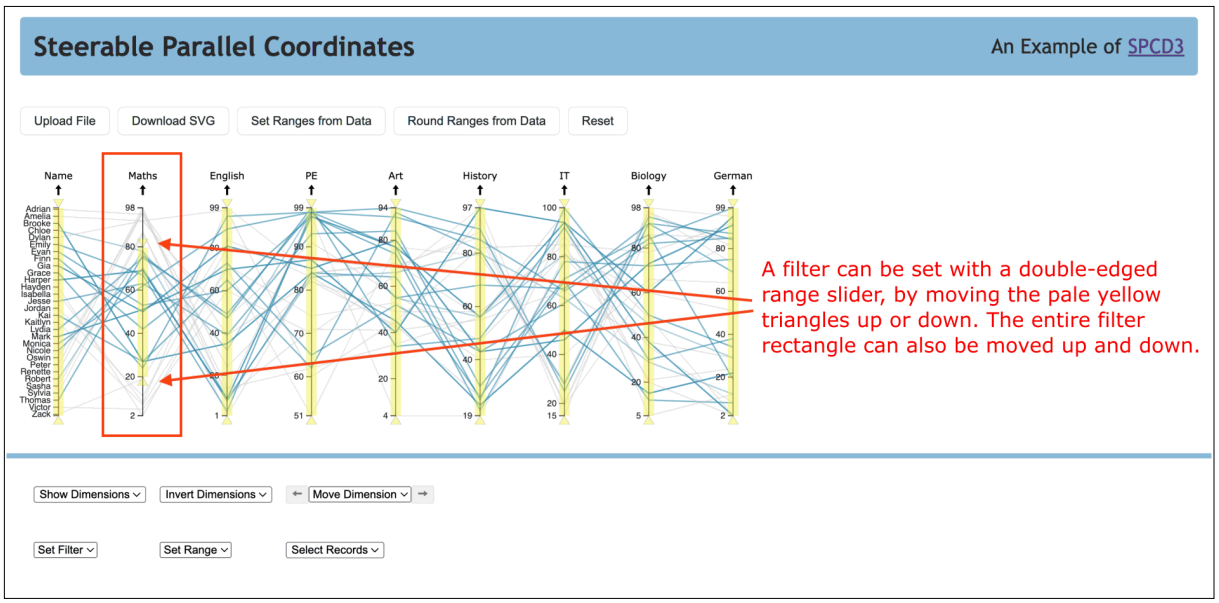

Figure 3.6: It is possible to filter records by adjusting the double-edged range slider for a dimension. All records outside of the filtered area become inactive and are greyed out. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

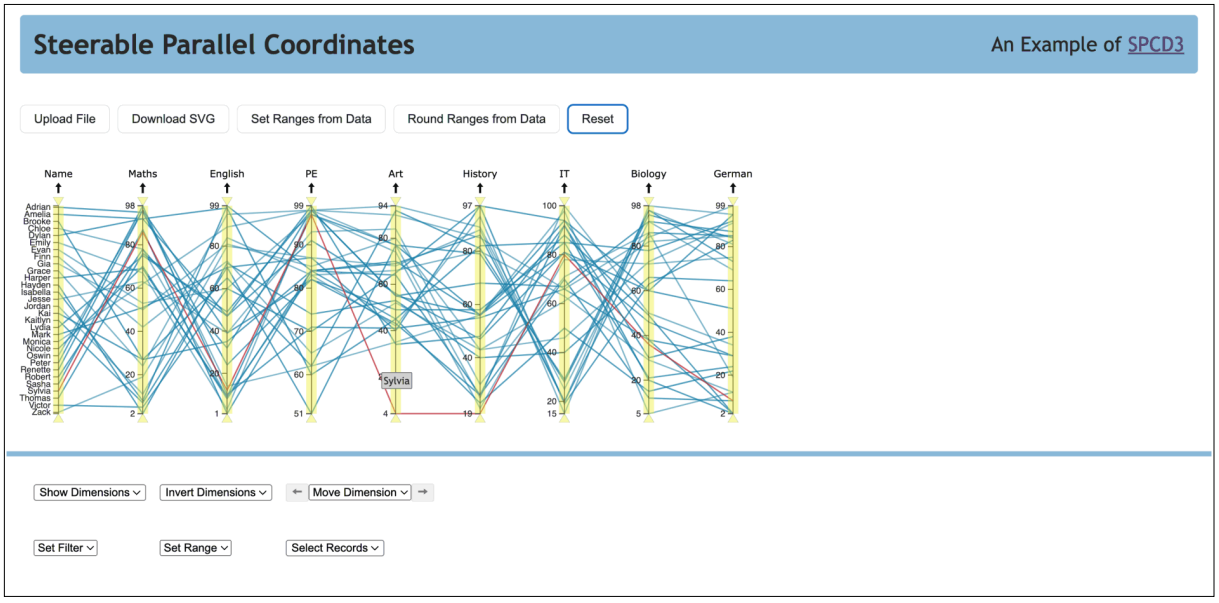

Figure 3.7: When hovering over polylines in the chart, the corresponding records are highlighted in red and their names are shown in a tooltip to identify them. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

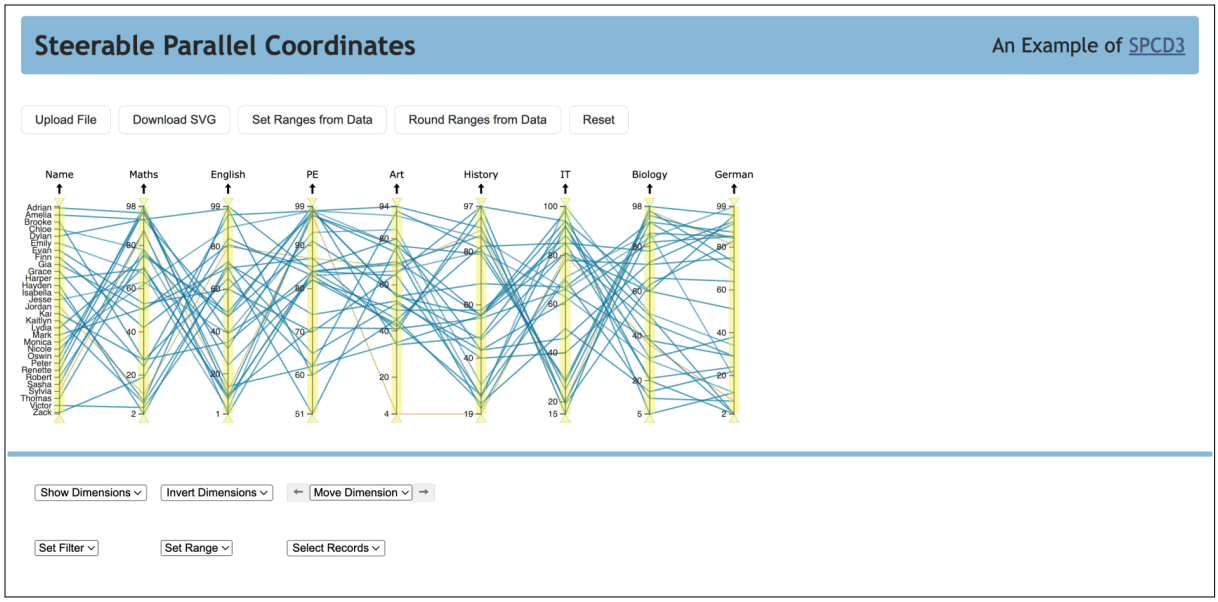

Figure 3.8: Two records have been selected and are highlighted in orange. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

- Download SVG: Download the plot as an SVG file.
- Set Ranges from Data: (Re)set the ranges of all dimensions to the minimum and maximum values which appear in the dataset for each dimension. This is the default for the example chart.
- Round Ranges from Data: Round the ranges of all dimensions up and down to “nice” values.
- Reset Chart: Reset all filters, ranges, inversions, and selections to the initial state.
- Show Dimensions: Open a dropdown, where each dimension can be hidden or shown.
- Invert Dimensions: Open a dropdown, where each dimension can be inverted.
- Move Dimensions: Open a dropdown, where a dimension can be selected and then moved left or right via the arrow buttons.
- Set Filter: Open a dropdown, where a dimension can be selected and then a filter set on that dimension.
- Set Range: Open a dropdown, where a dimension can be selected and then the range set on that dimension.
- Select Records: Open a dropdown, where each record can be selected or unselected.

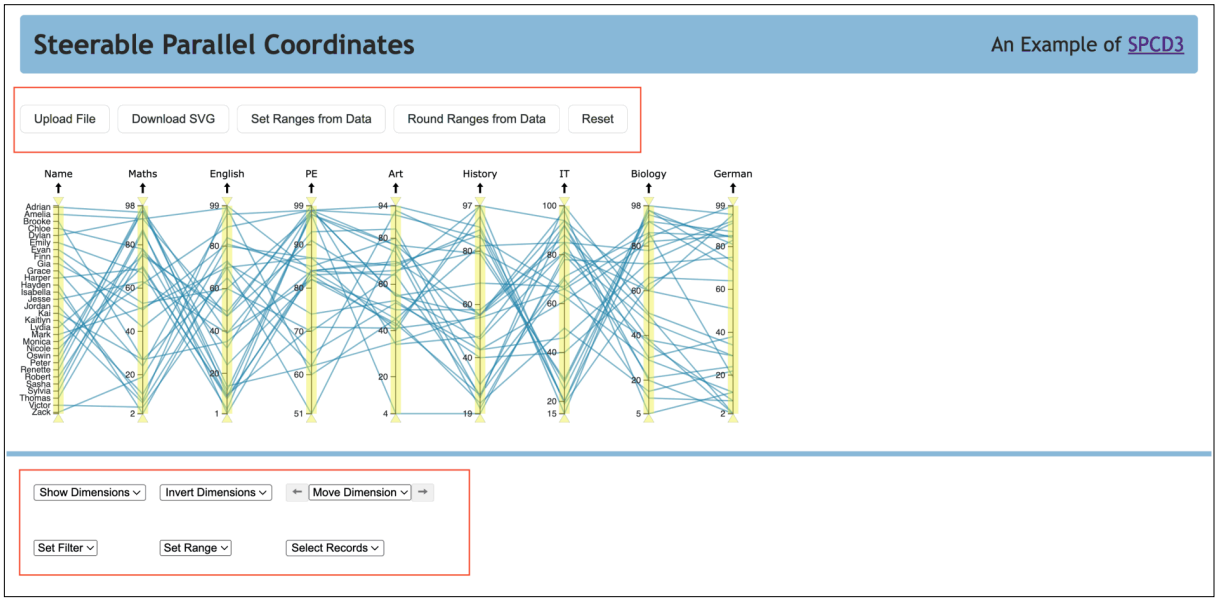

Figure 3.9: Functionality provided through the SPCD3 API. Above the plot, five buttons provide features to upload a dataset, download the chart as SVG, reset the ranges, and reset the entire chart. Beneath the plot, there are six additional UI controls to show, invert, and move dimensions, set filters, set ranges, and select records. [Screenshot made by the author of the report using the Demo of Gruber [2024a].]

Chapter 4

Concluding Remarks

Parallel coordinates are a widely-used visualisation technique for multidimensional datasets. They can assist an analyst to identify clusters, outliers, and correlations within a multidimensional dataset. Interactivity such as showing and hiding dimensions, adjusting a dimension's range, moving and inverting dimensions, brushing and linking records, and filtering records is essential to support the analysis process.

The SPCD3 library was implemented in TypeScript using D3v7 to provide steerable parallel coordinate plots with these important techniques. The library has both built-in interactivity and is steerable programmatically from the outside via API method calls. The library is open source and is available on GitHub [Gruber 2024b].

Further improvements might include adding histograms to the dimensions or providing colour-coding for specific groupings of records. In the near future, the SPCD3 library will be used to build an explorable explainer, a kind of interactive tutorial to explain how parallel coordinates work.

Bibliography

Bostock, Mike [2024]. *D3*. 25 Apr 2024. https://d3js.org/ (cited on page 5).

Bottin, Chris [2024]. *XML-formatter*. 20 Jan 2024. https://github.com/chrisbottin/xml-formatter (cited on page 5).

d'Ocagne, Maurice [1885]. *Coordonnées Parallèles et Axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèlles*. In French. Gauthier-Villars, 1885. 93 pages. https://archive.org/details/coordonnesparal00ocaggoog (cited on page 1).

draw.io [2024]. *draw.io*. 27 Jun 2024. https://drawio.com/ (cited on page 7).

Drescher, Philipp, Jeremias Kleinschuster, Sebastian Schreiner, and Burim Vrella [2023a]. *Explorable Explainers*. Survey Paper. Information Visualisation, SS 2023, Graz University of Technology, 19 May 2023. https://courses.isds.tugraz.at/ivis/surveys/ss2023/ivis-ss2023-g1-survey-explorables.pdf (cited on page 3).

Drescher, Philipp, Jeremias Kleinschuster, Sebastian Schreiner, and Burim Vrella [2023b]. *Steerable Parallel Coordinates in JavaScript*. Project Report. Information Visualisation, SS 2023, Graz University of Technology, 16 Jun 2023. https://courses.isds.tugraz.at/ivis/projects/ss2023/ivis-ss2023-g1-project-steerable-parcoords.pdf (cited on pages 1, 11–12).

Drescher, Philipp, Jeremias Kleinschuster, Sebastian Schreiner, and Burim Vrella [2023c]. *SteerableParallelCoordinates*. 20 Jul 2023. https://github.com/burimvrella/SteerableParallelCoordinates (cited on pages 1, 3).

Ellis, Geoffrey and Alan Dix [2006]. *Enabling Automatic Clutter Reduction in Parallel Coordinate Plots*. Transactions on Visualization and Computer Graphics 12.5 (Sep 2006), pages 717–724. doi:10.1109/TVCG.2006.138. https://eprints.lancs.ac.uk/id/eprint/12830/ (cited on page 1).

Gannett, Henry and Fletcher Willis Hewes [1883]. *Scribner's Statistical Atlas of the United States, Showing by Graphic Methods their Present Condition and their Political, Social and Industrial Development.* New York, 1883. https://loc.gov/item/a40001834/ (cited on page 1).

Graham, Martin and Jessie Kennedy [2003]. *Using Curves to Enhance Parallel Coordinate Visualisations*. Proc. 7th International Conference on Information Visualization (IV 2003) (London, UK). IEEE, 16 Jul 2003, pages 10–16. doi:10.1109/IV.2003.1217950. https://napier-repository.worktribe.com/preview/267520/IV03_G1087_GrahamKennedyParallelCurves.pdf (cited on page 1).

Gruber, Romana [2024a]. *Steerable Parallel Coordinates*. 30 Oct 2024. https://tugraz-isds.github.io/spcd3 (cited on pages 13–18).

Gruber, Romana [2024b]. *Steerable Parallel Coordinates in D3 (SPCD3)*. 30 Oct 2024. https://github.com/tugraz-isds/spcd3 (cited on pages 1, 5, 19).

Gulp [2024]. *Gulp*. 29 Mar 2024. https://gulpjs.com/ (cited on page 5).

Hunt, Taylor [2022]. *Mini SVG data: URI*. 09 Mar 2022. https://github.com/tigt/mini-svg-data-uri (cited on page 5).

Inselberg, Alfred [1985]. *The Plane with Parallel Coordinates*. The Visual Computer 1.2 (01 Aug 1985), pages 69–91. doi:10.1007/BF01898350. https://www.asc.ohio-state.edu/statistics/statgen//j_spr2013/Inselberg_1985_Parallel-Coordinates.pdf (cited on page 1).

Inselberg, Alfred [2009]. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer, 08 Oct 2009. 554 pages. ISBN 0387215077 (cited on page 1).

Lu, Liangfu, Mao Huang, and Jinson Zhang [2016]. *Two Axes Re-ordering Methods in Parallel Coordinates Plots*. Journal of Visual Languages & Computing 33.1 (01 Apr 2016), pages 3–12. doi:10.1016/j.jvlc.2015.12.001. https://opus.lib.uts.edu.au/bitstream/10453/41014/1/1-s2.0-S1045926X15300379-main.pdf (cited on page 1).

Microsoft [2024]. *TypeScript*. 09 May 2024. https://typescriptlang.org/ (cited on page 5).

Nature [1885]. *Coordonnées Parallèles et Axiales: Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèlles*. Nature 31 (16 Apr 1885). Book review, pages 551–552. doi:10.1038/031551b0 (cited on page 1).

Palmas, Gregorio, Myroslav Bachynskyi, Antti Oulasvirta, Hans Peter Seidel, and Tino Weinkauf [2014]. *An Edge-Bundling Layout for Interactive Parallel Coordinates*. Proc. 2014 IEEE Pacific Visualization Symposium (PacificVis 2014) (Yokohama, Japan). 04 Mar 2014, pages 57–64. doi:10.1109/PacificVis.2014.40. https://www.csc.kth.se/~weinkauf/publications/documents/palmas14a.pdf (cited on page 1).

Rollup [2024]. *rollup.js - The JavaScript Module Bundler*. 22 May 2024. https://rollupjs.org/ (cited on page 6).

Victor, Bret [2011]. *Explorable Explanations*. 10 Mar 2011. https://worrydream.com/ExplorableExplanations/ (cited on page 3).