Bachelor Project Report

Research Data Visual Analytics

Lorenzo Ferri

Advisor:
Prof. Dr. Michele Lanza
Assistents:
Dr. Andrea Mocci, Dr. Luca Ponzanelli

Contents

Abstract

Data analytics is an important practice that allows to better understand data and, in some cases, take strategic decisions accordingly. Many institutions, including USI, store data in a raw format like Excel, and lack a platform to perform data analytics. Data visualization is one of the preferred means for analyzing data and it helps to communicate information clearly and efficiently, making it easier to have a general overview and combine information across data sets. The goal of this project is to produce a visual analytics web-based platform for the interactive visualization of the research funding obtained by the USI researchers.

Chapter 1

Introduction

1.1 Motivation

At USI the data about the research projects is managed in an Excel file. Excel is a great tool for analyzing data but it can be hard for everyday users to leverage it for interactive data visualization, since it would require them to know its specific macro language. With a web application instead we can offer a ready-made solution where users can freely navigate and interact with real time updated data, without needing them to write any code or learn any specific skill. Also users can access the web application from wherever they want and from any type of device provided with a browser.

1.2 Goal

The goal of the project is to create a web-based application to visualize the funding obtained by the USI researchers. Given the list of all research projects, the application should display different charts in order to analyze the distribution of the funding, and it should also offer a way to interact with the displayed data and navigate between different views. Finally, the application will be integrated with the platform MyUSI, a platform that aims to offer some services to the university, including support for the research projects.

1.3 Technologies

The application is developed as part of the frontend of MyUSI. The web application is built on top of Polymer 2.4[1] and TypeScript[2], and it uses D3.js[3] to create the different charts. It relys on a Scala[4] backend that serve a GraphQL[5] endpoint in order to fetch the projects data. GraphQL is a query language for the API.

[1]Polymer 2.4: `https://www.polymer-project.org`
[2]Typescript: `http://www.typescriptlang.org`
[3]D3.js: `https://d3js.org`
[4]Scala: `https://www.scala-lang.org`
[5]GraphQL: `https://graphql.org`

Chapter 2

Related Work

Our project aims to create a tool for visual data analytics, using all the principles of data visualization. In history there are a lot of key figures in the field of data visualization. In this chapter we are going to see a few of them.

2.1 The Minard Map (1869)

The Minard Map [1], drawn by Charles Joseph Minard, shows the depiction of numerical data on a map of Napoleon's disastrous losses suffered during the Russian campaign of 1812. The map, shown in Figure 2.1, is known for being one of the best charts ever created.

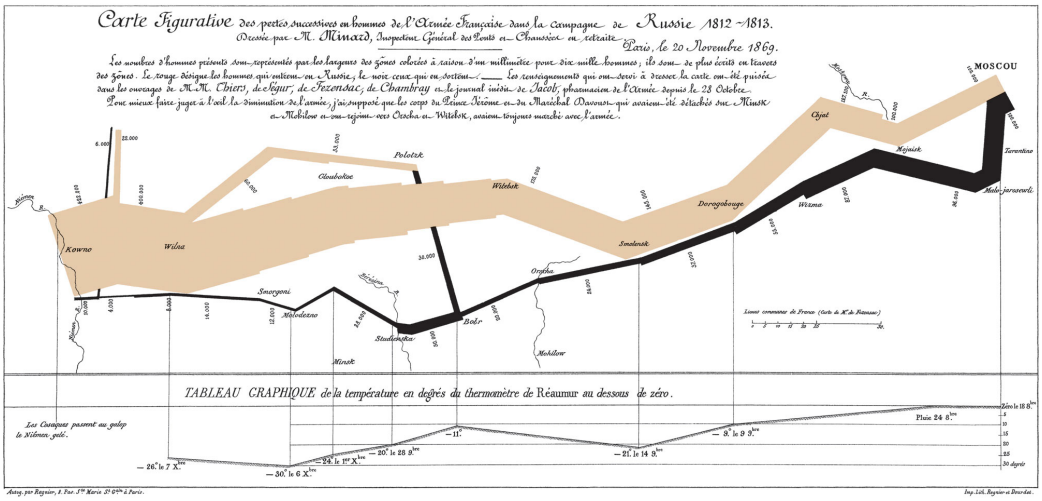


Figure 2.1: The Minard Map

2.2 The Commercial and Political Atlas (1786)

The Commercial and Political Atlas [2], written by William Playfair, featured a variety of graphs including the image shown below in Figure 2.2. In this famous example, he compares exports from England with imports into England from Denmark and Norway from 1700 to 1780. William Playfair is considered the father of statistical graphics, having invented the line and bar chart we use so often today. He is also credited with having created the area chart and pie chart.

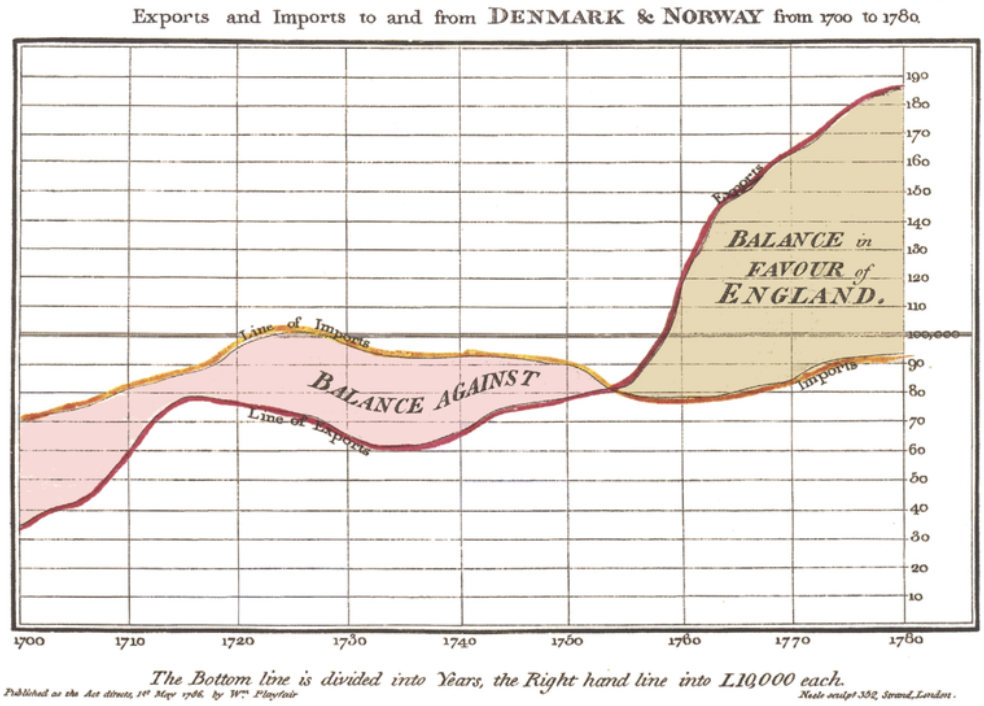
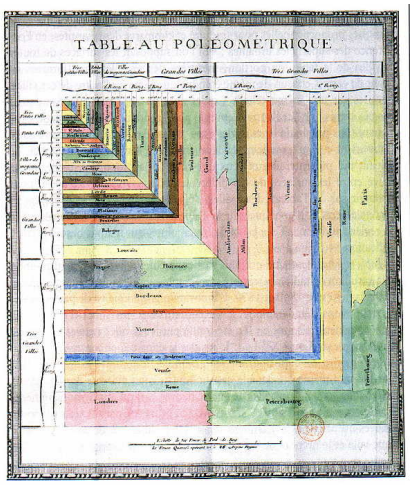


Figure 2.2: Exports and imports into England from Denmark and Norway from 1700 to 1780.

2.3 Essai dune table poléométrique (1782)



Essai d'une table poléométrique [3] is a book written by Charles Louis de Fourcroy that gives an analysis of the urban growth of European cities, which are graphically compared in a diagram, called Table Poléomètrique or Poleometric Table. Charles Louis de Fourcroy use of geometric shapes predates the modern treemap, which is widely used nowadays to display hierarchical data.

Figure 2.3: The Poleometric Table

2.4 Population pyramid (1878)

The Population pyramid [4] is a diagram drawn by Luigi Perozzo, and it is one of the first 3D representations of data, showing the age group of the Swedish population between the 18th and 19th centuries. Luigi Perozzo was an Italian mathematician and statistician who stood out for being the first to introduce 3D graphical representations, showing the relationships between three variables on the same graph.

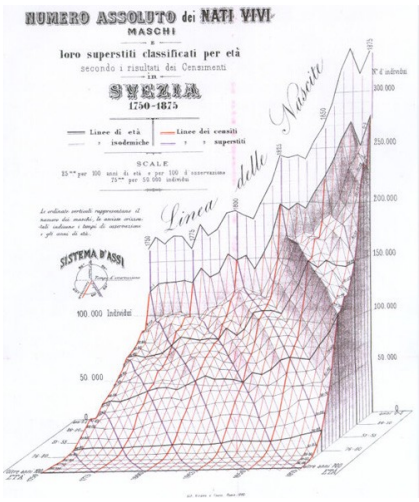


Figure 2.4: The Population pyramid

Chapter 3

Approach

3.1 Creating the charts

To create the web application the first step is to create the charts that will allow the users to visualize the data. We choose to build the charts by ourselves since we wanted them to fit our needs. In particular we focused our attention into interaction: we wanted the charts to interact with each other, and update if the data change. For example if we click into a specific part of a chart we want to change view and display some other data, this action should update all charts and their data. To achieve all of those requirements we decide to use `D3.js` as charting library, since is highly customizable.

3.1.1 Donut Chart

The first chart that we build is the donut chart. This chart is able to display a set of elements and compare their value.

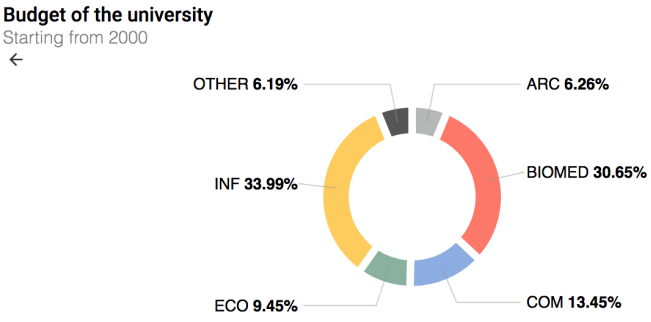


Figure 3.1: An example of donut chart.

This chart expect as input an array of data with the shape shown in Listing 3.1.

```
1 public label = '';
2 public value = 0;
3 public hidden = false;
4 public color?: string | number;
5 public callback?: () => void;
```

Listing 3.1: Input structure for the donut chart

The rendering logic of our donut chart is divided in three parts:

Slices: Slices are the main components of the chart. Slices are defined by the `d3.arc` function. This function describes the size of each slice in degrees, and also allow to set additional options. In particular, it was important to us to specify that if one element had the property `hidden` set to true, the value of that element should be considered 0. The color of the element is either given or randomly assigned from a predefined list.

Labels: The labels of the charts provide two main purposes: tell us the name of each element and tell us the percentage of that element. The name is simply passed to the chart, while the percentage is calculated based of how many degrees that element takes in the donut chart. In order to correctly place the labels we take the midpoint of the slice and place the label towards the outside of the circle from that point, then we align the text to the left or to the right based on which side the label is.

Lines: The lines tell us which label correspond to which slice. We draw a line from the slice to the outside of the donut, and the we connect it to the label.

In addition, it is important to notice that in the structure of the data we can set a callback that will be called when we click on the slice of the respective element.

3.1.2 Bar Chart

Similarly to the donut chart, the goal of the bar chart is to visualize a set of elements and compare their sizes. The main difference with the donut chart is that the bar chart supports also negative values, but it will not show the percentage of an element compared to the total.

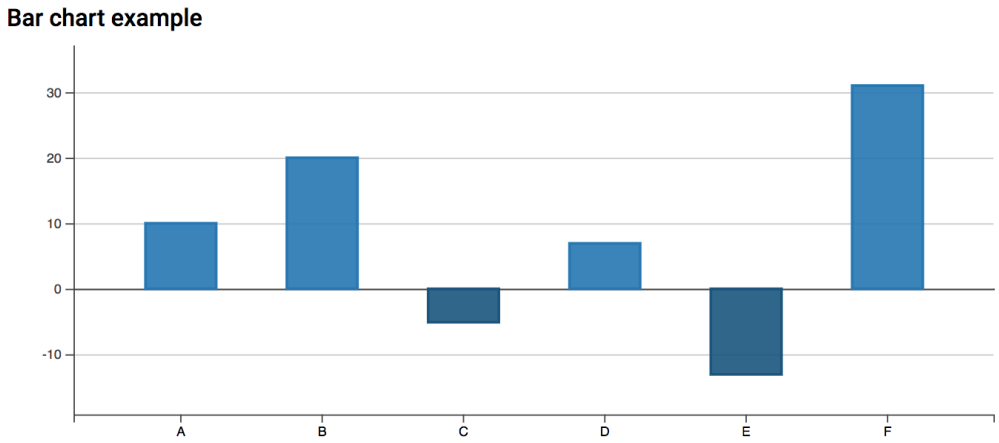


Figure 3.2: An example of bar chart.

This chart expects a data structure similar to the donut chart as shown in Listing 3.2.

```
1 public label = '';
2 public value = 0;
3 public hidden = false;
4 public callback?: () => void;
```

Listing 3.2: Input structure for the bar chart

The rendering of the bar chart is divided in 3 parts:

Axes: We draw 2 axes. On the `x` axis we place one label for each element, while the `y` axis represents the value of an element. D3 provides some ready to use functions to draw axes with `d3.axisLeft` and `d3.axisBottom` which just require a minimal configuration for the domain

Ticks: Along the `y` axis we display some ticks that identify the value at that specific point. In order to make the chart more clear we added some line that start from that ticks and go trough all the chart. Those line were drawn using `d3.axisLeft` but with a different configuration compared to the axes. Also, we make the `0 tick` bigger, so that it is easier to notice it.

Bars: Finally, we draw the bars which are the main part of the chart. To draw them we use some svg `rect` and we bind the height to the value of the element that we want to display. `d3.scaleLinear` will help us with all the transformation from the real value to the height in pixels. For the color we look if the value is positive or negative and choose the color accordingly.

3.1.3 Line Chart

The goal of the line chart is to show the evolution of multiple values over time.

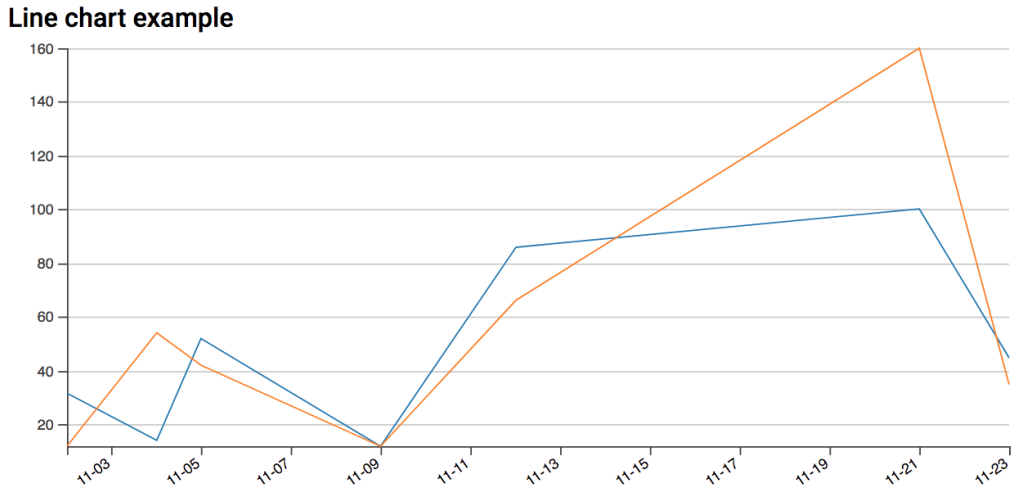


Figure 3.3: An example of line chart.

Since this type of chart supports multiple lines the data structure that it expect is an array structured as shown in Listing 3.3:

```
1 public label = '';
2 public data: { date: string, value = 0 }[];
3 public color?: string | number;
4 public hidden = false;
```

Listing 3.3: Input structure for the line chart

Drawing a line chart is similar to the bar chart, but instead of the bars, for each element we draw a `polyline`. It is important to notice that along the `x` axis we have time. In fact to render the `x` axis we to use use `d3.scaleTime`, and we also parse the `date` attribute into a JS date format.

3.1.4 Legend

The legend component displays the a list of the elements that are shown in the charts. In contrast to the other components it is also the only one that actually modifies the data.

Figure 3.4: An example of legend.

Since this component is meant to show different type of data it does not have a fixed data structure, but it expects to receive an array in which each element contains the properties described in Listing 3.4.

```
1 public label = '';
2 public value = 0;
3 public hidden = false;
4 public color?: string | number;
```

Listing 3.4: Input for legend component

As we said before, the legend will also modify the data. If we click on an element in the list, the legend component will toggle the `hidden` property. This event will hide the respective element from all the charts.

3.1.5 Stacked Charts

There are two types of stacked charts: bar charts and area charts. They are respectively an extension of the standard bar chart and line chart.

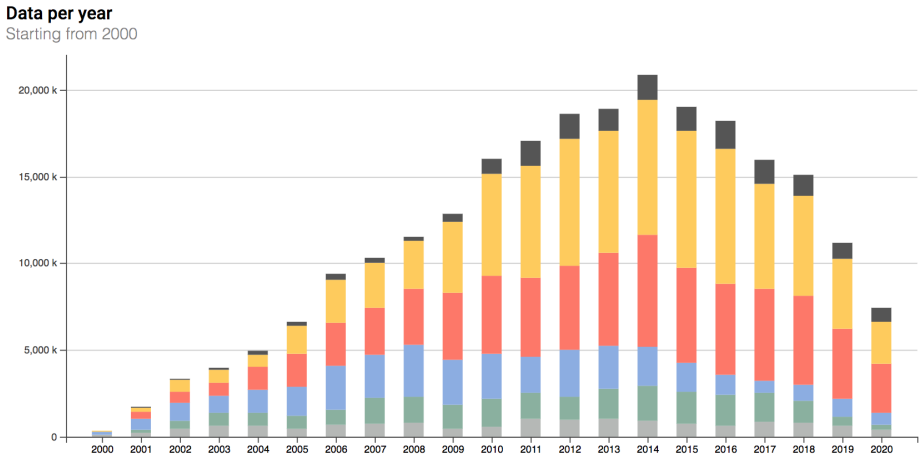


Figure 3.5: An example of stacked bar chart.

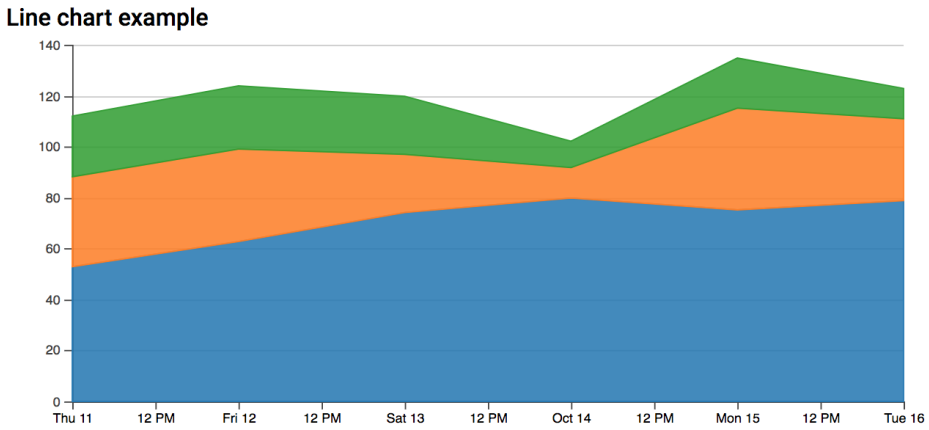


Figure 3.6: An example of stacked area chart.

In order to draw them the steps are the same as the bar chart and line chart, but we are going to use `d3.layout.stack` to handle the position of different stacked items. The expected data structure for those data is different from the previous ones as they expect an array of elements with the structure shown in Listing 3.5 and Listing 3.6:

```
1 public label = '';
2 public [key: string]: {
3     value = 0,
4     hidden = false,
5     color?: string | number
6 }
```

Listing 3.5: Input for stacked bar chart

```
1 public date: string;
2 public [key: string]: {
3     value = 0,
4     color?: string | number
5 }
```

Listing 3.6: Input for stacked area chart

The parameter `[key: string]` means it can receive any other property that has as the structure specified above. Listing 3.7 shows a list of valid input for the structure of the stacked bar chart.

```
[
    {
        label: 'A',
        p1: { value: 10, hidden: false },
        p2: { value: 35, hidden: false },
        p3: { value: 24, hidden: false }
    },
    {
        label: 'B',
        p1: { value: 63, hidden: false },
        p2: { value: 36, hidden: false },
        p3: { value: 25, hidden: false }
    },
]
```

Listing 3.7: Valid input for the stacked bar chart

3.2 Data Manipulation

Once we defined our charts, we need to feed them with the projects data. This is not a trivial task since the structure of the projects does not match the expected input from the charts. A research project is structured like in Listing 3.8.

```
title: string;
acronym: string | null;
accountingIdentifier: string | null;
affiliation: IAffiliation;
principalInvestigators: IInvestigator[];
coInvestigators: IInvestigator[];
usiAmount: IMonetaryAmount | null;
funding: IFunding;
startDate: ILocalDate;
endDate: ILocalDate | null;
spendingReview: ISpendingReviewEntry[];
notes: string | null;
```

Listing 3.8: Structure of a research project

Given this structure we can take the `principalInvestigators` where we do not only store the name of the investigator, but also his quota in the project and his affilitation. With this, we can map that quota to the affiliaton, and then with the charts compare all the different affiliations.

3.3 Building the web app

We now have all the tools we need to build the actual web app. We use a donut chart to show the overall distribution of the funding, and a stacked bar chart to show the same distribution but over the years. We also insert a legend component to hide and show specific data. Figure 3.7 shows the final result.

Figure 3.7: Screenshot of the final look of the web application

To place all the component in the page we take advantage of `CSS Grid` with the specification shown in Listing 3.9:

```
display: grid;
margin: auto;
grid-template-areas:
    "donut donut donut legend"
    "stack stack stack stack";
grid-template-columns: repeat(4, 1fr);
grid-template-rows: 2fr 3fr;
background: transparent;
grid-gap: 15px;
```

Listing 3.9: The specification for the style of the grid

Chapter 4

Use Case Scenario

In this use case scenario, Alice wants to know which institute of the Faculty of Informatics received more funding during the last year. First, she clicks on the slice `USI` of the donut chart which brings her to the faculty view. Here she can see a view that displays the funding received by each faculty.

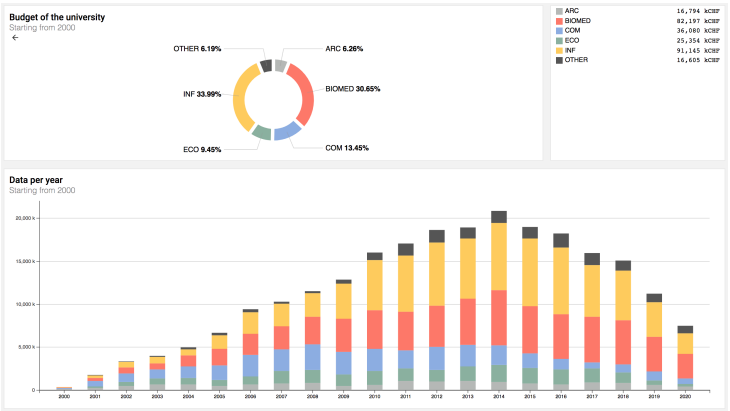


Figure 4.1: View of all faculty

Since she is interested in the Faculty of Informatics, she clicks on the respective slice. Now the web application will display all the institutes of Informatics and their respective funding. Here she can see that the institute `A` received more funding overall, but during the last year it was `B` which received more funding.

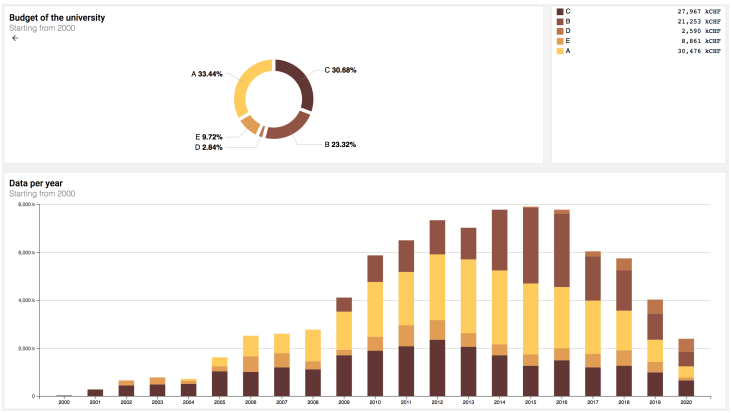


Figure 4.2: View of all the institute of Informatics

Now she is wondering which professors are affiliated with the institute `B`, so she clicks on the respective slice and she is presented with the division of the funding inside the institute `B`.

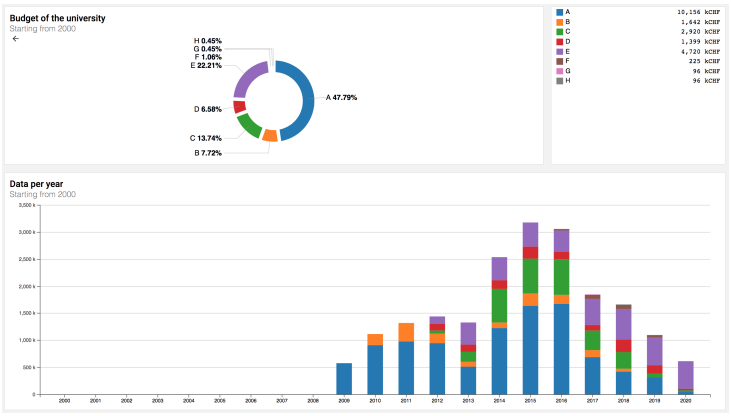


Figure 4.3: View od the professors of the institute `B`

With just a few clicks Alice was able to understand how the funding were split among the faculty of Informatics.

Chapter 5

Conclusion

We presented a visual analytics web-based platform for interactive visualization of the research funding obtained by the USI researchers We started by creating all the charts components that then we used in the visualize the data. Then we had to manipulate the data to fit our charts specification, and finally we could build the actual web application.

The part that took most of the time was building the charts, since we had to use `D3.js`, a technology new to us with a very steep learning curve. Even if building the charts took a lot of time, the most challenging part was manipulating the data, since we had to `map` the data back and forward between one structure and the other.

We are satisfied with the outcome of this project. The results obtained can be further extended in the future. For example, additional views and charts can be implemented, and we could allow users to visualize additional aspects of the research funding. We could also allow users to export some views to other formats or access the web application offline.

Bibliography

[1] Charles Joseph Minard. The minard map, 1869.

[2] W. Playfair. *The Commercial and Political Atlas: Which Represents at a Single View, by Means of Copper Plate Charts, the Most Important Public Accounts of Revenues, Expenditures, Debts, and Commerce of England. By William Playfair. To which are Added, Charts of the Revenues and Debts of Ireland, Done in the Same Manner, by James Corry, Esq.* John Stockdale, Piccadilly, 1787.

[3] Charles de Fourcroy. *Essai d'une table poléométrique, ou amusement dun amateur de plans sur la grandeur de quelques villes.* 1782.

[4] Luigi Perozzo. *Statistica Grafica - Della rappresentazione grafica di una collettivit di individui nella successione del tempo, e in particolare dei diagrammi a tre coordinate - Memoria di Luigi Perozzo.* 1782.