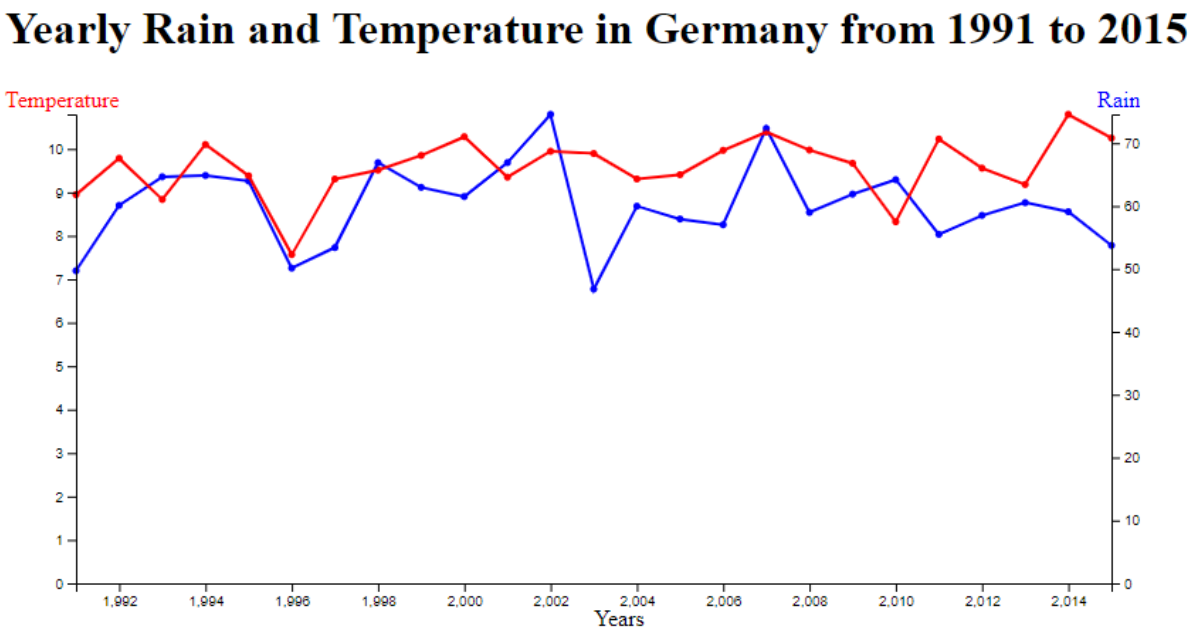
Exercise 2: Create a Line-Chart with D3.js (10 points)

Due: 2.5.2023 8:00AM (1st of May is a public holiday)



Goal of this exercise is to implement a line-chart with D3.js. Attached to this exercise you will find a folder called *linechart* which contains an unfinished implementation. Your task is to finish the implementation such that opening the *index.html* shows a line-chart as depicted in figure above. The figure shows trends (averaged by year) of the temperature (red line) and rainfall (blue line) in Germany from 1991 to 2015. To finish the implementation, follow the steps described as comments within the dedicated file. Each comment starting with *Task* indicates a position you have to add code. Within the folder *linechart* there are 4 files:

- **index.html (1 Point)**
  The main entry point of the visualization. When your implementation is finished, opening this file with a browser should show a *linechart*. Currently, opening the *index.html* will show a heading but no visualization.
- **index.js (9 Points)**
  The main JavaScript entry point which implements a line-chart.
- **index.css**
  Implements CSS Rules for specific elements.
- **data.js**
  Reflects the dataset we want to visualize. It represents monthly rainfall and temperature of Germany from 1991-2015. Attribute 'pr' is Rainfall in (mm) while 'tas' is the average temperature.

Wherever possible, try to use a function provided by the d3 library. In particular, do not use JS-function like *"document.select…", but use d3 selection instead.*
A helpful reference for this exercise: https://www.d3indepth.com/selections/

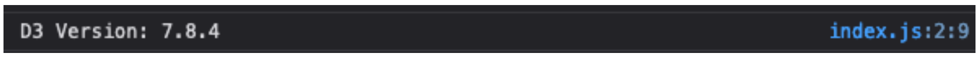
Task 1: Your first steps with D3.js (3 Points)

D3 is a JavaScript-library, so we can exclusively work in the JavaScript file. There is only one exception: We first need to load the D3-library inside of the HTML-file.

Task 1 a) - Install/Include the d3.js library in the HTML Header. Tipp: You can import the file like any other JavaScript file.

Now, we switch over to our main JavaScript file (*index.js*), and can start using functions from d3.

Task 1 b) - Check that D3 is loaded correctly by logging the library version to the console. You should get an output like:



```
D3 Version: 7.8.4                                                  index.js:2:9
```

Similarly, we also check that the data actually was loaded correctly. You should see a log output for the raw data.

Now, we want to dynamically add a SVG element to the HTML page. We first need to select an existing element by it's id. We will then append an SVG to the selected div.

Task 1 c) - Select the existing visualization container by it's id via d3.select. Tip: Remember how we select id's in CSS.

Task 1 d) – Append a new SVG element to the selected div with d3. You can use the dimensions we have specified in the JavaScript file.

Task 2: Some simple data processing (3 Points)

D3 core function is manipulating SVG elements, adding and moving lines, rectangles, or circles. But it also provides some functionality for data processing.

For this task, you do NOT need to code yourself. Instead, it's sufficient to describe the three code snippets we give you in the JavaScript file.

Task 2 a), b), c) - Find the respective line in the code and try to explain what is happening in the code block above. You can also add comments individual lines of the given code. A good tool here is to add console logging of intermediate results to understand the processing steps.

Task 3: Drawing the Line chart with D3 (10 Points)

Finally, it's time to use D3 to manipulate our SVG to create the line chart we show above.

Task 3 a) - First, we want to move our entire plot to keep some margins around the plot. Do this by appending a group to the SVG and apply a translation that respects the margins we have specified in the code.

Task 3 b) – Define scales to your D3 plot. We need these scales to position other elements like lines or circles. Save these scales to separate variables timeScale, *temperatureScale* and *rainfallScale*.
Task 3 c)/d) – Add circles for each rain and temperature data point. Make use of the initialized scales to determine the position of circles. You can use an arbitrary radius for the circles, e.g. r=3.

Task 3 e) - To plot data lines, you first need to create two line generator functions by calling d3.line with the respective data arrays.

Task 3 f) – Draw these lines by passing these line generator functions to SVG *path* object (which you add to the viewport). Tip: Check the slides of Tutorial Session 1 to see how you specify data for an SVG path.

Task 4: Adding Axes to our line chart (4 Points)

Finally, we want to draw some axes to the chart. We can do this by adding d3.axis objects based on the scales (timeScale, temperatureScale and rainfallScale) which we have created before. We have already added the axis for the time scale, displayed on the x-axis, for you.

Task 4 a) – Append an axis on the left side of the plot showing the temperature. Make sure to choose the correct scale.

Task 4 b) – Append on axis on the right side showing the rainfall.

Submission: Zipped linechart folder including all files. Please do not forgot to specify the names of both your team-partners in the submission file.