Programming Assignment 2

Due on Feb 12, 6pm.

This programming assignment assumes that you have fulfilled all course prerequisites and followed along with the tooling videos and readings.

Template and Submission

We will use **git** repositories for all programming assignments and project milestones. You should already be well acquainted with the process and all necessary git commands from the first programming assignment.

We created a git repository in your github student account (https://github.students.cs.ubc.ca/) that contains a template for all three tasks that should help you get started. The datasets are included in the /data folder. You will need to modify the .html, .js, and .css files as described below.

Submit the programming assignment by updating the given repository (`git push`). You can commit/push changes as often as you want before the deadline. Do not rename files or create new ones. Work alone.

We will also evaluate code readability and structure (add comments, indent code, use functions, ...).

Task 1: Interactive Scatter Plot

1. Familiarize yourself with the given template `task1_2.html` and the world indicator dataset in the */data* folder. See Technical Hints on the website for more details on git and instructions for starting a local web server.
    - The CSV file contains population, life expectancy, and GDP per capita statistics for 142 countries between 1952 and 2002, in a 10-year interval.
    - `task1_2.html` contains an HTML5 input slider that allows users to select a specific year. We have also included the D3 and jQuery libraries, and an external stylesheet.
2. **Data**
    - Load the CSV file with D3.
    - Select the top 50 countries by their population in 2002.
3. **Scatter plot**

- Create a scatter plot to visualize the data. We strongly recommend that you implement charts as JavaScript classes (see reading about reusable D3 components) as it will make it much easier to add interactivity and to link views afterwards. We provide a basic boilerplate `scatterplot.js` as part of the starter template.
- Each circle symbol corresponds to a country (y: life expectancy, x: GDP per capita, radius: population, hex color code: `#4682b4`).
- Use D3's enter-update-exit pattern for drawing the circles.
- Add SVG axis labels.

4. Connect the input slider (year) with the scatter plot.

   - Every time the user changes the year, the view must be updated accordingly.
   - We included jQuery—one of the most popular JavaScript libraries for DOM manipulation and event listeners—in the project. We will leave implementation details up to you but you can, for example, use the following jQuery code snippet to listen to slider changes instead of using plain JavaScript:

     ```
     $("#year-slider").on("input", function() {
         year = $(this).val();
         // todo
     });
     ```

     Use D3's `.transition()` method to show a smooth movement of circles every time the year selection changes.

5. **Element-wise interaction**

   - Change the circle fill color on hover (color code: `#71361c`).
   - Change the circle fill color on click (color code: `#b35227`).

     - The goal is to illustrate how the properties of one country change over time when the user changes the year selection.
     - A click on an active country (highlighted circle) should reset its state.
     - Max. 1 country can be active at a time.
     - When you hover over an active circle, it should show the hover color.

   - Show a tooltip with the country's name on hover.
     - There are many ways to display a text label next to a circle. One option is to detect mouse coordinates with D3 (`d3.event.pageX` and `d3.event.pageY`) when the user selects a circle and to use these coordinates to position a tooltip HTML element.

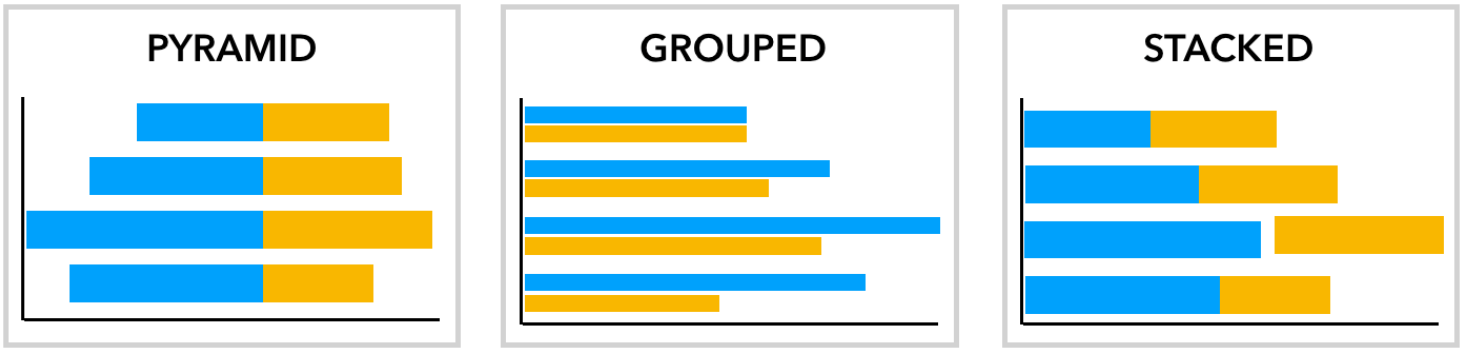
Task 2: Linked Views

1. Implement a column chart to visualize the population of the 5 most populated countries.
   - Sort descending by population.
   - `x` : country name, `y` : population
   - Format the tick labels on the y-axis as millions.
   - Use the D3 enter-update-exit pattern to draw the rectangles.
   - Rectangle color code: `#4682b4` .
2. Connect the bar chart to the input slider.
   - Show transitions, in the same way as for the scatter plot, every time the year selection changes.
   - The selected year might influence the top 5 populated countries and its order.
3. Change the rectangle fill color on hover (color code: `#71361c`).
4. Connect the views bidirectionally
   - Change the circle fill color (`#4682b4`) in the scatter plot every time the user hovers over a bar in the column chart, and vice versa.
   - Hovering over a circle or rectangle should not affect the active country in the scatter plot.

See exemplary result for task 1 and 2:

https://www.students.cs.ubc.ca/~cs-436v/20Jan/images/P2_task1_2.gif

Task 3: Chart Layouts



1. Familiarize yourself with the given template (task3.html, task3.js) and the canada_population.csv dataset in the /data folder.
   - The CSV file contains data about the age and sex structure of the population of Canada from the

2011 and 2016 Census. Source: https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/index-eng.cfm
    - `task3.html` contains <select> tags that allow users to select a specific year and layout. We have also included the D3 and jQuery libraries, and reuse the external stylesheet from Task 1-2.
    - `task3.js` contains D3 source code for loading the CSV file and for converting all values to numbers.

2. The goal of Task 3 is to visualize Canada's population by age and sex with D3.

    - Instead of showing a static visualization, you should allow users to switch between three different layouts: pyramid chart, grouped bar chart, and stacked bar chart.
    - Each rectangle/bar in your visualization must encode the female or male population. x-axis: population, y-axis: age-group
    - The demographics change rapidly and, therefore, you should allow users to inspect data from 2011 and 2016.
    - Use D3's enter-update-exit pattern and position all rectangles depending on the chosen layout. Don't remove all elements and redraw the chart from scratch every time the user changes the year or layout selection. D3's `.transition()` function ensures smooth transitions.
    - Draw x- and y-axes. You don't need axis transitions and you can just redraw an axis every time the user changes the layout. Make sure to include an x-axis for both the female and the male population group in the pyramid chart.
    - Add a color legend (either HTML elements or within SVG) for the female and male population.
    - Look at the source code of similar D3 blocks before you start with your implementation. Examples:
        - https://bl.ocks.org/HarryStevens/3d001d5e7ac8ea1eedc56bda01189008 *(missing axes)*
        - https://bl.ocks.org/ADJMLyon/db038850d5890d6aff43c145591c6f90
        - http://bl.ocks.org/RaphAllier/d37a74cf0b960a723ece44c1937ac6f4

Recommendation 1: Data wrangling

- Filter the data by the selected year at the beginning.
- Each row in the CSV file or object in the JavaScript array *data* corresponds to a 1-year age group and contains values for the female and male population. We recommend that you create two child objects for the male and female group for each year because each population group needs its own mark (rectangle).
- Use JS functions like .map(), .filter(), .reduce(), etc.

Recommendation 2: Rendering

- Create a nested D3 selection.
- First, create SVG groups for each age-group (1-year interval) and position them along the y-axis.
- Second, within each group, create two rectangles (male and female group) and change the position and size dynamically depending on the chosen layout.