CS49000-VIZ - Fall 2020  
Programming Assignment 2: User Interaction

**Key Dates**

|  |  |
| --- | --- |
| **Handed out:** | September 10, 2020 |
| Due date: | March 26, 2023 *(before 11:59 PM)* |

**Objectives**

This second programming assignment builds on the results of the first assignment and focuses this time on the interactivity of the visualization.

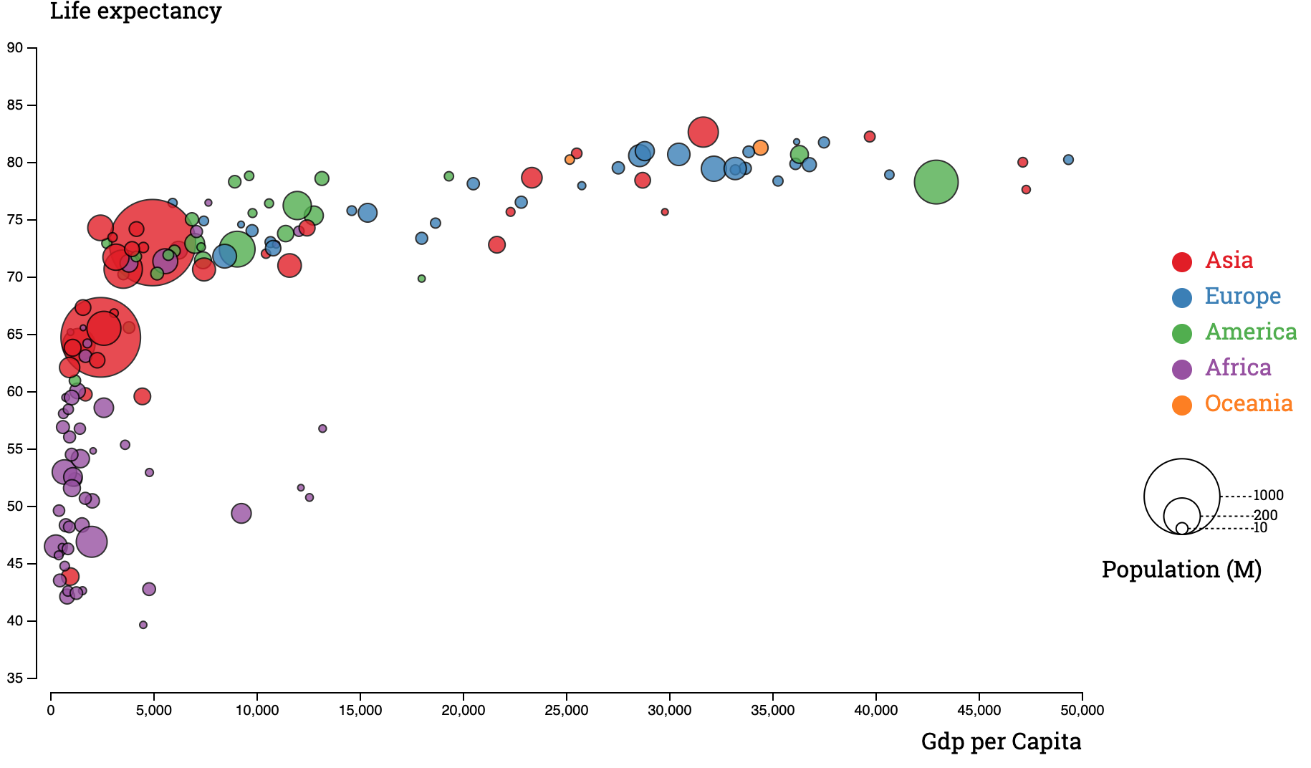
**Context**

The dataset for this project is a *country factbook* dataset that compiles a number of interesting attributes about many countries: birth rate, energy consumption, gross domestic product (GDP), health, national debt, and many more. Like the dataset of the first project, this is a tabular dataset. In this project, we will tackle the challenge posed by the visualization of this high-dimensional dataset through several interaction mechanisms.

**Tasks**

**Task 1: Bubble Chart**

In his famous 2006 [TED Talk](https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen) Hans Rosling [visualized](https://www.gapminder.org/tools/#$chart-type=bubbles) global health statistics in impressive and engagin fashion. For this first task, I am asking you to implement a basic bubble chart visualization, in other words a scatter plot in which individual data points are drawn as circles whose *area* and *color* vary as a function of two variables (in addition to the two variables that control the horizontal and vertical position of each circle).

[](https://www.d3-graph-gallery.com/bubble)

To demonstrate your bubble chart implementation, you will create following visualization: The *horizontal* (x) coordinate corresponds to 'GDP per capita', the *vertical* (y) coordinate corresponds to 'Life expectancy at birth', the *area* corresponds to 'Population', and the *color* corresponds to 'Birth rate'. Make sure to select scales, ranges, and color maps appropriately.

**Deliverable:** p2\_bubbles.py

**Task 2: Widgets**

In the [previous task](https://www.cs.purdue.edu/CS49000-VIZ/projects/assignment2.html#bubble%20chart) you were asked to assign specific (and arbitrary) attributes to each degree of freedom of your bubble chart. In reality, the visualization would be much more useful if the user could decide at runtime what attributes should be displayed and how. The goal of the second task is to create a user interface that makes this dynamic selection possible. Practically, you will augment the visualization of the first task with [widgets](https://en.wikipedia.org/wiki/Widget_(GUI)) that collectively allow one to select for each of *x*, *y*, *radius*, and *color*, what attribute of the dataset should be used. To do so, your widget should display a list of the existing attributes. Here, the selection of each displayed attribute should be made through a drop down menu. An example integrating a drop down menu in a Bokeh program is shown [here](https://docs.bokeh.org/en/latest/docs/user_guide/interaction/widgets.html#dropdown). Things are a bit less straightforward for Matplotlib and it is usually easier to use a third party GUI library (such as [TkInter](https://docs.python.org/3/library/tkinter.html) or [PyQt5](https://pypi.org/project/PyQt5/)) and to manually bind the visualization to the UI widgets. [Here](https://pythonspot.com/tk-dropdown-example/) is an example of dropdown menu in TkInter, and [here](https://pythonprogramming.net/drop-down-button-window-styles-pyqt-tutorial/) is another one in PyQt5.

A second aspect of the visualization that the user might want to control interactively is the scaling of the area with respect to the selected attribute. To enable this, you will add a second widget to your visualization, namely a slider bar whose value selection will be tied to the area of the circles in your bubble chart visualization. An example of slider bars associated with a Bokeh visualization is demonstrated [here](https://docs.bokeh.org/en/latest/docs/gallery/slider.html). Similar examples exist for [TkInter](https://pythonbasics.org/tkinter-scale/) and [PyQt5](https://www.geeksforgeeks.org/pyqt5-qslider-python/) as well.

Note that your visualization must be automatically updated every time a change is made to any widget selection.

**Deliverable:** p2\_widgets.py

**Task 3: Linked Brushing**

We saw in the [first project](https://www.cs.purdue.edu/CS49000-VIZ/projects/assignment1.html) that a matrix of scatter plots (plotting all attributes vs. one another) offers a comprehensive view of the data through a set of a 2D projections. For this third task, we will follow a similar approach and display four bubble charts in a 2-by-2 array to visualize more attributes than what is possible with a single bubble chart. The user should be free to select different attributes to control the appearance of each chart. For that, you will need to quadruple the number of the widgets created in the previous task.

One issue that you will face with this visualization is to understand how data points and patterns present in each chart relate to each other across charts. One way to mitigate this problem consists in selecting the same attributes to control the radius and/or color of the bubbles in all charts. An alternative, and more flexible, solution is too use what is known as linked brushing. Using linked brushing, the local selection of some data points in a visualization triggers the selection of the same data points in another visualization. You will implement this method for this third task and make sure that a selection made in any of the four windows is replicated in the other three. Practically, your implementation should allow for data points to be selected within a rectangular region drawn by the user and this selection should be visually accompanied by a change in the appearance of the selected items across all four windows to make them easy to spot. As always, you can find [examples of linked brushing](https://docs.bokeh.org/en/latest/docs/user_guide/interaction/linking.html#linked-brushing) in Bokeh. [Third party libraries](http://docs.glueviz.org/en/stable/index.html) seem needed here as well to achieve the same result within a Matplotlib implementation.

**Deliverable:** p2\_brushing.py

**Task 4: Details on Demand**

Though four bubble charts allow one to display a substantial number of attributes at once, they can only convey a relatively small portion of the information available in this dataset. A typical means to overcome this limitation is to give the user the option to query additional details about the items included in the visualization (so-called "details-on-demand" in the language of [Shneiderman's mantra](http://www.ifp.illinois.edu/nabhcs/abstracts/shneiderman.html)).

To afford access to additional details, you will use the *hover* event mechanism to trigger the display of all the attributes associated with a given data point in a tooltip. Specifically, upon moving the mouse over a particular data point, a window should appear to the side of the corresponding location that displays the name of the country along with a series of (attribute name, value) pairs that show a subset of the information available for this country. In addition, the country selected should be highlighted across all four charts. Your fourth task is to add this feature to the visualization tool that you created in [Task 3](https://www.cs.purdue.edu/CS49000-VIZ/projects/assignment2.html#brushing). As a starting point for this task, you should find helpful to refer to the explanations given [here](http://docs.glueviz.org/en/stable/index.html) for Bokeh and [here](https://stackoverflow.com/questions/7908636/possible-to-make-labels-appear-when-hovering-over-a-point-in-matplotlib) for Matplotlib.

**Deliverable:** p2\_tooltip.py

**Data Set**

The dataset is available in CSV format [here](https://www.cs.purdue.edu/CS49000-VIZ/projects/data/project2/factbook.csv).

**Submission**

Submit your solution for this project on *Brightspace* before March 26, 2023 at 11:59 pm. The instructions below are the same as the ones provided for the first programming assignment.

* Include all the requested python code files: p2\_bubbles.py, p2\_widgets.py, p2\_brushing.py, p2\_tooltip.py
* Include all submitted files in a single directory named **<myLogin>\_p2**, where **<myLogin>** is your Purdue login. Do not include the CSV file in your submission.
* Do not use an absolute path to access the dataset. Assume instead that it will be present in the same directory as your html files.

*Last modified March 20, 2023*