cse6242-2019-spring-hw2

CSE 6242 / CX 4242: Data and Visual Analytics Georgia Tech, Spring 2019

Homework 2 : D3 Graphs and Visualization

Prepared by our 30+ wonderful TAs of CSE6242A,Q,OAN,O01,O3/CX4242A for our 1200+ students

Submission Instructions and Important Notes:

It is important that you read the following instructions carefully and also those about the deliverables at the end of each question or **you may lose points**.

- ❏ Always check to make sure you are using the most up-to-date assignment (version number at bottom right of this document).
- ❏ Submit a single zipped file, called “HW2-{GT username}.zip”, containing all the deliverables including source code/scripoints, data files, and readme. Example: “HW2-jdoe3.zip” if GT account username is “jdoe3”. Only .zip is allowed (no other format will be accepted). Your GT username is the one with letters and numbers.
- ❏ You may discuss high-level ideas with other students at the "whiteboard" level (e.g., how cross validation works, use hashmap instead of array) and review any relevant materials online. However, each student must write up and submit his or her own answers.
- ❏ All incidents of suspected dishonesty, plagiarism, or violations of the Georgia Tech Honor Code will be subject to the institute’s Academic Integrity procedures (e.g., reported to and directly handled by the Office of Student Integrity (OSI)). Consequences can be severe, e.g., academic probation or dismissal, grade penalties, a 0 grade for assignments concerned, and prohibition from withdrawing from the class.
- ❏ At the end of this assignment, we have specified a folder structure **you must use** to organize your files in a single zipped file. 5 points will be deducted for not following this strictly.
- ❏ In your final zip file, do not include any intermediate files you may have generated to work on the task, unless your script is absolutely dependent on it to get the final result (which it ideally should not be).
- ❏ We may use auto-grading scripts to grade some of your deliverables, so it is extremely important that you strictly follow our requirements.
- ❏ Wherever you are asked to write down an explanation for the task you perform, stay within the word limit or you may lose points.
- ❏ Every homework assignment deliverable and every project deliverable comes with a 48-hour "grace period". Any deliverable submitted after the grace period will get zero credit. We recommend that you plan to finish by the beginning of the grace period in order to leave yourself time for any unexpected issues which might arise.
- ❏ We will not consider late submission of any missing parts of a homework assignment or project deliverable. To make sure you have submitted everything, download your submitted files to double check. You may re-submit your work before the grace period expires. Canvas automatically appends a “version number” to files that you re-submit. You do not need to worry about these version numbers, and there is no need to delete old submissions. We will only grade the most recent submission.

Grading

The maximum possible score for this homework is 100 points.

Students in the CX4242 undergraduate section can choose to complete any 85 points worth of work to receive the full 15% of the final course grade. For example, if a CX4242 student scores 100 pts, that student will receive (100 / 85) * 15 = 17.65 points towards the final course grade. To receive the full 15% score, students in the CSE6242 sections will need to complete all 100 points.

===== Important Prerequisites =====

Download the HW2 Skeleton that contains files you will use in this homework.

We highly recommend that you use the latest Firefox browser to complete this homework. We will grade your work using **Firefox 64.0 (or newer)**.

For this homework, you will work with version 5 of D3, provided to you in the **lib** folder. You must NOT use any d3 libraries (d3*.js) other than the ones provided.

You may need to setup an HTTP server to run your D3 visualizations (depending on which web browser you are using, as discussed in the D3 lecture (OMS students: the video "Week 5 - Data Visualization for the Web (D3) - Prerequisites: Javascript and SVG". Campus students: see lecture PDF.). The easiest way is to use http.server for Python 3.x, or SimpleHTTPServer for Python 2.x. **You should run your local HTTP server in the root (hw2-skeleton) folder.**

All d3*.js files in the **lib** folder must be referenced using **relative paths**, e.g., **"../lib/<filename>**" in your html files (e.g., those in folders Q2, Q3, etc.). For example, suppose the file "Q2/graph.html" uses d3, its header should contain:
<script type="text/javascript" src="../lib/d3.v5.min.js"></script>
It is incorrect to use an absolute path such as:
<script type="text/javascript" src="http://d3js.org/d3.v5.min.js"></script>

All questions that require reading from a dataset require you to submit the dataset in the deliverables too. In your html/js code**,** use a **relative path** to read in the dataset file**.** For example, since Q4 requires reading data from the `heatmap.csv` file (which should be submitted as part of the deliverables in the Q4 folder), the path should simply be 'heatmap.csv' and **NOT** an absolute path such as "C:/Users/polo/HW2-skeleton/Q4/heatmap.csv".

You can and are encouraged to decouple the style, functionality and markup in the code for each question. That is, you can use separate files for css, javascript and html.

==========

Q1 [10 points] Designing a good table. Visualizing data with Tableau.

Imagine you are a data scientist working with data that documents employment outcomes for various college majors. Perform *task a* below to help your organization analyze the relationship between the major categories

Humanities & Liberal Arts and *Computers/Mathematics* in relation to employment outcomes. Perform *task b* to help your organization better understand overall graduate employment outcomes in relation to particular areas of study within these major categories.

a. **[5 points] Good table design.** Create a table to display the details of the data contained in *all-ages.csv*#. You can use any tool (e.g., Excel, HTML) to create the table.

- The table should contain data from the following columns: *Major Category, Major, Employed, Unemployed, Unemployment rate*

- **Save the table as table.png.**

You may reorder the columns while creating the table. Keep suggestions from lecture in mind when designing your table. You are not required to use only the techniques described in lecture. For OMS students, the online lecture video pertaining to this topic is *Week 4 - Fixing Common Visualization Issues - Fixing Bar Charts, Line Charts*). For campus student, please review slide 43 and onwards of the lecture slides.

b. **[5 points] Tableau:** Visualize the *grad-students* data as a *stacked bar chart*. Your chart should display the counts for columns *Grad employed, Grad Total,* and *Grad unemployed* using the dataset *grad-students.csv*[1] (in Q1 folder). (Optional reading: the effectiveness of stacked bar charts is often debated --- sometimes, they can be confusing, difficult to understand, and may make data series comparison challenging.)

Our main goal here is for you to try out Tableau, a popular information visualization tool. Thus, we keep this part more open-ended, so you can practice making design decisions. **We will accept most designs from you all.** We show one possible design in the figure below, based on the tutorial from Tableau, and you are not limited to the techniques presented there.

Please follow the instructions below:

- Your design should visualize the values of the columns *Grad employed, Grad Total,* and *Grad unemployed* for each of the majors within the major categories *Humanities & Liberal Arts* and *Computers/Mathematics*
- Your design should utilize a stacked bar chart to show the count for each of the aforementioned columns
- Your design should have clear label axes and a clear chart title. Include a legend for your chart.
- **Save the chart as barchart.png.**

Tableau has provided us with student licenses for *Tableau Desktop*, available for Mac and Windows. Go to tableau activation and select “Tableau Desktop”. After the installation, you will be asked to provide an activation key, which you can find on Canvas. (For OMS students: visit here. For campus students: visit here.). This key is for your use in this course only. **Do not share the key with anyone.**

If you do not have access to a Mac or Windows machine, please use the 14-day trial version of *Tableau Online*:

1. Visit https://www.tableau.com/trial/tableau-online
2. Enter your information (name, email, GT details, etc)
3. You will then receive an email to access your Tableau Online site
4. Go to your Site and create a workbook

One final option, if neither of the above methods work, is to take advantage of Tableau for Students**.** Follow the link and select “Get Tableau For Free”. You should be able to receive an activation key which offers you a one-year use of Tableau Desktop at no cost by providing a valid Georgia Tech email. Note that it is unclear whether Tableau intends for these licenses to be renewable, so you may only be eligible to receive one in the event that you have never used a *Tableau for Students* license before.

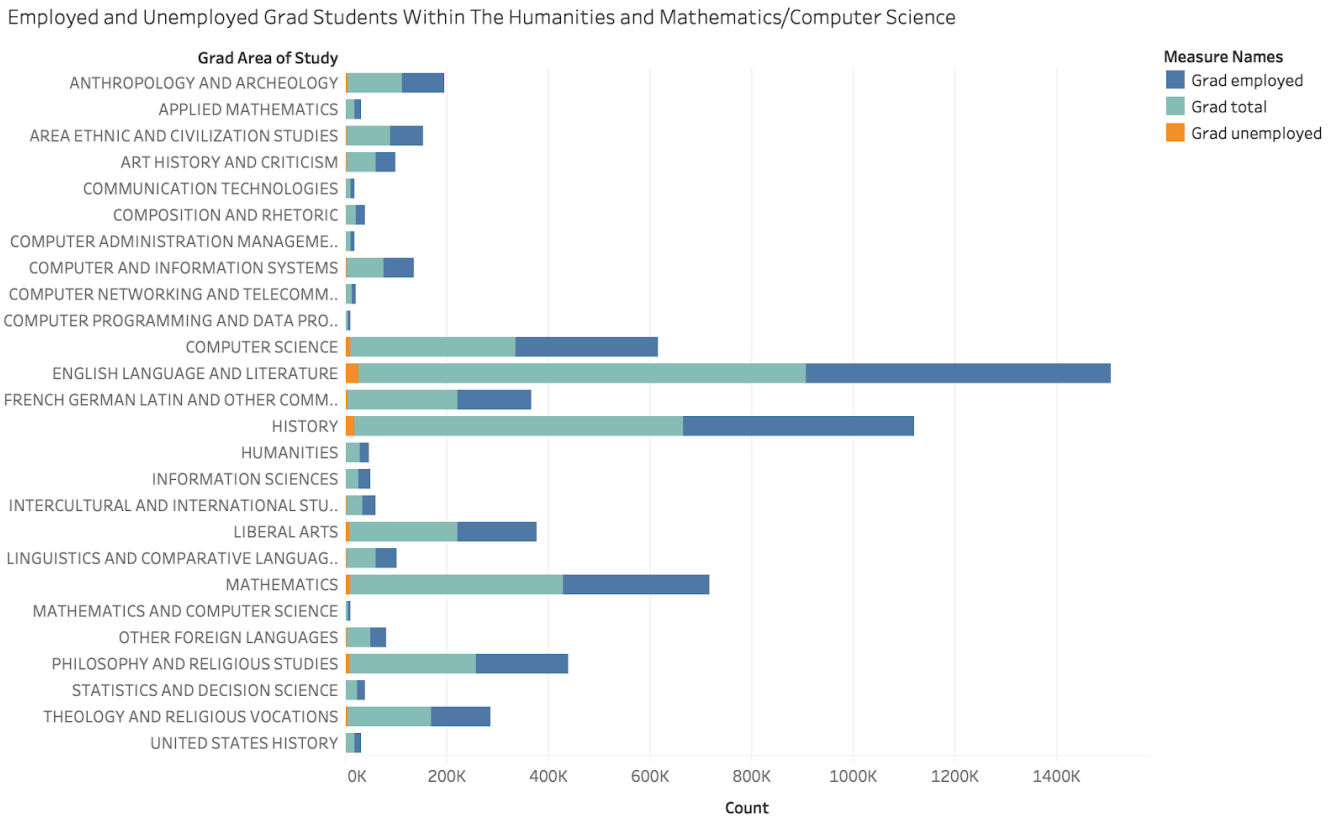

Figure 1: Example of a stacked bar chart

Q1 Deliverables:
The directory structure should be as follows:

Q1/
table.png
barchart.png
all-ages.csv
grad-students.csv

- **table.png** - An image/screenshot of the table in Q1.a (png format **only**).
- **barchart.png** - An image of the chart in Q1.b (png format **only)**, Tableau workbooks will not be graded!). The image should be clear and of high-quality.
- **all-ages.csv** and **grad-students.csv** - the datasets

Q2 [15 points] Force-directed graph layout

You will experiment with many aspects of D3 for graph visualization. To help you get started, we have provided the graph.html file (in the Q2 folder).

Note: You are welcome to split graph.html into graph.html, graph.css, and graph.js. Please also make certain that any paths in your code are **relative paths**. Nonfunctioning code will result in a **five point deduction.**

a. **[3 points] Adding node labels**: Modify graph.html to show a node label (the node name, i.e., the source) below each node. If a node is dragged, its label must move with it.

b. **[3 points] Styling links**: Style the links based on the "value" field in the links array. Assign the following styles:

If the value of the edge is equal to 0, the link should be **green** and **thick**.

If the value of the edge is equal to 1, the link should be **blue** and **thin**.

c. **[3 points] Scaling node sizes:**

1. Scale the radius of each node in the graph based on the degree of the node (you may try linear or squared scale, but you are not limited to these choices).

Note: Regardless of which scale you decide to use, you should avoid extreme node sizes (e.g., nodes that are mere points, or barely visible, as well as very large nodes). Failure to do so will result in a poor quality visualization.

d. **Pinning nodes** (fixing node positions):

1. **[2 points]** Modify the code so that when you double click a node, it pins the node's position such that it will not be modified by the graph layout algorithm (note: pinned nodes can still be dragged around by the user but they will remain at their positions otherwise). Node pinning is an effective interaction technique to help users spatially organize nodes during graph exploration.
2. **[2 points]** Mark pinned nodes to visually distinguish them from unpinned nodes, e.g., pinned nodes are shown in a different color, border thickness or visually annotated with an "asterisk" (*), etc.
3. **[2 points]** Double clicking a pinned node should unpin (unfreeze) its position and unmark it.

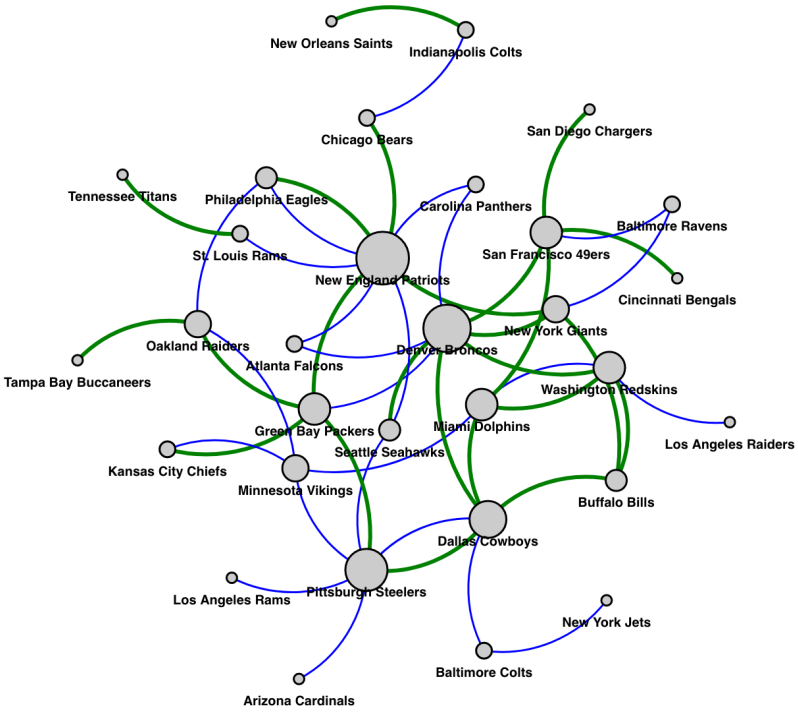

Figure 2a. Example Visualization

Q2 Deliverables:

The directory structure should be as follows:

Q2/

graph.html

graph.js, graph.css (if not included in graph.html)

- **graph.html** - the html file created.
- **graph.(js / css)** - the js / css files if not included in graph.html

Q3 [15 points] Scatter plots

Use the dataset[2] provided in the file *movies.csv* (in the Q3 folder) to create a scatter plot.

Refer to the tutorial for scatter plot here.
Attributes in the dataset:

Feature 1: Id
Feature 2: Title
Feature 3: Year
Feature 4: Runtime
Featurn 5: Country
Feature 6: Rating
Feature 7: Votes
Feature 8: Budget
Feature 9: Gross
Feature 10: Wins and nominations
Feature 11: Is good rating? (value 1 means “good”, value 0 - “bad”)

a. [8 points] Creating scatter plots:

1. **[6 points] Create two scatter plots**, one for each feature combination specified below. In the scatter plots, visualize “good rating” class instances as blue crosses, and “bad rating” instances as red circles. Add a legend to the top right corner showing the symbols’ mapping to the classes.
   - Feature 10 (Wins and nominations) vs. Feature 6 (Rating)
     - Figure title: Wins+Nominations vs. Rating
     - X axis (horizontal) label: Rating
     - Y axis (vertical) label: Wins+Noms
   - Feature 8 (Budget) vs. Features 6 (Rating)
     - Figure title: Budget vs. Rating
     - X axis (horizontal) label: Rating
     - Y axis (vertical) label: Budget

2. **[2 points]** In **explanation.txt**, use no more than 50 words to discuss which feature combination is better at separating the classes and why.

Note: Your scatter plots should be placed one after the other **on a single HTML page**, similar to the example image below (Figure 3). Note that your design need NOT be identical to the example.

b. [3 points] Scaling symbol sizes. Create a scatter plot (append to the HTML page) using the feature combination specified below. Set the size of each symbol to be proportional to the value of Feature 10 (Wins and nominations); use a good scaling coefficient to make the scatter plot legible, visually attractive and meaningful. Visualize “good rating” class instances as blue crosses, and “bad rating” instances as red circles.
   - Feature 7 (Votes) vs. Feature 6 (Rating) sized by Feature 10 (Wins+Nominations)
     - Figure title: Votes vs. Rating sized by Wins+Nominations
     - X axis (horizontal) label: Rating
     - Y axis (vertical) label: Votes

c. [4 points] Axis scales in D3. Create two plots for this part (append to the HTML page) to try out two axis scales in D3: the first plot uses the square root scale for its y-axis (only), and the second plot uses the log scale for its y-axis (only). In **explanation.txt,** explain when we may want to use square root scale and log scale in charts, in no more than 50 words.

Note: the x-axes should be kept in linear scale, and only the y-axes are affected.
Hint: You may need to carefully set the scale domain to handle the 0s in data.

   - First Figure: uses the square root scale for its y-axis (only)
     - Figure title: Wins+Nominations (square-root-scaled) vs. Rating
     - X axis (horizontal) label: Rating
     - Y axis (vertical) label: Wins+Noms
   - Second Figure: uses the log scale for its y-axis (only)
     - Figure title: Wins+Nominations (log-scaled) vs. Rating
     - X axis (horizontal) label: Rating
     - Y axis (vertical) label: Wins+Noms

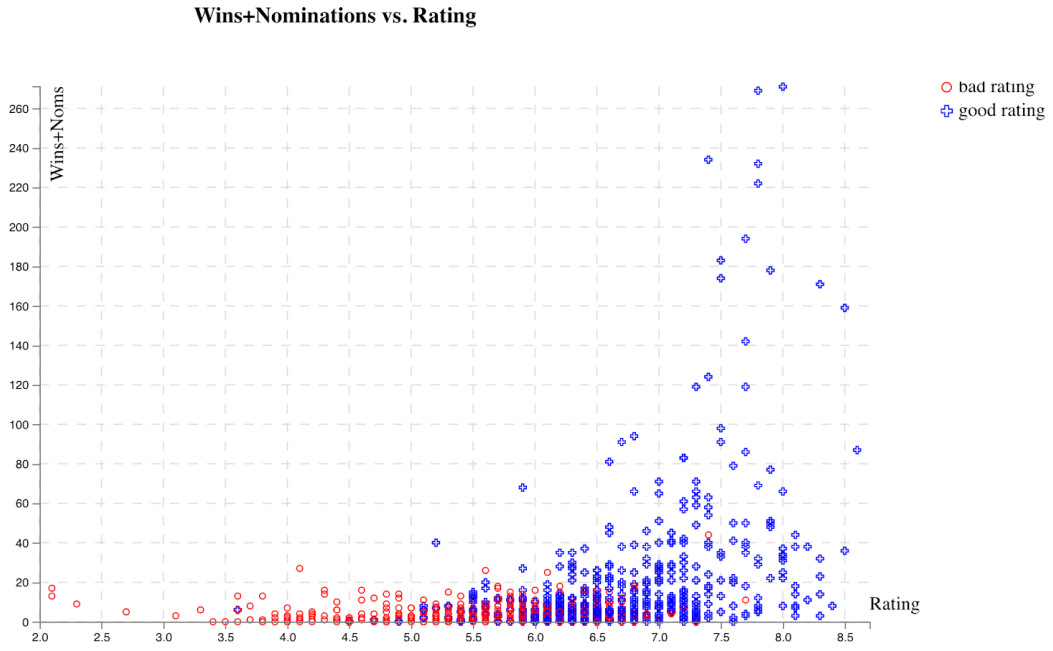

Figure 3a: Example for scatter plots.

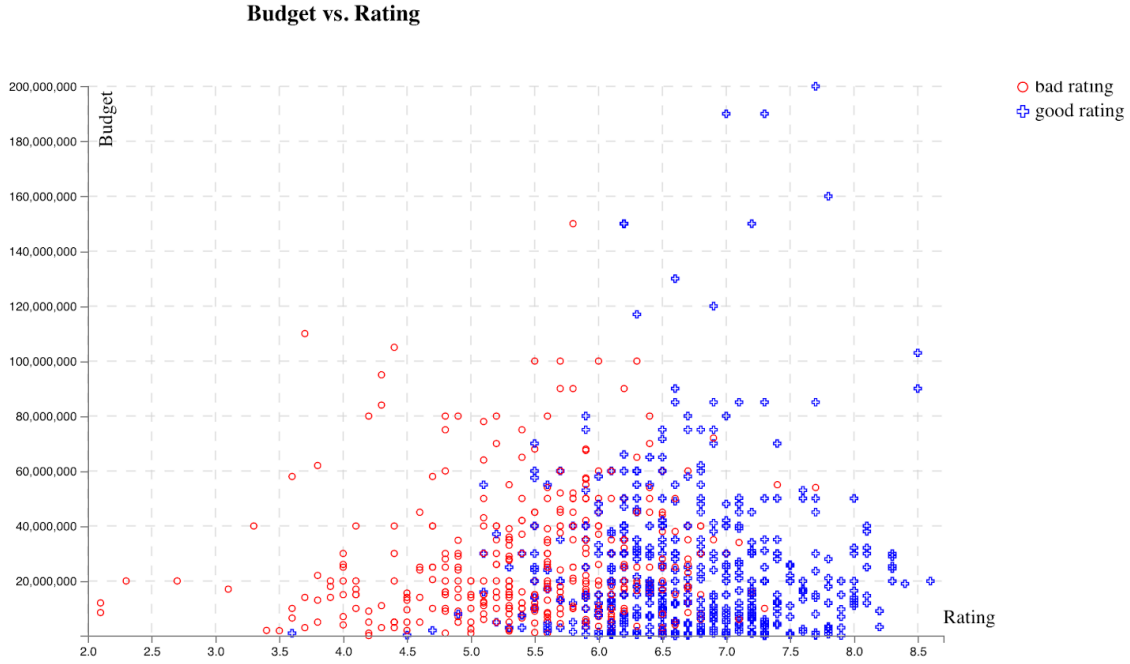

Figure 3b: Example for scatter plots.

Q3 Deliverables:
The directory structure should be organized as follows:

Q3/
scatterplot.(html / js / css)
explanation.txt
scatter_plots.pdf
movies.csv

- **scatterplot.(html / js / css)** - the html / js / css files created.
- **explanation.txt** - the text file explaining your observations for Q3.a.2 and Q3.c.

● **scatter_plots.pdf** - a PDF document showing the screenshots of the five scatter plots created above (two for Q3.a.1, one for Q3.b and two for Q3.c). You may print the HTML page as a PDF file, and each PDF page shows one plot (**hint:** use CSS page break). Clearly title the plots as instructed (see examples in Figure 3).
● **movies.csv** - the dataset.

Q4 [15 points] Heatmap and Select Box

Example: 2D Histogram, Select Options

Use the dataset provided in *heatmap.csv* (in the Q4 folder) that describes the number and type of crimes in each of the 5 boroughs of New York City. Visualize the data using D3 heatmaps.

a. **[5 points]** Create a file named *heatmap.html*. Within this file, create a heatmap of the number of crimes for each crime type that occured in each borough for the year 2011. Place the crime type on the heatmap's horizontal axis and the name of the borough on its vertical axis.

b. [**1 point**] The color scheme of a heatmap is a very important part of its design. The number of crimes for each borough should be represented by colors in the heatmap. Pick a meaningful color scheme (hint: color gradients) with 9 color gradations for the heatmap.

c. **[3 pt]** Add axis labels and a legend to the chart. Place the name of the boroughs ("Manhattan", "Brooklyn", "Queens", etc.) on the vertical axis in alphabetical order (i.e. top → bottom: A → Z). Place the crime type ("Murder", "Assault", "Shooting", etc.) on the horizontal axis also in alphabetical order (i.e. left → right: A → Z).

d. **[6 pt]** Now create a drop down select box with D3 that is populated with the years (2011, 2012, 2013, 2014, 2015). When the user selects a different year in this select box, the heatmap and the legend should both be updated with values corresponding to the selected year. Note the differences in the legends for years 2011 and 2012 in Figure 4a and Figure 4b below. **While the 9 color gradations in the legend remain the same, the thresholds values are different.** The default year when the page loads should be 2011.

Note:

1. The NYC Crime Statistics being used here have been synthetically generated. They do not accurately represent the actual NYC crime records for the stated time period.
2. The data provided in heatmap.csv would need to be “reshaped” in such a way that it can produce the expected output. **All data reshaping must only be performed in javascript; you must not modify heatmap.csv**. That is, your code should first read the data from heatmap.csv file as is, then you may reshape that loaded data using javascript, and then use it to create the heatmap.
3. **The threshold values should not be hardcoded**. They do not necessarily have to match the ones provided in the screenshots below.

The screenshots provided below serve as an example only. You are not expected to produce an exact copy of the screenshots. Please feel free to experiment with fonts, placement, colors, etc., as long as the output looks reasonable for a heatmap.

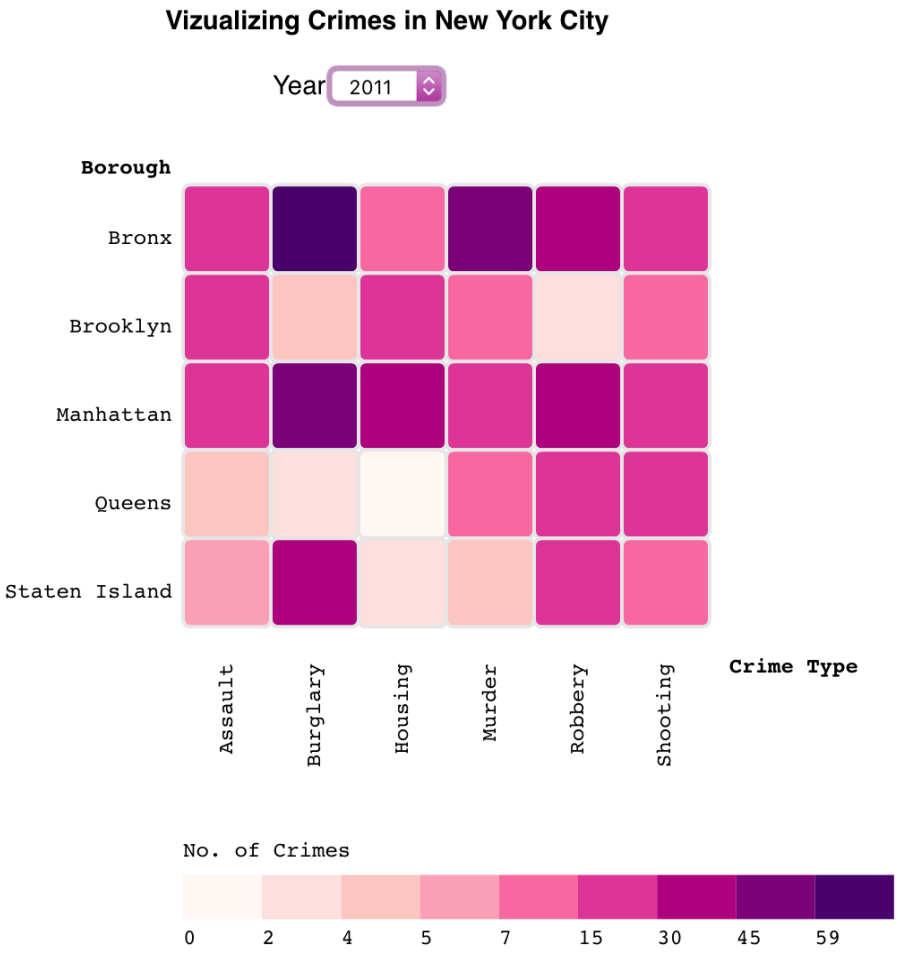

Figure 4a: Number of crimes of each type that happened in year 2011.

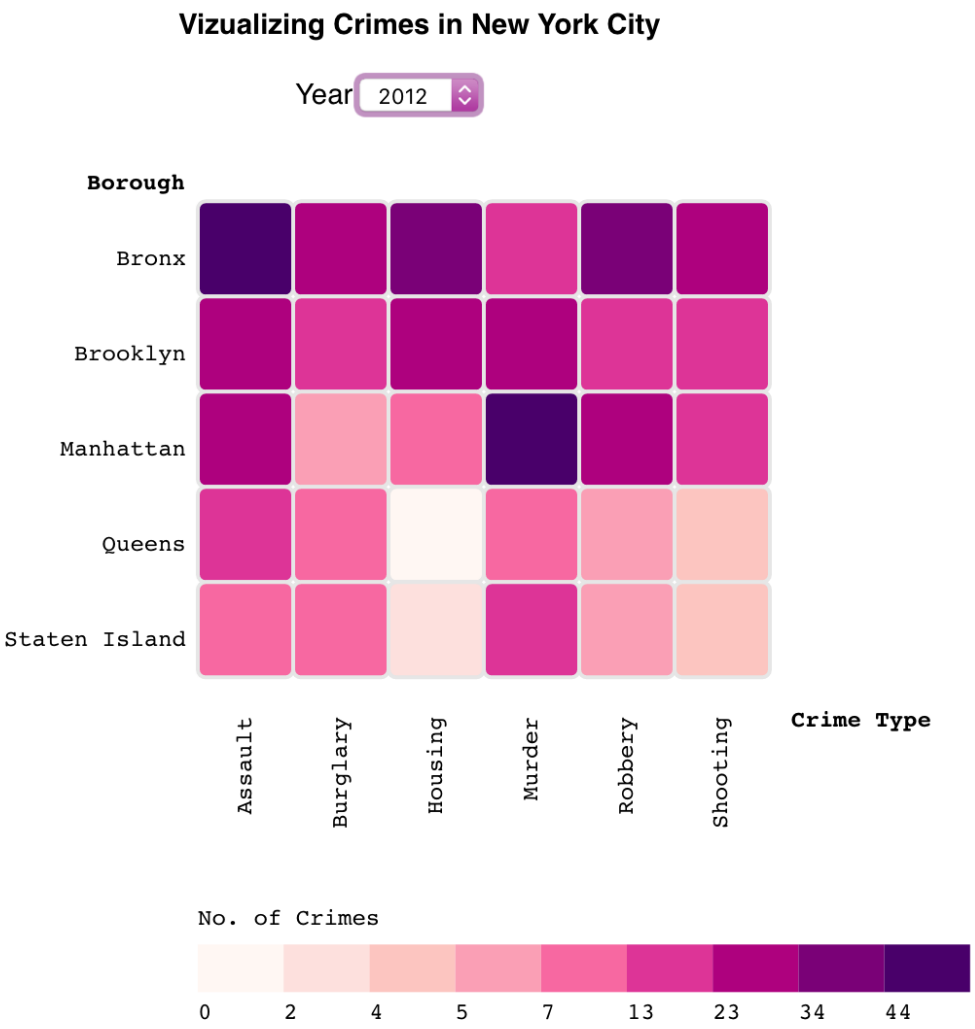

Figure 4b: Number of crimes of each type that happened in year 2012.

Q4 Deliverables:
The directory structure should look like:

Q4/
heatmap.(html / js /css)
heatmap.csv

- **heatmap.(html / js/ css)** - the html / js / css files created.
- **heatmap.csv** - the dataset

Q5 [20 points] Interactive Visualization

Use the dataset[3] provided in the dataset.txt file (in the Q5 folder) to create an interactive bar chart. Each line in the file represents rural population growth (per year) of countries over the past five years, starting with total rural population of year 2012.
You will copy the data contained in dataset.txt and paste it, directly into your code as is, as an array variable named **data**, similar to what is shown below.

Note: You must NOT modify or reorder the content of the data file; what you paste into your code should be the same content that the data file contains. If you believe you want to sort or order the data in any way (e.g., by population), do so using javascript.

Example: `<script> var data=[<paste data file content here>];</script>`

a. **[5 points]** Create a **horizontal bar chart** with its vertical axis denoting the country names (ordered by population) and its horizontal axis denoting the total cumulative population (with "," as the thousand separator) at the end of 2017. Each bar should have its associated total population shown on top of it. Refer to the example shown in Figure 5a.
Note: The vertical axis of the chart should use country names as labels.

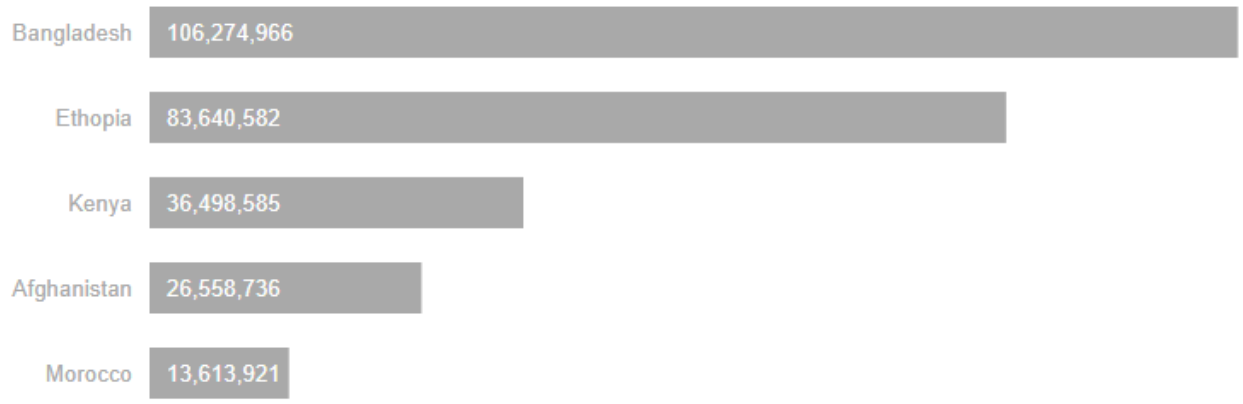

Figure 5a. Bars representing **total** rural population of each country

b. **[10 points]** When hovering the mouse over a bar, a smaller line chart representing the population growth of that country for each year (2013-2017) should be displayed in the top right corner. For example, **Bangladesh** has a growth value of 0.04%, 0.0008%, 0.05%, 0.10%, 0.15% for the years 2013, 2014, 2015, 2016 and 2017 respectively. On hovering over the bar representing **Bangladesh**, a line chart depicting these 5 values in % is displayed. See Figure 5b for an example.

The calculation to show the percentage of population growth is:

$$Population\ Growth\ \% = (Change\ in\ Population\ /\ Previous\ Year\ Population\ ) * 100$$

Figure 5b. On hovering over the bar for **Bangladesh**, a smaller line chart representing its percentage of growth per year in decimal over the past 5 years is displayed at the top right corner.

c**. [3 points]** On mouse out, the line chart should no longer be visible.

d. **[2 points]** On hovering over any horizontal bar representing a country, the color of the bar should change. You can use any color that is visually distinct from the regular bars. On mouseout, the color should be reset.

Q5 Deliverables:
The directory structure should be as follows:

Q5/
interactive.(html/js/css)

interactive.(html/js/css) - The html, javascript, css to render the visualization in Q5 (dataset.txt is *NOT* required to be included in the final directory structure as the data provided in dataset.txt should have already been integrated into the "data" variable in your code).

Q6 [20 points] Choropleth Map of County Data

Example of choropleth map: Unemployment rates

Use the provided dataset[4] in ***county_poverty.csv***, ***county_detail.csv, and us.json*** (in the Q6 folder) and visualize them as a choropleth map.

- Each record in *county_poverty.csv* represents a county and is of the form `<CensusId,State,County,Poverty>`, where
  - `CensusId`: the id of one county in the US. e.g., 01001.
  - `State`: the name of the state which the county belongs to. e.g., Alabama .
  - `County`: the county name. e.g., Autauga
  - `Poverty`: is poverty rate (i.e. the percentage of poverty people living in that county). e.g., 5.86 mean the poverty rate in this county is 5.86% (the poverty rates in the county_poverty.csv file have been slightly modified from the original values and do not represent the official figures).

- The *county_detail.csv* file contains a list of records, each having 3 fields in the form `<CensusId,TotalPop,IncomePerCap>`, where:
  - `CensusId`: the id of one county in the US
  - `TotalPop`: total population (i.e., the total number of people living in the county)
  - `IncomePerCap`: the income per capita for people who live in the county.
- The *us.json* file is a TopoJSON *topology* containing three geometry collections: *counties*, *states*, and *nation*.

a. **[15 points]** Create a choropleth map using the provided datasets, use Figure 6 below as reference.

1. **[10 points]** The color of each county should correspond to poverty rate in that county (`Poverty field` in *county_poverty.csv*.). i.e., darker colors correspond to higher poverty rate in that county and lighter colors correspond to lower poverty rate in that county. Use gradients of only **one** particular hue. Use promises (part of the d3.v5.min.js file present in the lib directory; there is no need to download or install anything) to easily load data from multiple files into a function. Use topojson (present in **lib** folder) to draw the choropleth map.
2. **[5 points]** Add a **vertical** legend showing how colors map to the poverty rate. (In the example shown in Figure 6, there are 9 color gradations and you must use exactly 9 in your submission as well.)

b. **[5 points]** Add a tooltip using the d3-tip.min library (in the **lib** folder) on hovering over a county. The tooltip shows the following information on each line: (1) state name, (2) county name, (3) poverty rate, (4) total population, and (5) income per capita. The tooltip should appear when the **mouse** hovers over the county. On mouseout, the tooltip should disappear. Use Figure 6 below as reference. We recommend that you position the tooltip some distance away from the **mouse cursor**, which will prevent the tooltip from "flickering" as you move the mouse arou nd quickly (the tooltip disappears when your mouse leaves a county and enters the tooltip's bounding box). Please ensure that the tooltip is fully visible (i.e., not clipped, especially near the page edges).
Note: You **must** create the tooltip by **only** using **d3-tip.min.js** in the **lib** folder.

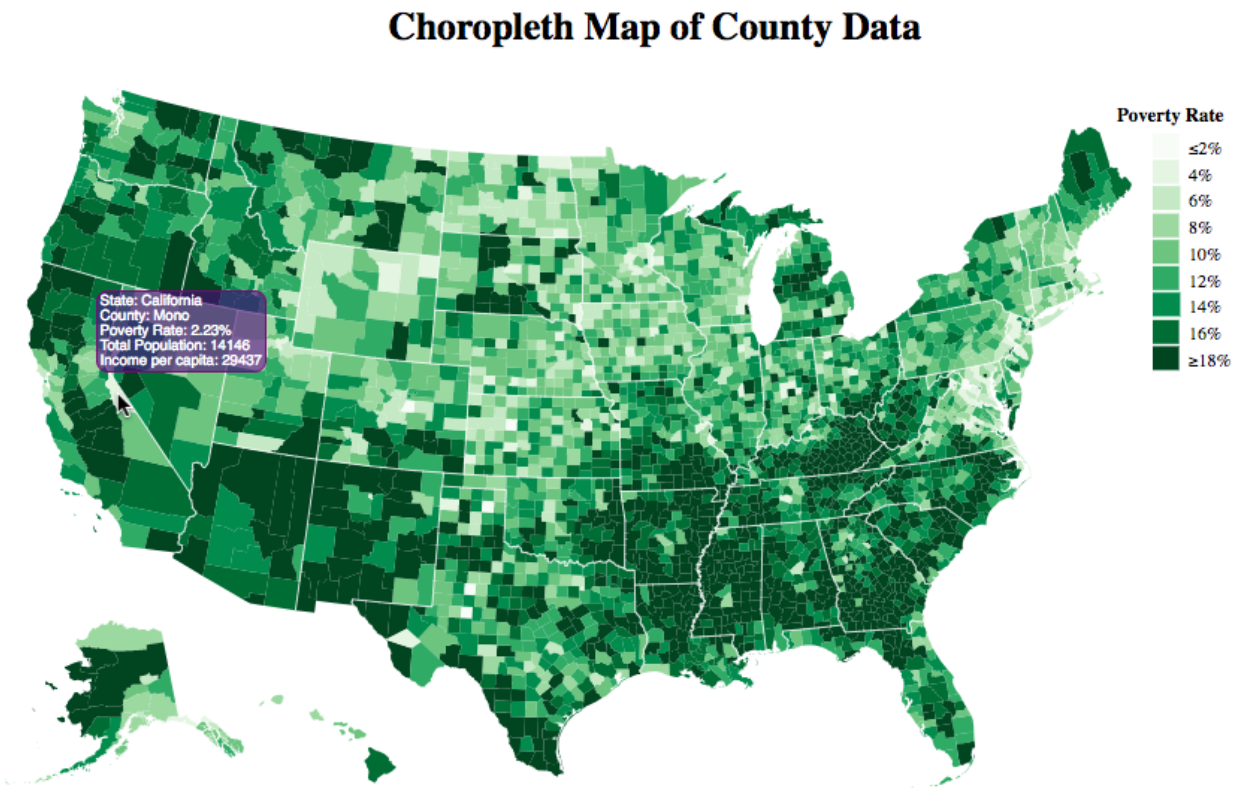

Figure 6. Reference example for Choropleth Maps

Q6 Deliverables:
The directory structure should be organized as follows:

Q6/
q6.(html/js/css)
county_poverty.csv
county_detail.csv
us.json

- **q6.(html /js /css**)- The html/js/css file to render the visualization.
- **county_poverty.csv, county_detail.csv** - The datasets used to show the information of each county.
- **us.json** - Dataset needed to draw the map.

Q7 [5 points] Pros and Cons of Visualization Tools

This is an open-ended question. Your answer will depend on what you have learned from working through the questions in this assignment, and your personal experience.

Pick a visualization system/tool/library/framework that you are familiar with (R, R Shiny, Python, Plotly, Excel, JMP, Matlab, Mathematica, Julia, etc.). Then, using no more than 220 words in total, compare it with Tableau and D3 in terms of:

1. Ease to develop for developers
2. Ease to maintain the visualization for developers (e.g., difficulty of the maintenance of the product as the requirements change, the data changes, the hosting platform changes, etc.)
3. Usability of visualization developed for end users
4. Scalability of visualization to “large” datasets
5. System requirements to run the visualization (e.g., browsers, OS, software licensing) for end users

Note: Your claims should be well justified, supported with compelling reasons. Simply stating that a tool is better (or worse) than D3 without justifications will receive a low (or no) score.

We recommend formatting your answers as bullet lists for better readability. For example:

1. Ease to develop
   Seaborn: …
   Tableau: …
   D3: …
2. Ease to maintain the visualization
   Seaborn: …
   Tableau: …
   D3: …

...

Q7 Deliverables:
The directory structure should be as follows:

Q7/
analysis.txt

- **analysis.txt** - comparison of visualization tools.

Important: folder structure of the zip file that you submit

You are submitting a single zip file HW2-GTUsername.zip (e.g., HW2-jdoe3.zip, where “jdoe3” is your GT username), which must unzip to the following directory structure (i.e., a folder “HW2-jdoe3”, containing folders “Q1”, “Q2”, etc.). The files to be included in each question’s folder have been clearly specified at the end of each question’s problem description above.

```
HW2-GTUsername/
    lib/
        d3.v5.min.js
        d3-tip.min.js
        d3-scale-chromatic.v1.min
        topojson.v2.min.js
    Q1/
        table.png
        barchart.png
        all-ages.csv
        grad-students.csv
    Q2/
        graph.html
        graph.(js / css) - if not included in graph.html
    Q3/
        scatterplot.(html / js / css)
        explanation.txt
        scatter_plots.pdf
        movies.csv
    Q4/
        heatmap.(html / js /css)
        heatmap.csv
    Q5/
        interactive.(html / js / css)
    Q6/
         q6.(html / js / css)
```

```
        county_poverty.csv
        county_detail.csv
        us.json
    Q7/
        analysis.txt
```

Version 1

---

[1] Source: https://www.kaggle.com/the-guardian/olympic-games
[2]Source: derived from a “movies” dataset prepared by Dr. Guy Lebanon, for an earlier version of OMSCS CSE 6242 (the source raw data is available at the following URL; you do not need to download it when working on this question https://s3.amazonaws.com/content.udacity-data.com/courses/gt-cs6242/project/movies_merged)
[3] Source: WorldBank Rural Populations
[4] Source: Derived from USDA.