cse6242-2018fall-hw2

CSE 6242 / CX 4242: Data and Visual Analytics | Georgia Tech | Fall 2018

Homework 2 : D3 Graphs and Visualization

Prepared by (in alphabetical order): Anmol Chhabria, Arathi Arivayutham, Brendon Duprey, Fan Zhou, Jennifer Ma, Keith Adkins, Kunal Agarwal, Mansi Mathur, Matthew Hull, Matthew Keezer, Michael Petrey, Nilaksh Das, Priyank Madria, Siddharth Gulati, Shruti Shinde, Sneha Venkatachalam, Susanta Routray, Svetlana Afanaseva, Vineet Vinayak Pasupulety, Polo Chau

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 about how 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 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 62.0 (or newer)**.

For this homework, you will work with version 3 of D3, provided to you in the **lib** folder. You must NOT use any other 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 SimpleHTTPServer in Python (for Python version 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.v3.min.js"></script>
```

It is incorrect to use an absolute path such as:

```
<script type="text/javascript" src="http://d3js.org/d3.v3.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 the data for the Men’s Gymnastics World Championships and the Summer Olympics. Perform the *task a* to help the Championships committee analyze the Men’s Gymnastics data and *task b* to help the Olympics Committee better understand the overall Medals by Country.

a. **[5 points] Good table design.** Create a table to display the details of Men’s Gymnastics World Championships provided in *worldchamps_mens_gymnastics.csv*[1]. You can use any tool (e.g., Excel, HTML) to create the table.

- The table should contain data from the following columns: *Name, Overall Rank, Nationality, Apparatus, Total Score*
- **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 Summer Olympics data as a *stacked bar chart*. Your chart should display the Count of Medals for Olympic using the dataset *summer_olympics.csv*[2] (in Q1 folder). (Optional reading: the effectiveness of stacked bar charts is often debated --- sometimes, they can be confusing 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 Count of Medals during the years 1992-2012 (inclusive) for the countries: France, Germany, United Kingdom, United States, and China.
- Your design should design a stacked bar chart to show the count for each type of medal (Bronze, Silver, Gold) for the following Sports: Aquatics, Athletics, Football, Gymnastics, Rowing.
- For each country, there should be two stacked bars (one for each gender). Make sure the bars are grouped by country.
- Your design should have clear label axes and chart title. Include a legend for your chart.
- **Save the chart as barchart.png.**

Tableau has provided us with student licenses. Go to tableau activation and select “Tableau Desktop”. After the installation, you will be asked to add an activation key. The Desktop Key for activation is available on the course github repository as “Tableau Desktop Key-Fall 2018” at download. This key is for your use in this course only. **Do not share the key with anyone.**

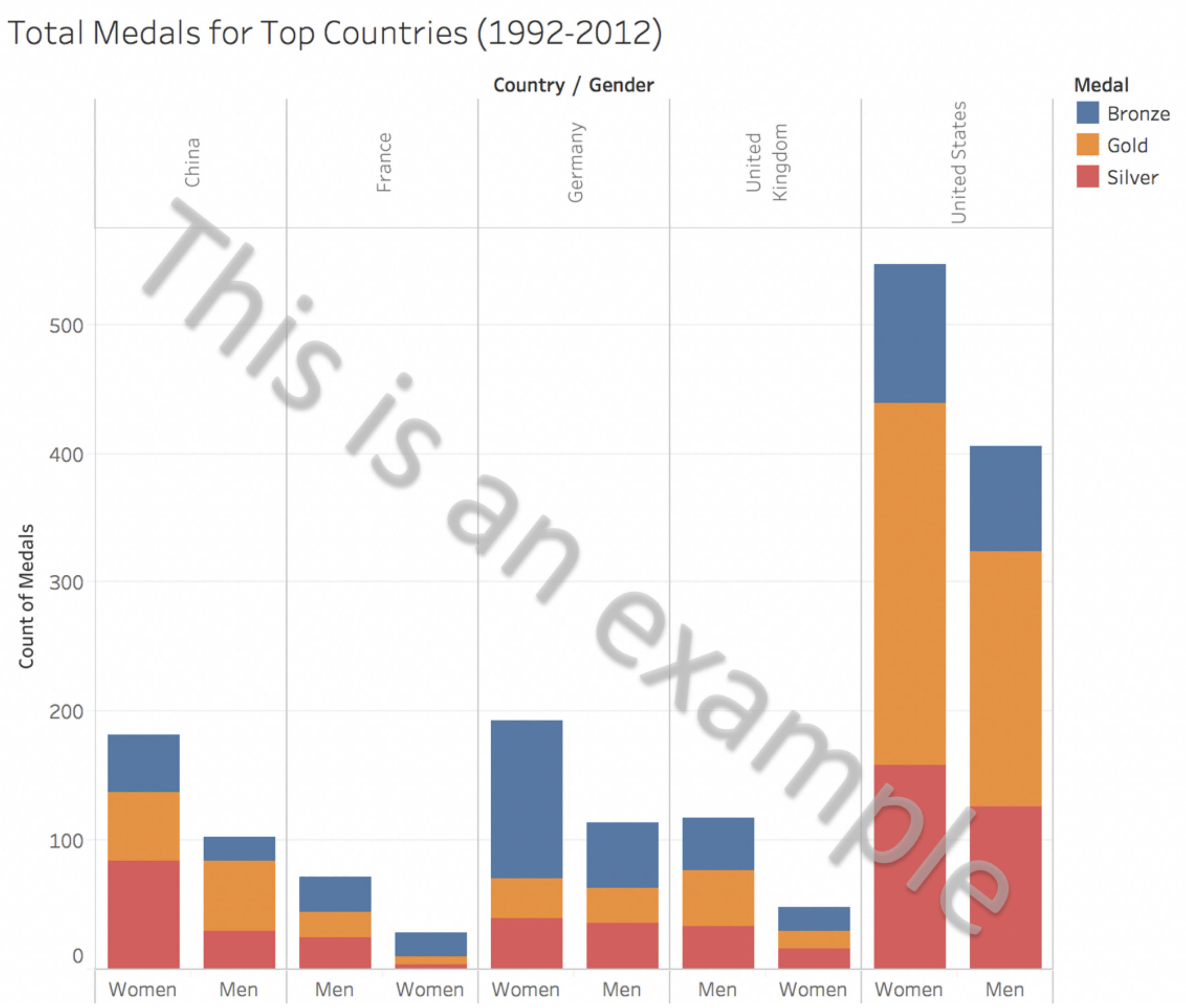

Figure 3: Example for scatter plots, on a single HTML page.

Q1 Deliverables:
The directory structure should be as follows:

```
Q1/
    table.png
    barchart.png
    worldchamps_mens_gymnastics.csv
    summer_olympics.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.
- **worldchamps_mens_gymnastics.csv** and **summer_olympics.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*) to the right of each node. If a node is dragged, its label must also move with the node.

b. **[3 points] Coloring links**: Color the links based on the "value" field in the links array. Assign the following colors:

If the value of the edge is equal to 0, assign the color **blue** to the link.
If the value of the edge is equal to 1, assign the color **red** to the link.

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 due 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.

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[3] 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 of the movie
Feature 2: Title
Feature 3: Year
Feature 4: Runtime (minutes)
Feature 5: Country
Feature 6: IMDb Rating
Feature 7: IMDb Votes
Feature 8: Budget (in USD)
Feature 9: Gross (in USD)
Feature 10: Wins and nominations
Feature 11: Is good rating? (value 1 means “good”, value 0 - “bad”)

Optional: to learn more about IMDb, visit https://en.wikipedia.org/wiki/IMDb

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 (IMDb Rating)
     - Figure title: Wins+Nominations vs. IMDb Rating
     - X axis (horizontal) label: IMDb Rating
     - Y axis (vertical) label: Wins+Noms

   - Feature 8 (Budget) vs. Features 6 (IMDb Rating)
     - Figure title: Budget vs. IMDb Rating
     - X axis (horizontal) label: IMDb 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 (IMDb Votes) vs. Feature 6 (IMDb Rating) sized by Feature 10 (Wins+Nominations)
  - Figure title: Votes vs. IMDb Rating sized by Wins+Nominations
  - X axis (horizontal) label: IMDb Rating
  - Y axis (vertical) label: IMDb 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. IMDb Rating
  - X axis (horizontal) label: IMDb 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. IMDb Rating
  - X axis (horizontal) label: IMDb Rating
  - Y axis (vertical) label: Wins+Noms

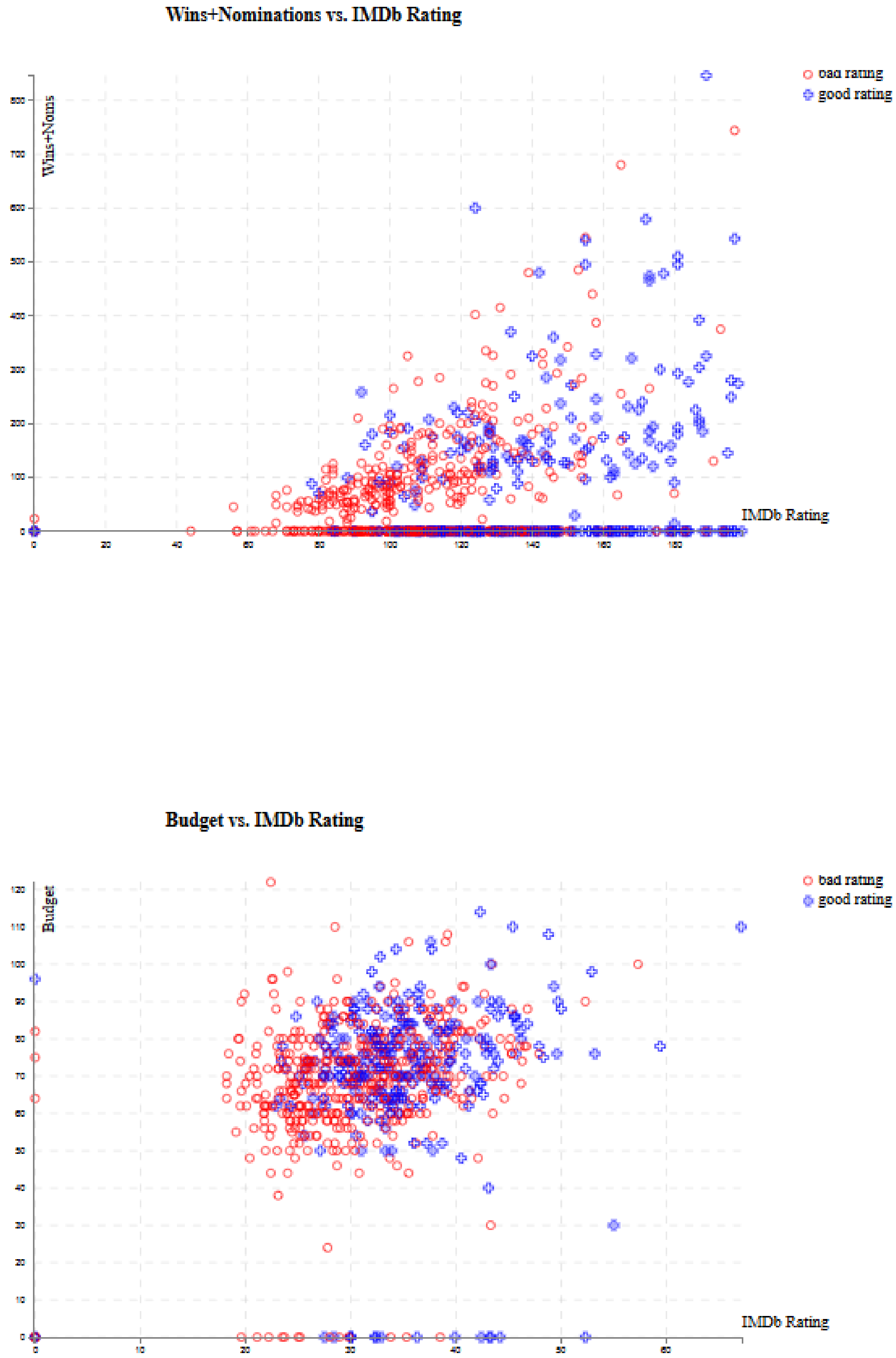

Figure 3: Example for scatter plots, on a single HTML page.

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 types of Charms cast by each of the wizarding houses across J.K. Rowling's 7 Harry Potter books. 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 spells, for each spell type used in each book for the wizarding house 'Gryffindor'. Place the Spell Type on the heatmap's horizontal axis and the Book on its vertical axis.

b. [**1 point**] The color scheme of a heatmap is a very important part of its design. The number of Spell Types for each Book 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 Book ("Sorcerer's Stone", "Chamber of Secrets", "Prisoner of Azkaban", etc.) on the vertical axis in the order of publication (the earliest book goes to the top). Place the Spell Type ("Hex", "Counter Spell", "Jinx", etc.) on the horizontal axis. Order the Spell Types alphabetically, from left to right.

d. **[6 pt]** Now create a drop down select box with D3 that is populated with the wizarding house names ("Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin"). When the user selects a different house in this select box, the heatmap and the legend should both be updated with values corresponding to the selected house. Note the differences in the legends for house "Gryffindor" and "Ravenclaw" 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 house when the page loads should be "Gryffindor".

Note:

1. The Harry Potter dataset being used here has been synthetically generated. It doesn't accurately resemble the count of spell types used per Wizarding House across different books in the Harry Potter world.
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.

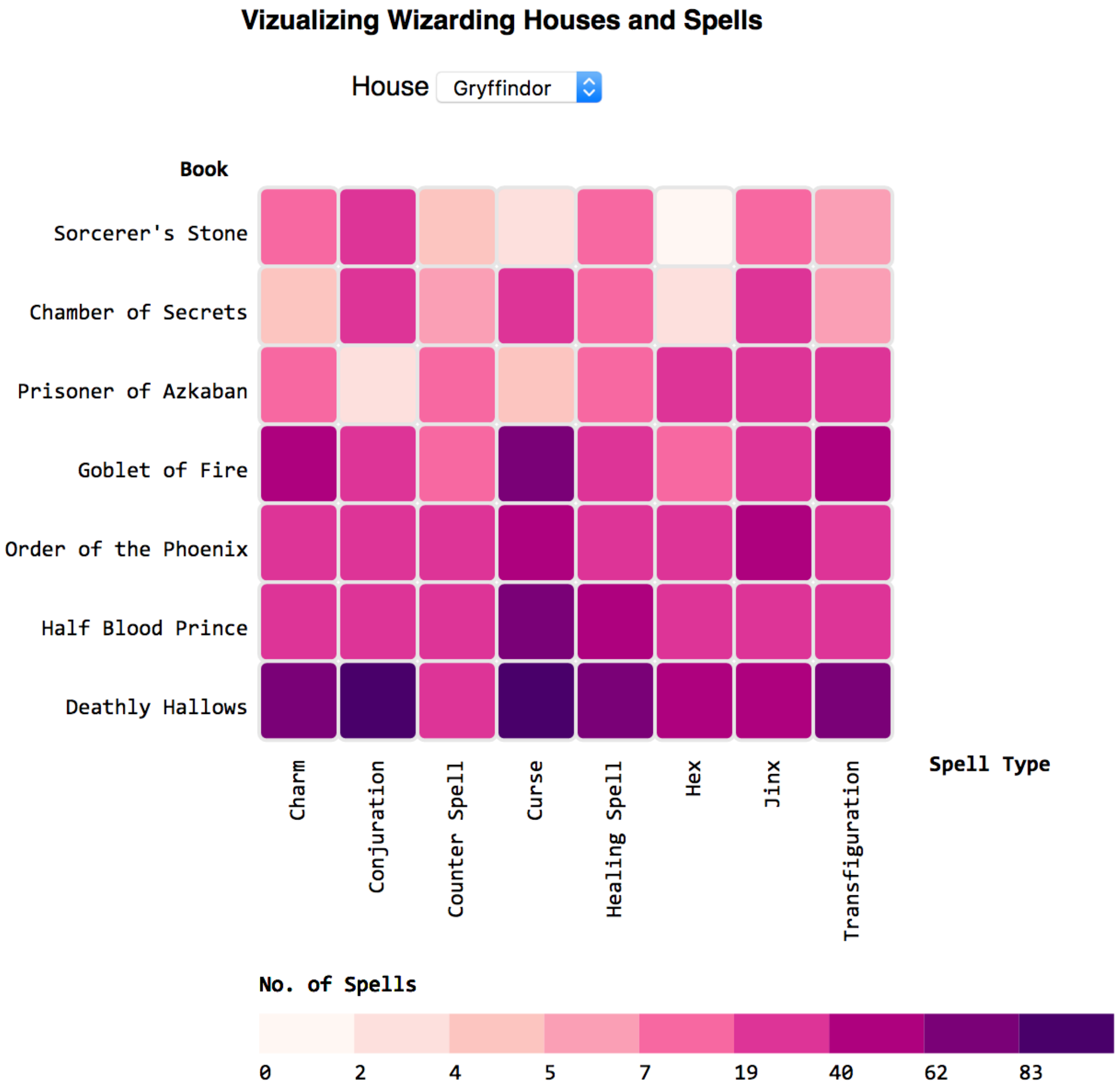

Figure 4a: Number of spells of each type used by house Gryffindor for all books.

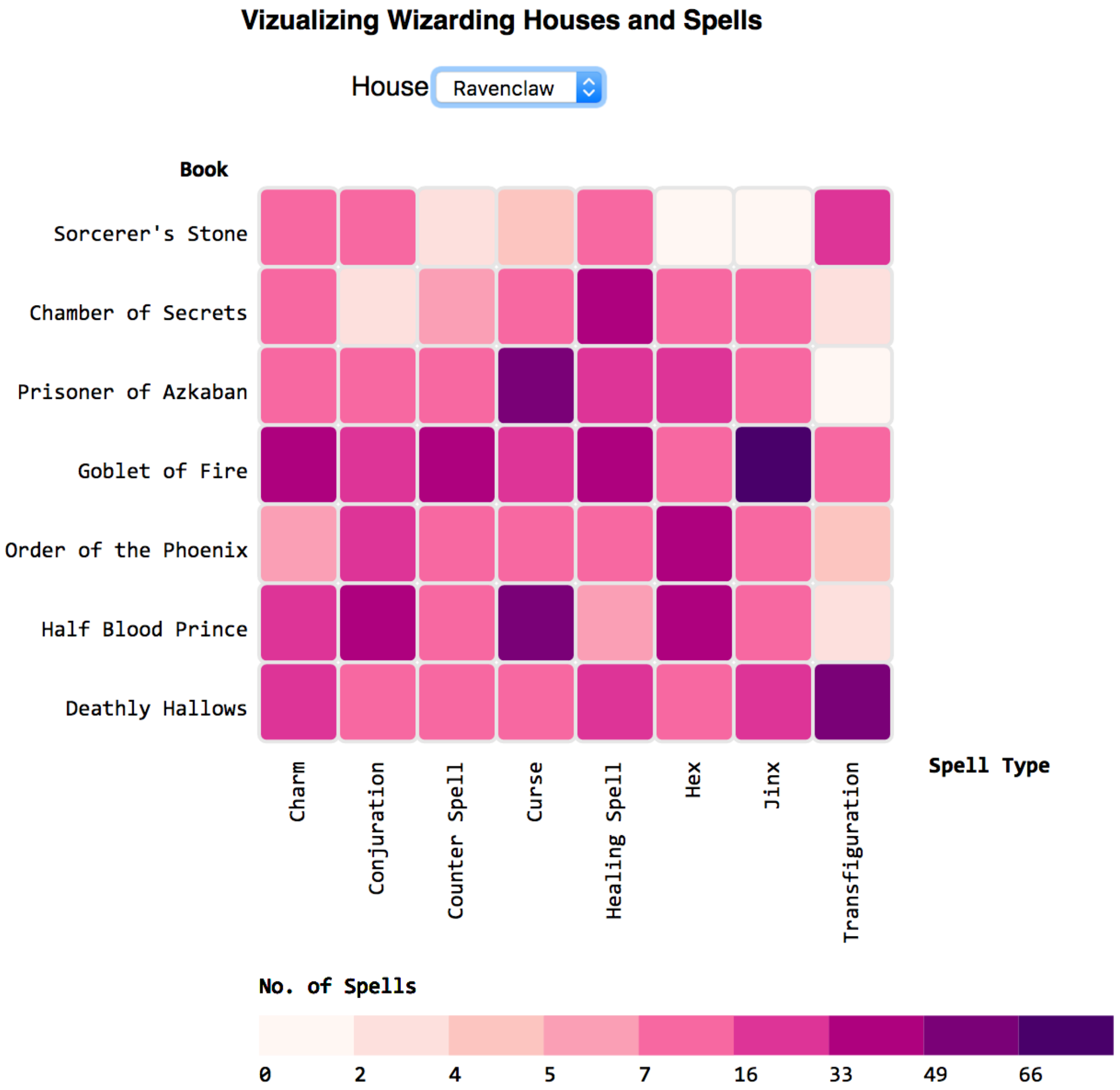

Figure 4b: Number of spells of each type used by house Ravenclaw for all books.

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[4] provided in the dataset.txt file (in the Q5 folder) to create an interactive bar chart. Each line in the file represents population growth (per year) of an US city over the past five years, starting with total 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 city 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 city names as labels.

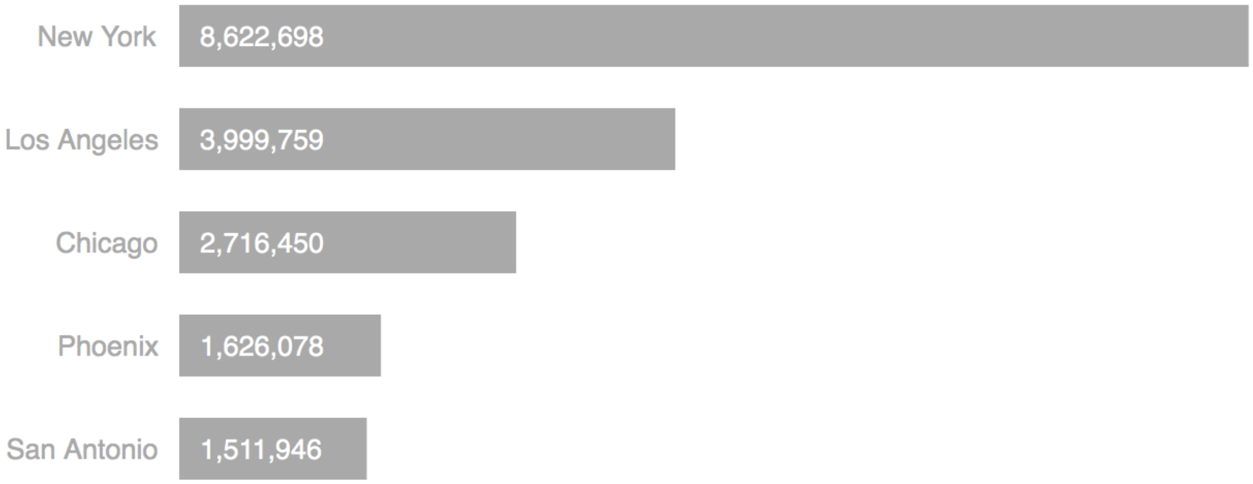

Figure 5a. Bars representing **total** population of each city

b. **[10 points]** When hovering the mouse over a bar, a smaller line chart representing the population growth of that city for each year (2013-2017) should be displayed in the top right corner. For example, **Los Angeles** has a growth value of 0.84%, 0.79%, 0.78%, 0.70%, 0.47% for the years 2013, 2014, 2015, 2016 and 2017 respectively. On hovering over the bar representing **Los Angeles**, 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:

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

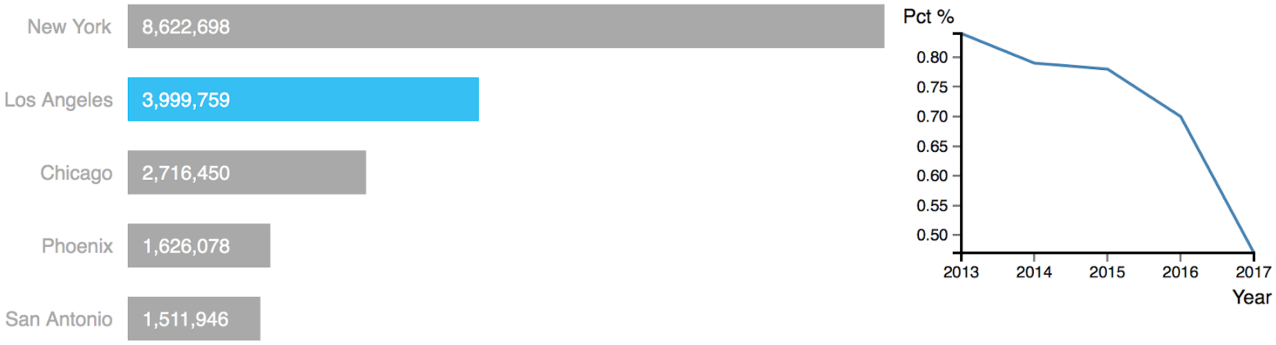

Figure 5b. On hovering over the bar for **Los Angeles**, a smaller line chart representing its percentage of growth per year in decimal over the past 5 years is displayed at the top right corner.

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 city, 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: Unemployment rates

Use the provided dataset[5] in *education.csv*, *us.json* and *education_details.csv* (in the Q6 folder) and visualize them as a choropleth map.

- Each record in *education.csv* represents a county and is of the form `<id,name,percent_educated>`, where
  - ○ `id` corresponds to the county id
  - ○ `name` is the county name
  - ○ `percent_educated` is the percentage of educated people living in that county

- The *education_details.csv* file contains a list of records, each having four fields:
  - ○ an `id` field corresponding to a county in the United States,
  - ○ a `qualified_professionals` field corresponding to the number of professionals in the county,
  - ○ a `high_school` and a `middle_school_or_lower` fields corresponding to the number of high school students and middle school students respectively.

- 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 a reference.

1. **[10 points]** The color of each county should correspond to the percentage of educated people in that county, i.e., darker colors correspond to higher percentage in that county and lighter colors correspond to lower percentage in that county. Use gradients of only **one** particular hue. Use d3-queue (in the lib folder) to easily load data from multiple files into a function[6]. Use topojson (present in lib) to draw the choropleth map.
2. **[5 points]** Add a legend showing how colors map to percentage of educated people. Create the legend using a threshold scale as shown in the figure. The domain used for the legend should be [0,90] (inclusively) with a step size of 10.

b. **[5 points]** Add a tooltip using the d3.tip library (in the **lib** folder) that, on hovering over a county, shows the (1) county name, (2) percentage of educated people, (3) number of qualified professionals, (4) high school graduates, and (5) middle school or lower graduates, each on a separate line each. The tooltip should appear when the mouse hovers over the county. On mouseout, the tooltip should disappear. Use Figure 6 below as a 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 around 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).

Note: You must create the tooltip by only using **d3.tip.v0.6.3.js** present in the **lib** folder.

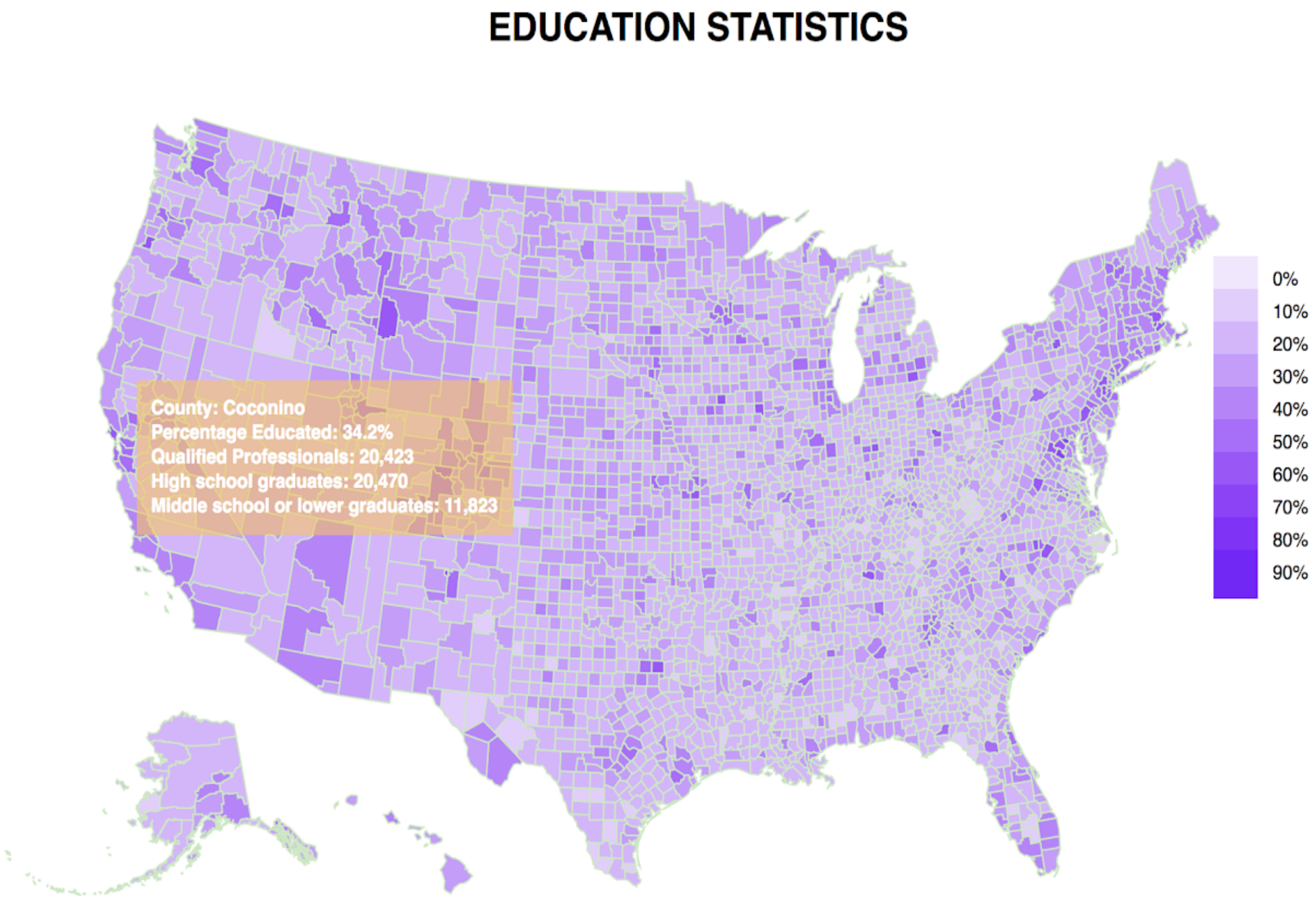
EDUCATION STATISTICS
County: Coconino
Percentage Educated: 34.2%
Qualified Professionals: 20,423
High school graduates: 20,470
Middle school or lower graduates: 11,823
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%

Figure 6. Reference example for Choropleth Maps

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

Q6/
q6.(html/js/css)
education.csv
education_details.csv
us.json

- **q6.(html /js /css)**- The html/js/css file to render the visualization.
- **education.csv and education_details.csv -** The datasets used.
- **us.json** - Dataset needed to draw the map.

Q7 [5 points] Pros and Cons of Visualization Tools

This 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 150 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)
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)

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

1. Ease to …
    Seaborn: …
    Tableau: …
    D3: …
2. Ease to …
    ...

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 (e.g., HW2-jdoe3.zip), 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-{GT account username}/
    lib/
        d3.v3.min.js
        d3.tip.v0.6.3.js
        d3-queue.v3.min.js
        topojson.v1.min.js
    Q1/
        table.png
        barchart.png
        worldchamps_mens_gymnastics.csv
        summer_olympics.csv
    Q2/
        ...
    Q3/
        scatterplot.(html / js / css)
        explanation.txt
        scatter_plots.pdf
        movies.csv
    Q4/
        heatmap.(html / js /css)
        heatmap.csv
    Q5/
        interactive.html
    Q6/
        ...
    Q7/
        analysis.txt
```

Version 0

[1] Source: https://www.kaggle.com/cjdaffern/gymnastic-champs-mens-all-round
[2] Source: https://www.kaggle.com/the-guardian/olympic-games
[3]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)
[4] Source: US City Populations
[5] Source: Derived from USDA.

[6] d3-queue evaluates a number of asynchronous tasks concurrently -- in this question, each task would be loading one data file. When all tasks have finished, d3-queue passes the results to a user-defined callback function.