CSE 6242 / CX 4242: Data and Visual Analytics | Georgia Tech | Fall 2023

HW 2: Tableau, D3 Graphs, and Visualization

Important Notes

1. Submit your work by the due date on the course schedule.
    a. Every assignment has a 48-hour grace period. You may use it without asking us.
    b. Before the grace period expires, you may resubmit as many times as you need to.
    c. The grace period is a lenient buffer for resolving last minute issues. We do not recommend starting new work or modifying existing work during the grace period.
    d. TA assistance is not guaranteed during the grace period.
    e. Submissions during the grace period will display as "late" and will not incur a penalty.
    f. We will not accept any submissions after the grace period.
2. Always use the **most up-to-date assignment** (version number at the bottom right of this document).
3. You may discuss 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 the student's own answers**.

Submission Instructions

Carefully read and follow the high-level instructions below, and the detailed instructions in each question.

1. **Submit ALL deliverables via Gradescope.** We will not accept submissions anywhere else.
2. Submit all required files, as specified at the beginning of each question. We will not grade any submissions that deviate from the specified format (extra files, misnamed files, etc.).
3. Each submission and its score will be recorded and saved by Gradescope. **By default, Gradescope uses your last submission for grading.** To use a different submission, **you MUST "activate" it** (click "Submission History" button at bottom toolbar, then "Activate").

Grading and Feedback

The maximum possible score for this homework is 100 points. Students can choose to complete any 90 points worth of work to receive 100% for this assignment, and can receive more than 100% if additional work is submitted. For example, if a student scores 100 points, that student will receive 111% for the assignment.

We will grade all questions using the Gradescope platform. Question 1 will be manually graded after the grace period. Questions 2-5 are auto-graded. Keep the following in mind:

1. You can access Gradescope through Canvas.
2. You may upload your code periodically to Gradescope to obtain feedback on your code. Gradescope will auto-grade your submission using the same test cases that we use to grade your work.
3. You must **not** use Gradescope as the primary way to test your code's correctness, since it provides only a few test cases, and error messages may not be as informative as local debuggers. Iteratively develop and test your code locally, write more test cases, and follow good coding practices. Use Gradescope mainly as a "final" check.
4. **Gradescope cannot run code that contains syntax errors.** If Gradescope is not running, verify:
    a. Your code is free of syntax errors (by running locally)
    b. All methods have been implemented
    c. You have submitted the correct file with the correct name
5. Do not share links to your Gradescope submissions in public posts
6. When many students use Gradescope simultaneously, it may slow down or fail to communicate with the tester. It can become even slower as the submission deadline approaches. You are responsible for submitting your work on time.

Download the HW2 Skeleton before you begin

Homework Overview

"Visualization gives you answers to questions you didn't know you have" - Ben Schneiderman

This homework focuses on exploring and creating data visualizations using two of the most popular tools in the field. Data visualization is an integral part of exploratory analysis and communicating key insights. All questions use data on the same topic to highlight the uses and strengths of different types of visualizations. The data comes from BoardGameGeek and includes games' ratings, popularity, and metadata.

Below are some terms you will often see in the questions:

- **Rating** – a value from 0 to 10 given to each game. *BoardGameGeek* calculates a game's overall rating in different ways including **Average** and **Bayes**, so make sure you are using the correct rating called for in a question. A higher rating is better than a lower rating.
- **Rank** – the overall rank of a boardgame from 1 to *n,* with ranks closer to 1 being better and *n* being the total number of games. The rank may be for all games or for a subgroup of games such as abstract games or family games.

Q1 [25 points] Designing a good table. Visualizing data with Tableau.

Goal	Design a table, a grouped bar chart, and a stacked bar chart with filters in Tableau.
Technology	Tableau Desktop
Deliverables	**Gradescope:** After selecting HW2 - Q1, click **Submit Images**. You will be taken to a list of questions for your assignment. Click **Select Images** and submit the following four PNG images under the corresponding questions: ● **table.png**: Image/screenshot of the table in Q1.a ● **grouped_barchart.png**: Image of the chart in Q1.b ● **stacked_barchart_1.png**: Image of the chart in Q1.c after filtering data for Max.Players = 2 ● **stacked_barchart_2.png**: Image of the chart in Q1.c after filtering data for Max.Players = 4 **Q1 will be manually graded after the grace period.**

Setting Up Tableau

Install and activate Tableau Desktop by following "HW2 Instructions" on Canvas. The product activation key is for your use in this course only. **Do not share the key with anyone.** If you already have Tableau Desktop installed on your machine, you may use this key to reactivate it.

If you do not have access to a Mac or Windows machine, 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

If neither of the above methods work, use 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.

Connecting to Data

1. It is optional to use Tableau for Q1a. Otherwise, complete all parts using **a single Tableau workbook**.
2. Q1 will require connecting Tableau to two different data sources. You can connect to multiple data sources within one workbook by following the directions here.
3. For Q1a and Q1b:
   a. Open Tableau and connect to a data source. Choose `To a File` – **`Text file`**. Select the ***popular_board_game.csv*** file from the skeleton.
   b. Click on the graph area at the bottom section next to "Data Source" to create worksheets.
4. For Q1c:
   a. You will need a *data.world* account to access the data for Q1c. Add a new data source by clicking on `Data` – **`New Data Source`**.
   b. When connecting to a data source, choose `To a Server` – **`Web Data Connector`**.
   c. Enter this URL to connect to the data.world data set on board games. You may be prompted to log in to *data-world* and authorize Tableau. If you haven't used data.world before, you will be required to create an account by clicking "Join Now". Do not edit the provided SQL query.

      NOTE: If you cannot connect to *data-world*, you can use the provided csv files for Q1 in the skeleton. The provided csv files are identical to those hosted online and can be loaded directly into Tableau.

   d. Click the graph area at the bottom section to create another worksheet, and Tableau will automatically create a data extract.

Table and Chart Design

a. **[5 points] Good table design.** Visualize the data contained in *popular_board_game.csv* as a data table (known as a text table in Tableau). In this part (Q1a), you can use any tool (e.g., Excel, HTML, Pandas, Tableau) to create the table.

We are interested in grouping popular games into "support solo" (min player = 1) and "not support solo" (min player > 1). Your table should clearly communicate information about these two groups simultaneously. For each group (Solo Supported, Solo Not Supported), show:

1. Total number of games in each category (fighting, economic, ...)
2. In each category, the game with the highest number of ratings. If more than one game has the same (highest) number of ratings, pick the game you prefer. **NOTE:** Level of Detail expressions may be useful if you use Tableau.
3. Average rating of games in each category (use simple average), rounded to 2 decimal places.
4. Average playtime of games in each category, rounded to 2 decimal places.
5. In the bottom left corner below your table, include your GT username (In Tableau, this can be done by including a caption when exporting an image of a worksheet or by adding a text box to a dashboard. If you use Tableau, refer to the tutorial here).
6. Save the table as **table.png**. (If you use Tableau, go to Worksheet/Dashboard → Export → Image). **NOTE:** Do not take screenshots in Tableau since your image must have high resolution. You can take a screenshot If you use HTML, Pandas, etc.

Your learning goal here is to practice good table design, which is not strongly dependent on the tool that you use. Thus, we do not require that you use Tableau in this part. You may decide the most meaningful column names, the number of columns, and the column order. You are not limited to only the techniques described in the lecture. For OMS students, the lecture video on this topic is *Week 4 - Fixing Common Visualization Issues - Fixing Bar Charts, Line Charts*. For campus students, review lecture slides 43 and 44.

b. **[10 points] Grouped bar chart.** Visualize *popular_board_game.csv* as a grouped bar chart in Tableau. Your chart should display game category (e.g., fighting, economic,...) along the horizontal axis and game count along the vertical axis. Show game playtime (e.g., <=30, (30, 60]) for each category. **NOTE:** Do not differentiate between “support solo” and “non-support solo” for this question.

1. Design a vertically grouped bar chart. For each category, show the game count for each playtime.
2. Include clearly labeled axes, a clear chart title, and a legend.
3. In the bottom left corner of your image, include your GT username. **NOTE:** In Tableau, this can be done by including a caption when exporting an image of a worksheet or by adding a text box to a dashboard. Refer to the tutorial here.
4. **Save the chart as grouped_barchart.png (**go to Worksheet/Dashboard → Export → Image. **NOTE:** Do not take screenshots in Tableau since your image must have high resolution.

The main goal here is for you to get familiarized with Tableau. Thus, we kept this open-ended, so you can practice making design decisions. **We will accept most designs.** We show one possible design in Figure 1b, based on the tutorial from Tableau.

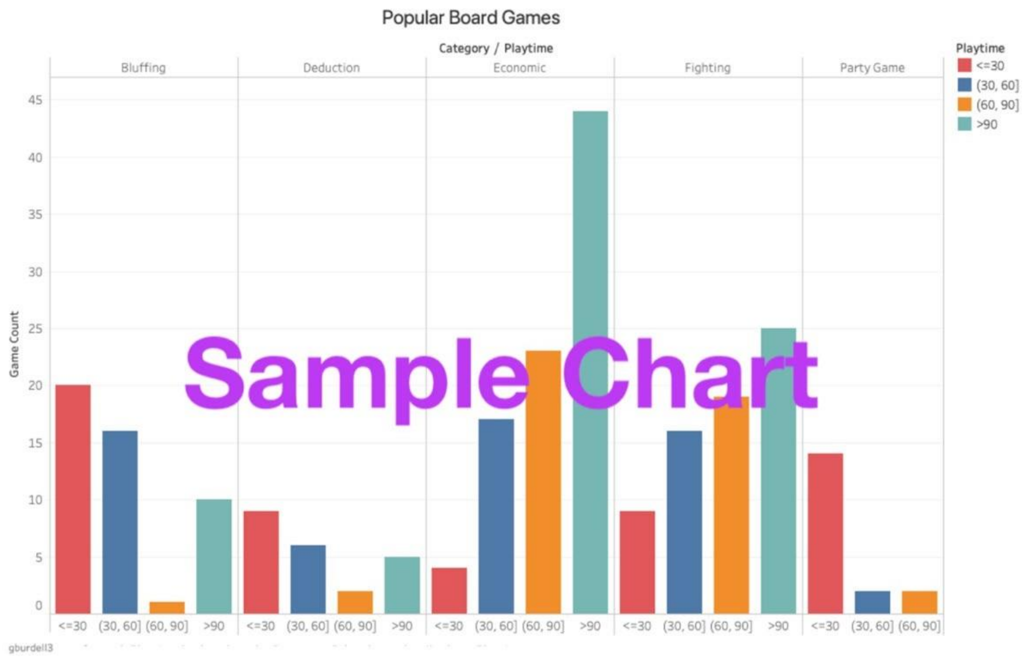

Figure 1b: Example of a grouped bar chart. Your chart may appear different and can earn full credit if it meets all the stated requirements. Your submitted image should include your GT username in the bottom left.

c. **[10 points] Stacked bar chart.** Visualize the *data.world* dataset (or *games_detailed_info_filtered.csv* if using the local files in the skeleton) as a stacked bar chart. Showcase the count of games in different categories and the relationship between game categories, their mechanics, and max player size.

1. Create a **Worksheet** with a stacked bar chart that shows game counts for each playing mechanic (sub-bars) for each game category. **NOTE:** This data contains duplicate rows, as each row represents a distinct game. Do not remove duplicate rows from the data.
2. Display game counts along the vertical axis and category along the horizontal axis.
3. Include clear axes labels, a clear chart title, and a legend.
4. Create a ***Dashboard*** using the worksheet you created in Step 1.
5. Add a filter for number of 'Max.Players' allowed in each game. Update the chart using this filter to generate the following chart images (Refer to the tutorial here on how to add a filter in a dashboard. Make sure to add 'Max.Players' in the filter shelf in the Worksheet first, like this):
    a. Select "2 Players" only in the filter. Save the resulting chart as '**stacked_barchart_1.png**'
    b. Select "4 Players" only in the filter. Save the resulting chart as '**stacked_barchart_2.png**'
    c. Both images must include your GT username in the bottom left. This can be added using a text box. Refer to the tutorial here.https://youtu.be/fRwQenvBJ6I
    d. In each image, the filter must be visible. If you are using Tableau Online, you may need to add your worksheet containing the chart to a dashboard and then download an image of the dashboard that contains both the filter and the chart.

Note: To save a dashboard image, go to Dashboard - Export Image. Do not submit screenshots.

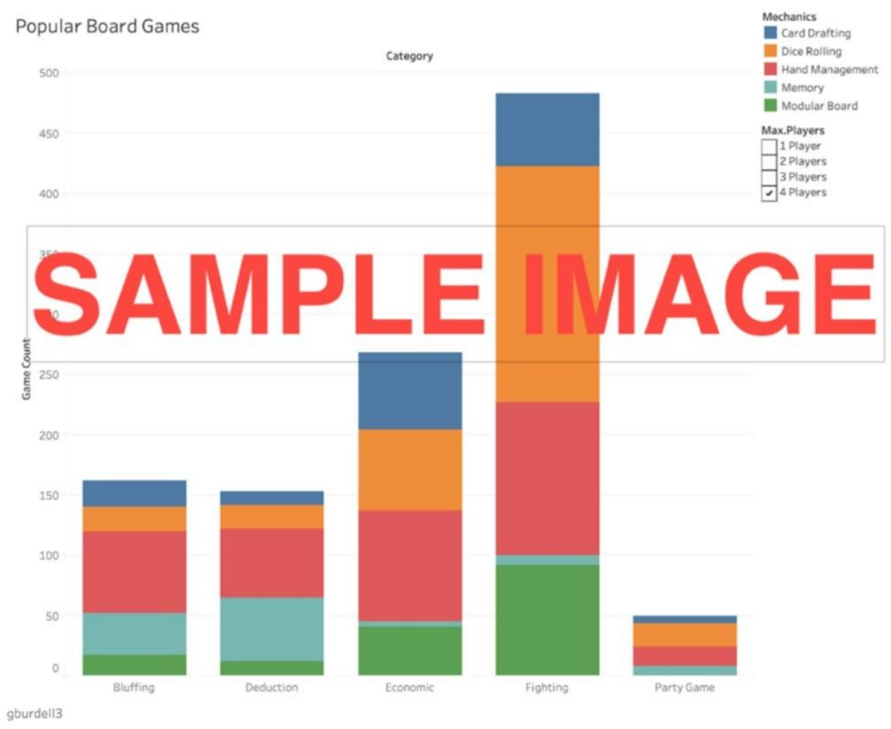

Figure 1c: Example of a stacked bar chart after selecting "4 Players" in Max.Players filter. Your chart may appear different and can earn full credit if it meets all the stated requirements. Your submitted image should include your GT username in the bottom left.

Optional Reading: The effectiveness of stacked bar charts is often debated—sometimes, they can be confusing, difficult to understand, and may make data series comparisons challenging.

Important Points about Developing with D3 in Questions 2–5

1. We highly recommend that you use the latest Chrome browser to complete this question. We will grade your work using Chrome v92 (or higher)**.**

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

3. For Q3–5, your D3 visualization **MUST** produce a DOM structure as specified at the end of each question. Not only does the structure help guide your D3 code design, but it also enables your code to be auto-graded (the auto-grader identifies and evaluates relevant elements in the rendered HTML). We highly recommend you review the specified DOM structure **before** starting to code.

4. **You need to setup a local HTTP server in the root (hw2-skeleton) folder to run your D3 visualizations**, 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. (for more details, see link).

5. **All d3*.js files in the lib folder must be referenced using relative paths**, e.g., **"../lib/<filename>**" in your html files. For example, if the file "Q2/submission.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="C:/Users/polo/hw2-skeleton/lib/d3.v5.min.js"></script>

6. For questions that require reading from a dataset, use a **relative path** to read in the dataset file**.** For example, suppose a question reads data from `earthquake.csv`, the path should simply be "earthquake.csv" and **NOT** an absolute path such as "C:/Users/polo/hw2-skeleton/Q/earthquake.csv".

7. 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.

Q2 [15 points] Force-directed graph layout

Goal	Create a network graph shows relationships between games in D3. Use interactive features like pinning nodes to give the viewer some control over the visualization.
Technology	D3 Version 5 (included in the lib folder) Chrome v92.0 (or higher): the browser for grading your code Python http server (for local testing)
Allowed Libraries	D3 library is provided to you in the **lib** folder. You must **NOT** use any D3 libraries (d3*.js) other than the ones provided. On Gradescope, these libraries are provided for you in the auto-grading environment.
Deliverables	[Gradescope] Q2**.(html/js/css):** The HTML, JavaScript, CSS to render the graph. Do not include the D3 libraries or board_games.csv dataset.

You will experiment with many aspects of D3 for graph visualization. To help you get started, we have provided the Q2.html file (in the Q2 folder) and an undirected graph dataset of boardgames, board_games.csv file (in the Q2 folder). The dataset for this question was inspired by a reddit post about visualizing boardgames as a network, where the author calculates the similarity between board games based on categories and game mechanics where the edge value between each board game (node) is the total weighted similarity index. This dataset has been modified and simplified for this question and does not fully represent actual data found from the post. The provided Q2.html file will display a graph (network) in a web browser. The goal of this question is for you to experiment with the visual styling of this graph to make a more meaningful representation of the data. Here is a helpful resource (about graph layout) for this question.

Note: You can submit a single Q2.html that contains all the css and js components; or you can split Q2.html into Q2.html, Q2.css, and Q2.js.

a. **[2 points] Adding node labels**: Modify Q2.html to show the node label (the node name, e.g., the source) at the **top right** of each node in **bold**. If a node is dragged, its label must move with it.

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

- If the value of the edge is equal to 0 (similar), the edge should be gray, thick, and **solid** (The dashed line with zero gap is not considered as solid).
- If the value of the edge is equal to 1 (not similar), the edge should be green, thin, and **dashed**.

c. [3 points] Scaling nodes:

1. **[1.5 points]** 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, which will likely lead to low-quality visualization (e.g., nodes that are mere points, barely visible, or of huge sizes with overlaps).

   Note: D3 v5 does not support d.weight (which was the typical approach to obtain node degree in D3 v3). You may need to calculate node degrees yourself. Example relevant approach is here.

2. **[1.5 points]** The degree of each node should be represented by varying colors. Pick a meaningful color scheme (hint: color gradients). There should be at least 3 color gradations and it must be visually evident that the nodes with a higher degree use darker/deeper colors and the nodes with lower degrees use lighter colors. You can find example color gradients at Color Brewer.

d. **[6 points] Pinning nodes**:

1. **[2 points]** Modify the code so that dragging a node will fix (i.e., "pin") the node's position such that it will not be modified by the graph layout algorithm (Note: pinned nodes can be further dragged around by the user. Additionally, pinning a node should not affect the free movement of the other nodes). Node pinning is an effective interaction technique to help users spatially organize nodes during graph exploration. The D3 API for pinning nodes has evolved over time. We recommend reading this post when you work on this sub-question.
2. **[1 points]** Mark pinned nodes to visually distinguish them from unpinned nodes, i.e., show pinned nodes in a different color.
3. **[3 points]** Double clicking a pinned node should unpin (unfreeze) its position and unmark it. When a node is no longer pinned, it should move freely again.

IMPORTANT:

1. For part 1 to consistently pass the autograder (which tests that a dragged node becomes pinned and retains its position), you may need to increase the radius of the highly-weighted nodes and reduce their label sizes, so that the nodes can be more easily detected by the autograder's webdriver mouse cursor.
2. To avoid timeout errors on Gradescope, complete the double click function in part 3 before submitting.
3. If you receive timeout messages for all parts and your code works locally on your computer, verify that you are indeed using the appropriate ids provided in the "add the nodes" section in the skeleton code.
4. D3 v5 does not support the d.fixed method (it was deprecated after D3 v3). For our purposes, it is used as a Boolean value to indicate whether a node has been pinned or not.

e. **[1 points] Add GT username:** Add your Georgia Tech username (usually includes a mix of letters and numbers, e.g., gburdell3) to the top right corner of the force-directed graph (see example image). The GT username must be a `<text>` element having the `id: "credit"`

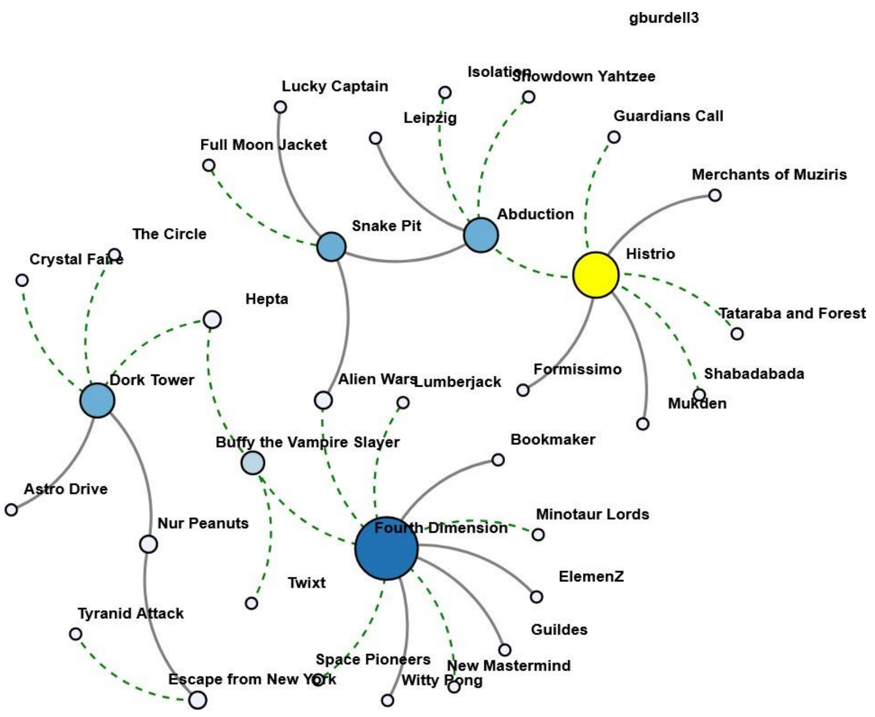

Figure 2a: Example of Visualization with pinned node (yellow). Your chart may appear different, and can earn full credit if it meets all the stated requirements.

Q3 [15 points] Line Charts

Goal	Explore temporal patterns in the *BoardGameGeek* data using line charts in D3 to compare how the number of ratings grew. Integrate additional data about board game rankings onto these line charts and explore the effect of axis scale choice.
Technology	D3 Version 5 (included in the lib folder) Chrome v92.0 (or higher): the browser for grading your code Python http server (for local testing)
Allowed Libraries	D3 library is provided to you in the **lib** folder. You must **NOT** use any D3 libraries (d3*.js) other than the ones provided. On Gradescope, these libraries are provided for you in the auto-grading environment.
Deliverables	[Gradescope] **Q3.(html / js / css)**: The HTML, JavaScript, CSS to render the line charts. Do not include the D3 libraries or boardgame_ratings.csv dataset.

Use the dataset provided in the file *boardgame_ratings.csv* (in the Q3 folder) to create line charts.

Refer to the tutorial for line chart here or here.

Note: You will create four charts in this question, which 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; however, the submission must follow the DOM structure specified at the end of this question.

IMPORTANT: use the Margin Convention guide for specifying chart dimensions and layout. The autograder will assume this convention has been followed for grading purposes.

a. [5 points] Creating line chart. Create a line chart (Figure 3a) that visualizes the number of board game ratings from November 2016 to August 2020 (inclusively), for the eight board games: ['Catan', 'Dominion', 'Codenames', 'Terraforming Mars', 'Gloomhaven', 'Magic: The Gathering', 'Dixit', 'Monopoly']. Use d3.schemeCategory10() to differentiate these board games. Add each board game's name next to its corresponding line. For the x-axis, show a tick label for every three months. Use D3 axis.tickFormat() and d3.timeFormat() to format the ticks to display abbreviated months and years. For example, Jan 17, Apr 17, Jul 17. (See Figure 3a and its x-axis ticks).

- Chart title: Number of Ratings 2016-2020
- Horizontal axis label: Month. Use D3.scaleTime().
- Vertical axis label: Num of Ratings. Use a linear scale (for this part a).

VERY IMPORTANT — Beware of "Silent Date Conversion": Opening the csv file in an application like Excel may silently modify date strings without warning you, e.g., converting hyphen-separated date strings (e.g., 2016-11-01) into slash-separated date strings (e.g., 11/01/16). Impacted students would see a "correct" line chart visualization on their local computers, but when they upload their code to Gradescope, test cases will fail (e.g., tick labels are not found, lines are not drawn) because the x-scale cannot be computed (as the dates are parsed as NaN). **To view the content of a csv file, we recommend you only use text editors (e.g., sublime text, notepad) that do not silently modify csv files.**

b. [5 points] Adding board game rankings. Create a line chart (Figure 3b) for this part (append to the same HTML page) whose design is a variant of what you have created in part a. Start with your chart from part a. Modify the code to visualize how the rankings of ['Catan', 'Codenames', 'Terraforming Mars', 'Gloomhaven'] change over time by adding a symbol with the ranking text on their corresponding lines. Show the symbol for every three months, similar to the x-axis ticks in part a. (See Figure 3b). Add a legend to explain what this symbol represents next to your chart (See the Figure 3b bottom right).

- Chart title: Number of Ratings 2016-2020 with Rankings

c. [5 points] Axis scales in D3. Create two line charts (Figure 3c-1,2) for this part (append to the same HTML page) to try out two axis scales in D3. Start with your chart from part b. Then modify the vertical axis scale for each chart: the first chart uses the square root scale for its vertical axis (only), and the second chart uses the log scale for its vertical axis (only). Keep the symbols and the symbol legend you implemented in part b. At the bottom right of the last chart, add your **GT username** (e.g., gburdell3, see Figure 3c-2 for example).

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

- First chart (Figure 3c-1)
    - Chart title: Number of Ratings 2016-2020 (Square root Scale)
    - This chart uses the **square root** scale for its vertical axis (only)
    - Other features should be the same as part b.

- Second chart (Figure 3c-2)
    - Chart title: Number of Ratings 2016-2020 (Log Scale)
    - This chart uses the **log scale** for its vertical axis (only). Set the y-scale domain minimum to 1.
    - Other features should be the same as part b.

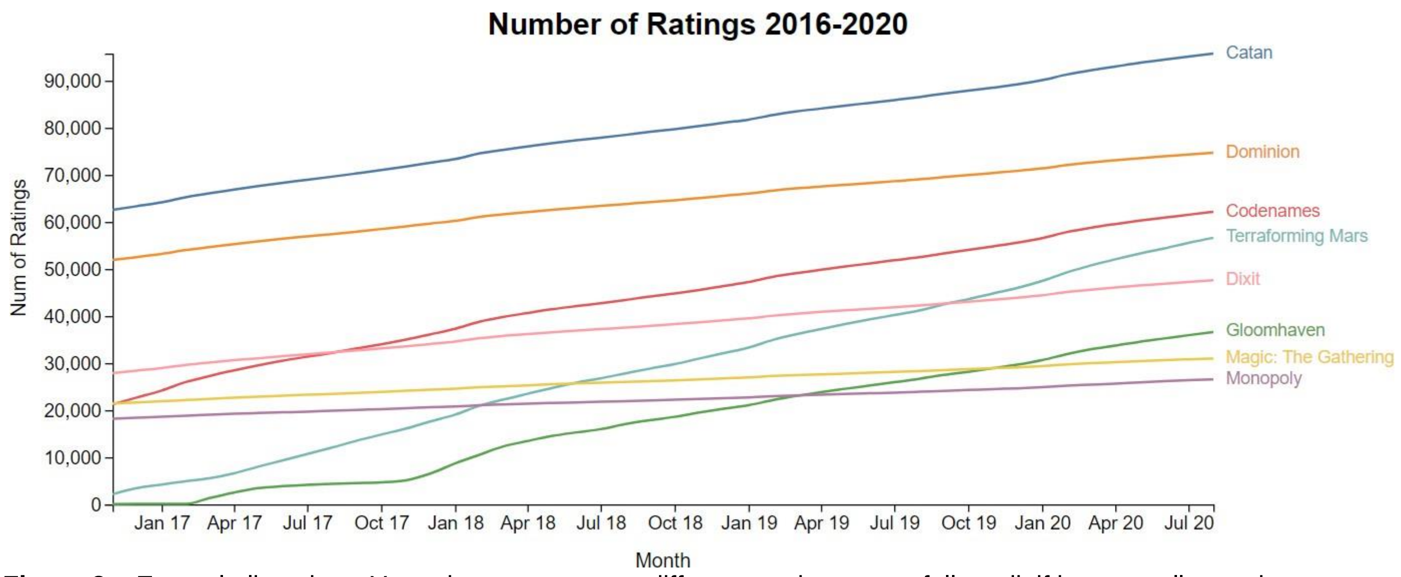

Figure 3a: Example line chart. Your chart may appear different, and can earn full credit if it meets all stated requirements.

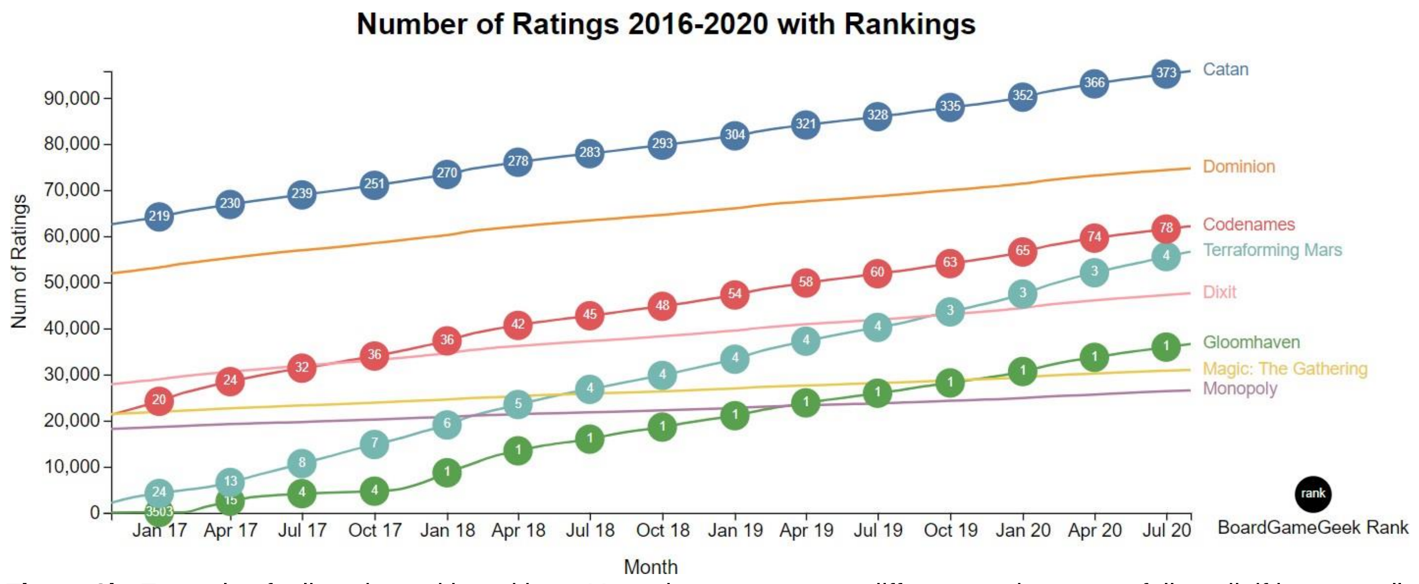

Figure 3b: Example of a line chart with rankings. Your chart may appear different, and can earn full credit if it meets all stated requirements.

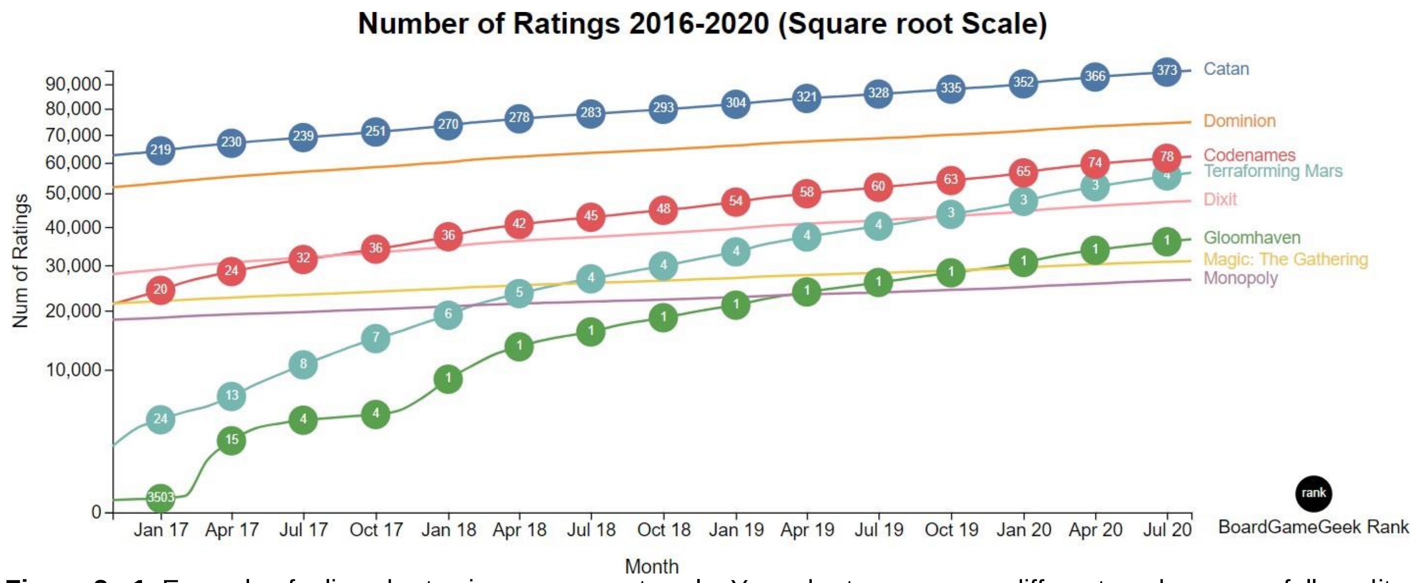

Figure 3c-1: Example of a line chart using square root scale. Your chart may appear different, and can earn full credit if it meets all stated requirements.

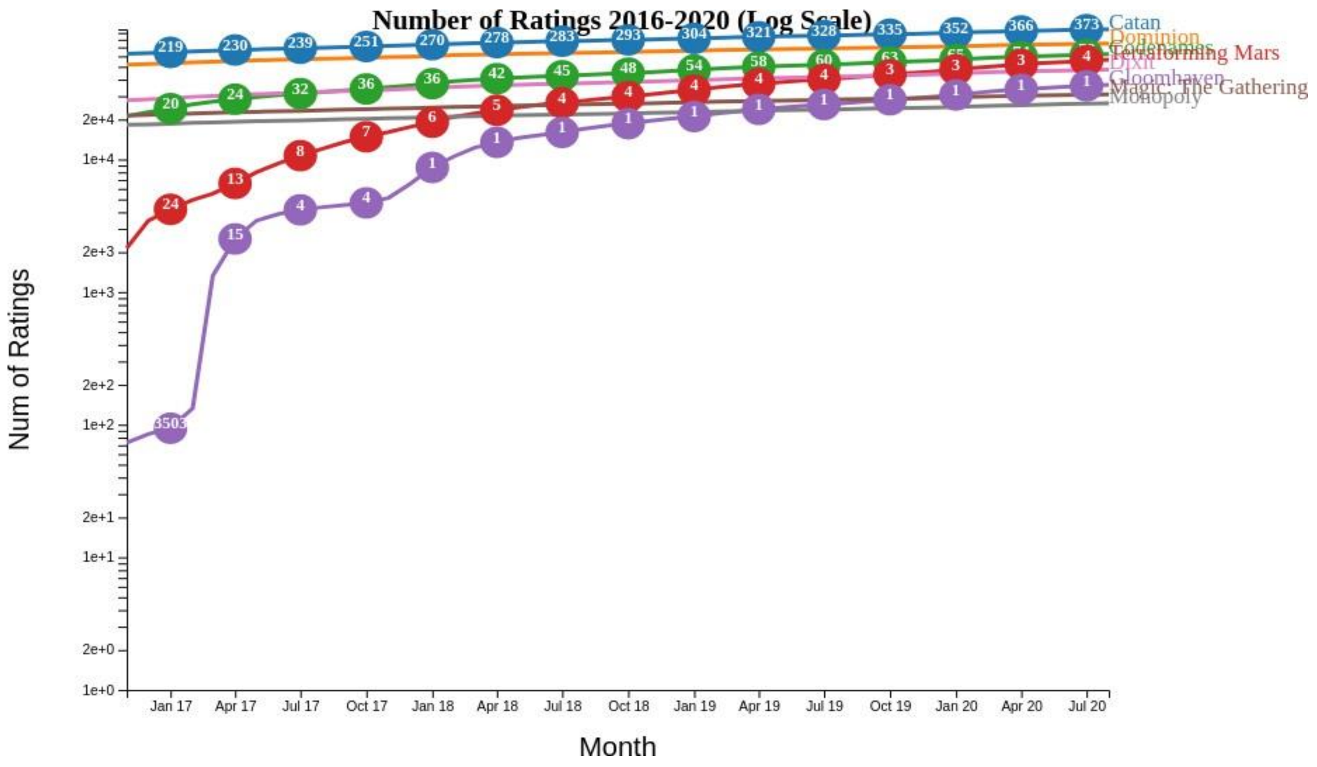

Figure 3c-2: Example of a line chart using log scale. Your chart may appear different, and can earn full credit if it meets all stated requirements.

Note: Your D3 visualization **MUST** produce the following DOM structure.

```
<svg id="svg-a"> plot (Q3.a)
|
+-- <text id="title-a"> chart title
|
+-- <g id="plot-a"> containing Q3.a plot elements
     |
     +-- <g id="lines-a"> containing plot lines, line labels
     |
     +-- <g id="x-axis-a"> x-axis
     |    |
     |    +-- (x-axis elements)
     |    |
     |    +-- <text> x-axis label
     |
     +-- <g id="y-axis-a"> y-axis
          |
          +-- (y-axis elements)
          |
          +-- <text> y-axis label

<svg id="svg-b"> plot (Q3.b)
|
+-- <text id="title-b"> chart title
|
+-- <g id="plot-b"> containing Q3.b plot elements
|    |
|    +-- <g id="lines-b"> containing plot lines, line labels
|    |
|   +-- <g id="x-axis-b"> for x-axis
|   |   |
|   |   +-- (x-axis elements)
|   |   |
|   |   +-- <text> x-axis label
|   |
|   +-- <g id="y-axis-b"> for y-axis
|   |   |
|   |   +-- (y-axis elements)
|   |   |
|   |   +-- <text> for y-axis label
|   |
|   +-- <g id="symbols-b"> containing plotted symbols, symbol labels
|
+-- <g id="legend-b"> containing legend symbol and legend text element(s)
```

`<svg id="svg-c-1"> plot (Q3.c1):` *same as format for Q3.b, with c-1 in ids (e.g., id="svg-c-1", etc.)*

`<svg id="svg-c-2"> plot (Q3.c2):` *same as format for Q3.b, with c-2 in ids (e.g., id="svg-c-2", etc.)*

`<div id="signature"> containing GT username`

Q4 [20 points] Interactive Visualization

Goal	Create line charts in D3 that use interactive elements to display additional data. Then implement a bar chart that appears when you mouse over a point on the line chart.
Technology	D3 Version 5 (included in the lib folder) Chrome v92.0 (or higher): the browser for grading your code Python http server (for local testing)
Allowed Libraries	D3 library is provided to you in the **lib** folder. You must **NOT** use any D3 libraries (d3*.js) other than the ones provided. On Gradescope, these libraries are provided for you in the auto-grading environment.
Deliverables	[Gradescope] **Q4.(html/js/css):** The HTML, JavaScript, CSS to render the visualization in Q4. Do not include the D3 libraries or average-rating.csv dataset.

Use the dataset `average-rating.csv` provided in the Q4 folder to create an interactive frequency polygon line chart. This dataset contains a list of games, their ratings and supporting information like the numbers of users who rated a game and the year a game was published. In the data sample below, each row under the header represents a game name, year of publication, average rating, and the number of users who rated the game. Helpful resource to work with nested data in D3: https://gist.github.com/phoebebright/3176159

```
name,year,average_rating,users_rated
Codenames,2015,7.71148,51209
King of Tokyo,2011,7.23048,48611
```

a. **[3 points] Create a line chart.** Summarize the data by displaying the count of board games by rating for each year. Round each rating down to the nearest integer, using Math.floor(). For example, a rating of 7.71148 becomes 7. For each year, sum the count of board games by rating. Display one plot line for each of the 5 years (2015-2019) in the dataset. **Note**: the dataset comprises year data from 2011 to 2020; this question asks to plot lines for the years 2015-2019. If some of the datapoints in the chart do not have ratings, generate dummy values (0s) to be displayed on the chart for the required years.

   All axes must start at 0, and their upper limits must be automatically adjusted based on the data. **Do not hard-code the upper limits. Note:** if you are losing points on Gradescope for axis or scale, ensure that you are using the proper margin convention without any additional paddings or translations.
   - The vertical axis represents the count of board games for a given rating. Use a linear scale.
   - The horizontal axis represents the ratings. Use a linear scale.

b. **[3 points] Line styling, legend, title and username.**
   - For each line, use a different color of your choosing. Display a filled circle for each rating-count data point.
   - Display a legend on the right-hand portion of the chart to show how line colors map to years.
   - Display the title "Board games by Rating 2015-2019" at the top of the chart.
   - Add your GT username (usually includes a mix of lowercase letters and numbers, e.g., gburdell3) beneath the title (see example figure 4b).

Figure 4b shows an example line chart design. Yours may look different, but can earn full credit if it meets all stated requirements.

Note: The data provided in `average-rating.csv` requires some processing for aggregation. **All aggregation must only be performed in JavaScript; you must NOT modify `average-rating.csv`.** That is, your code should first read the data from `.csv` file as is, then you may process the loaded data using

JavaScript. If you are getting a MoveTargetOutOfBoundsException, (a) check that your margin convention is correct, and (b) make sure to check the Dropbox linked screenshot of your graph to get a good idea of how the plot could be Improved compared to the sample graph provided.

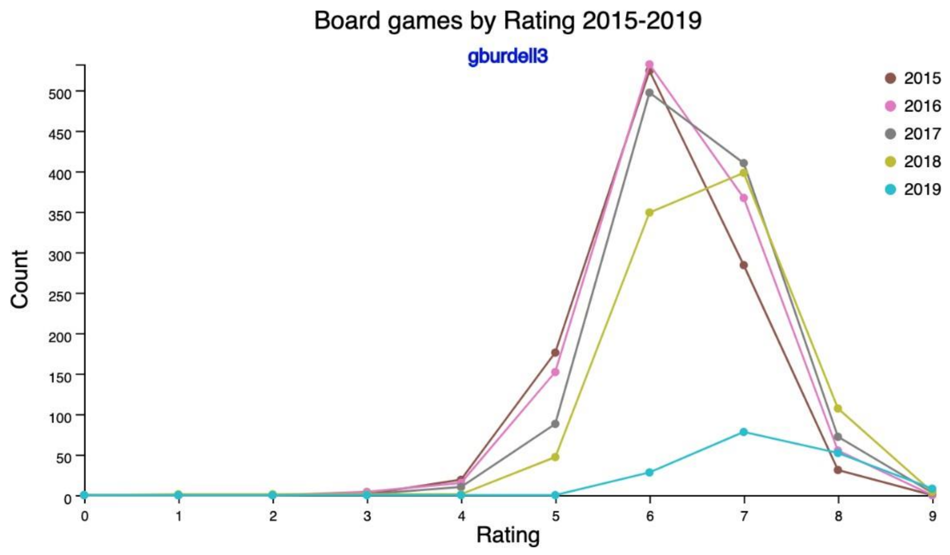

Figure 4b: Line chart representing count of board games by rating for each year. Your chart may appear different, but can earn full credit if it meets all stated requirements.

Interactivity and sub-chart. In the next few sub-questions, you will create event handlers to detect mouseover and mouseout events over each circle that you added in Q4.b.

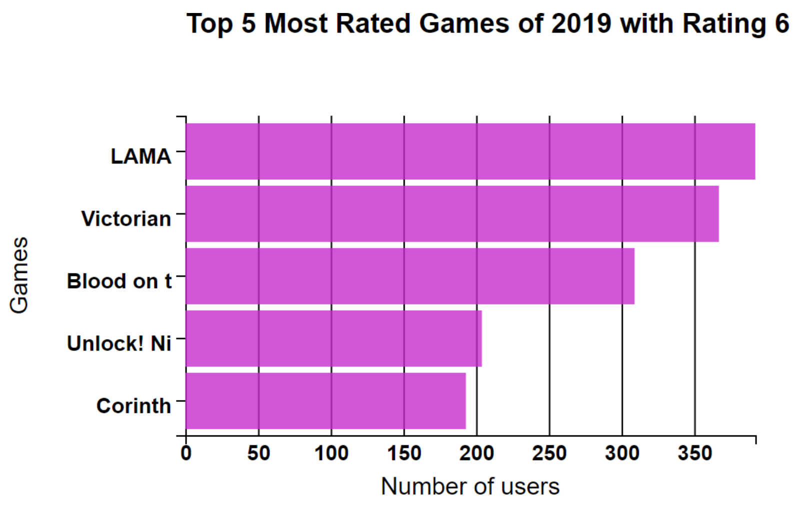

Figure 4c: Bar chart representing the number of users who rated the top 5 board games with the rating 6 in year 2019. Your chart may appear different, but can earn full credit if it meets all stated requirements.

c. **[5 points] Create a horizontal bar chart**, so that when hovering over a circle, that bar chart will be shown *below* the line chart. The bar chart displays the top 5 board games that received the highest numbers of user ratings (`users_rated`), for the hovered year and rating. For example, hovering over the rating-6 circle for 2019 will display the bar chart for the number of users who rated the top 5 board games. If a certain year/rating combination has fewer than 5 entries, it should display as many as

there are. Figure 4c shows an example design. Show one bar per game. The bar length represents the number of users who rated the game.
Note: No bar chart should be displayed when the count of games is 0 for hovered year and rating.

Axes: All axes should be automatically adjusted based on the data. Do not hard-code any values.

- The vertical axis represents the board games. Sort the game names in descending order, such that the game with the highest users_rated is at the bottom, and the game with the smallest users_rated is at the top. Some boardgame names are quite long. For each game name, display its first 10 characters (if a name has fewer than 10 characters, display them all). A space counts as a character. The horizontal axis represents the number of users who rated the game (for the hovered year and rating). Use a linear scale.
- Set horizontal axis label to 'Number of users' and vertical axis label to 'Games'.

d. [3 points] Bar styling, grid lines and title

- Bars: All bars should have the same color regardless of year or rating. All bars for the specific year should have a uniform bar thickness.
- Grid lines should be displayed.
- Title: Display a title with the format "Top 5 Most Rated Games of <Year> with Rating <Rating>" at the top of the chart where <Year> and <Rating> are what the user hovers over in the line chart. For example, hovering over rating 6 in 2015, the title would read: "Top 5 Most Rated Games of 2015 with Rating 6"

e. [3 points] Mouseover Event Handling

- The bar chart and its title should only be displayed during mouseover events for a circle in the line chart.
- The circle in the line chart should change to a larger size during mouseover to emphasize that it is the selected point.
- When count of games is 0 for hovered year and rating, no bar chart should be displayed. The hovered-over circle on the line graph should still change to a larger size to show it is selected.

Hint: .attr() is generally used for describing the size, shape, location, etc. of an element, whereas .style() is used for other design aspects like color, opacity, etc.

f. [3 points] Mouseout Event Handling

- The bar chart and its title should be hidden from view on mouseout and the circle previously mouseover-ed should return to its original size in the line chart.

The graph should exhibit interactivity similar to Figure 4f where the mouse is over the larger circle.

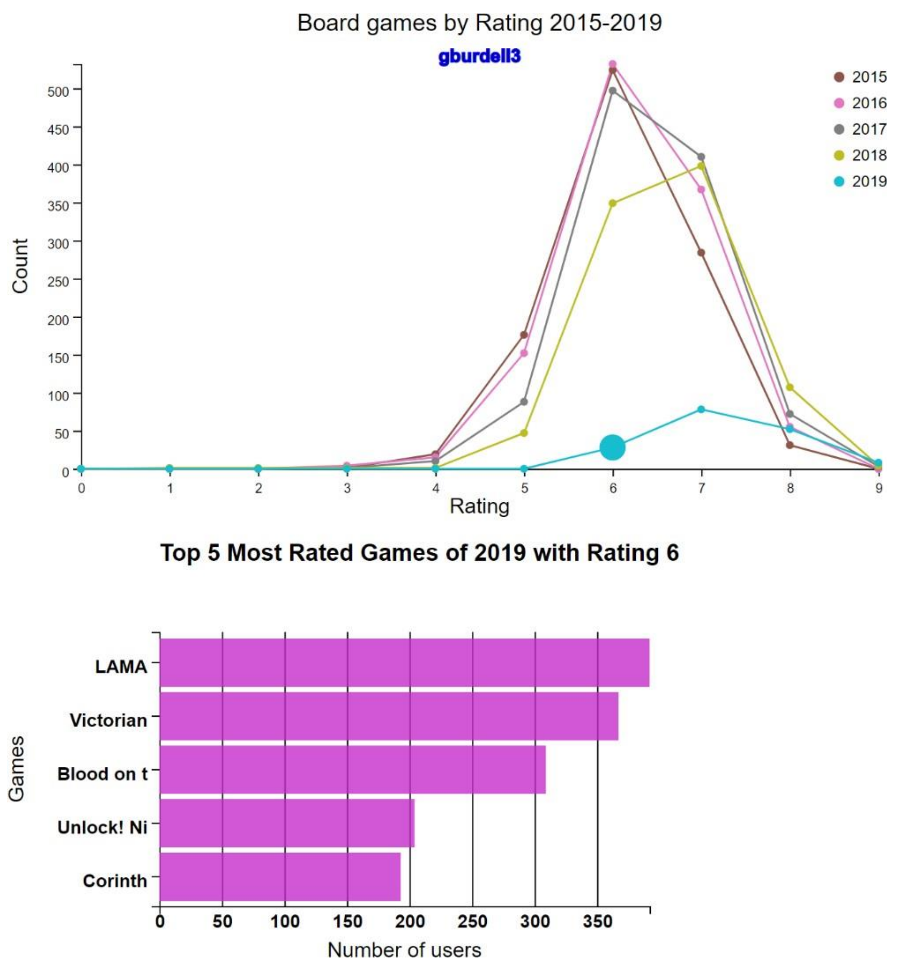

Figure 4f: Line chart and bar chart demonstrating interactivity. Your chart may appear different, but can earn full credit if it meets all stated requirements.

Note: Your D3 visualization **MUST** produce the following DOM structure.

```
<svg id=”line_chart”> containing line chart
|
+-- <g id="container">
       |
       +-- <g id=”lines”> element containing all line elements
       |      |
       |      +-- <path> elements for plotted lines
       |
       +-- <g id=”x-axis-lines"> element for x-axis
       |
       +-- <g id=”y-axis-lines"> element for y-axis
       |
       +-- <g id=”circles”> element for all circular elements
       |      |
       |      +-- <circle> elements
       |
       +-- <text id=”line_chart_title”> element for line chart title
       |
       +-- <text id=”credit”> element for GT username
       |
       +-- <g id=”legend”> element for legend
       |      |
       |     +-- (<circle> elements for legend)
       |     |
       |     +-- (<text> elements for legend)
       |
       +-- <text> element for x axis label
       |
       +-- <text> element for y axis label

 <div id=”bar_chart_title” > containing bar chart title

 <svg id=”bar_chart” > containing bar chart
 |
 +-- <g id=”container_2”>
       |
       +-- <g id=”bars”> element for bars
       |     |
       |     +-- <rect> elements for bars
       |
       +-- <g id=”x-axis-bars"> element for x-axis
       |
       +-- <g id=”y-axis-bars"> element for y-axis
       |
       +-- <text id=”bar_x_axis_label”> element for x axis label
       |
       +-- <text id=”bar_y_axis_label”> element for y axis label
```

Q5 [25 points] Choropleth Map of Board Game Ratings

Goal	Create a choropleth map in D3 to explore the average rating of board games in different countries.
Technology	D3 Version 5 (included in the lib folder) Chrome v92.0 (or higher): the browser for grading your code Python http server (for local testing)
Allowed Libraries	D3 library is provided to you in the **lib** folder. You must **NOT** use any D3 libraries (d3*.js) other than the ones provided. On Gradescope, these libraries are provided for you in the auto-grading environment.
Deliverables	[Gradescope] **`Q5.(html/js/css)`**: Modified file(s) containing all html, javascript, and any css code required to produce the plot. Do not include the D3 libraries or csv files.

Choropleth maps are a very common visualization in which different geographic areas are colored based on the value of a variable for each geographic area. You have most probably seen choropleth maps showing quantities like unemployment rates for each county in the US, or the number of confirmed COVID-19 cases per 10,000 people at county level.

We will use choropleth maps to examine the popularity of different board games across the world. We have provided two files in the Q5 folder, ***ratings-by-country.csv*** and ***world_countries.json***.

- Each row in *ratings-by-country.csv* represents about a game's information for a country, in the form of `<Game,Country,Number of Users,Average Rating>`, where
  - `Game`: the name of a game, e.g., Catan.
  - `Country`: a country in the world, e.g., United States of America.
  - `Number of Users`: the number of users who have rated `Game` who are from `Country`.
  - `Average Rating`: the mean rating given to `Game` by users who are from `Country`.

  This dataset has been preprocessed and filtered to include only those games that have been rated by more than 1000 users in the world.
- The *world_countries.json* file is a geoJSON, containing a single geometry collection: *countries*. You can find examples of map generation using geoJSON here.

a. **[20 points]** Create a choropleth map using the provided data, and use Figure 5a and 5b as references.

1. **[5 points]** Dropdown lists are commonly used on dashboards to enable filtering of data. Create a dropdown list (see example in Figure 5a) to allow users to select which game's data are displayed.
   - The list options should be obtained from the `Game` column of the csv file.
   - Sort the list options in (case-sensitive) alphabetical order. Set the default display value to the first option.
   - Selecting a different game from the dropdown list should update both the choropleth map (see part 2) and the legend (see part 3) accordingly.

   Hint: If failing to load map, you may try this method.

2. **[10 points]** Load the data from *ratings-by-country.csv* and create a choropleth map such that the color of each country in the map corresponds to the **average rating of the game selected in the dropdown** in each country. Use a "Natural Earth" projection of the geoJSON file. **Name the function for calculating path as 'path', to help the auto-grader locate it**. Create a projection first and use it as an input for the path.

Promise.all() is provided within the skeleton code and you can use it to read in both the world json file and game data csv file. Usage example: https://datawanderings.com/2018/08/15/d3-js-v5-promise-syntax-examples/.

Many countries have no ratings for some games - these should be **colored gray**.

For those countries that do have ratings for the selected game, use a quantile scale to generate the color scheme based on the average rating by country. Color them along a gradient of exactly 4 gradations from a single hue, with darker colors corresponding to higher rating values and lighter colors correspond to lower values (see gradient examples at Color Brewer).

About Scaling Colormaps: In order to create effective visualizations that highlight patterns of interest, it is important to carefully think about the relationship between the range and distribution of values being displayed (the domain) and the color scale the values are mapped to (the range). Many types of mapping functions are possible, e.g., we could use a linear mapping where the lowest game rating is mapped to the first value in the color scheme, the highest game rating is mapped to the highest value in the color scheme, and intermediate ratings are mapped to hues in the middle. This articleIllustrates the value of choosing appropriate endpoints for linear color maps, or log-scaling the domain so that large but relatively infrequent values do not cause differences between smaller values to be washed out. In our case, most board games have similar average ratings across countries, e.g. Catan has an average rating close to 9.3 in almost all countries, making it challenging to perceive relative differences in popularity. To address this, we can compute quantiles of the domain data—game valu es that divide the ordered list of average ratings per country into roughly equally-sized groups. Here, we will get 4 groups, a special case of quantiles called "quartiles" since the data are divided into quarters.

Hint: You can verify the correctness of the quartiles generated by using the 'quartile' function in Excel. Open ratings-by-country.csv and filter the data for one game (say Catan). Then use the quartile function to get the 0th, 1st, 2nd, 3rd and 4th quartile values from the Average Rating column. Here [0th quartile, 1st quartile), [1st quartile, 2nd quartile), [2nd quartile, 3rd quartile), [3rd quartile, 4th quartile] will represent the 4 groups of values generated by the d3 quantile scale. Use all the countries listed in ratings-by-country.csv to generate your quartiles depending on the selection, including ones which may not appear in the geoJSON.

3. **[5 points]** Add a **vertical** legend showing how colors map to the average rating for a particular game. The legend must update for the quartiles of the selected game, and display values formatted to show precision up to 2 decimal places. You must use **exactly 4** color gradations in your submission. It is recommended, but not required, to use d3-legend.min.js (in the **lib** folder) to create the legend for the scale you use. Also, **display your GT username** (e.g., gburdell3) beneath the map.

b. **[5 points]** Add a tooltip using the d3-tip.min library (in the **lib** folder). On hovering over a country, the tooltip should show the following information on separate lines:

- *Country* name
- *Game*
- *Avg Rating* for that game in that country
- *Number of Users* from the Country who rated the game

For countries with no data, the tooltip should display "N/A" for *Avg Rating* and *Number of Users*.

The tooltip should appear when the **mouse** hovers over the country. Figure 5b demonstrates this for Catan. On mouseout, the tooltip should disappear. You can position the tooltip a small distance away from the **mouse cursor and have it follow the cursor**, which will prevent the tooltip from "flickering" as you move the mouse around quickly (the tooltip disappears when your mouse leaves a state and enters the tooltip's bounding box). Alternatively, you may position the tooltip at a location (picked by you) such that it is close to the country the cursor is currently at. Please ensure that the tooltip is fully visible (i.e., not clipped, especially near the page edges). If the tooltip becomes clipped you may lose points.

Note: You **must** create the tooltip by **only** using **d3-tip.min.js** in the **lib** folder.

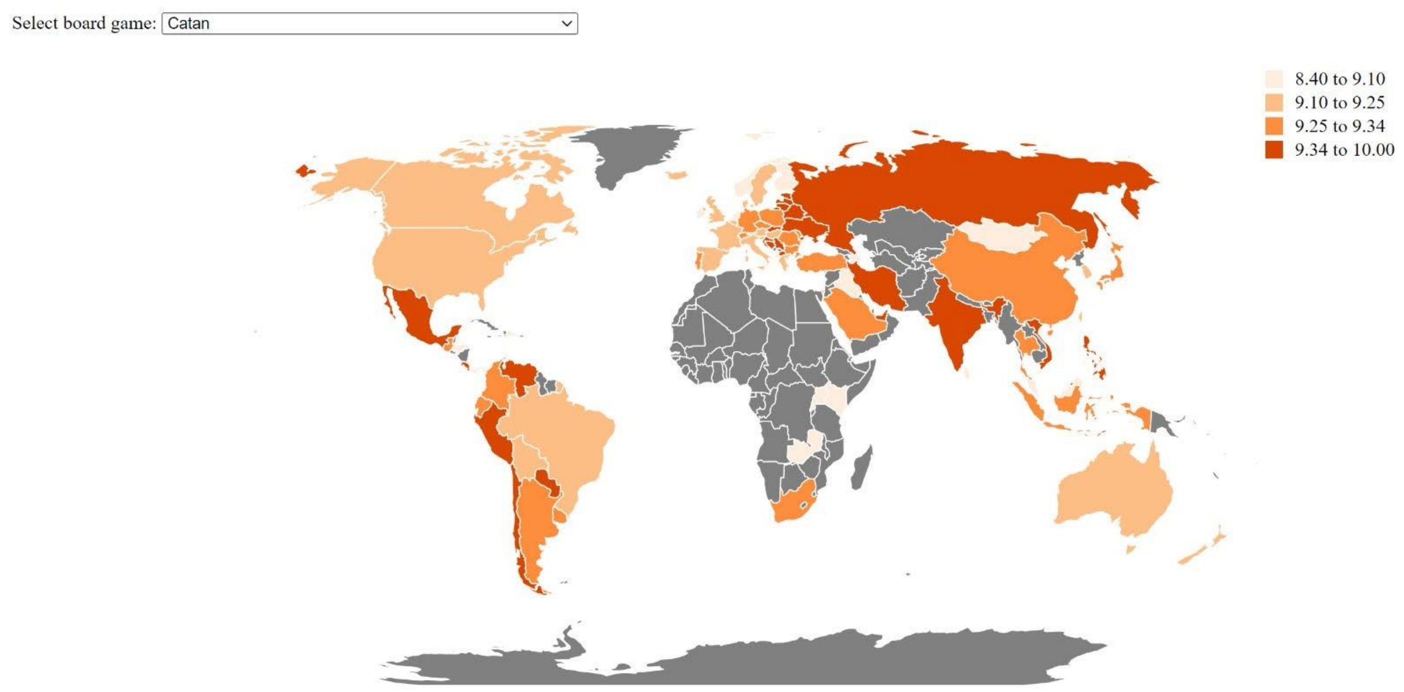

Figure 5a: Reference example for Choropleth Map showing average rating of Catan. Your chart may appear different and earn full credit as long as it meets all stated requirements.

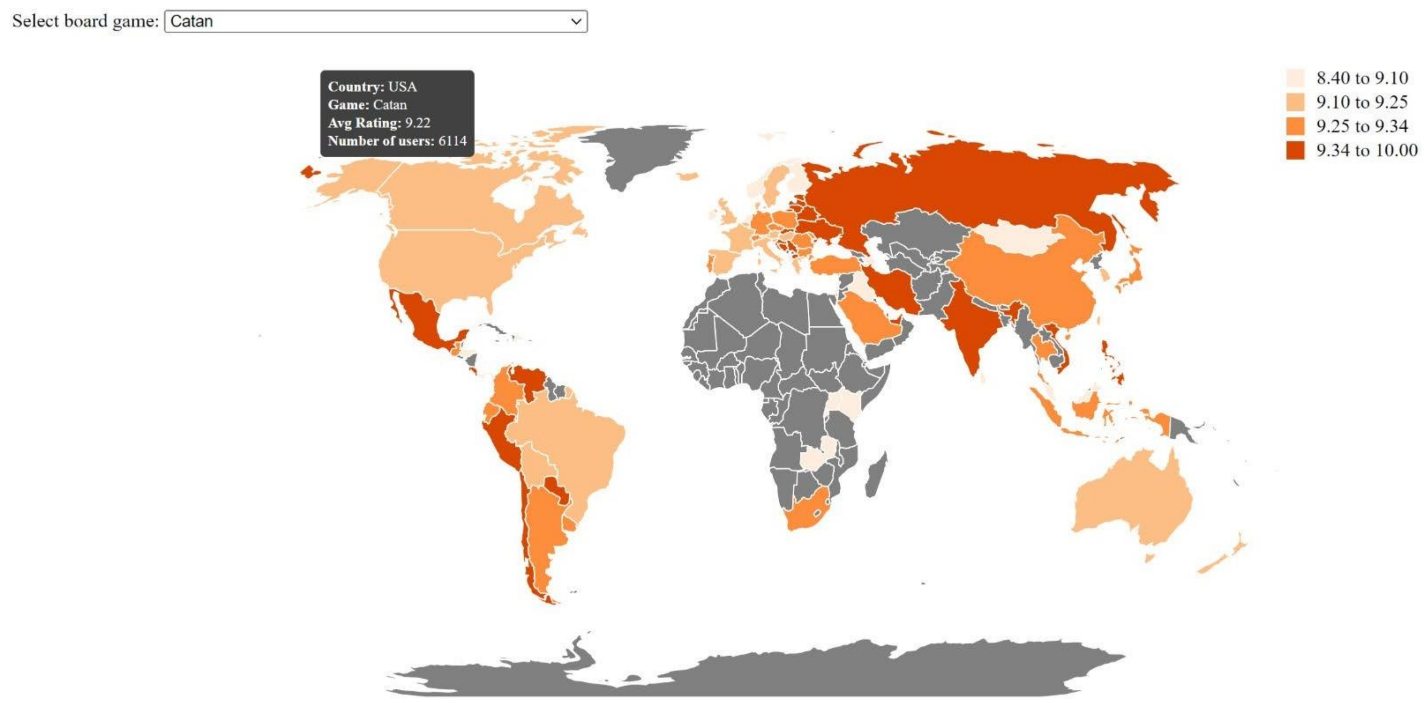

Figure 5b: Reference example for Choropleth Map showing tooltip. Your chart may appear different and earn full credit as long as it meets all stated requirements.

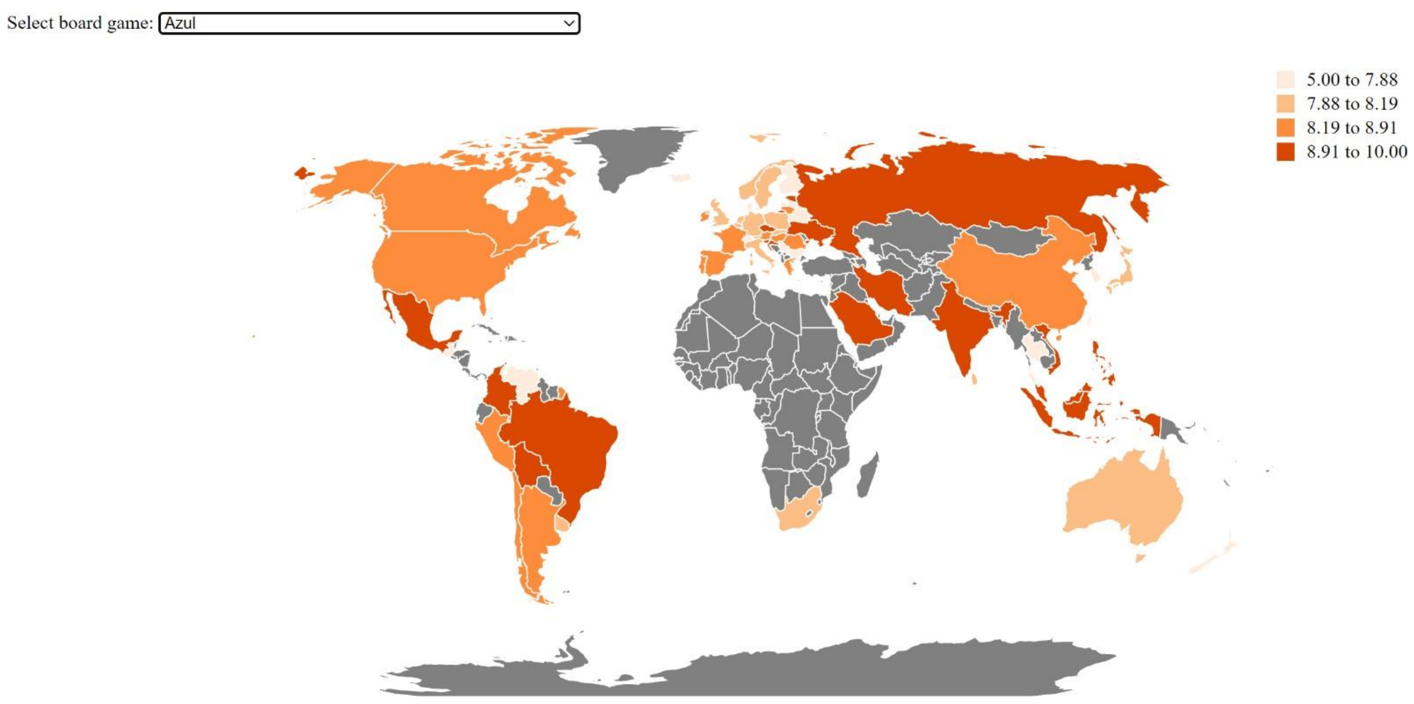

Figure 5c: Reference example showing updated Choropleth and legend for Azul. Your chart may appear different and earn full credit as long as it meets all stated requirements.

Hints

- Countries without data should be colored gray. These countries can be found using a condition that compares the country's average rating with 'undefined'.
- It is optional for your visualization to show (or not show) Antarctica.
- D3-tip warning may be ignored if it does not break the code.
- You may consider clearing the SVG and create a new map when selecting a new game.

Note: You may change the given code in choropleth.html as necessary. Your D3 visualization **MUST** produce the following DOM structure.

```
<select id=”gameDropdown”> sorted list of game options
|
+-- <option> (one option for each game; value = game name)
     |
     +-- text (= game name)

<svg id=”choropleth”> contains choropleth map
|
+-- <g id=" countries">
|    |
|    +-- <path>s for each country
|
+-- <g id=”legend”> legend

<div id=”tooltip”> contains tooltip to display (Q5.b)
|
+-- (text for tooltip)
```