CSE6242 / CX4242: Data and Visual Analytics | Georgia Tech | Fall 2015

Homework 2
D3 Graphs and Visualization

Due: October 2, 2015, 11:55PM EST

Prepared by Meera Kamath, Gopi Krishnan, Siddharth Raja, Ramakrishnan Kannan, Akanksha, Polo Chau

Submission Instructions:

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**.

- Submit a **single zipped file**, called "HW2-{YOUR_LAST_NAME}-{YOUR_FIRST_NAME}.zip", containing all the deliverables including source code/scripts, data files, and readme. Example: 'HW2-Doe-John.zip' if your name is John Doe. Only .zip is allowed (no .rar, etc.)
- You may collaborate with other students on this assignment, but **each student must write up their own answers in their own words**, and must write down the collaborators' names on T-Square's submission page. Any suspected plagiarism and academic misconduct will be reported and directly handled by the Office of Student Integrity (OSI).
- If you use any "*slip days*", you must write down the number of days used in the T-square submission page. For example, "Slip days used: 1"
- At the end of this assignment, we have provided a folder structure that describes how we expect the files to be contained in that single zipped file. Please make sure you submit your files in the format specified. 5 points will be deducted for not following this strictly.
- Wherever you are asked to write down an explanation for the task you perform, please stay within the word limit or you may lose points.
- In your final zip file, please 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).

Part 0: Prerequisites

Please download this zip file, which contains all the datasets that are to be used in this assignment.

You may need to setup your own HTTP server to properly run your D3 visualizations. The easiest way is using SimpleHTTPServer in Python. See this for details.

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

Please include all the files related to d3*.js in the lib folder, and use a relative path to reference these files from html files in other folders i.e. Q1, Q2, etc.

Part 1: Visualizing Data using D3 [60 pts]

1. **[10 pts]** Use the dataset[1] provided in the file *iris.tsv* (in the folder Q1) to create a scatterplot visualization.
   *Anticipated time needed: 3 hours.

   Refer the tutorial for scatterplot here.
   Features/ Attributes the dataset:
   1. Sepal length in cm
   2. Sepal width in cm
   3. Petal length in cm
   4. Petal width in cm
   5. Class: Iris Setosa, Iris Versicolor, Iris Virginica

a. [6 pts] **Creating scatter plots**:
   i. [5 pts] Create two scatterplots using the features/attributes below. Visualize the classes in the plot using different symbols(circle, square and triangle) and add a legend of how symbols map to the classes
      1. Feature 1 and 2
      2. Feature 3 and 4
   ii. [1 pts] In no more than 40 words discuss the plot that separates the classes more appropriately in the file **explanation.txt**

Note: Use the graph created using feature 1 and 2 for the following questions:

b. [1 pt] **Scaling symbol sizes.** Set the size (side, radius in pixels) of each symbol in the plot to be proportional to square root of the the length parameter.
c. [2 pts] **Scales.** Plot the graphs using any two other scales available in D3. Explain in no more than 40 words which scale works best for this dataset in **explanation.txt**

Q1 Deliverables:

- **The directory structure should be as follows:**

```
Q1/
   scatterplot.(html / js / css)
   explanation.txt
   scatter_plots.yyy
   scales.yyy
   iris.tsv
```

- **scatterplot.(html / js / css)** - the html/js/css files created.
- **explanation.txt** - the text file explaining your approach for Q1.a-ii and Q1.c. Keep it succinct.

[1] Source: https://archive.ics.uci.edu/ml/datasets/Iris

- **scatter_plots.yyy** - a png, jpg, or pdf screenshot of the two scatterplots created in Q1.a
- **scales.yyy** - a png, jpg, or pdf screenshot of the different scaled plots from Q1.c.

2. **[10 pts]** Use the dataset[2] in the file *hourly_heatmap.json* for glucose readings throughout the day in the folder Q2 inside the zipped folder and visualize it using a D3 heatmap. To get started, refer to the heatmap example here.
*Anticipated time needed: 3 hours.

a. [6 pts] Plot the glucose readings against the time of the day (Hint: Use the glucose readings as a "**z**" parameter in the given example)
b. [3 pts] Now use the file *day_heatmap.json* (in the folder Q2) to plot the glucose readings against the day of the week (use the day names instead of numbers as the steps on the axis, day 1 being Monday) on the heatmap.
c. [1 pt] A pattern should emerge from the visualizations. Explain the pattern and why it occurs in no more than 40 words, in the **explanation.txt**

Q2 Deliverables:

- **The directory structure should look like (remember to include the d3 library):**

```
Q2/
   heatmap.(html / js /css)
   heatmap.yyy
   explanation.txt
   day_heatmap.json
   hourly_heatmap.json
```

- **heatmap.(html / js/ css)** - the html / js / css files created.
- **explanation.txt** - the text file explaining with your answer for Q2.c. Keep it succinct (each answer should be no more than 40 words).
- **heatmap.yyy** - a png, jpg, or pdf screenshot of the plots of heatmap created in Q2.b.

[2] Source: https://github.com/jebeck/iPancreas-archive

3. [20 pts] Data Pre-processing - generate json file for Graph visualization.

*Anticipated time needed : 2 hours

You will prepare a json file which will be the source for the next question Q4, where you create a Force-Directed Graph Layout visualization. You are given a csv file[3] *arsenal_players.csv* (in the folder Q3) which contains the data about players' history from the English Premier League Soccer Club Arsenal FC.

The data in the csv file contains the following fields:
- name: name of a player.
- position: what position the player played in (e.g. F = forward, G = goalkeeper, etc.).
- start_season: year in which the player joined Arsenal FC (first season at Arsenal).
- end_season: year *after* which the player left Arsenal FC i.e. last season at Arsenal.
- transferred_from: the club the player was at before joining Arsenal.
- transferred_to: which club the player left Arsenal for.
- appearances: number of matches played as an Arsenal FC player.
- goals: goals scored for Arsenal FC.

The json file you create from the csv file should follow the following format:

```
{
“nodes”:
      [ { node1 }, { node2 }, { node3 }, ……., { node n } ],
“links”:
      [ { link 1}, { link 2 }, ……., { link m } ]
}
```

[8 pts] **Create nodes**: Your job is to take this CSV file and create 50 nodes for the graph where each node corresponds to one player. The csv file has 359 players. You will choose the top 50 players that have the highest number of appearances. If two or more players have same number of appearances break the tie using alphabetical ordering from the name field. Choose name, position, appearances, goals as the attributes for the nodes.

Format for each node:

```
{"name":"O'LEARY David","position":"CD","appearances":558,"goals":11}
```

[10 pts] **Create links**: Now that you have created the nodes, you will create links between those nodes. There is a link between any two players if their careers at Arsenal FC overlapped by at least one year.

[3] Source: http://www.neilbrown.newcastlefans.com/

For example if player X played at the club from 1990 (START_SEASON = 1990) to 1995 (END_SEASON = 1995) and Y played at the club from 1995 to 1997 then they both played for Arsenal in 1995. So they have an overlap of 1 year. Hence there will be a link between them and the weight (value) of that link will be 1. In another example, X (1990-1994) and Y (1995-2004) do not have any overlap and hence no link will be drawn between these two nodes. However, A (2005-2013) and B (2002- 2007) will have a link between them since they were both at the club from years 2005-2007 and hence the value of this link is 3 years (seasons 2005, 06, 07).

The links should be of the following format:

```
{"source":0,"target":1,"value":11}
```

where "source":0 indicates player at index 0 in the node list and "target":1 indicates player is at index 1 in the node list. The "value": 11 indicates that they had an overlap of 11 years in their Arsenal FC career.

Remove *duplicate links*. If a link has already been drawn from 0 to 1, there shouldn't exist another link entry going from 1 to 0. We suggest that you always draw links from a smaller ID to a larger ID (e.g.: 0 to 3 or 24 to 28 and not 3 to 0 or 28 to 24).

Format of your overall json object:

```
{
"nodes":[
{"name":"O'LEARY David","position":"CD","appearances":558,"goals":11},
{"name":"ADAMS Tony","position":"CD","appearances":504,"goals":32},
        ..........
        ..........
{"name":"LJUNGBERG Freddie","position":"W","appearances":216,"goals":46},
{"name":"SAMMELS Jon","position":"M","appearances":215,"goals":39}
],
"links":[
        {"source":0,"target":1,"value":11},
        {"source":0,"target":2,"value":3},
        ..........
        ..........
        {"source":44,"target":46,"value":2},
        {"source":44,"target":48,"value":1},
        {"source":45,"target":47,"value":8}
                                                                              ]
}
```

Name this file as **afc.json.** This is the file you will use as the input for the question Q4.

We expect you to write a script/program to pre-process this csv file and create the afc.json. The script can be in any language (java, js, python, shell script, etc.). The script should take as input the csv file and produce afc.json in the format given above.

[2 pts] **explanation.txt:** Finally add an explanation.txt file describing the steps you used in your script file to generate the json in *not more than 40 words*.

Q3 Deliverables:

- **The directory structure should look like :**

  Q3/

  Q3Script.xxx (your custom script in the language of your choice)
  afc.json
  explanation.txt (40 words max)

4. [20 pts] Force-Directed Graph Layout in D3

*Anticipated time needed: 7 hours.

You will experiment with many aspects of D3 for graph visualization. To get you started, we provide *graph.html* (in the folder Q4). For this question you will use the afc.json you created in Q3.

a. [2 pts] **Adding node labels.** Modify the html to *show a label* to the right of each node in the graph. The label should be the name of the player that node represents. If a node is dragged, its label must also move with the node.

b. [2 pts] **Coloring nodes.** Color the nodes based on the "position" field in the json file (i.e., all nodes from the same position have the same color). You *get to choose the colors*. The goal is to make the groups visually distinguishable from each other. We suggest using a color scheme (can be grayscale) that also works for people with colorblindness. (Hint and color brewer.)

c. [3 pts] **Scaling node sizes.** Adjust the radius of each node in the graph based on how many goals a player has scored. Choose the json file's "goals" field for each player.
   i. [1 pts] Use this metric to scale the radii of the nodes *linearly*. This means players with higher number of goals will be represented as larger nodes. Take a screenshot of the whole graph with linearly scaled node size (Polo recommends Skitch for taking screenshots).
   ii. [1 pts] Now scale the radii by the *square root* of goals scored. Take a screenshot of the whole graph with square root scaled node size.
   iii. [1 pts] In no more than 40 words, discuss which approach works better for this problem and why. Place this response in **explanation.txt.**

d. [1 pts] **Filtering node labels.** Only show the labels for "experienced" players, who have made more than 250 appearances.

e. [3 pts] **Pinning nodes (fixing node positions).** Modify the html so that when you

double click a node it fixes the node's position. Mark fixed nodes so that they are visually distinguishable from unfixed nodes, e.g., pinned nodes can be shown in a different color, or border thickness, or visually annotated with a "star" (*), etc. The rest of the nodes' positions should remain unfixed. Double clicking a fixed node should unfreeze its position and unmark it.
Hint: the easiest way to do this is to add a doubleclick event-listener to the nodes.

f. [5 pts] **Tooltips.** Using the d3-tip library, add a tooltip for each node. A "**mouseover"** event on the node, should display a tooltip containing the name, position and appearances associated with the node (player).
Note: You should also add tooltips for nodes whose labels have already been filtered out in Q4.d.

g. [4 pts] Improve the visualization by making it easier to read the text labels. There are several possible routes to making the text more readable (Hint: balancing font size and the graph layout parameters to reduce clutter).
    i. [3 pts] The changes you make should be included in the final version of **graph.html** that you turn in.
    ii. [1 pts] In 40 words or fewer tell us what you did to make your graph less cluttered in **explanation.txt**.

Q4 Deliverables:

- **The directory structure should be as follows:**

```
        Q4/
          graph.(html / js / css)
          afc.json
          linear_nodes.yyy
          squareroot_nodes.yyy
          explanation.txt
```

- **graph.(html /js /css)** - the html file based on the initial code that contains the changes made in (a-g) above and js/css files created (if any).
- **explanation.txt** - the text file explaining your approach for Q4.c-iii, Q4.g-ii. Keep it succinct (each answer should be no more than 40 words).
- **linear_nodes.yyy** - a png, jpg, or pdf screenshot of the linearly scaled nodes from Q4.c-i.
- **squareroot_nodes.yyy** - a png, jpg, or pdf screenshot of the square root scaled nodes from Q4.c-ii.

Part 2: Visualizing Statistics of Refugees in Europe [50 pts]

*Anticipated time needed for Q5 : +12 hours, Q6 : 5 hours.

5. **[35 pts]** After the training in Part 1, assume that you are analyzing data for United Nations High Commissioner for Refugees (UNHCR) and need to perform the following tasks to aid in the UNHCR's understanding of the persons of concern, considering the ongoing European migrant crisis.

a. [5 pts] Convert the dataset[4] provided in the file *unhcr_persons_of_concern.csv* (in the folder Q5) to a json file (poc.json). This dataset represents a subset of the information available from UNHCR about Persons of Concern during the timeframe 2005-2015 residing in select European countries . The conversion to json file can be done by hand, a tool, or a script of your choice. Only the json file will be graded so you don't need to turn in the script.

b. [5 pts] **Table.** Create a table to display details of the refugees in the year 2005. You can use **any tool** (e.g., Excel, HTML, Tableau, D3) you want to make the table. Keep suggestions from class in mind when designing your table (see lectures slides for what to and what not to do, but you are not limited to the techniques in the slides). Your visualization needs to convey the data clearly and effectively to the reader (displaying all rows that match the filter may not be an effective visualization). No user interaction is required. Describe your reason for choosing the techniques you use in **explanation.txt** in not more than 50 words**.**

c. [25 pts] **Bar charts. *Using D3 & Tableau***, visualize attributes like country of origin and number of refugees.

   i. [20 pts] **D3:** Construct two bar charts as follows: The first chart should contain the origin country vs total number of refugees with a dropdown of the year (as shown in the example bar chart below). Take special care in the first chart to calculate the sum of refugees across all residence countries while creating the first chart. In the second chart, pick any 5 residence countries from the above list and show the refugee count in each for the given year. Please note that the **data in both graphs should change based on the year selected in the dropdown of the first chart**. All charts must be shown on the same webpage.

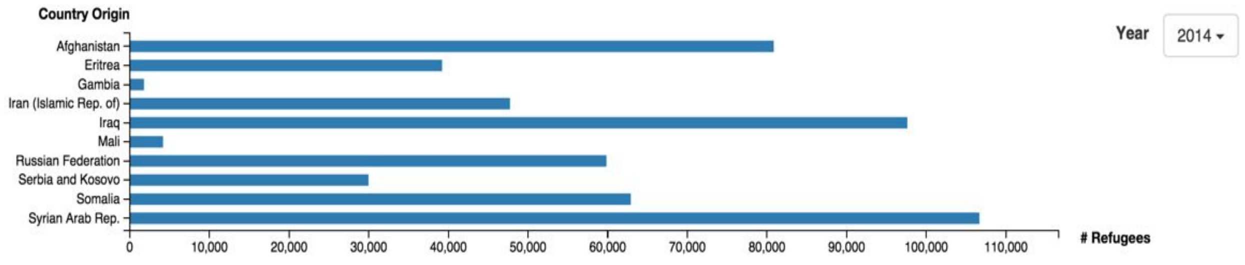
   Below are example **bar charts** we would like you to use as a reference. For all other properties of the chart, use what you have learned from class (see lectures slides) to make it effective in conveying information and visually pleasing (e.g., include axis labels, choose a good color scheme, etc).

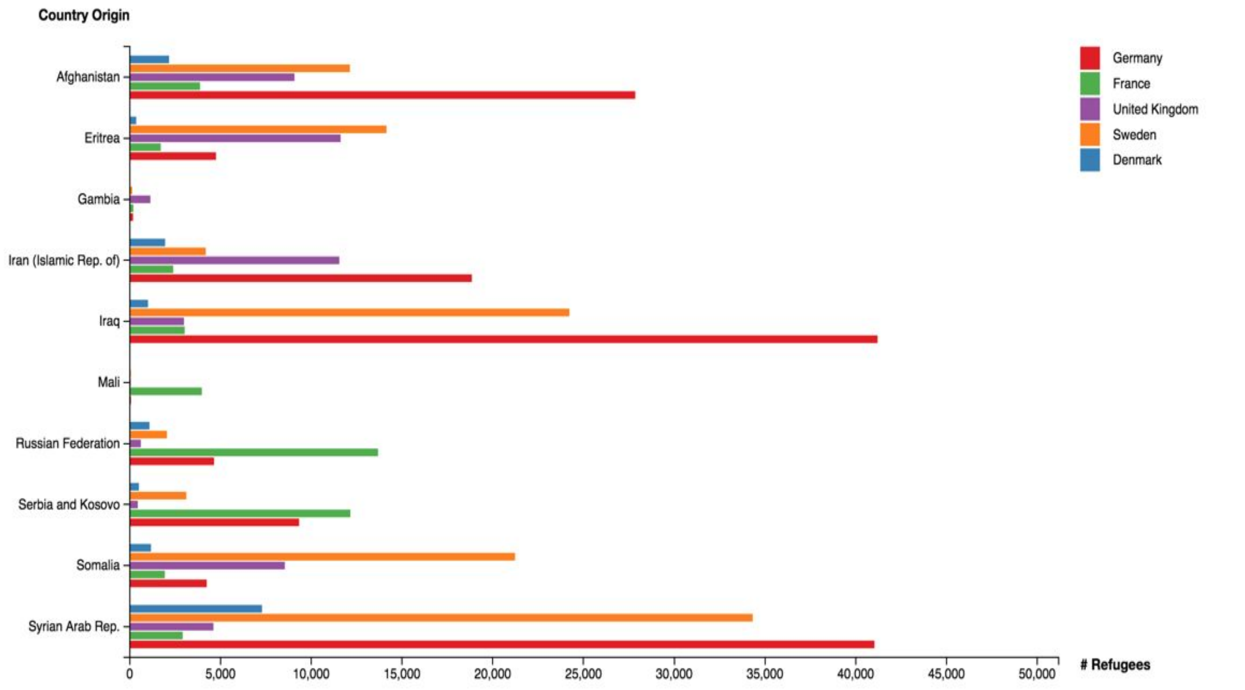
[4] Source: http://popstats.unhcr.org/en/persons_of_concern

Note:
Please use d3.js only. You cannot use libraries built on top of d3.js or other charting libraries for this assignment.

You must use the json file(poc.json) created in Q5a as the datasource while creating the charts.

Hints*:* You may find the following functions useful - d3.nest(), array.filter(), array.map()





ii. [5 pts] **Tableau:** Visualize the demographic attributes(age, sex, country of origin, asylum seeking country) in the file *unhcr_popstats_demographics.csv* (in the folder Q5) for any given year in one chart. Tableau is a popular InfoViz tool and the company has provided us with student licenses. Go to http://www.tableau.com/tft/activation , and select Get Started. On the form, enter your Georgia Tech email address for “Business email” and “Georgia Institute of technology” for "Organization". The Desktop Key for activation is on T-Square. Do not share the key with anyone.

Provide a rationale for your design choices in this step in the file **explanation.txt** in not more than 50 words.

Q5 Deliverables:

- **The directory structure should be as follows:**
  Q5/
  poc.json
  table.yyy
  bars.(html / js/ css)
  bars.xxx (from Tableau)
  explanation.txt
- **table.yyy** - Any modern image format (e.g., jpg, png, pdf) showing the table created in Q5.b.
- **bars.html / js / css** - The html, javascript, css to render the bar graphs requested in Q5.c.i.
- **bars.xxx** - The figure for bar charts generated from Tableau in Q5.c.ii. You can use formats like png and pdf, but be sure to make it a high-quality and clear image).
- **explanation.txt** - Write the explanation for parts Q5.b and Q5.c.ii in this file. Keep it succinct.

6. **[15 pts] Visualization.** Using D3, construct a visualization that addresses the plight of refugees / ongoing refugee crisis in Europe.

You can have one large visualization or multiple small ones. (The visualization does **not** need to support any interactions.)

You could use these data sources, or any others you may find relevant to this topic:
http://popstats.unhcr.org/en/
http://data.worldbank.org/indicator/SM.POP.REFG

- **Do not turn in a bar graph or table.** The point of this task is for you to take a design idea you've created yourself and implement it in D3 as best you can. Points will be awarded for functionality, and also for interesting ideas.
- Discuss your idea behind the visualization in **explanation.txt** in not more than 50 words**.**

Q6 Deliverables:

- **The directory structure should be as follows:**
  Q6/
  q6.(html /jss /css)
  *.json
  explanation.txt

- **q6.(html /js /css**)- The html, javascript, css files to render the visualization made in Q6.
- ***.json** - Include all the json file(s) used as data sources.
- **explanation.txt** - As described in the question.

Survey

We would greatly appreciate it if you would spend a few minutes to complete this survey related to the time spent by you while completing this assignment
https://docs.google.com/forms/d/18CCYwo8GFUqhwmF2_gooqfDcIPtaN225CQDxwheC3j0/viewform
Your responses would help us give a more reliable estimate of the time needed to complete the assignment in the future. Thanks!

Important Instructions on Folder structure

We will be executing the following commands to validate the submission is all right. If some files are missing in our query of the zip folder, marks will be deducted. For example, if Q1/explanation.txt is missing, points could be deducted.

unzip -l HW2-LastName-FirstName.zip | grep -c txt should give **X** files.
Similarly for html, json, csv, js files as well.

The directory structure should be as follows:

```
                    HW2-LastName-FirstName/
                                  |--- lib/
                                          |----   d3/
                                                   |----    d3.v3.min.js
                                                   |----     d3-tip.js
                                  |--- Q1/
                                          |----   scatterplot.html
                                          |----   explanation.txt
                                          |----   scatter_plots.yyy
                                          |----   scales.yyy
                                          |----   iris.tsv
                                  |--- Q2/
                                          |----   heatmap.html
                                          |----   explanation.txt
                                          |----   heatmap.yyy
                                          |----   day_heatmap.json
                                          |----   hourly_heatmap_json

                                  |--- Q3/
```

```
        |----   Q3Script.xxx (your custom script)
        |----   explanation.txt
        |----   afc.json
|--- Q4/
        |----   graph.html
        |----   explanation.txt
        |----   linear_nodes.yyy
        |----   squareroot_nodes.yyy
        |----   afc.json
|--- Q5/
        |----    poc.json
        |----    table.yyy
        |----    bars.(html / js /css)
        |----    bars.xxx (from Tableau)
        |----    explanation.txt
|--- Q6/
        |----    *.json
        |----    q6.(html / js/ css)
        |----    explanation.txt
```