CS/INFO 3300; INFO 5100
Homework 6
Due 11:59pm Wednesday, October 4

Goals: Practice using d3 to create some simple charts. Get more experience importing data and working with data joins.

Your work should be in the form of an HTML file called index.html or index.htm with one **<p>** element per problem. Put your netID at the top of the file in a <p> tag. If you must add any SVG canvases programmatically, we suggest that you add a ** ** element to the HTML to hold each SVG to make positioning a bit easier. For this homework we will be using d3.js. **In the <head> section of your file, please import d3 using this tag: <script src="https://d3js.org/d3.v7.min.js"></script>**

Create a zip archive containing your **HTML file and all associated data files** (such as wines.json) and upload it to CMS before the deadline. Submissions that do not include data files may be penalized. Your submission will be graded using a Python web server run in a parent directory containing your zip file contents along with many other students' submissions.

1. You've now seen a few scatterplots in class as well as developed your own in previous assignments. In this problem you will once again be developing a scatterplot, however this time many of the specific **design features of the plot will be left to you**. In the homework ZIP file you will find **wines.json**, a subsample of a larger dataset of wine reviews sampled from the web (source). We have sampled wine scores from four US wine regions: New York, Washington, Oregon, and California. Please use this dataset to complete this homework. For each of the following sub-problems, use a **<p>** tag as directed to briefly **explain your procedure or design rationale for that step** (e.g. How did you decide on axis scale labels? What made you choose a linear/log scale? What compromises did you make?).

Your goal in this problem is to create a plot that can help to answer two questions:

- Is there a relationship between the `price` of wines and their 0-100 scores on a quality `point` scale?
- Are some wine regions better or more expensive than others?

(next page)

A. This data file isn't exactly perfect. In fact, we've gone ahead and **added some fake points with confusing, missing, or bad data values**. We used a Python script to make the fake points, so there is a **pattern that all bad points share** (hint: while the fake names are silly, there are other factors you can examine which have patterns; you don't need to be familiar with wine types and brands to find the fake data). Worse, we've not bothered fixing any types or standardizing values. Begin by loading the data file using `d3.json`. Feel free to use `d3.autotype` if you'd like, but it may not convert everything cleanly. You may need to loop through and manually convert strings into numbers -- after cleaning up some pesky characters (such as dollar signs or commas) so you don't get NaNs.

Please note that some issues may not become obvious until you begin working on the scatterplot. For now, we recommend that you correct number parsing issues but continue creating your basic chart so that you can spot data issues more easily. Save filtering until after step C. At each step of development, check back and make sure that the data you are working with appear to be correct. For instance, if you continue working only to find that an axis looks extremely skewed, inspect the points that are skewing your axis to see if they follow a pattern (hint: filtering purely based on `points` or `price` is not going to work, but we have provided 3 consistent attributes that reveal all bad points).

Use **filter** and/or **forEach** to **hide or correct any important data quality issues**. Describe what data issues you **found** and **how you fixed them** in your <p> tag. Please write an **exact count** of the number of bad data items you removed. Additionally, after filtering, please print the length of your data array to the console to prove that you've got the right number of remaining points.

B. Create an SVG object that is **800 pixels in width and 500 pixels in height**. Give it an ID so that you can use selectors to find it later. You are going to be graphing **`price` on the x-axis**, showing **`points` on the y-axis**, and **coloring them by state**. In this step, please **find extents and construct scales for your chart**. Create **<g> elements**, use **translate** to move them to an appropriate place, and **populate them with `d3.axis` labels**. Use a **second set of `d3.axis` objects** to create **corresponding x- and y-axis gridlines** in a light color. You must use `d3.axis` for your gridlines.

Feel free to choose whatever margin/padding values, domain, range, log/linear scales, colors, and axis formatting styles you like. Use the <p> tag to **explain the choices you made** in designing the axes/scales. We will reward scales/axes that are legible, show the distribution of data clearly (or as clearly as possible), use color effectively, and avoid visual clutter. **A good rationale in the <p> for your design will outweigh problems in these areas.**

C. Create a **<g> element** and **translate** it so that it can act as your **main chart region**.
Begin by **creating a new function, `jitter()`, which returns a random number between -4 and 4.** Now, populate the chart with **<circle> elements corresponding to valid data points**. Move, scale, and color them as necessary using the structures you built in step B. When you are positioning the circles, **add a random number from your jitter function to both the x and y position**. This will reduce the amount of overlap when circles have the same point and price. You may still want to reduce the opacity of circles if there is heavy overlap. In addition to **writing about your design rationale for the choices you made in creating circle elements in your <p> tag**, please also **write 2-3 sentences identifying 1 advantage and 1 disadvantage of this jittering approach** to managing overlap.

D. Using `d3.on("mouseover", ...)` and `d3.on("mouseout", ...)`, provide users a way to move their mouse **onto points** and see the **"title" of the wine they are hovering over**. At minimum, points should **grow in size** and a **floating <text> label should appear nearby** when the mouse enters the inside of a circle. Your label can be either in a fixed position in the canvas or a variable position near where your mouseover is located. A corner is fine, so long as the label and a circle do not experience a race between `mouseover` and `mouseout` due to overlap (this will appear as a flashing label). Do not use HTML elements; your event actions must happen entirely within SVG canvas elements. When the mouse leaves the circle, it must **return to its normal appearance**. Feel free to add more complexity to make the highlighting of points more obvious or the text more legible, but this is not explicitly required. **Write 2-3 sentences in your <p> tag** describing the possible **benefits to users** from this approach and identifying places in the chart where it may be **ineffective or confusing** (or, if you fixed them, what you did to improve the user experience).

Part E of this problem is completely optional.
Students who successfully complete part E will receive one free quiz grade drop. If you've missed a quiz, this is an easy chance to win back one of those 0 grades.

E. Finally, create a simple legend for the chart with mouseover filters. First, **add a element underneath your SVG** for your legend labels. For each of the different states in the dataset, add a ** tag for that state. Color the text** with the corresponding color in your chart. (hint: if you use a scaleOrdinal to set the colors, you can access scaleOrdinal.domain() after you finish adding the circles – you might consider a *loop* that adds elements, colors them, and adds interactivity)

Using `d3.on("mouseover", ...)`, make it so that **whenever you mouse over a label, the corresponding circles on the charts appear with high opacity** and the **other circles have reduced opacity** (hint: data joins give you access to the data embedded every circle – think about how to configure a `.style(...)` call to use data *conditionally*). Additionally, **add a final element labeled "Clear"** in black that **returns all of the circles to their original opacity** when you mouse over it. Note: you do not need to do anything special with your mouseover labels when a filter is applied – it is okay if you can see the names of points that are filtered. You can talk about design elements in your <p> tag, but you are not required to add anything for this sub-problem.