CS/INFO 3300; INFO 5100
Homework 3
Due 11:59pm Monday February 26

Goals: Practice using d3 to create SVG elements and set their aesthetic properties. Recognize the effect of data transformations through direct data changes and through scale functions. Create data-generated line plots. Practice mapping.

1. In HW3 you recreated Fig. 2 from the Wickham reading using SVG elements. Now create the same plot again, but this time using d3 functions. First create x and y scale functions that map from data coordinates to SVG coordinates (10 pts). Add circles and rectangles, with positions given by the x and y scales. You don't need to use `data()` or `enter()` functions: it's fine if you do a separate command for each shape (10 pts). Add d3 axes, again using the x and y scale functions (10 pts). Now add an event listener that changes the color of a circles or rectangles to blue when it is clicked, using d3 selections (10 pts).

2. In this problem we're going to plot some data about English word frequencies from Google Books. The file `words.json` contains a JSON block that defines an array of objects. Each object represents a word, sorted by the number of pages that contain at least one instance of the word. The most frequent word, "of", occurs 15 billion times. The 512th most frequent word, "middle", occurs 45 million times. Add appropriate d3 axes for each figure.

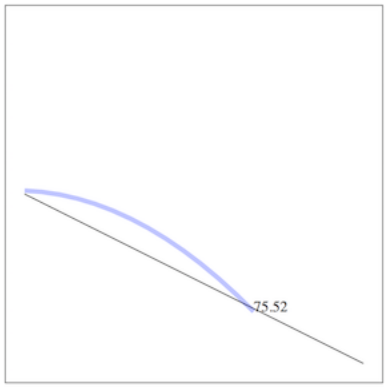
A. Load the data file using an asynchronous request with `d3.json`. Implement the rest of this problem in the callback function. Save the data array in a variable `wordData` that is defined outside the scope of the callback function. (5 pts)

B. Create a 200x200 pixel SVG element using d3 functions. Create two linear scale functions: an x scale for the "rank" and a y scale for the "count". Choose the "range" attributes to be appropriate for "rank" and "count". Use d3 to add text elements to the plot for each word in the data set. Use a loop or a "forEach" statement; you may not use a separate command for each word. Is this visualization useful? Why or why not? (5 pts)

C. In this section we'll transform the data as you create the text elements. Create a second 200x200 SVG element. Add the same points, but this time calculate the natural log of word's rank (use `Math.log()`), convert that value to a pixel value with a linear scale, and set "x" to that scaled log value. Similarly set "y" to the scaled log of the count. You will need to create new x and y scale functions using appropriate values for the "domain". How does this version differ from the previous version? (5 pts)

D. Now rather than transforming the data, let's change the scale functions. Create a third 200x200 SVG element, and create two **log** scale functions using the same "domain" values as in part A. See the d3js.org API documentation as necessary. Use d3 to add text elements to this new plot, again using the original values for "rank" and "count". (5 pts)

3. Line plots. In this problem you will simulate projectile motion under the influence of gravity using a finite approximation, where we estimate a ski jumper's position every 0.2 seconds. (This method was the original use for the ENIAC electronic computer.) Physics review: keep track of the position (displacement), velocity, and acceleration for the x and y dimensions separately. The finished work should look similar to the diagram below.



A. Contextualize the data. Usually we do this with axes and other guides. Here we will show the slope of a simplified ski-jump hill. Real hills have a long track (the inrun) where the jumper picks up speed and a curved surface in the landing area; we'll ignore the inrun altogether and treat the landing area as a smooth descent from 50m high at the take-off point, with a horizontal width of 100m. To make the calculations in the next sections easier, treat the top of the hill as (0,0), and the position of the ground to the right as (x, -0.5x). The take-off point will therefore be (0, 1), and 10 meters to the right of the take-off and three meters down will be (10, -2). Create an SVG 400 pixels high and 400 pixels wide. Create linear scales for the x and y dimensions that map meters relative to the top of the hill to pixels. You can choose how to define the range of pixel values, but keep the aspect ratio fixed so that one pixel up represents the same distance as one pixel to the right. (5 pts)

B. Generate a data array. Create a function `trajectory` that takes an `initialVelocity` in meters per second, an initial `angle`, and an `initialY` displacement (assume initial x displacement is 0), and returns an array of objects. Each object in this array should have seven variables: `ground`, `x`, `y`, `xVelocity`, `yVelocity`, `xAcceleration`, and `yAcceleration`. You will need to set the initial conditions (at array index 0) specially: set the initial x velocity to the initial velocity times the cosine of the initial angle ~~cosine of the initial velocity~~, use the sine for the y velocity. The acceleration in both dimensions will be constant, and represent the change in the skier's speed every fifth of a second. In the x dimension acceleration should be zero. Acceleration in the y dimension should be -9.8 / 5, to account for gravity. For each additional object, set velocity in the x and y dimensions equal to the previous velocity plus the current acceleration. The x and y positions should be equal to their previous values plus 0.2 × their current velocities. Finally, to see whether the skier has landed, calculate the height of the sloping hillside by setting the `ground` variable to the x position times -0.5. The array should comprise exactly as many elements as you need to hit the ground (i.e. y <= ground). (5 pts)

C. Display data. Create a function `plotTrajectory` that takes an array of the format created by the function in part A. This function should use `d3.line()` to create a 25% opacity 5-pixel-wide blue `<path>` element tracing this trajectory. Place a `<text>` element near the point of impact showing the distance from the take-off point when the jumper hits the ground (consult Pythagoras). (5 pts)

D. Display the trajectory for a ski jumper leaves the jump at 26 meters per second, -7 degrees angle and initial y displacement of 1m above the ground. Show two additional trajectories with velocities of your choice. (5 pts)

4. Map yourself! Find the longitude and latitude coordinates of three places that have meaning to you. Two must be within 30 miles of each other, the third must be at least 1000 miles away. Use d3 to create a map of the world, the US, or any relevant continent or region (10 pts). Use the JSON geographic files included with the class notes on GitHub or find your own. Select a projection for the map. Consult the d3 documentation for options. If you choose, you may want to use one of the projections from the d3-geo-projection package, which will require an additional javascript library file, available at https://github.com/d3/d3-geo-projection/. Place colored circles on the map in the locations you selected. Add text labels describing the meaning of these places. For each location connect the circle to the text with a line. (10 pts)