CS/INFO 3300; INFO 5100
Homework 3
Due 11:59pm Monday February 18

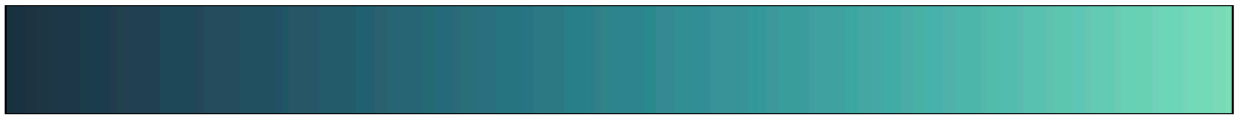
Goals: Practice using d3 to create SVG elements and set their aesthetic properties. Recognize the effect of data transformations through direct data changes and through scale functions. Practice working with color scales.

Your work should be in the form of an HTML file called index.html with one <p> element per problem. Wrap any SVG code for each problem in a <svg> element following the <p> element. 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.v5.min.js"></script>**

Create a zip archive containing your HTML file plus associated data files (such as diamonds.json) and upload it to CMS before the deadline.

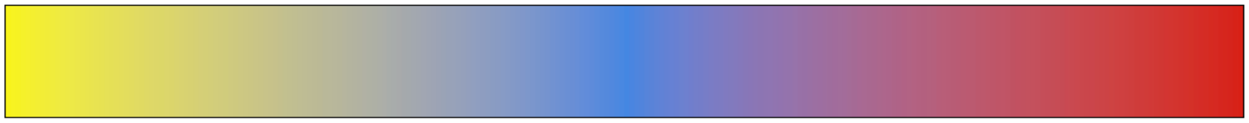
1. Instead of a <p> element, for this question please create a <ul> element. For each of the following scales, create a <li> sub-element and answer the following questions (5pts each):

A:



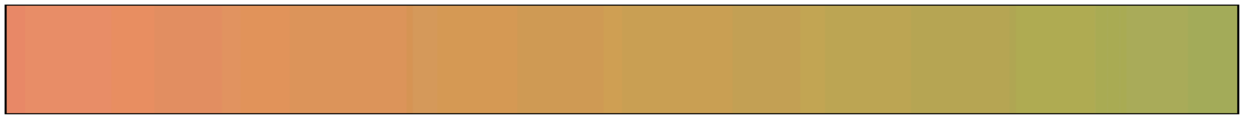
Is this a **sequential** or a **divergent** scale?
Do you think this an **effective color scale**? Justify your answer in **1-2 sentences**.

B:



Assume that this scale is being used as a scale for a **numeric data attribute**, with negative values moving towards yellow and positive values moving towards red. Middle values remain blue. Is this an **effective color scale for this task**? Justify your answer in **1-2 sentences**.

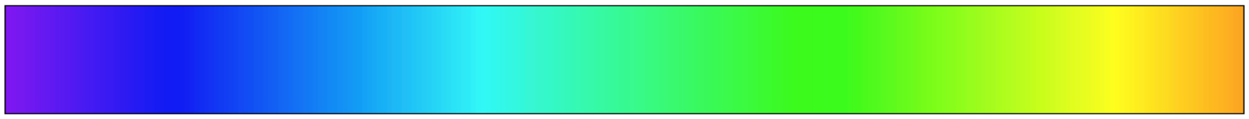
C:



Besides being rather ugly, this color scale poses serious **usability issues** for individuals with color vision deficiencies. Use an online color blindness image testing tool to identify and list **which color vision deficiencies** would render this scale hard to interpret (file included in ZIP).

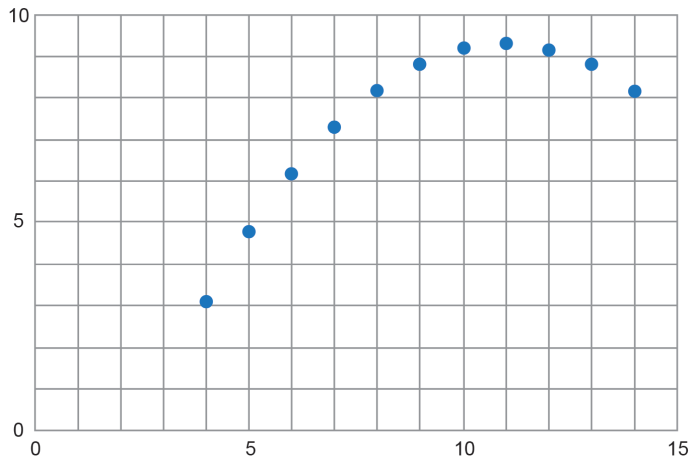
(next page)

D:



A data scientist is designing a choropleth map for a new **continuous, numeric county-by-county income data attribute** they developed. **Would you recommend that they use this (admittedly rather fetching) rainbow scale?** Justify your answer in 1-2 sentences.

2. In HW2 you recreated Anscombe's Quartet #2 from scratch in SVG. 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, using the same minimum and maximum values as the chart (10 pts). Create a variable containing the dataset (helpfully included in list form on the last page of the assignment). Add **d3 axes**, again using the x and y scale functions, with lines for every integer and labels for every 5 lines (hint: use the .tick() call on d3 axes) (5 pts). We know the default d3 axes will not look exactly like the image below. Using the dataset, add **circles** with positions provided by the scales. You don't need to use data() or enter() functions: it's fine if you just create circles one-by-one (5 pts). Finally add an **event listener** that changes the color of a circle to red when the circle is clicked (10 pts).



3. In this problem we're going to plot some data about diamond sales. The file **diamonds.json** contains a JSON block that defines an array of objects. Each object represents a particular diamond sold in a marketplace. These have been randomly sampled from a much larger dataset. In addition to numeric columns for price and size (carats), the dataset contains a color rating value where 1 is the best value and additional measures of quality.

A. Load the data file using an asynchronous request with **d3.json**. Implement the rest of this problem in the promise function. Save the data array in a variable diamondData that is defined **outside** the scope of the callback function. Make sure to handle any **error conditions** by appropriately using catch to throw a new Error object. (10 pts)

B. Create a 400x400px SVG element using d3 functions. Create two linear scale functions: an x scale for "carat" and a y scale for the "price". Make the domain start at 0 and end at the maximum value for each respective attribute. Choose the "range" attributes to be appropriate for the size of your plot. The plot can use the entire SVG canvas, but reserve 5 pixels at the top and right as **padding** using your range to make space for the maximum points so they do not clip at the edges. Remember to consider for the way SVG coordinates handle Y when using your y scale; we don't want any upside-down charts. Using any kind of loop you prefer, use d3 and the scales you build to construct <line> element gridlines, **including 0**, for **every integer** for the x axis and **every $1,000** for the y axis on the SVG canvas. Do not create any text labels. Style lines however you want. (10 pts)

C. Using a forEach loop in Javascript, create a **3px radius circle** for each point in the dataset, located at its proper place on the chart. Use your scales to place the points and employ Math.floor() to make sure you place each circle on an **integer pixel location**. Fill each point in a **dark blue** color of your choice. Set the **opacity** SVG attribute of each circle to 0.4 to make them translucent and show point density. (10 pts)

D. Now let's look at the color rating of each gem. Create a **second** 400x400px SVG element using d3 functions. Instead of charting "carat" on the x axis, chart the "color_rating" variable using an appropriate scale (hint: modify a copy of step C rather than starting from scratch). Recall that color_rating=1 is the *best* value. While color_rating ranges from 1 to 7, set the domain of your new x scale to [0,8] so that you have more space for columns of points. Create vertical gridlines for integers in range [1,7]. You will not need any padding on this axis since you reserved space in the domain. As in part B, create horizontal gridlines for every $1000 on the y axis. Finally, create a **3px radius circle** for each point in the dataset. To locate the points, use your scales and **add a random 5px jitter to each point location** (i.e. (Math.random()*10) - 5). Make sure to Math.floor() that result so that you place each circle on an **integer pixel location**. Fill each point in a **dark green** color of your choice. Set the **opacity** SVG attribute of each circle to 0.3 to make them translucent. (10 pts)

E. There's not much of a trend to see in this chart? This is because the new plot combines points of all sorts of different carats together. A low quality gem might be priced the same as a rank 1 gem if it is absolutely gigantic. Create a **third** 400x400px SVG element using d3 functions. Instead of charting "price" on the y axis, create **a new measure that shows the price per carat** of each point (hint: modify a copy of step D rather than starting from scratch). You do not need to record this new measure into the data; rather, just build it on the fly as you construct scales, gridlines, and circles. Create a new y scale to account for this new measure, and draw gridlines every $1000 per carat, starting at 0. Create a **3px radius circle** for each point in the dataset. To locate the points, use your scales and **add a random 5px jitter to each point location** (i.e. (Math.random()*10) - 5). Make sure to Math.floor() that result so that you place each circle on an **integer pixel location**. Fill each point in a **dark red** color of your choice. Set the **opacity** SVG attribute of each circle to 0.3 to make them translucent. This view ought to **suggest a relationship** between price per carat and quality. Unfortunately, statistical testing shows no such relation. In the <p> tag of your submission, briefly discuss **why this chart might mislead a viewer into seeing a pattern** that isn't present. (10 pts)

Anscombe’s Quartet -- #2

x	y
10	9.14
8	8.14
13	8.74
9	8.77
11	9.26
14	8.1
6	6.13
4	3.1
12	9.13
7	7.26
5	4.74

For easier import into your code:

[[10, 9.14], [8, 8.14], [13, 8.74], [9, 8.77], [11, 9.26], [14, 8.1], [6, 6.13], [4, 3.1], [12, 9.13], [7, 7.26], [5, 4.74]]