

CS/INFO 3300; INFO 5100
Homework 6
Due 11:59pm **Tuesday** March 26

Goals: Practice creating widgets and other interactive elements using d3. Get a bit more experience working with color in d3. Explore an example integrating a canvas and image.

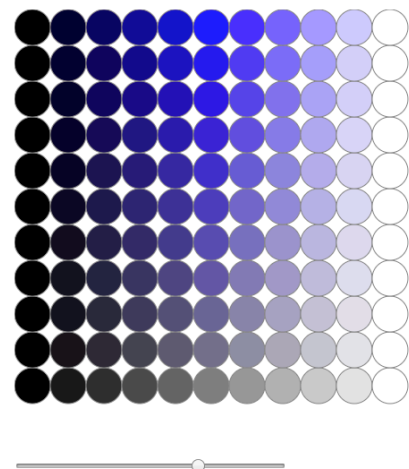
Your work should be in the form of an HTML file called index.html with one `<p>` element per (sub)problem. Wrap any SVG code for each problem in a `<svg>` element following the `<p>` element(s). For this homework we will be using d3.js. **In the `<head>` section of your file, please import d3 using this tag:**

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

Create a zip archive containing your HTML file plus **ALL associated data files** and upload it to CMS before the deadline. You will be penalized for missing data files or non-functional code.

1. For this problem, we will be making use of a color space available in d3, HSL. Unlike traditional HSV (hue, saturation, value), HSL (hue, saturation, lightness) acts more like a perceptual scale that accurately accounts for how individuals perceive color luminosity. You will design an interactive tool for exploring different colors.

A. In a `<p>` tag, place a **square SVG element** 500px in height and width. In a `<script>` tag, First write code which creates a single array containing Objects. Each object should have a **"saturation"** property (i.e. key/value pair) ranging in value from **0 to 100** and a **"lightness"** variable also **ranging from 0 to 100**, evenly spaced in **multiples of 10 (including 0 and 100)**. Every combination of saturation and lightness ought to be represented in the array, which will give you a total of **121 objects** in your array.



B. Create a function, **showCircles(hue)** that uses the d3 **"data join"** (i.e. `selectAll()`, `data()`, `enter()`, `attr()`, and `style()` functions) to create or modify one circle for each object in the list. Set the **radius of each circle to 20** and give each circle a **1px grey stroke**. Set the location of each circle to **create a grid based on the associated values: lightness for x, saturation for y, with centers 40px apart** so that the circles barely touch each other. Set the fill of each circle to an HSL color specified by the circle's lightness and saturation, and the hue value passed to the function. You may want to use `d3.hsl()` to create the color. **Check out the d3-color documentation** (hint: be careful of the range of values d3 expects for hsl's parameters).

C. Add a **slider input** so the user can choose a **hue**. This slider should range in value from **0 to 360 with a step size of 1**. Use d3 to attach an event listener functions to the **"input"** event for the slider to call your **showCircles** function with the **current hue value of the slider**.

2. For this problem you will make use of a **template provided with the assignment PDF**. Open `hw6_template.htm` in a plain text editor. **Copy the entire file's contents into your submission file after Problem 1**. It contains a `<p>` tag for the problem, some `<svg>` elements to use when you code, and some helper functions already nested in a `<script>` tag. **All of your changes to this template should occur in the marked section in the middle of the `<script>` tag**. Make sure you include `nighthawks.png` in your final submitted ZIP file.

The goal of this problem is to create an interactive tool that allows you to find the average color of a region of an image (in this case the painting, *Nighthawks*, by Hopper). The template includes a helper function that gives you a 2-d array of the RGB values for the pixels in the image. The script will automatically load the image and then run the `imgLoaded` function.

A. Create a **d3 2-dimensional brush that extends over the entire 800x437px region** of the `#container` `<svg>` element. Create an empty function, `"brushed"`, which will trigger on the `"brush"` and `"end"` events of your brush. Using d3's `.call()` function, install your brush into the `#brush` `<g>` element inside of this `<svg>`. Verify that your brush works before you proceed using `"console.log(d3.event.selection);"`

B. Inside of your `"brushed"` function, **compute the average color of all of the pixels contained within that brush using the LAB color space**. We use LAB rather than RGB because it will produce a more useful average color. First, use `d3.event.selection` and some **C-style for loops** to go through **all of the pixels that lie within the brushed region** (hint: brush selections may not start on integers, yet array indices **must** be integers to work). For each individual pixel, use `d3.rgb` and `d3.lab` to **convert the RGB pixel values to LAB color space values**. As you go through the pixels, **keep a running total of L, A, and B values** of pixels so that you can later compute averages. Once you have finished looping through the pixels, **create a variable "averageColor"** which contains a `d3.lab` color object for the average color of all of the pixels in the brushed region (hint: divide your running totals by the total number of pixels). Using `.attr`, **assign this new color to the "fill" attribute of the `#color` element in the template**.

C. The template also contains a `#label` `<text>` element. Use this element to provide printed output on the average color. Using `d3.lab`'s `hex` function, **change the text of the `#label` tag to show the hexadecimal web color value for the average color you found** (e.g. `#1A3933`). The black text color might be hard to read if the average color is darker. So, also **adjust the "fill" attribute of this `<text>` element** so that it is `"white"` if the L value of the average color is **below 50** and `"black"` if it is **above 50**.

