Lab 2

Student Name

YYYY-MM-DD

The purpose of this lab is to use color to your advantage. You will be asked to use a variety of color palettes, and use color for its three main purposes: (a) distinguish groups from each other, (b) represent data values, and (c) highlight particular data points.

Data

We'll be working with the honey production data from #tidytuesday. The #tidytuesday repo contains the full data, but we'll work with just the cleaned up version, using the **honeyproduction.csv** file, which is posted on the website.

The data is in under Dataset tab of Week 4 module on Canvas.

You can import the dataset using the code below.

```
d <- read.csv("./honeyproduction.csv",header=TRUE)
```

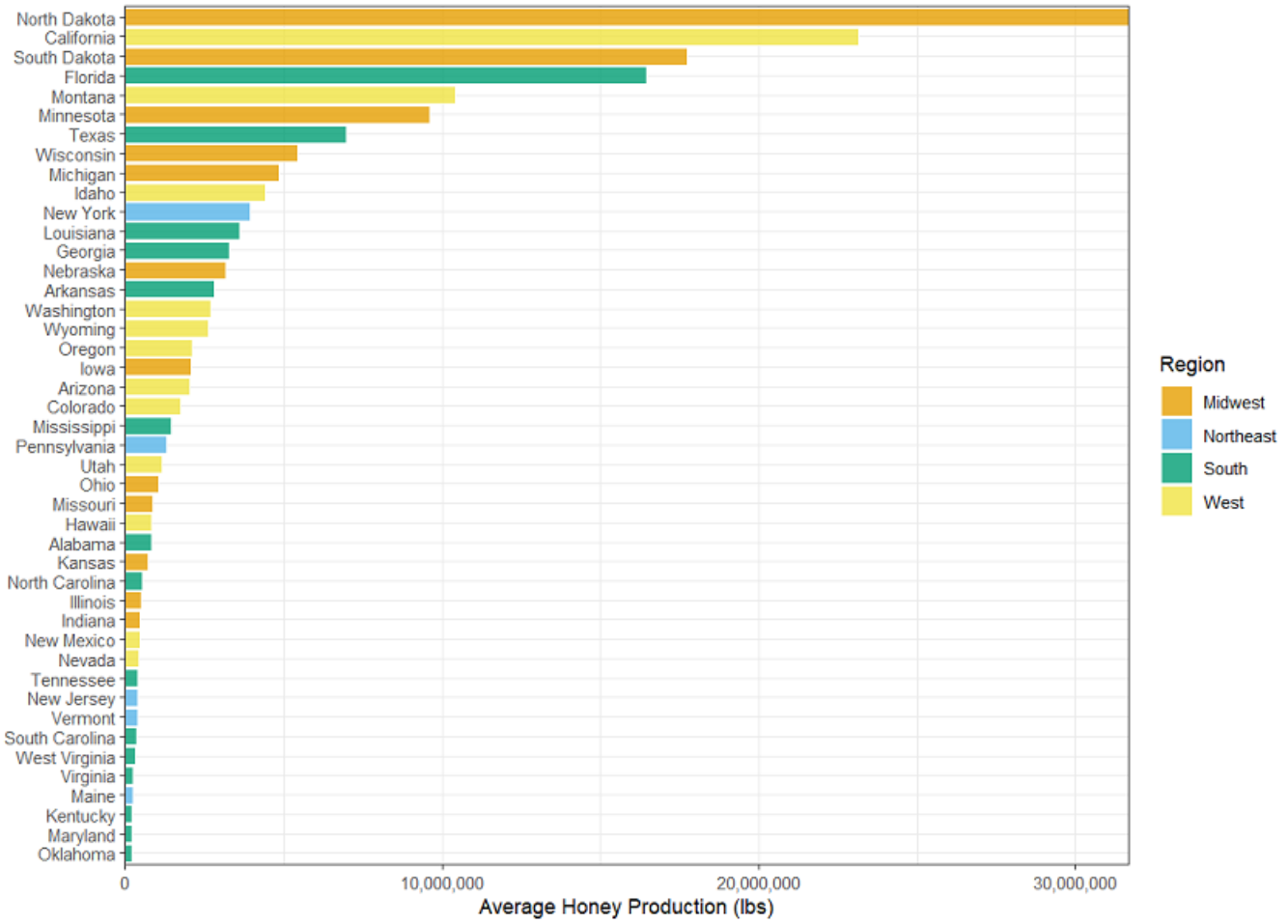
1. Visualize the total production of honey (**totalprod**) across years (**year**) by state (**state**). Use color to highlight the west coast (Washington, Oregon, and California) with a different color used for each west coast state.

- **Hint 1**: I'm not asking for a specific kind of plot, just one that does the preceding. But if you're trying to visualize change over time, a bar chart is likely not going to be the best choice.
- **Hint 2**: To get each state to be a different color you should either map state to color (for your layer that adds the west coast colors) or use the gghighlight package.

2. Reproduce the plot according three different kinds of color blindness using the `cvd_grid` package from the `colorblindr` package.
3. Reproduce the plot using a color blind safe palette of your choice.
4. Download the file **'us census bureau regions and divisions.csv'** from the course website denoting the region and division of each state.

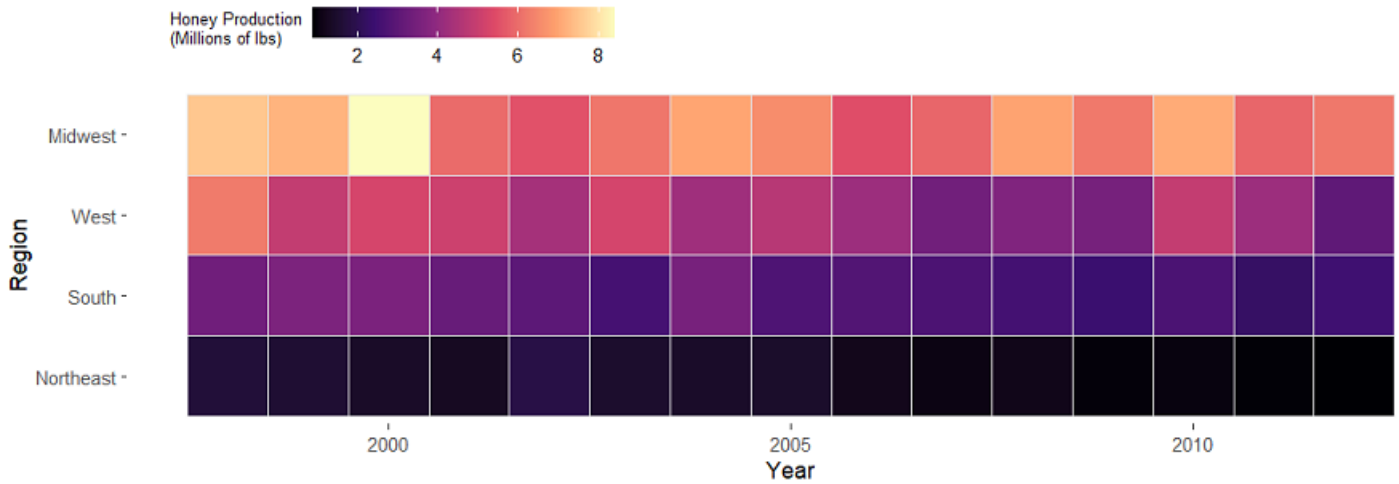
- Join the file with your honey file.
- Produce a bar plot displaying the average honey for each state (collapsing across years).
- Use color to highlight the region of the country the state is from.
- Note patterns you notice.

The plot should look like similar to the following plot.



5. Create a heatmap displaying the average honey production across years by *region* (averaging across states within region).

The plot should look like similar to the following plot.



6. Create at least one more plot of your choosing using color to distinguish, represent data values, or highlight. If you are interested in producing maps, I suggest grabbing a simple features data frame of the US using the Albers projection by doing the following:

```
remotes::install_github("hrbrmstr/albersusa")
library(albersusa)
us <- usa_sf()
```

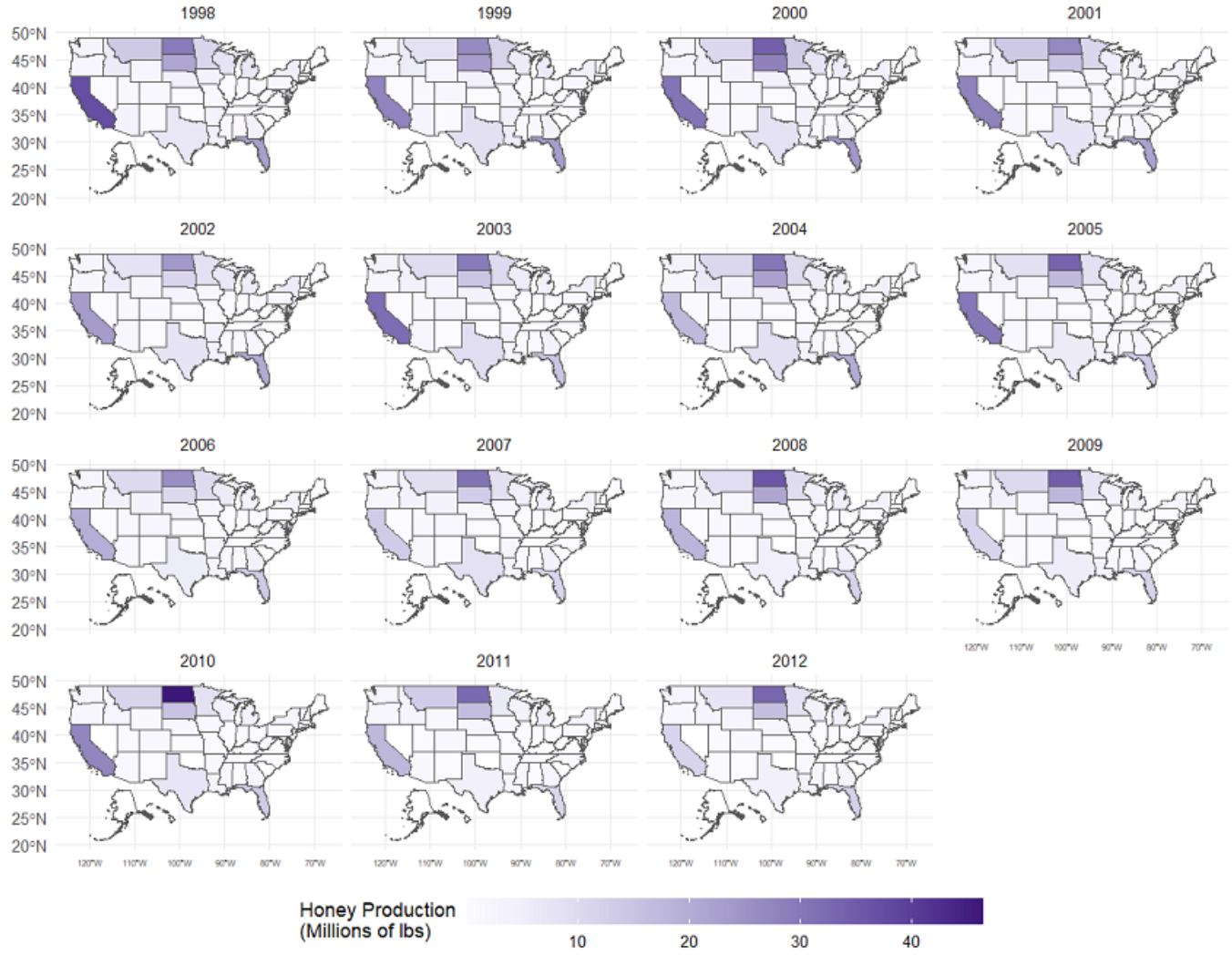
You can then join your honey data with this dataset. We'll talk about geographic data more later in the course, but it is pretty easy to work with. For example, you could use the data frame above to create a map of the US with:

```
ggplot(us) +
  geom_sf()
```

You will likely get a few warnings but they are most likely ignorable. You could also theme it more from here, but if you join it with your honey data, you should be able to `fill` and/or `facet` by specific variables.

Note - this is a little trickier than it initially seems because you can "lose" states in your map if they don't have any data (there are only 44 states represented in your honey dataset). You should still plot all states, but just have them not be filled according to your fill scale if there is no data.

For instance, below is a plot created based on this data for inspiration.




```
honey_subset <- honey %>%
  select("iso_3166_2" = "state", year, totalprod)

# Missing some years in the honey dataset, so create them here

full_set <- expand.grid(iso_3166_2 = unique(us$iso_3166_2),
                        year = 1998:2012)

# Add them into the honey dataset as missing values
honey_subset <- left_join(full_set, honey_subset)

# Now join it to the USA dataset
honey_geo <- left_join(us, honey_subset)

ggplot(honey_geo) +
  geom_sf(aes(fill = totalprod/1e6)) +
  facet_wrap(~year) +
  colorspace::scale_fill_continuous_sequential(
    palette = "Purples",
```

```
  na.value = "white",
  name = "Honey Production\n(Millions of lbs)\n"
) +
theme_minimal() +
theme(legend.direction = "horizontal",
      legend.position = "bottom",
      legend.key.size = unit(2, 'cm'),
      legend.key.height = unit(.5,"cm"),
      axis.text.x = element_text(size = 5))
```