Prep1

Sebastian Montesinos

Due by midnight, Monday Feb. 14

Reminder: Prep assignments are to be completed individually.

Reading

The associated reading for the week is Chapter 2, Chapter 3, and Section 8.2. This reading explores key aspects of visualizations, how to build them, and ethical issues around visualizations.

In addition to reading, I recommend you code along with the book examples. You'll see some of the problems below use data from the text. You can try out the code yourself - just be sure to load the mdsr package and any other packages referenced. You can get the code in R script files (basically, files of just R code, not like a .Rmd) from the book website.

Git Workflow Review

1. Before editing this file, verify you are working on the copy saved in *your* repo for the course (check the filepath and the project name in the top right corner).
2. Before editing this file, make an initial commit of the file to your repo to add your copy of the assignment.
3. Change your name at the top of the file and get started!
4. You should *save, knit, and commit* the .Rmd file each time you've finished a question, if not more often.
5. You should occasionally *push* the updated version of the .Rmd file back onto GitHub. When you are ready to push, you can click on the Git pane and then click **Push**. You can also do this after each commit in RStudio by clicking **Push** in the top right of the *Commit* pop-up window.
6. When you think you are done with the assignment, save the pdf as "*YourFirstInitialYourLastName_thisfilename*.pdf" before committing and pushing (this is generally good practice but also helps me in those times where I need to download all student homework files). For example, I would save this file as AWagaman_Prep1.pdf.

1 - Some basics

Chapter 2 describes Yao's taxonomy for graphics (which is very similar to what ggplot2 uses). The four basic elements are visual cues, coordinate systems, scale, and context.

part a

Solution: Two of the best visual cues are position and length.

part b

Solution: The polar coordinate system, a radial analog with a point identified by radius and angle, is a non-cartesian coordinate system.

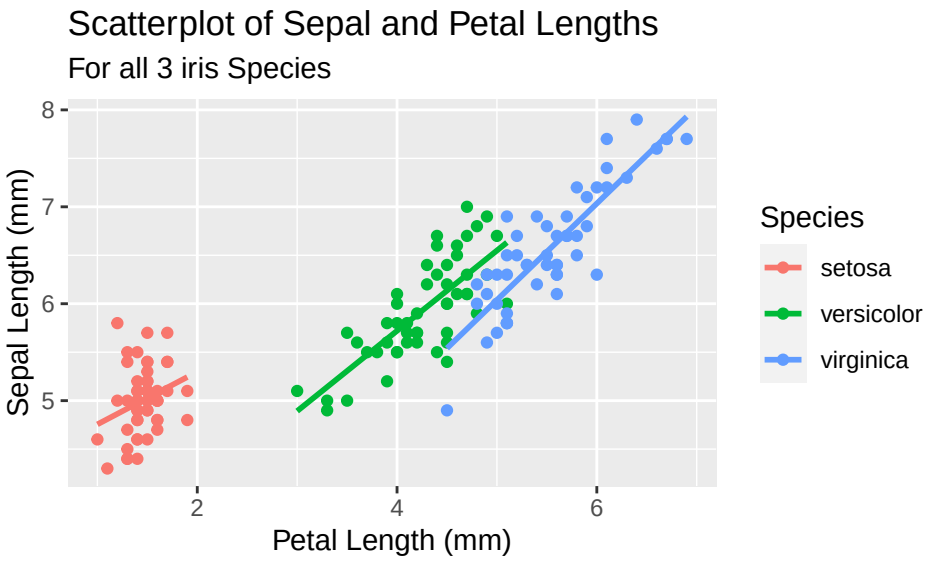
part c - Name a numeric scale other than the usual linear scale.

Solution: A logarithmic scale.

part d - Consider the following plot made based on the iris data. What aspects of the plot contribute to its context?

```
data(iris)
ggplot(iris, aes(x = Petal.Length, y = Sepal.Length, color = Species))+
  geom_point()+
  geom_smooth(method = 'lm', se = FALSE)+
  labs(y = "Sepal Length (mm)",
       x = "Petal Length (mm)",
       title = "Scatterplot of Sepal and Petal Lengths",
       subtitle = "For all 3 iris Species")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Solution: The axis labels that specify sepal length and petal length both contribute to context. The title and subtitle both also contribute since they let the viewer know basic information about the graph. Finally, the key on the right indicating what species is associated with what color contributes to context.

2 - GDP and education

part a - Figure 3.3 in Section 3.1.1 shows a scatterplot that uses both location and label as aesthetics. Reproduce this figure.

Hint: you'll need to define 'g' based on code from earlier in Section 3.1.1. Also, make sure you load the packages in the `setup` *chunk!*

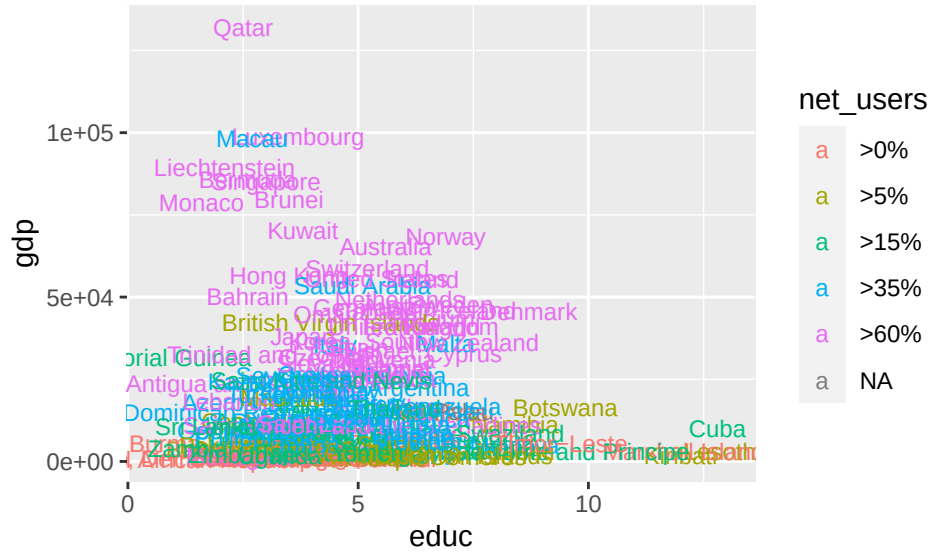
Solution:

```
data(CIACountries)

# define the plot object
g <- ggplot(data = CIACountries, aes(y = gdp, x = educ))

# print the plot
g + geom_text(aes(label = country, color = net_users), size = 3)
```

```
## Warning: Removed 64 rows containing missing values (geom_text).
```



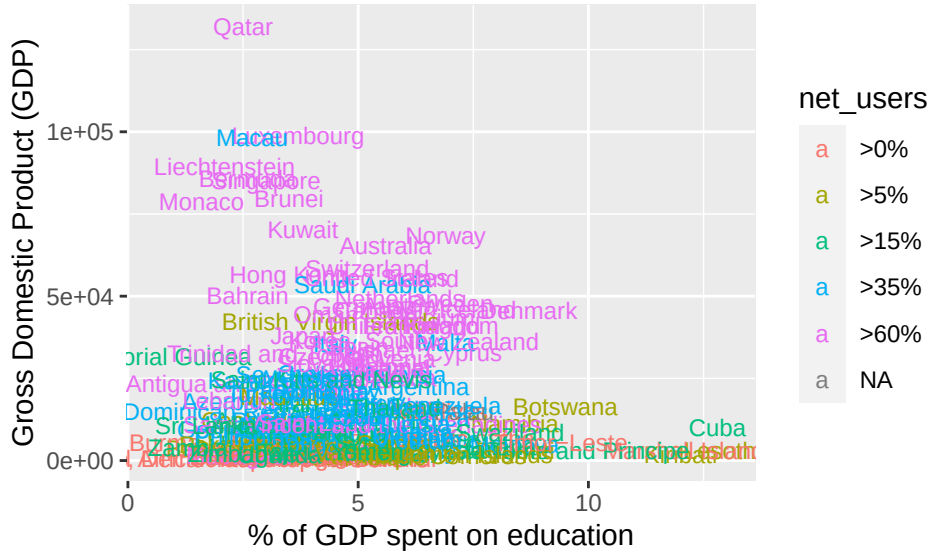
part b - Now, update the plot with more informative labels. Label the x-axis "% of GDP spent on education" and the y-axis "Gross Domestic Product (GDP)".

Hint: see Section 3.2.1 for an example of one way to label the axes.

Solution:

```
g +
  geom_text(aes(label = country, color = net_users), size = 3) +
  labs(x = "% of GDP spent on education", y = "Gross Domestic Product (GDP)")
```

```
## Warning: Removed 64 rows containing missing values (geom_text).
```



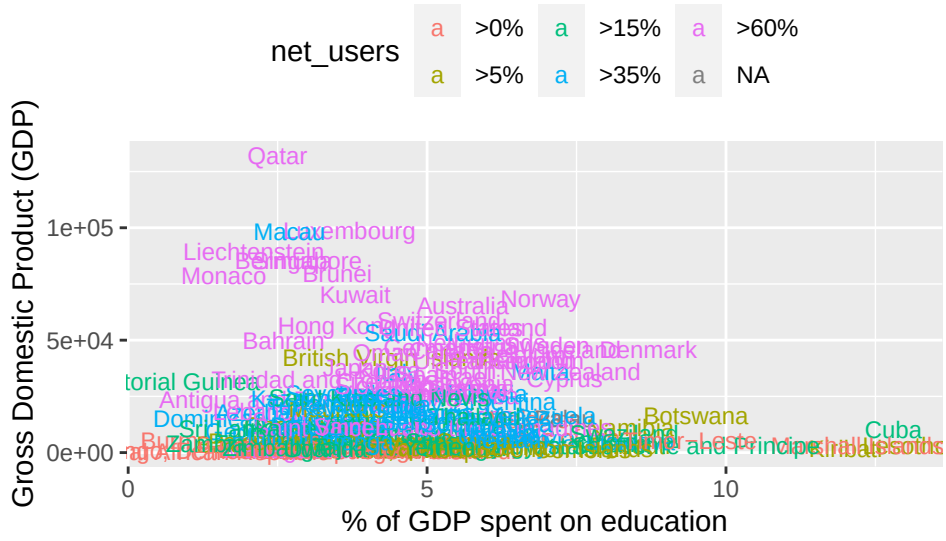
part c - Next, move the legend so that it's located on the top of the plot as opposed to the right of the plot.

Hint: see Section 3.1.4 for an example on how to change the legend position.

Solution:

```
g +
  geom_text(aes(label = country, color = net_users), size = 3) +
  labs(x = "% of GDP spent on education", y = "Gross Domestic Product (GDP)") +
  theme(legend.position = "top")
```

```
## Warning: Removed 64 rows containing missing values (geom_text).
```

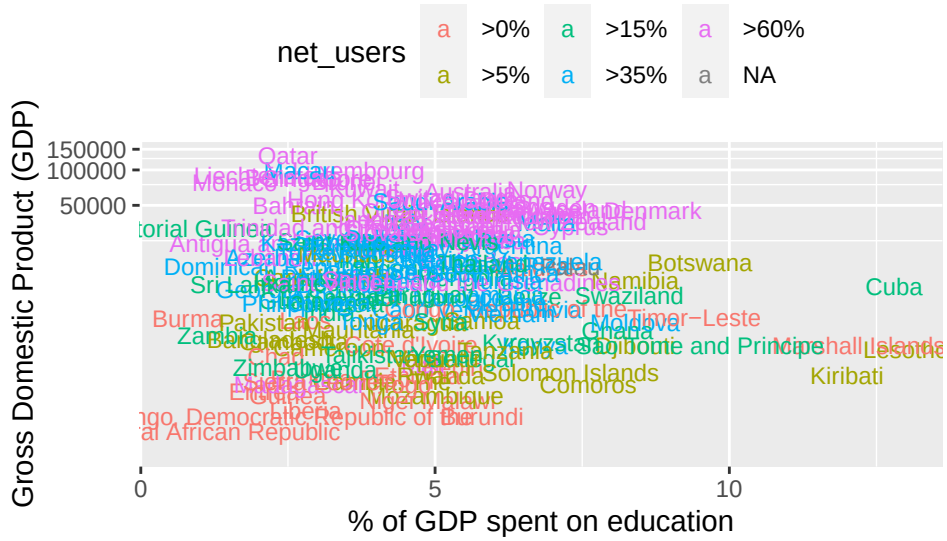


part d - Lastly, Section 3.1.2 discusses *scale*, and demonstrates how to display GDP on a logarithmic scale to better discern differences in GDP. Update the figure so GDP is on a log10 scale.

Solution:

```
g +
  geom_text(aes(label = country, color = net_users), size = 3) +
  labs(x = "% of GDP spent on education", y = "Gross Domestic Product (GDP)") +
  theme(legend.position = "top") +
  coord_trans(y = "log10")
```

```
## Warning: Removed 64 rows containing missing values (geom_text).
```



3 - Demos with Iris

The *iris* data set contains 5 variables on 3 species of iris. There are 4 measurement variables and the Species variable. We can look at a data set to get a sense of its structure with *glimpse*, *str* or *summary* - all provide a quick overview of the data set. We can look at the first few observations with *head*. To make plots, one must know what plots are appropriate for the variables involved.

Note: for this problem, you aren't actually making any of the plots.

> part a - Use one of data set overview commands to look at iris. List whether each variable is numeric or categorical (quantitative vs. qualitative).

Solution: The sepal length, sepal width, petal length, and petal width variables are all quantitative, while the species variable is categorical.

```
data(iris)
glimpse(iris)
```

```
## Rows: 150
## Columns: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

> part b - Suppose we want to examine the distribution of Petal.Width. What plot would you recommend for this?

Solution: Since this is a numeric variable, I would recommend a density plot that would allow us to get an idea of the distribution across the range of numeric quantities.

> part c - Suppose we want to examine the distribution of Species. What plot would you recommend for this?

Solution: Since this is a categorical variable, I would recommend a stacked bar chart. This would allow us to see the relative amount of data in each species category.

> part d - We want to examine the relationship between Petal.Width and Petal.Length across Species. However, our client insists the graphic be printed in black and white. What plot would you recommend for this? Be sure to name the plot and describe how all three variables are represented/included.

Solution: We could use a scatterplot with multiples, wherein each categorical variable corresponds to one plot. So, the specific species would be represented by the multiple one is looking at. On each graph, width and length would be represented on the y and x axis.

4 - Learning about R functions

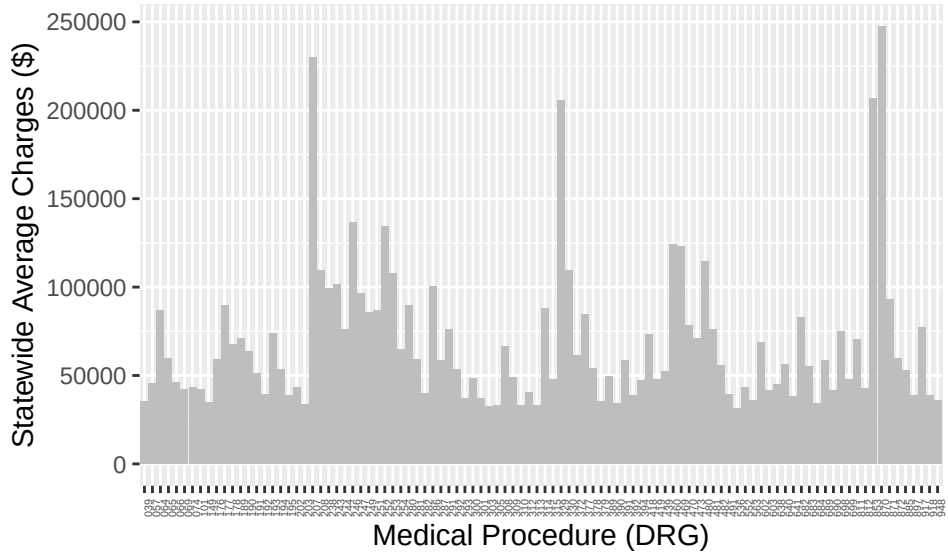
part a - Consider Figure 3.8 in Section 3.1.5. What does `reorder(drg, mean_charge)` do? To figure this out, recreate the plot, but use `x = drg` instead of `x = reorder(drg, mean_charge)`. What happens?

Solution: Based on the alternative graph, reorder seems to be taking the drg variable and ordering it in terms of the mean charge, so that the graph displays the data from smallest to largest. When you do not include the reorder function, the drg values are no longer ordered and so do not appear neatly from smallest to largest.

```
data(MedicareCharges)
ChargesNJ <- MedicareCharges %>%
  ungroup() %>%
  filter(stateProvider == "NJ")

# create the plot object
p <- ggplot(data = ChargesNJ,
  aes(x = drg, y = mean_charge)) +
  geom_col(fill = "gray") +
  ylab("Statewide Average Charges ($)") +
  xlab("Medical Procedure (DRG)") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = rel(0.5)))

# print the plot
p
```



part b - Another way to learn about a function is via the help menu. You can search directly (search window in the Help pane) or type ?functionname in the console. Try this with *reorder*. From the help results, report the package that *reorder* is from and the name of the data set used in the help menu example.

Hint: The package is in curly brackets after the name of the function in the help menu.

Solution: Reorder is in the stats package. In the help menu example, they use the 'InsectSprays' data set.

Note: Normally, you'd look through the help information to learn about the function, it's arguments, outputs, etc. This problem is designed so you learn about the help menu, not to really focus on this function. That said, you should know what *reorder* does at the end of the problem. You can also access R documentation via the internet, rather than through R.

> part c - Sometimes different packages have functions with shared names that don't do the same thing. We don't have a lot of packages loaded here, but we can demo this with *filter*. Try looking up *filter*. What two packages are listed as having results for a *filter* function?

Hint: You can also find this where the packages are loaded in above. If there are conflicts, the most recently loaded package's function masks the function from previously loaded packages. You should see a notice of masking here in regards to filter (and lag).

Solution: The stats package and dplyr package both use filter.

Note: If you are ever worried about a function conflict between packages, or want to use just one function (or data set) from a package without loading the entire rest of the package, you can reference the function this way - package::function. You will see this occasionally in our work this semester.