STAT 231: Problem Set 1A

Brandon Kwon

due by 5 PM on Monday, February 22

In order to most effectively digest the textbook chapter readings – and the new R commands each presents – series A homework assignments are designed to encourage you to read the textbook chapters actively and in line with the textbook's Pro Tip on page 33:

"**Pro Tip**: If you want to learn how to use a particular command, we highly recommend running the example code on your own"

A more thorough reading and light practice of the textbook chapter prior to class allows us to dive quicker and deeper into the topics and commands during class. Furthermore, learning a programming language is like learning any other language – practice, practice, practice is the key to fluency. By having two assignments each week, I hope to encourage practice throughout the week. A little coding each day will take you a long way!

Series A assignments are intended to be completed individually. While most of our work in this class will be collaborative, it is important each individual completes the active readings. The problems should be straightforward based on the textbook readings, but if you have any questions, feel free to ask me!

Steps to proceed:

1. In RStudio, go to File > Open Project, navigate to the folder with the course-content repo, select the course-content project (course-content.Rproj), and click "Open"
2. Pull the course-content repo (e.g. using the blue-ish down arrow in the Git tab in upper right window)
3. Copy ps1A.Rmd from the course repo to your repo (see page 6 of the GitHub Classroom Guide for Stat231 if needed)
4. Close the course-content repo project in RStudio
5. Open YOUR repo project in RStudio
6. In the ps1A.Rmd file in YOUR repo, replace "YOUR NAME HERE" with your name
7. Add in your responses, committing and pushing to YOUR repo in appropriate places along the way
8. Run "Knit PDF"
9. Upload the pdf to Gradescope. Don't forget to select which of your pages are associated with each problem. *You will not get credit for work on unassigned pages (e.g., if you only selected the first page but your solution spans two pages, you would lose points for any part on the second page that the grader can't see).*

1. GDP and education

a.

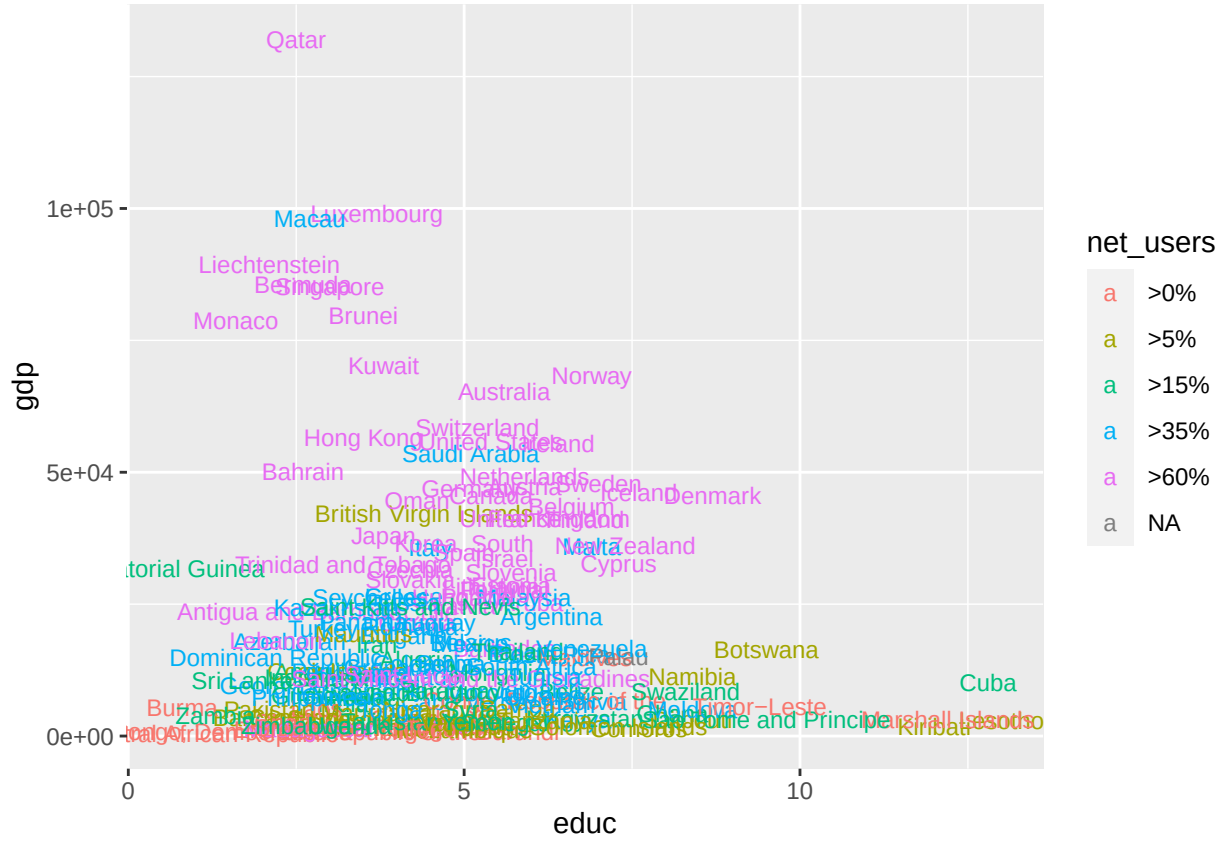
Figure 3.3 in Section 3.1.1 shows a scatterplot that uses both location and label as aesthetics. Reproduce this figure. Hint: you'll need to define 'g' based on code from earlier in Section 3.1.1.

```
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).
```



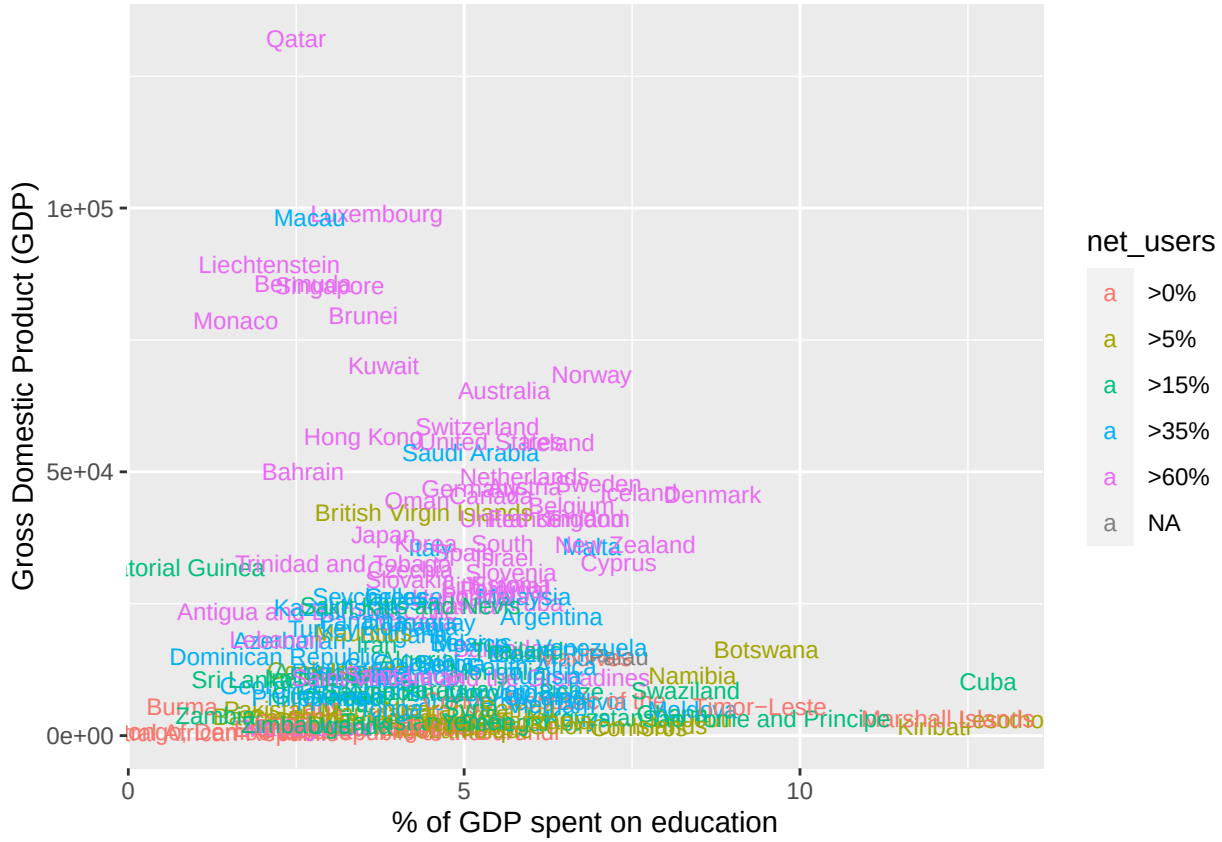
b.

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 3.2.2 for an example of one way to label the axes.

```
# 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) + xlab("% of GDP spent on education") +
```

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



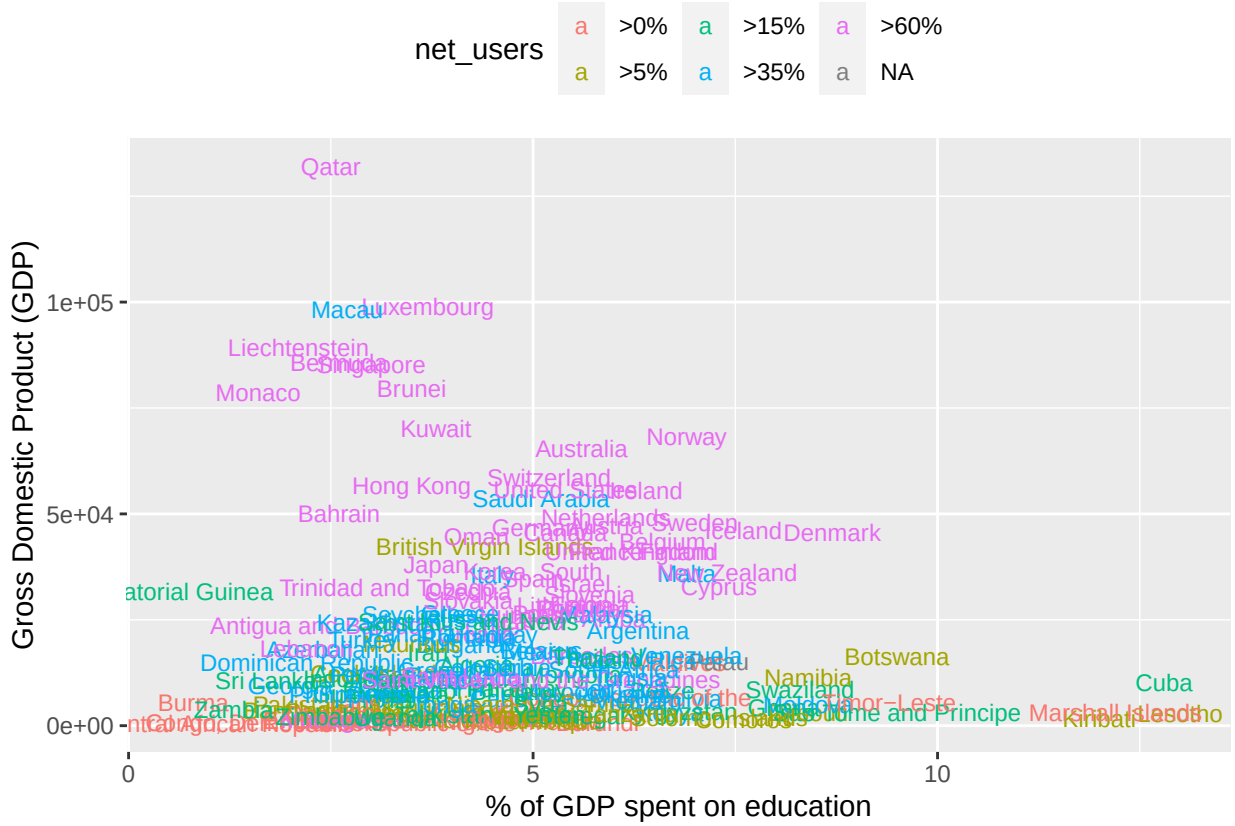
c.

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.

```
# 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) + xlab("% of GDP spent on education") +
```

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



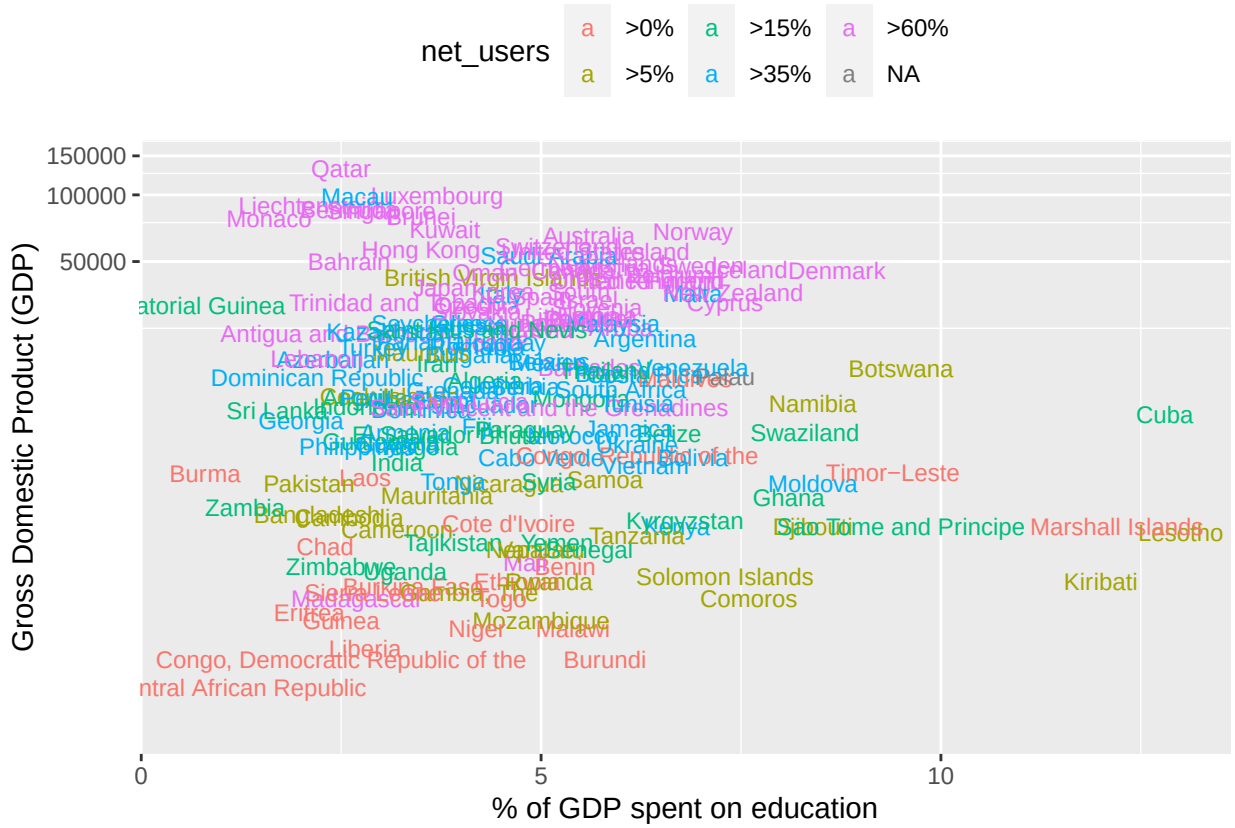
d.

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.

```
# 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) + xlab("% of GDP spent on education") +
```

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


net_users
a >0%
a >5%
a >15%
a >35%
a >60%
a NA
150000
100000
50000
Gross Domestic Product (GDP)
Qatar
Macau
Luxembourg
Monaco
Brunei
Kuwait
Australia
Norway
Hong Kong
Bahrain
Denmark
Cyprus
Malta
Argentina
Botswana
Namibia
Cuba
Swaziland
Timor−Leste
Moldova
Ghana
Marshall Islands
Lesotho
Kiribati
Solomon Islands
Comoros
Burma
Pakistan
Zambia
Cameroon
Chad
Zimbabwe
Uganda
Eritrea
Guinea
Liberia
Mozambique
Niger
Malawi
Burundi
Congo, Democratic Republic of the
0
5
10
% of GDP spent on education

2. Medical procedures

a.

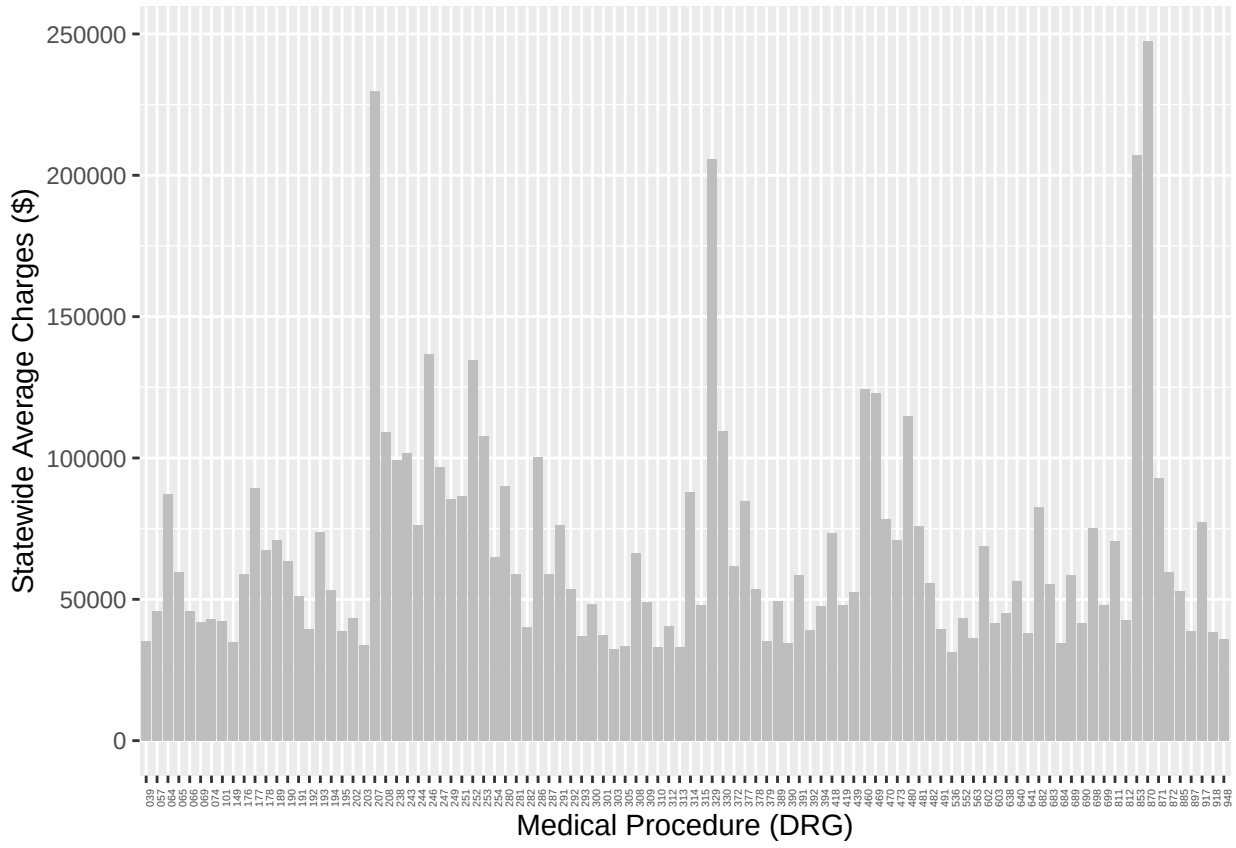
Consider Figure 3.7 in Section 3.2.1. What does `reorder(drg, mean_charge)` do? Recreate the plot, but use `x = drg` instead of `x = reorder(drg, mean_charge)`. What happens?

> ANSWER: The "reorder(drg, mean_charge)" function places the medical procedures (DRG) in a linear order that goes from the cheapest statewide average charges to the most expensive statewide average charges (aka in ascending order) for better aesthetic appeal and easier visual analysis. It also demonstrates a key appeal in terms of viewing data: direction. Due to this function, the viewer can now see the relationship between medical procedure and statewide average charges more clearly. When using "x = drg" instead of "x = reorder(drg, mean_charge)," the order of the medical procedures on the x-axis is based simply on how they were ordered in the original data set; as a result, in terms of direction, there is no direction but rather random/arbitrary peaks and dips in terms of the statewide average charges.

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

# create the plot object
p <- ggplot(data = ChargesNJ, aes(x = drg, y = mean_charge))


# print the plot
p + 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)))
```



b.

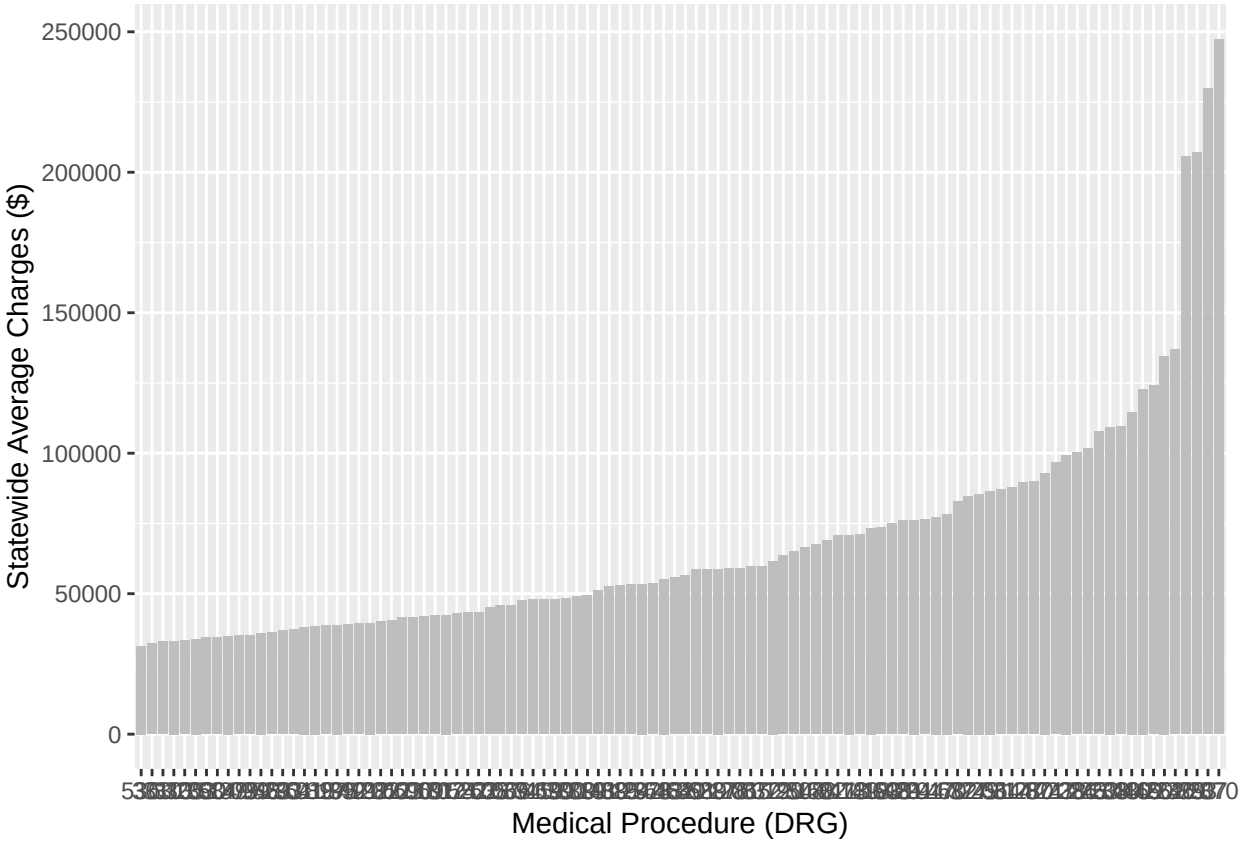
Replace `x = drg` with `x = reorder(drg, mean_charge)`, but also remove the `theme()` line. Now what happens? What was the purpose of the `theme()` line? Hint: You may need to knit the document and look at the pdf to better observe what's happening.

> ANSWER: When removing the "theme()" line, we see that that the x-axis is drastically different in terms of labelling. The "theme()" line essentially helps the x-axis become more "viewable" by reorganizing the labels in terms of angle and size. The "angle" of 90 suggests that each label is faced "vertically" to provide for more room between each label. The "hjust" allows for better centering, and the "size" suggests the font size of each label to allow for more readable labels.

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

# create the plot object
p <- ggplot(data = ChargesNJ, aes(x = reorder(drg, mean_charge), y = mean_charge))


# print the plot
p + geom_col(fill = "gray") +
  ylab("Statewide Average Charges ($)") +
  xlab("Medical Procedure (DRG)")
```


250000
200000
150000
100000
50000
0
Statewide Average Charges ($)
Medical Procedure (DRG)

3. Historical baby names

As you read through (and, better yet – code along with (not required, but useful practice!)) – the extended example on historical baby names in section 3.3.1, write down two questions you have about any of the R code used in that example. (Your questions could be about what a specific part of the code – ggplot or not – is actually doing, or a more general question about any of the commands used.) Please be thoughtful about your questions; we will use them (anonymously) in an exercise in class this week.

> ANSWER: 1) What is the exact function of "%>%"? 2) I understand that spacing/row filling when writing code is not really an issue when it comes to compiling. However, I was wondering if there is a specific type of spacing that you or any other data scientist prefers when writing code, especially when it comes to adding aesthetic aspects including fill = "#b2d7e9," color = "white," and size = 0.1?

```
# to get you started following along . . .
library(babynames)
BabynamesDist <- make_babynames_dist()

joseph <- BabynamesDist %>%
  filter(name == "Joseph" & sex == "M")

name_plot <- ggplot(data = joseph, aes(x = year)) +
  geom_bar(stat = "identity", aes(y = count_thousands*alive_prob)
           , fill = "#b2d7e9", color = "white")

name_plot
```