STAT 231: Problem Set 1A

Kevin Ma

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 lanugage 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).*

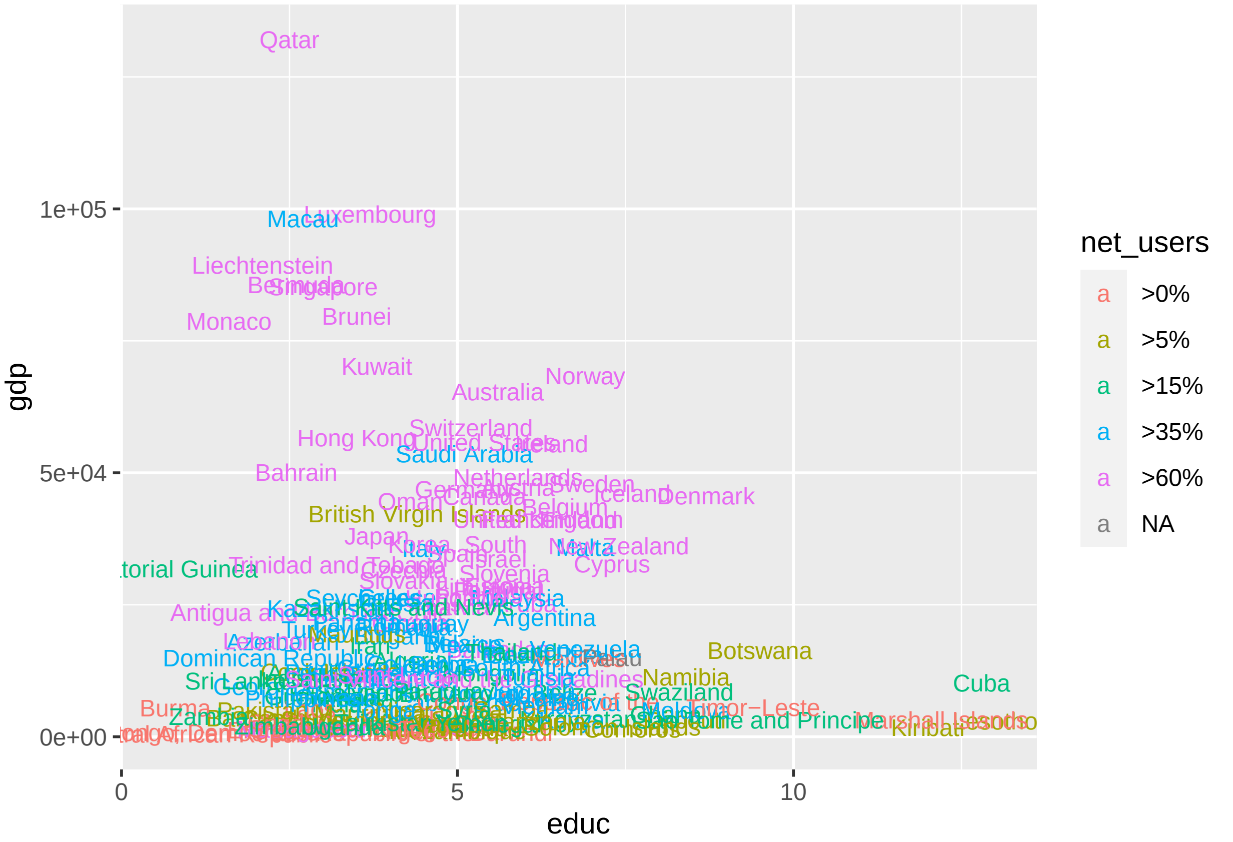
1. GDP and education

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))
g <- g + geom_text(aes(label = country, color = net_users), size = 3)

# print the plot
print(g)
```

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

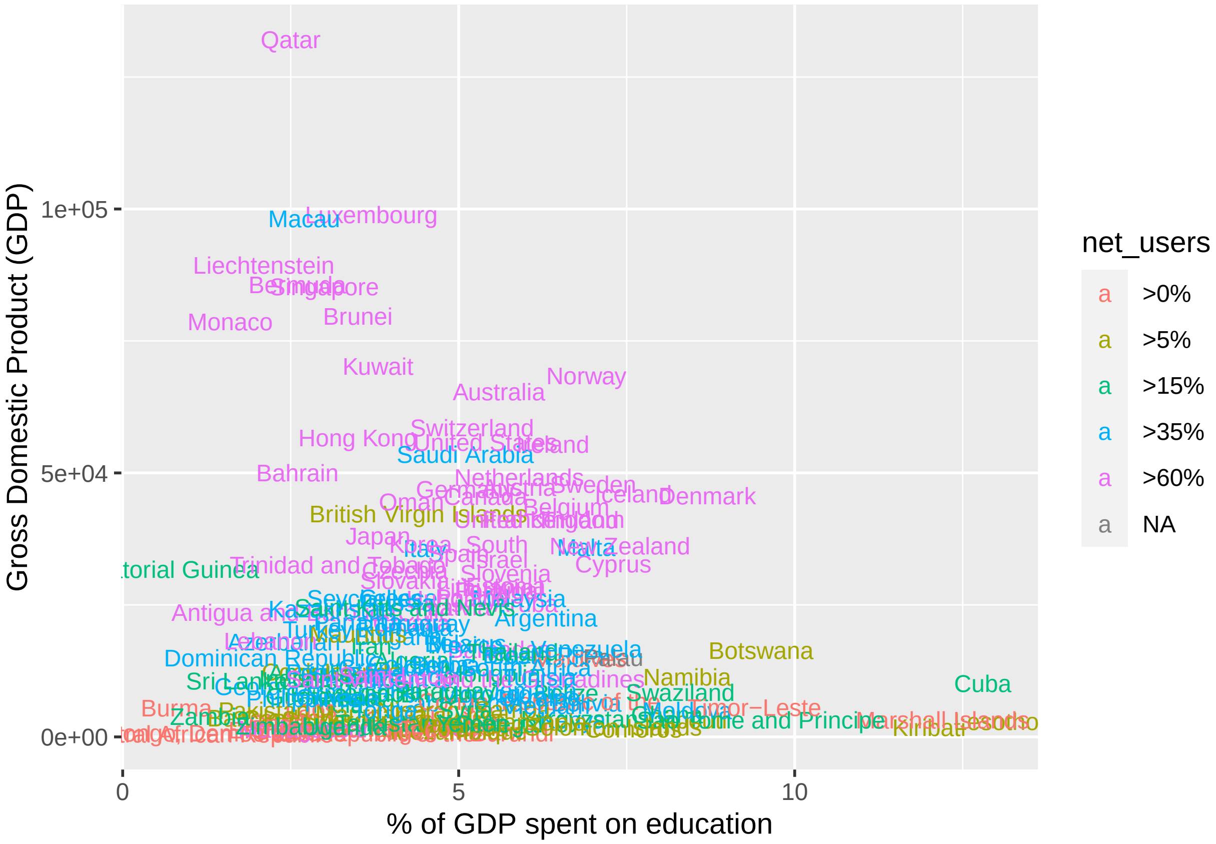


b.

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.

```
g <- g + xlab("% of GDP spent on education") + ylab("Gross Domestic Product (GDP)")
print(g)
```

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

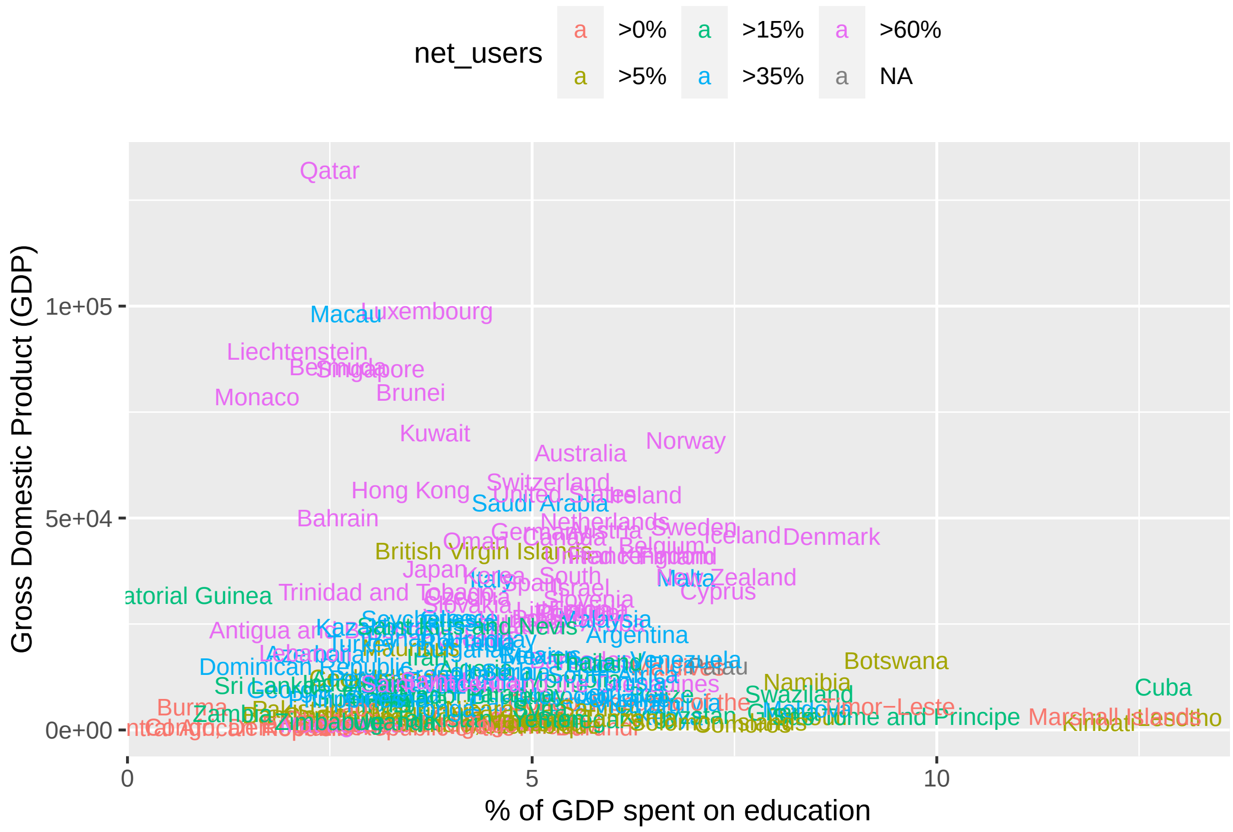


c.

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.

```
g <- g + theme(legend.position = "top")
print(g)
```

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

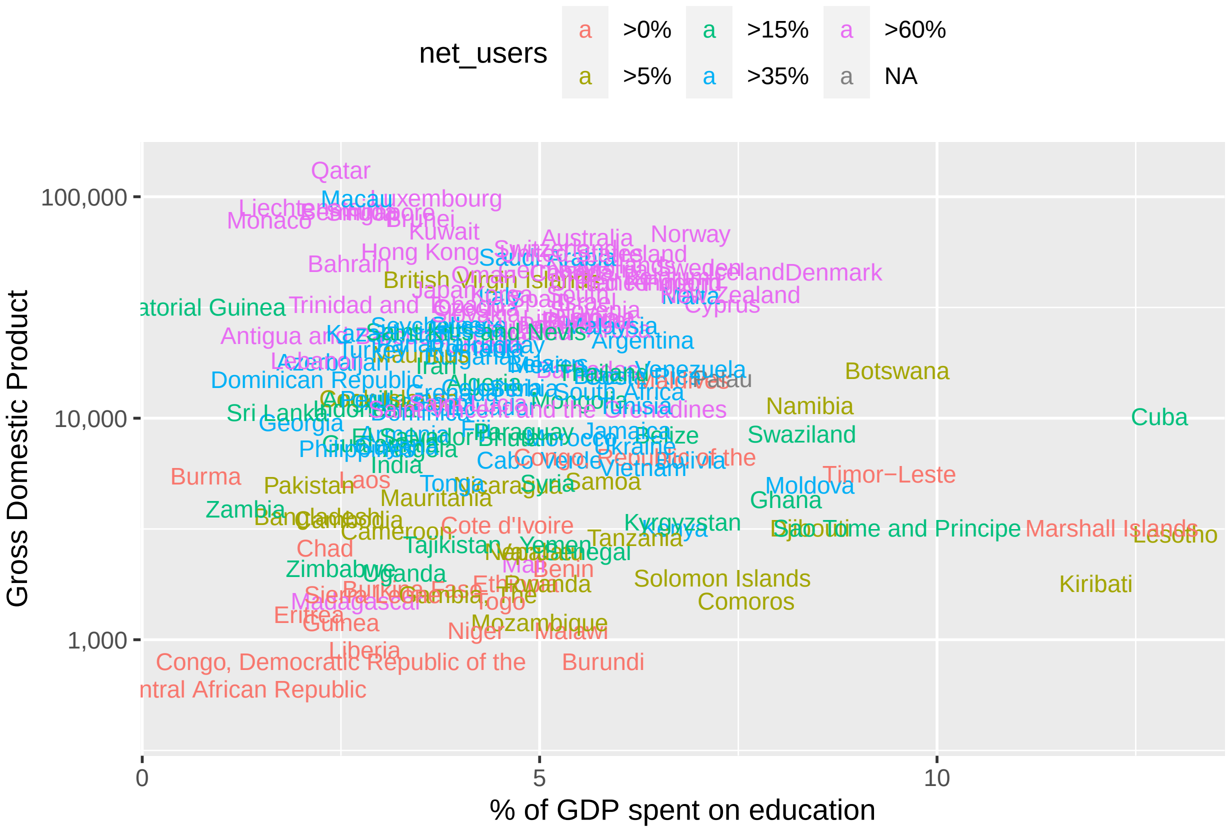


d.

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.

```
g <- g + scale_y_continuous(
    name = "Gross Domestic Product",
    trans = "log10",
    labels = scales::comma
  )
print(g)
```

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


net_users
a >0%
a >5%
a >15%
a >35%
a >60%
a NA
Gross Domestic Product
100,000
10,000
1,000
% of GDP spent on education
0
5
10
Qatar
Macau
Luxembourg
Monaco
Brunei
Kuwait
Australia
Norway
Hong Kong
Bahrain
Saudi Arabia
Denmark
British Virgin Islands
New Zealand
atorial Guinea
Trinidad and Tobago
Cyprus
Antigua and Barbuda
Argentina
Turkey
Lebanon
Azerbaijan
Iran
Venezuela
Botswana
Dominican Republic
Algeria
South Africa
Palau
Namibia
Sri Lanka
Georgia
Mongolia
Saint Vincent and the Grenadines
Cuba
Fiji
Paraguay
Jamaica
Belize
Swaziland
Armenia
El Salvador
Philippines
Ukraine
India
Cabo Verde
Congo, Republic of the
Bolivia
Vietnam
Burma
Pakistan
Laos
Tonga
Nicaragua
Syria
Samoa
Moldova
Timor−Leste
Ghana
Mauritania
Zambia
Bangladesh
Cambodia
Cote d'Ivoire
Kyrgyzstan
Kenya
Tanzania
Djibouti
Sao Tome and Principe
Marshall Islands
Lesotho
Cameroon
Chad
Tajikistan
Yemen
Nepal
Senegal
Zimbabwe
Uganda
Mali
Benin
Solomon Islands
Burkina Faso
Ethiopia
Rwanda
Kiribati
Sierra Leone
Gambia, The
Madagascar
Togo
Comoros
Eritrea
Guinea
Mozambique
Niger
Malawi
Liberia
Congo, Democratic Republic of the
Burundi
ntral African Republic

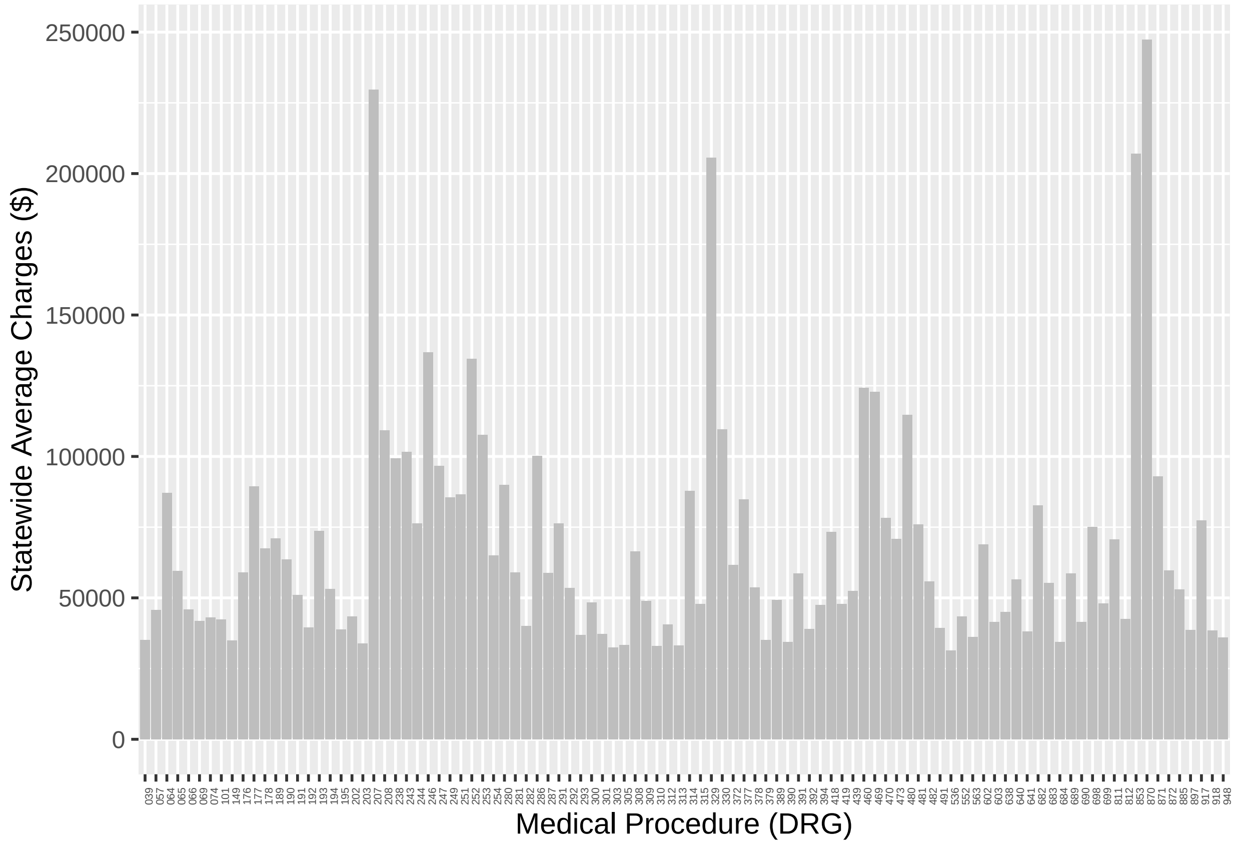
2. Medical procedures

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: ‘reorder(drg, mean_charge)’ sorts the drgs along the x axis by the mean_charge of the procedure. As such, the procedures are sorted from smallest to highest based on how expensive they are rather than by their numerical identification number. If we use `x = drg` instead of `x = reorder(drg, mean_charge)`, then the medical procedures are ordered by their drg number rather than by the average charge. Thus, they are no longer presented from lowest to highest in terms of average medical charge.

```r
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
```

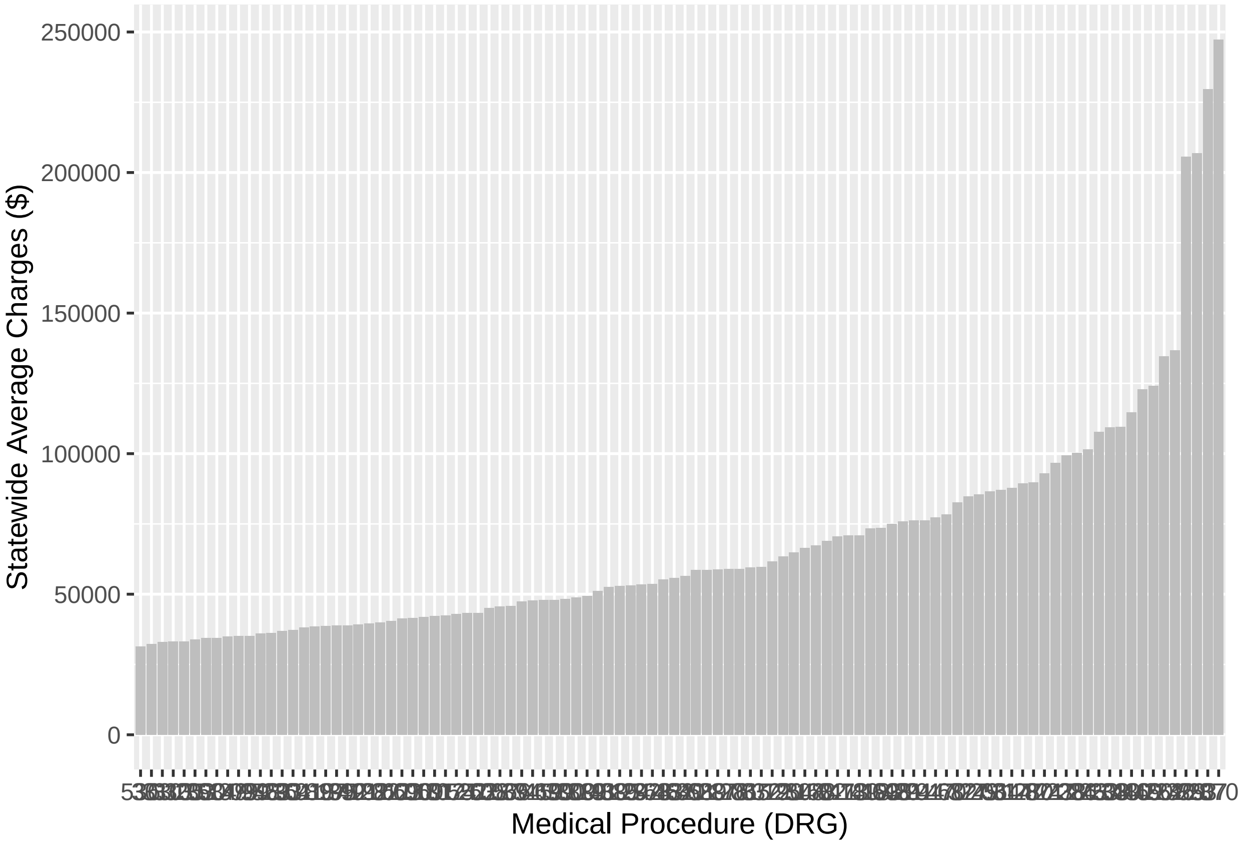


b.

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: If we remove the `theme()` line, the labels on the x axis are all bunched together. This is because the labels are now all horizontally oriented rather than vertically oriented. As such, the theme line first oriented the labe 90 degrees, due to `angle=90` and reduced the size of the lable with the instruction `size = rel(0.5)`.

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

# create the plot object
q <- ggplot(
  data = ChargesNJ,
  aes(x = reorder(drg, mean_charge), y = mean_charge)
) +
  geom_col(fill = "gray") +
  ylab("Statewide Average Charges ($)") +
  xlab("Medical Procedure (DRG)")

q
```

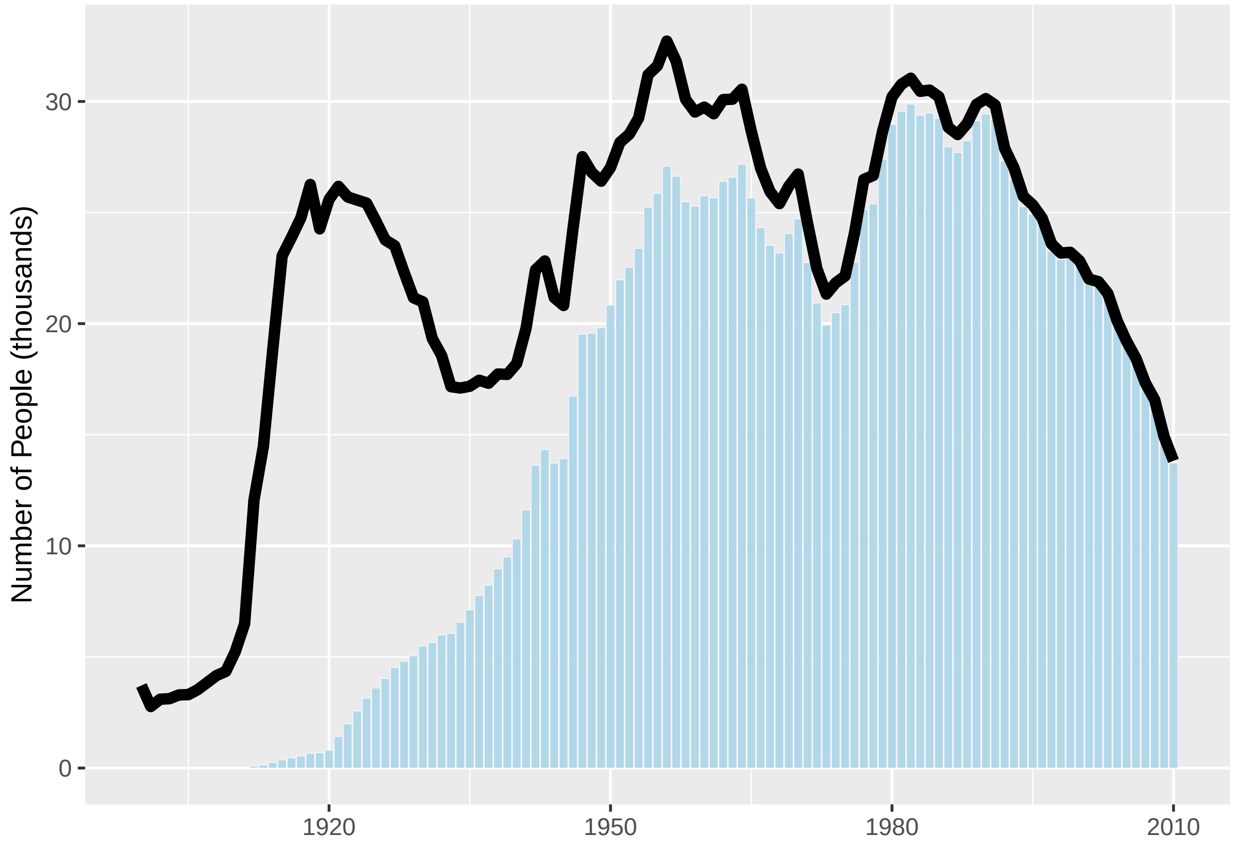

250000
200000
150000
100000
50000
0
Statewide Average Charges ($)
Medical Procedure (DRG)

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: My first question is about the section of code where the median year of birth is calculated using wtd.quartile. I'm not sure how that section of the code works, especially how the pipe symbols relate to `year = wtd_quantile(year, est_alive_today, probs = 0.5)`. My second question is about the "tribble" and what was the point of the numbers in the tribble. For example, there is ~year, ~num_people but where do we use these variables?

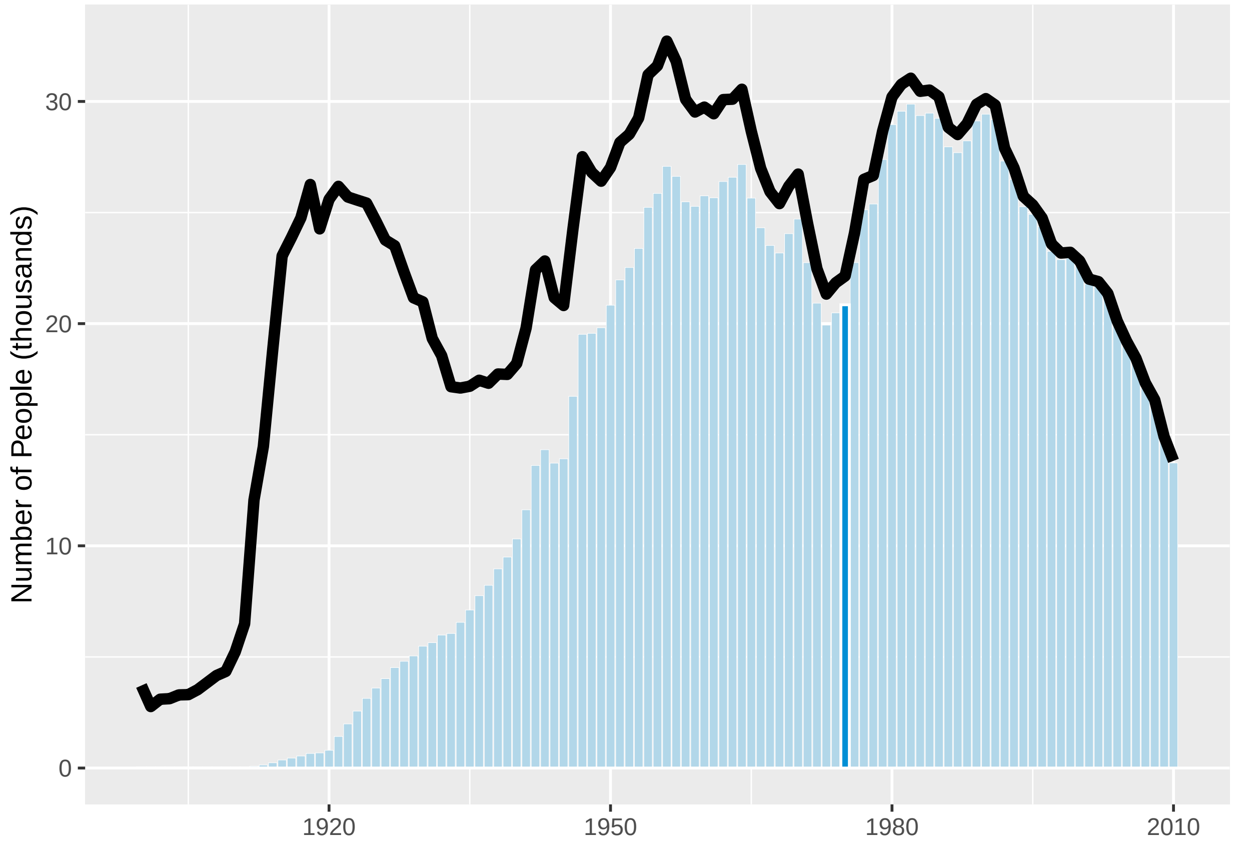
```
# to get you started following along . . .
library(babynames)
BabynamesDist <- make_babynames_dist()

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", size = 0.1)

name_plot <- name_plot +
  geom_line(aes(y = count_thousands), size = 2)
name_plot <- name_plot +
  ylab("Number of People (thousands)") +
  xlab(NULL)

name_plot
```



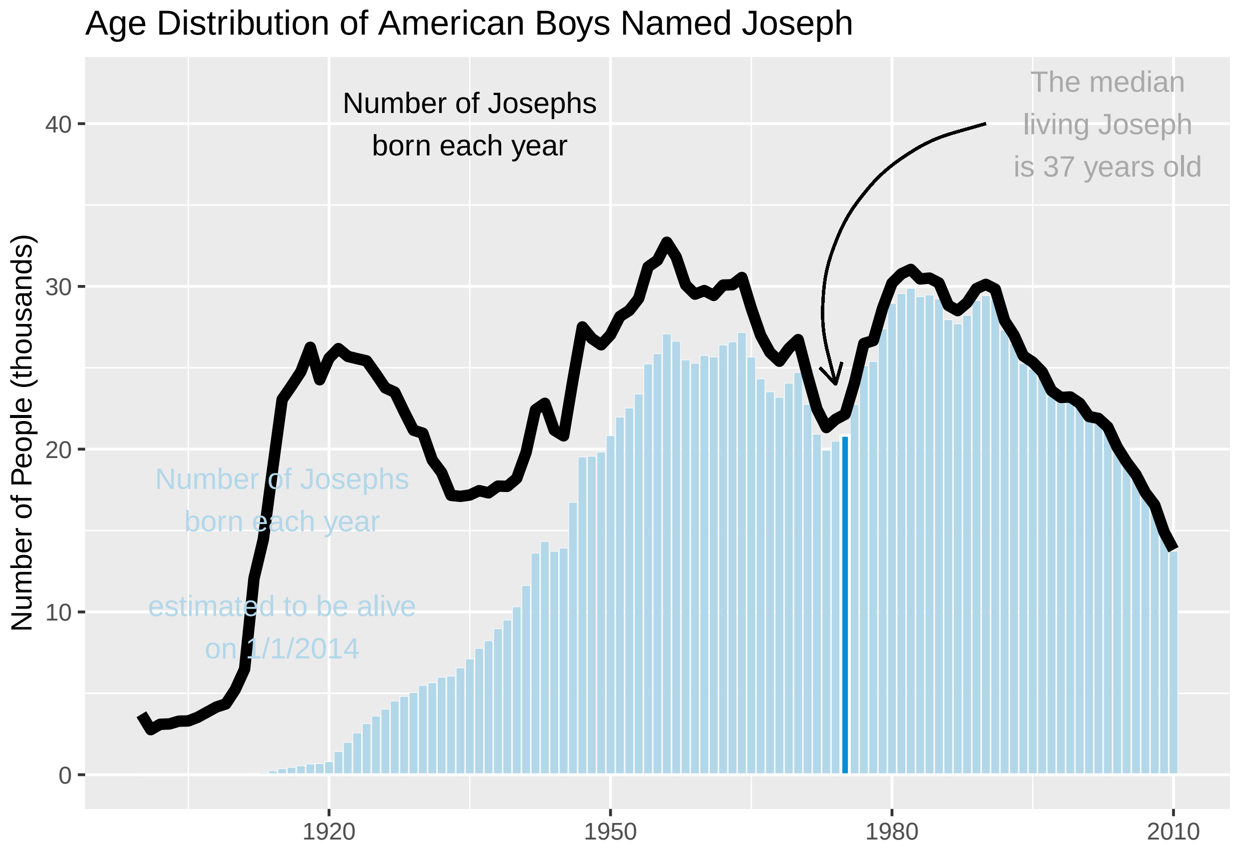
```
wtd_quantile <- Hmisc::wtd.quantile
median_yob <- joseph %>%
  summarize(
    year = wtd_quantile(year, est_alive_today, probs = 0.5)
  ) %>%
  pull(year)
median_yob
```

```
##  50%
## 1975
```

```
name_plot <- name_plot +
  geom_col(
    color = "white", fill = "#008fd5",
    aes(y = ifelse(year == median_yob, est_alive_today / 1000, 0))
  )
name_plot
```



```
context <- tribble(
  ~year, ~num_people, ~label,
  1935, 40, "Number of Josephs\nborn each year",
  1915, 13, "Number of Josephs\nborn each year
  \nestimated to be alive\non 1/1/2014",
  2003, 40, "The median\nliving Joseph\nis 37 years old",
)

name_plot +
  ggtitle("Age Distribution of American Boys Named Joseph") +
  geom_text(
    data = context,
    aes(y = num_people, label = label, color = label)
  ) +
  geom_curve(
    x = 1990, xend = 1974, y = 40, yend = 24,
    arrow = arrow(length = unit(0.3, "cm")), curvature = 0.5
  ) +
  scale_color_manual(
    guide = FALSE,
    values = c("black", "#b2d7e9", "darkgray")
  ) +
  ylim(0, 42)
```


Age Distribution of American Boys Named Joseph
Number of Josephs
born each year
The median
living Joseph
is 37 years old
Number of Josephs
born each year
estimated to be alive
on 1/1/2014
Number of People (thousands)
40
30
20
10
0
1920
1950
1980
2010