Lab Session 4 - Solutions

Leandro Grespan

8/26/2020

We expect you to **watch the class 4 material**, here prior to lab. In addition, **read the background and data section before lab**.

Background and data

Follow the tweet thread and you'll see that Prof. Damon Jones, of Harris, gets that data and does some analysis. In this lab, you're going to follow his lead and dig into traffic stop data from the University of Chicago Police Department, one of the largest private police forces in the world.

Download the data here.

Warm-up

1. Open a new `Rmd` and save it in your coding lab folder. If you have not yet, move your data file to your preferred data location.
2. In your `Rmd`, write code to load your packages. If you load packages in the console, you will get an error when you knit because knitting starts a fresh R session.

```r
library("tidyverse")
```

1. Load `data_traffic.csv` and assign it to the name `traffic_data`. This data was scrapped from the UCPD website and partially cleaned by Prof. Jones.

Note: This solution may vary depending on where your csv file is, compared to the Rmd file location. Please refer to Lab 3's Problem Set for more information

```r
traffic_data <- read_csv("../data/data_traffic.csv")
```

1. Recall that `group_by()` operates silently. How can you tell `grouped_data` different from `traffic_data`?

   You can use summarise() to check the grouped data:

   ```r
   grouped_data <-
     traffic_data %>%
       group_by(Race, Gender)

   summarise(grouped_data)
   ```

```
## # A tibble: 14 x 2
## # Groups:   Race [6]
##    Race                                     Gender
##    <chr>                                    <chr>
##  1 African American                         female
##  2 African American                         Female
##  3 African American                         Male
##  4 African American                         <NA>
##  5 American Indian/Alaskan Native           Female
##  6 American Indian/Alaskan Native           Male
##  7 Asian                                    Female
##  8 Asian                                    Male
##  9 Caucasian                                Female
## 10 Caucasian                                male
## 11 Caucasian                                Male
## 12 Hispanic                                 Female
## 13 Hispanic                                 Male
## 14 Native Hawaiian/Other Pacific Islander Male
```

a. How many groups (Race-Gender pairs) are in the data? (This information should be available without writing additional code!)

   14, the number of rows in the tibble.

b. Before running the code. Predict the dimensions (number of rows by number of columns) of the tibbles created by `traffic_data %>% summarize(n = n())` and `grouped_data %>% summarize(n = n())`. Now check you intuition by running the code.

   The traffic_data summary will be a 1x1 tibble and the grouped_data summary will be a 14x3 tibble

```r
traffic_data %>% summarize(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  4478
```

```r
grouped_data %>% summarize(n = n())
```

```
## # A tibble: 14 x 3
## # Groups:   Race [6]
##    Race                                     Gender     n
##    <chr>                                    <chr>  <int>
##  1 African American                         female     4
##  2 African American                         Female  1217
##  3 African American                         Male    2056
##  4 African American                         <NA>       1
##  5 American Indian/Alaskan Native           Female     2
##  6 American Indian/Alaskan Native           Male      10
##  7 Asian                                    Female    62
##  8 Asian                                    Male     164
##  9 Caucasian                                Female   263
## 10 Caucasian                                male       1
## 11 Caucasian                                Male     477
## 12 Hispanic                                 Female    68
## 13 Hispanic                               Male     149
## 14 Native Hawaiian/Other Pacific Islander Male       4
```

2. Use `group_by()` and `summarize()` to recreate the following table.

```
traffic_data %>%
  group_by(Race) %>%
  summarize(n = n())
```

```
## # A tibble: 6 x 2
##   Race                                       n
##   <chr>                                  <int>
## 1 African American                        3278
## 2 American Indian/Alaskan Native            12
## 3 Asian                                    226
## 4 Caucasian                                741
## 5 Hispanic                                 217
## 6 Native Hawaiian/Other Pacific Islander     4
```

3. Use `count()` to produce the same table.

```
traffic_data %>%
  count(Race)
```

```
## # A tibble: 6 x 2
##   Race                                       n
##   <chr>                                  <int>
## 1 African American                        3278
## 2 American Indian/Alaskan Native            12
## 3 Asian                                    226
## 4 Caucasian                                741
## 5 Hispanic                                 217
## 6 Native Hawaiian/Other Pacific Islander     4
```

Moving beyond counts

1. Raw counts are okay, but frequencies (or proportions) are easier to compare across data sets. Add a column with frequencies and assign the new tibble to the name `traffic_stop_freq`. The result should be identical to Prof. Jones's analysis on twitter.

   Try on your own first. If you're not sure how to add a frequency though, you could google "add a proportion to count with tidyverse" and find this stackoverflow post. Follow the advice of the number one answer. The green checkmark and large number of upvotes indicate the answer is likely reliable.

```
  traffic_stop_freq <- traffic_data %>%
    group_by(Race) %>%
    summarise(n = n()) %>%
    mutate(freq = n / sum(n))

  traffic_stop_freq
```

```
## # A tibble: 6 x 3
##   Race                                          n     freq
##   <chr>                                     <int>    <dbl>
## 1 African American                           3278 0.732
## 2 American Indian/Alaskan Native               12 0.00268
## 3 Asian                                       226 0.0505
## 4 Caucasian                                   741 0.165
## 5 Hispanic                                    217 0.0485
## 6 Native Hawaiian/Other Pacific Islander        4 0.000893
```

2. The frequencies out of context are not super insightful. What additional information do we need to argue the police are disproportionately stopping members of a certain group? (Hint: Prof. Jones shares the information in his tweets.)[1]

   Prof Jones compares these frequencies with two other frequencies: the demographic breakdown of Hyde Park and the breakdown of UChicago Students races.

3. For the problem above, your groupmate tried the following code. Explain why the frequencies are all 1.[2]

```
traffic_stop_freq_bad <-
traffic_data %>%
  group_by(Race) %>%
  summarize(n = n(),
            freq = n / sum(n))

traffic_stop_freq_bad
```

   As explained in the linked stackoverflow post, the last grouping variable is peeled off *after* the summarise function, by default. So, if you calculate frequencies within the summarize function, the data will still be grouped by race and therefore each frequency must be 1. However, if you calculate frequencies after the summarise function, the whole data will be ungrouped and frequencies can be properly calculated.

4. Now we want to go a step further than Prof. Jones.[3] Do outcomes differ by race? In the first code block below, I provide code so you can visualize disposition by race. "Disposition" is police jargon that means the current status or final outcome of a police interaction.

```
citation_strings <- c("citation issued", "citations issued", "citation  issued" )

arrest_strings <- c("citation issued, arrested on active warrant",
                "citation issued; arrested on warrant",
                "arrested by cpd",
                "arrested on warrant",
                "arrested",
                "arrest")

disposition_by_race <-
    traffic_data %>%
      mutate(Disposition = str_to_lower(Disposition),
             Disposition = case_when(Disposition %in% citation_strings ~ "citation",
```

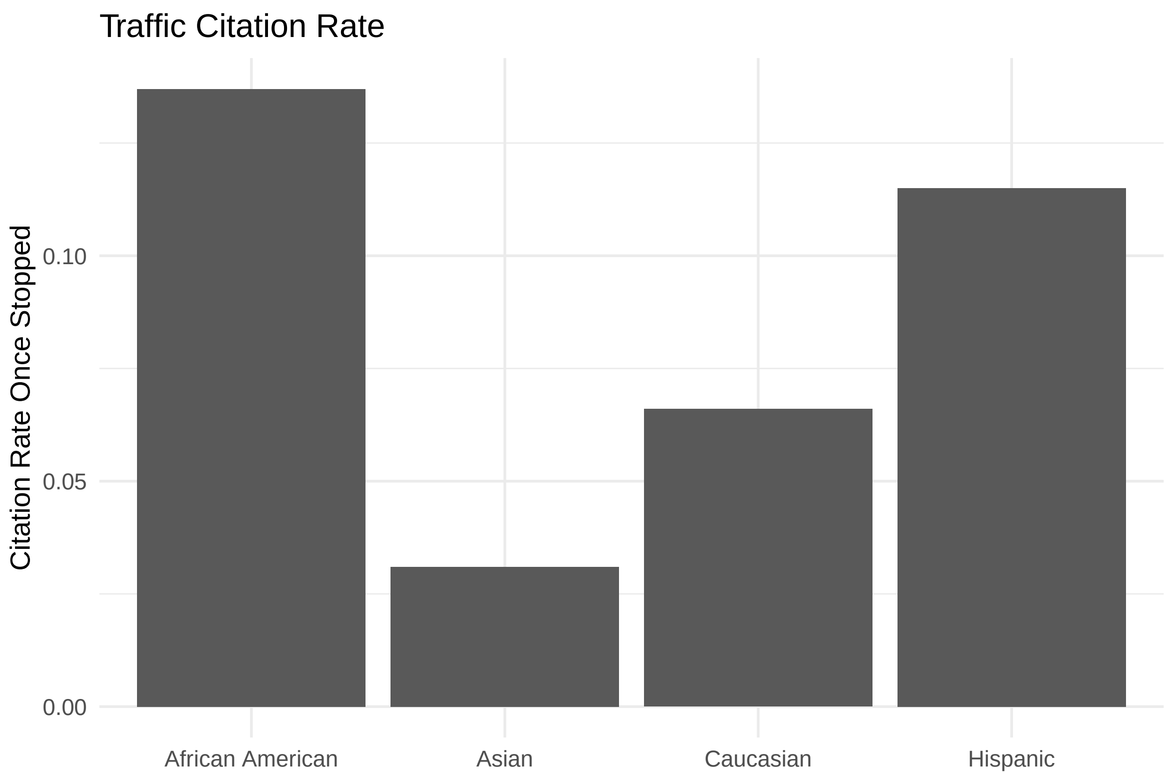
[1] To be fair, even with this information, this is crude evidence that can be explained away in any number of ways. One job of a policy analyst is to bring together evidence from a variety of sources to better understand the issue.

[2] Hint: This is a lesson about `group_by()`!

[3] The analysis that follows is partially inspired by Eric Langowski, a Harris alum, who was also inspired to investigate by the existence of this data (You may have seen Prof. Jones retweet him at the end of the thread.)

```
                                     Disposition %in% arrest_strings ~ "arrest",
                                     TRUE ~ Disposition)) %>%
      count(Race, Disposition) %>%
      group_by(Race) %>%
      mutate(freq = round(n / sum(n), 3))


disposition_by_race %>%
  filter(n > 5, Disposition == "citation") %>%
  ggplot(aes(y = freq, x = Race)) +
  geom_col() +
  labs(y = "Citation Rate Once Stopped", x = "", title = "Traffic Citation Rate") +
  theme_minimal()
```



Let's break down how we got to this code. First, I ran `traffic_data %>% count(Race, Disposition)` and noticed that we have a lot of variety in how officers enter information into the system.[4] I knew I could deal with some of the issue by standardizing capitalization.

a. In the console, try out `str_to_lower(...)` by replacing the `...` with different strings. The name may be clear enough, but what does `str_to_lower()` do?[5]

```
traffic_data %>%
  count(Race, Disposition)
```

```
## # A tibble: 31 x 3
```

[4]Try it yourself!
[5]This code comes from the `stringr` package. Checkout `?str_to_lower` to learn about some related functions.

```
##    Race             Disposition                                    n
##    <chr>            <chr>                                      <int>
##  1 African American Arrest                                         1
##  2 African American Arrested                                       1
##  3 African American Arrested by CPD                                1
##  4 African American Arrested on warrant                            1
##  5 African American Citation  Issued                               2
##  6 African American Citation issued                              127
##  7 African American Citation Issued                              297
##  8 African American Citation Issued, Arrested on Active Warrant    1
##  9 African American Citation issued; arrested on warrant           1
## 10 African American Citations issued                               5
## # ... with 21 more rows
```

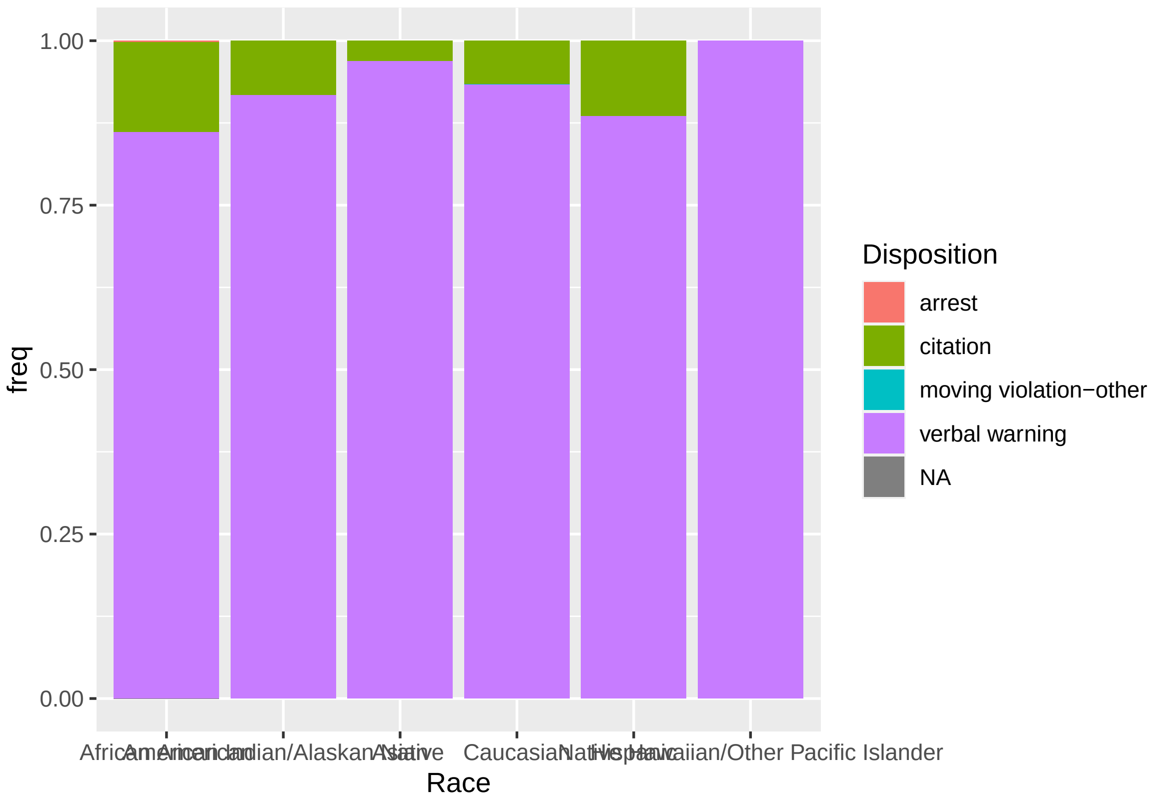
```
str_to_lower("Citation Issued")
```

```
## [1] "citation issued"
```

After using `mutate` with `str_to_lower()`, I piped into `count()` again and looked for strings that represent the same `Disposition`. I stored terms in character vectors (e.g. `citation_strings`). The purpose is to make the `case_when()` easier to code and read. Once I got that right, I added frequencies to finalize `disposition_by_race`. See code above.

5. To make the graph, I first tried to get all the disposition data on the same plot.

```
    disposition_by_race %>%
      ggplot(aes(y = freq, x = Race, fill = Disposition)) +
      geom_col()
```



By default, the bar graph is stacked. Look at the resulting graph and discuss the pros and cons of this plot with your group.

6. I decided I would focus on citations only and added the `filter(n > 5, Disposition == "citation")` to the code.[6] What is the impact of filtering based on `n > 5`? Would you make the same choice? This question doesn't have a "right" answer. You should try different options and reflect.

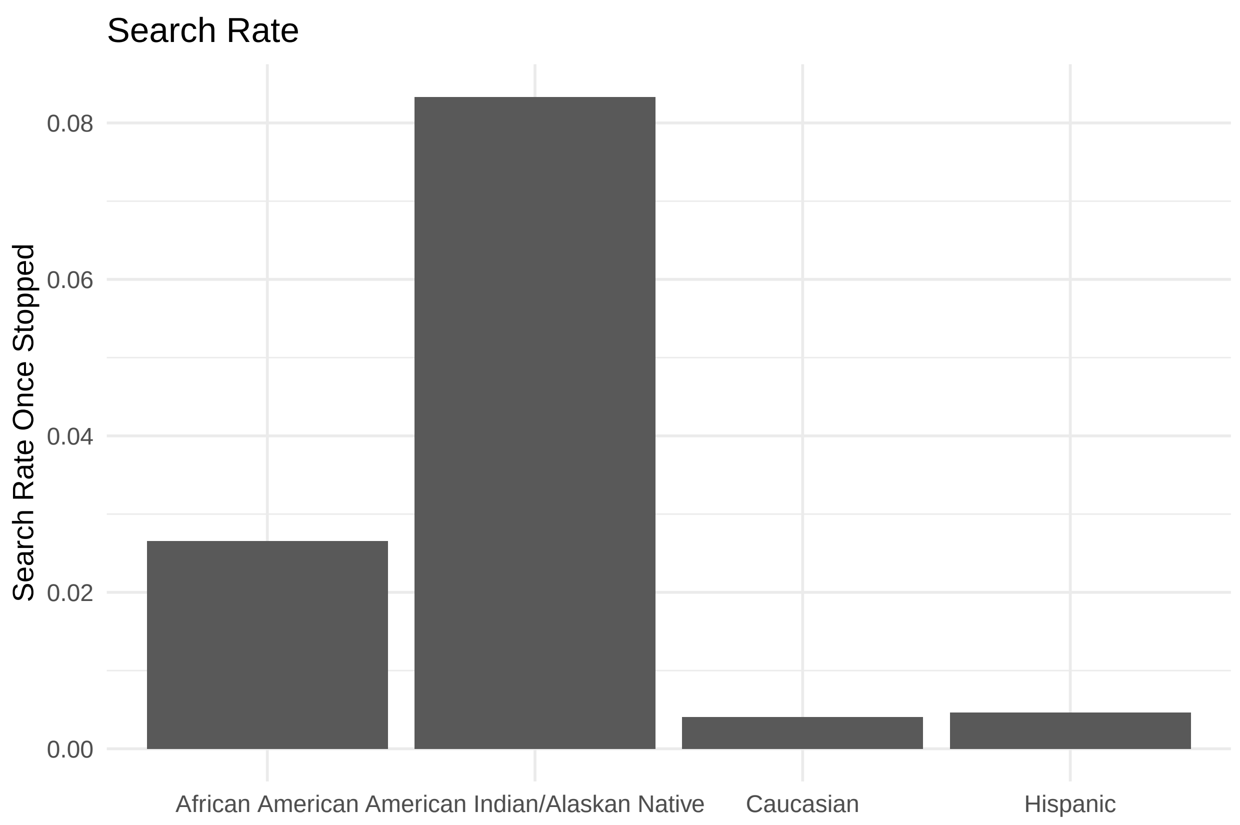
   Here are some arguments (not a comprehensive list): *Against + We throw away information. + `n` here is already subdivided based on "Disposition", but it would make more sense to filter based on number of observations for a given race rather than a race-disposition count.* For + small `n` groups can be misleading since one interaction can sway the result significantly. + An alternative is to create an "other" category, though that might bury heterogeneity across the smallest groups.

7. Now, you can create a similar plot based called "Search Rate" using the `Search` variable. Write code to reproduce this plot.

```
search <-
traffic_data %>%
  mutate(Search = str_to_lower(Search),
         Search = ifelse(is.na(Search) | Search == "N/A","No" , Search)) %>%
  count(Race, Search) %>%
  group_by(Race) %>%
  mutate(freq = n / sum(n))


search %>%
  filter(Search == "yes", n > 0) %>%
ggplot(aes(y = freq, x = Race)) +
      geom_col() +
      labs(y = "Search Rate Once Stopped", x = "", title = "Search Rate") +
      theme_minimal()
```

[6]Notice that I get the data exactly how I wanted it using `dplyr` verbs and then try to make the graph.



Extension: Revisiting world inequality data

When we explored the World Inequality Database data in lab 1, we mimicked grouped analysis by filtering the data to only show data for France and then repeated the analysis for Russia. Using `group_by()`, we can complete the analysis for each country simultaneously.

1. Read in the `wid_data`.[7]

```
wid_data_raw <-
    # You will likely have to adjust the file path
    readxl::read_xlsx("../data/world_wealth_inequality.xlsx",
                      col_names = c("country", "indicator", "percentile", "year", "value")) %>%
    separate(indicator, sep = "\\n", into = c("row_tag", "type", "notes"))

wid_data <- wid_data_raw %>%
              select(-row_tag) %>%
              select(-notes, everything()) %>%
              # some students had trouble because excel added "\r" to the end
              # of each string. mutate standardizes the string across platforms.
              mutate(type = ifelse(str_detect(type, "Net personal wealth"),
                                   "Net personal wealth", type)) %>%
              filter(type == "Net personal wealth")
```

[7]If you are still having trouble, you may want to re-download the file here and do not open with Excel!

2. Create a table that tells us the number of years observed per country and first and last year we have data[8] for each country.[9] For example, India has 6 years of observations and has data from 1961 to 2012.

```r
wid_data %>%
  filter(!is.na(value)) %>%
  group_by(country) %>%
  summarize(n_obs = as.integer(n()/4),
            first_year = min(year),
            last_year = max(year))
```

```
## # A tibble: 8 x 4
##   country            n_obs first_year last_year
##   <chr>              <int>      <dbl>     <dbl>
## 1 China                 38       1978      2015
## 2 France               107       1902      2014
## 3 India                  6       1961      2012
## 4 Korea                 12       2000      2013
## 5 Russian Federation    21       1995      2015
## 6 South Africa          25       1993      2017
## 7 United Kingdom        50       1900      2012
## 8 USA                   77       1913      2016
```

3. Create a table that provides the mean and standard deviation of the share of wealth owned by the top 10 percent and top 1 percent for each country. Call the resulting tibble `mean_share_per_country`.

```r
mean_share_per_country <-
wid_data %>%
  filter(percentile %in% c("p99p100", "p90p100")) %>%
  group_by(country, percentile) %>%
  summarize(mean_share = mean(value, na.rm = TRUE),
            sd_share = sd(value, na.rm = TRUE))

mean_share_per_country
```

```
## # A tibble: 16 x 4
## # Groups:   country [8]
##    country            percentile mean_share sd_share
##    <chr>              <chr>           <dbl>    <dbl>
##  1 China              p90p100         0.483   0.0939
##  2 China              p99p100         0.202   0.0546
##  3 France             p90p100         0.669   0.127
##  4 France             p99p100         0.329   0.131
##  5 India              p90p100         0.499   0.0808
##  6 India              p99p100         0.178   0.0798
##  7 Korea              p90p100         0.640   0.0239
##  8 Korea              p99p100         0.243   0.0181
##  9 Russian Federation p90p100         0.646   0.0448
## 10 Russian Federation p99p100         0.364   0.0578
## 11 South Africa       p90p100         0.860   0.0261
```

[8] i.e. not `NA`s.

[9] Hint: `?summarize` lists "Useful functions" for summarizing data. Look at the "Range" or "Position" functions. If you are going to use the "Position" functions, make sure the data is sorted properly.

```
## 12 South Africa       p99p100         0.513   0.0337
## 13 United Kingdom     p90p100         0.715   0.166
## 14 United Kingdom     p99p100         0.395   0.194
## 15 USA                p90p100         0.714   0.0697
## 16 USA                p99p100         0.321   0.0722
```

a. Which country has the smallest standard deviation in share of wealth owned by the top 10 percent? Use `arrange()` to order the countries by standard deviation. Compare the order to you results above about the number of observation and time horizon.[10]

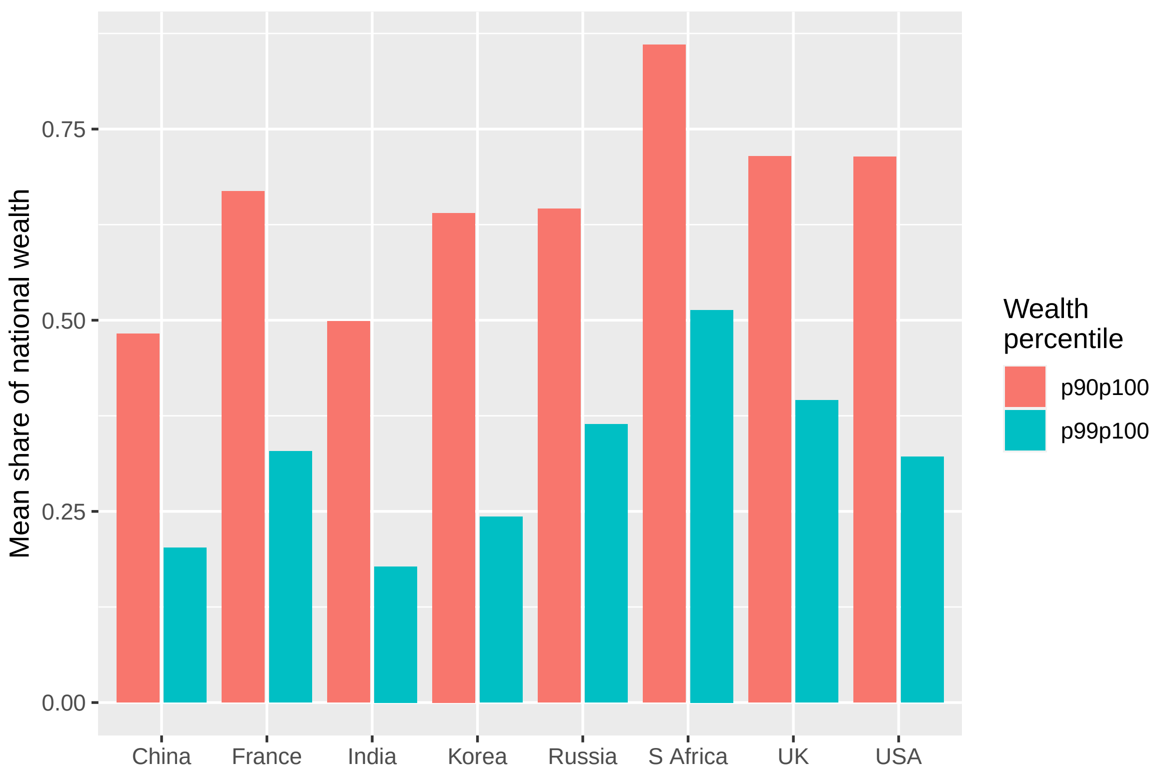
```
  mean_share_per_country %>%
    filter(percentile == "p90p100") %>%
    arrange(sd_share)
```

```
## # A tibble: 8 x 4
## # Groups:   country [8]
##   country            percentile mean_share sd_share
##   <chr>              <chr>           <dbl>    <dbl>
## 1 Korea              p90p100         0.640   0.0239
## 2 South Africa       p90p100         0.860   0.0261
## 3 Russian Federation p90p100         0.646   0.0448
## 4 USA                p90p100         0.714   0.0697
## 5 India              p90p100         0.499   0.0808
## 6 China              p90p100         0.483   0.0939
## 7 France             p90p100         0.669   0.127
## 8 United Kingdom     p90p100         0.715   0.166
```

a. If your code worked, you should be able to make this bar chart.

```
mean_share_per_country %>%
  mutate(country = case_when(country == "Russian Federation" ~ "Russia",
                             country == "United Kingdom" ~ "UK",
                             country == "South Africa" ~ "S Africa",
                             TRUE ~ country)) %>%
  ggplot(aes(x = country, y = mean_share, fill = percentile)) +
  geom_col(position = "dodge2") +
  labs(y = "Mean share of national wealth", x = "", fill = "Wealth\npercentile")
```

[10]Usually when we get more data we expect variances to decreased, but that reasoning assumes independence between observations. In this case, there is high temporal correlation, which means if the top 10 percent own 50 percent of wealth this year, they'll own some proportion near 50 percent in the next year. South Korean only has observations during a short and highly stable historical period, so that explains the low variance.

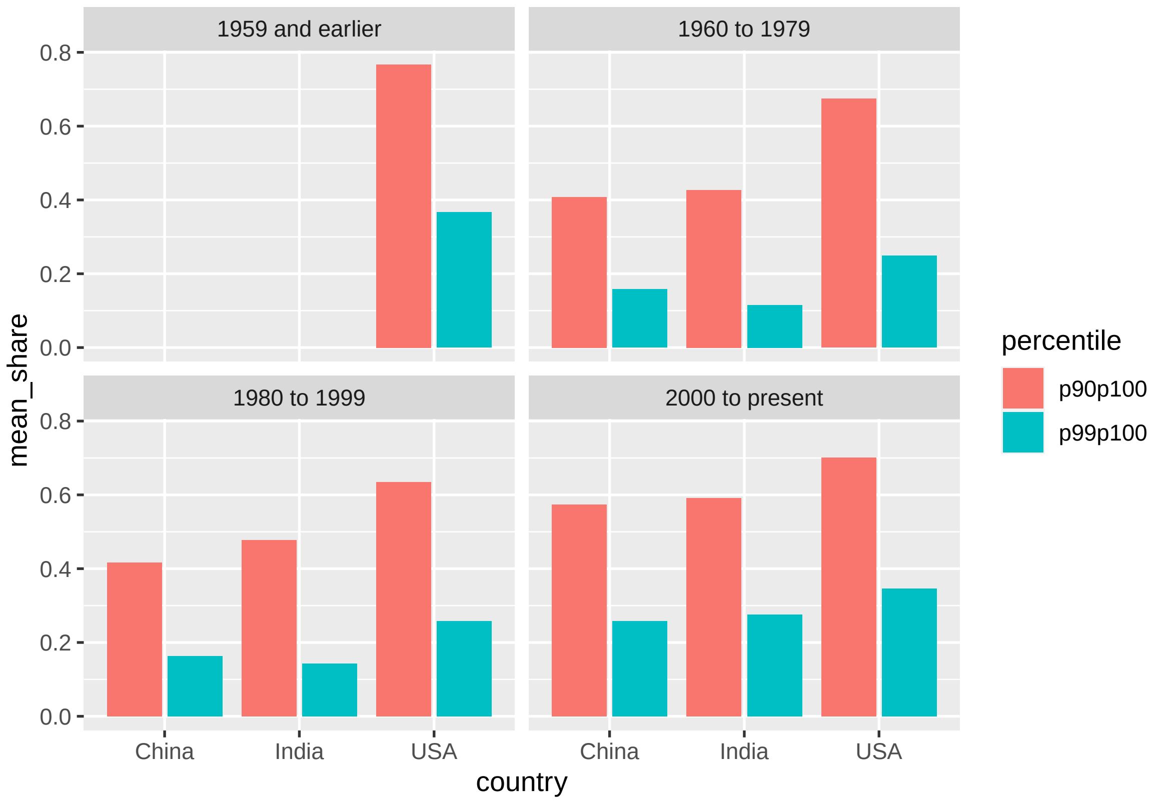


4. **Challenge** Write code to create `mean_share_per_country_with_time` a tibble that produces the following graph which lets us see how the share of national wealth held by the top 10 and 1 percent change over time.[11]

```
mean_share_per_country_with_time %>%
  ggplot(aes(x = country, y = mean_share, fill = percentile)) +
    geom_col(position = "dodge2") +
    facet_wrap(~time_period)
```

```
## Warning: Removed 4 rows containing missing values (geom_col).
```

[11]Hint: use `case_when` or several `ifelse` to create a new column called `time_period` that labels data as "1959 and earlier", "1960 to 1979", "1980 to 1999", or "2000 to present". Then, add `time_period` to your `group_by()` along with other relevant grouping variables.



1959 and earlier
1960 to 1979
1980 to 1999
2000 to present
mean_share
0.8
0.6
0.4
0.2
0.0
China
India
USA
country
percentile
p90p100
p99p100