Problem Set 1

Answer Key

ECON 480 — Fall 2021

Answers generally go above and beyond what I expect from you. They are meant to show you the correct answer, explain *why* it is correct, and potentially show *several methods* by which you can reach the answer.

The Popularity of Baby Names

Install and load the package `babynames`. Get help for `?babynames` to see what the data includes.

```
# install.packages("babynames")

# Note I've "commented" out some of these commands  (with a #) so they do not run when I knit this docu
# You should **never** install a package inside a .Rmd document, just do that in R Studio itself
# Of course, you do need to load everything with library() in a .Rmd document!

library(babynames)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# ?babynames()
```

Question 1

Part A

What are the top 5 boys names for 2017, and what *percent* of overall names is each?

```
# save as a new tibble
top_5_boys_2017 <- babynames %>% # take data
  filter(sex=="M", # filter by males
         year==2017) %>% # and for 2007
  arrange(desc(n)) %>% # arrange in largest-to-smallest order of n (number)
  slice(1:5) %>% # optional, look only at first 5 rows; head(., n=5) also works
  mutate(percent = round(prop*100, 2)) # also optional, make a percent variable rounded to 2 decimals

# look at our new tibble
top_5_boys_2017
```

```
## # A tibble: 5 x 6
##    year sex   name        n    prop percent
##   <dbl> <chr> <chr>   <int>   <dbl>   <dbl>
## 1  2017 M     Liam    18728 0.00954    0.95
## 2  2017 M     Noah    18326 0.00933    0.93
## 3  2017 M     William 14904 0.00759    0.76
## 4  2017 M     James   14232 0.00725    0.72
## 5  2017 M     Logan   13974 0.00712    0.71
```

The top 5 names are

```
top_5_boys_2017 %>%
  select(name,percent) %>%
  knitr::kable()
```

name	percent
Liam	0.95
Noah	0.93
William	0.76
James	0.72
Logan	0.71

Alternatively, you could just write what you found manually into an object like:

```
top_5_boys_2017_alt <- c("Liam", "Noah", "William", "James", "Logan")

top_5_boys_2017_alt
```

```
## [1] "Liam"    "Noah"    "William" "James"   "Logan"
```

```
# you could alternatively add a command,
# %>% pull(name) to the first chunk of code,
# and it would do the same thing, but we'd want to save it,
# for example:

top_5_boys_2017_alt <- babynames %>%
  filter(sex=="M",
         year==2017) %>%
  arrange(desc(n)) %>%
  slice(1:5) %>%
  mutate(percent = round(prop*100, 2)) %>%
  pull(name)

top_5_boys_2017_alt
```

```
## [1] "Liam"    "Noah"    "William" "James"   "Logan"
```

Part B

What are the top 5 *girls* names for 2017, and what *percent* of overall names is each?

```
# save as a new tibble
top_5_girls_2017 <- babynames %>% # take data
  filter(sex=="F", # filter by females
         year==2017) %>% # and for 2007
  arrange(desc(n)) %>% # arrange in largest-to-smallest order of n (number)
  slice(1:5) %>% # optional, look only at first 5 rows; head(., n=5) also works
  mutate(percent = round(prop*100, 2)) # also optional, make a percent variable rounded to 2 decimals

# look at our new tibble
top_5_girls_2017
```

```
## # A tibble: 5 x 6
##    year sex   name         n    prop percent
##   <dbl> <chr> <chr>    <int>   <dbl>   <dbl>
## 1  2017 F     Emma     19738 0.0105     1.05
## 2  2017 F     Olivia   18632 0.00994    0.99
## 3  2017 F     Ava      15902 0.00848    0.85
## 4  2017 F     Isabella 15100 0.00805    0.81
## 5  2017 F     Sophia   14831 0.00791    0.79
```

The top 5 names are

```
top_5_girls_2017 %>%
  select(name,percent) %>%
  knitr::kable()
```

name	percent
Emma	1.05
Olivia	0.99
Ava	0.85
Isabella	0.81
Sophia	0.79

Alternatively, you could just write what you found manually into an object like:

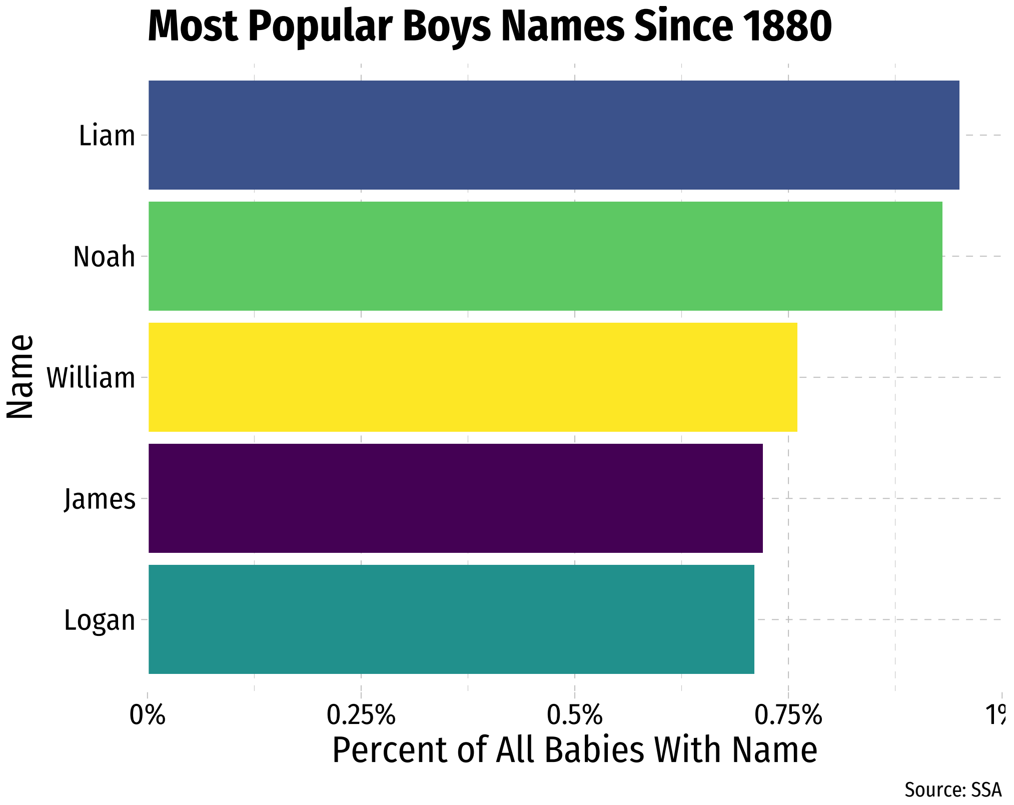
```
top_5_girls_2017_alt <- c("Emma", "Olivia", "Ava", "Isabella", "Sophia")
```

Question 2

Make two barplots of these top 5 names, one for each sex. Map aesthetics x to name and y to prop [or percent, if you made that variable, as I did.] and use geom_col (since you are declaring a specific y, otherwise you could just use geom_bar() and just an x.)

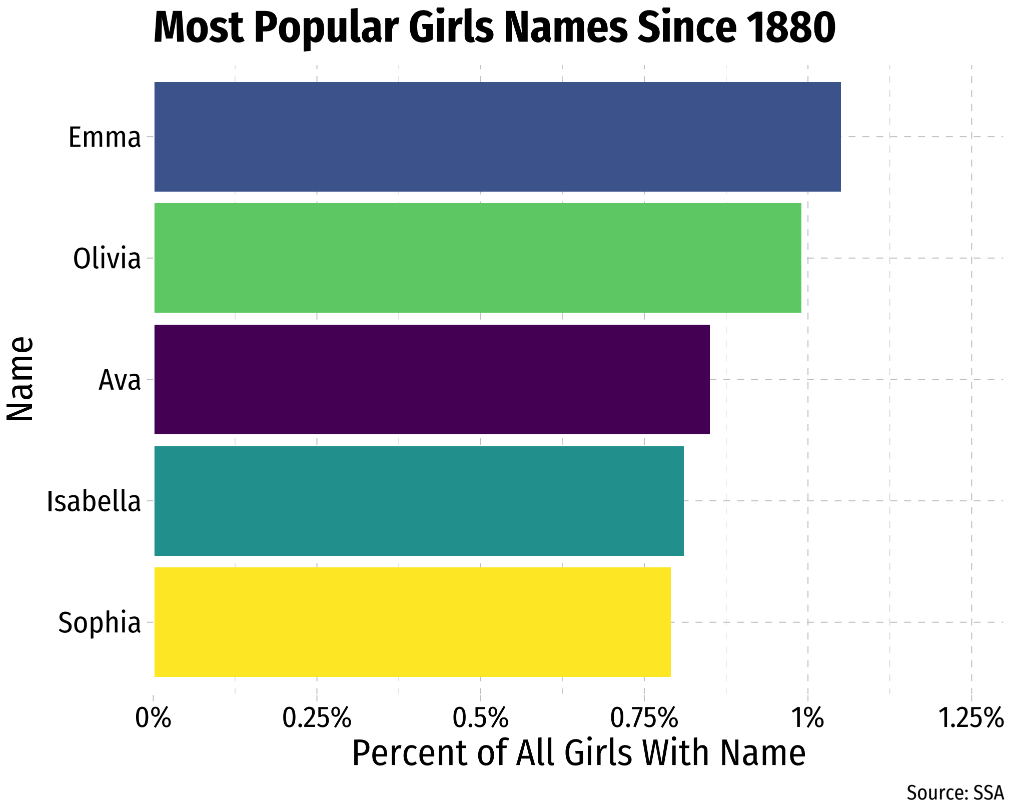
```
ggplot(data = top_5_boys_2017)+
  aes(x = reorder(name, n), #note this reorders the x variable from small to large n
      y = percent, # you can use prop if you didn't make a percent variable
      fill = name)+ # optional color!
  geom_col()+

  # all of the above is sufficient, now I'm just making it pretty
  scale_y_continuous(labels = function(x){paste0(x, "%")}, # add percent signs
                     breaks = seq(from = 0, # make line breaks every 0.25%
                                  to = 1,
                                  by = 0.25),
                     limits = c(0,1), # limit axis to between 0 and 1
                     expand = c(0,0))+ # don't let it go beyond this
  labs(x = "Name",
       y = "Percent of All Babies With Name",
       title = "Most Popular Boys Names Since 1880",
      fill = "Boy's Name",
      caption = "Source: SSA")+
  ggthemes::theme_pander(base_family = "Fira Sans Condensed", base_size=16)+
  coord_flip()+ # flip axes to make horizontal!
  scale_fill_viridis_d(option = "default")+ # use viridis discrete color palette
  theme(legend.position = "") # hide legend
```

```
## Warning in viridisLite::viridis(n, alpha, begin, end, direction, option): Option
## 'default' does not exist. Defaulting to 'viridis'.
```



```
ggplot(data = top_5_girls_2017)+
  aes(x = reorder(name, n), #note this reorders the x variable from small to large n
      y = percent, # you can use prop if you didn't make a percent variable
      fill = name)+ # optional color!
  geom_col()+
  # all of the above is sufficient, now I'm just making it pretty
  scale_y_continuous(labels = function(x){paste0(x, "%")}, # add percent signs
                     breaks = seq(from = 0, # make line breaks every 0.25%
                                  to = 1.25,
                                  by = 0.25),
                     limits = c(0,1.3), # limit axis to between 0 and 1.2
                     expand = c(0,0))+ # don't let it go beyond this
  labs(x = "Name",
       y = "Percent of All Girls With Name",
       title = "Most Popular Girls Names Since 1880",
       fill = "Girl's Name",
       caption = "Source: SSA")+
  ggthemes::theme_pander(base_family = "Fira Sans Condensed", base_size=16)+
  coord_flip()+ # flip axes to make horizontal!
  scale_fill_viridis_d(option = "default")+ # use viridis discrete color palette
  theme(legend.position = "") # hide legend
```

```
## Warning in viridisLite::viridis(n, alpha, begin, end, direction, option): Option
## 'default' does not exist. Defaulting to 'viridis'.
```



If you had gone the alternate route by saving an object of names (like I did above with `top_5_boys_2017_alt` and `top_5_girls_2017_alt`), you could filter the data using the `%in%` operator to use for your `data` layer of each plot, like so:

```
boys_data <- babynames %>%
  filter(name %in% top_5_boys_2017_alt) # this will only use data for the 5 names

ggplot(data = boys_data) #+... the rest of the plot code above
```

Question 3

Find your name. [If your name isn't in there :(, pick a random name.] `count` by `sex` how many babies since 1880 were named your name. [Hint: if you do this, you'll get the number of *rows* (years) there are in the data. You want to add the number of babies in each row (`n`), so inside `count`, add `wt = n` to weight the count by `n`.] Also add a variable for the percent of each sex.

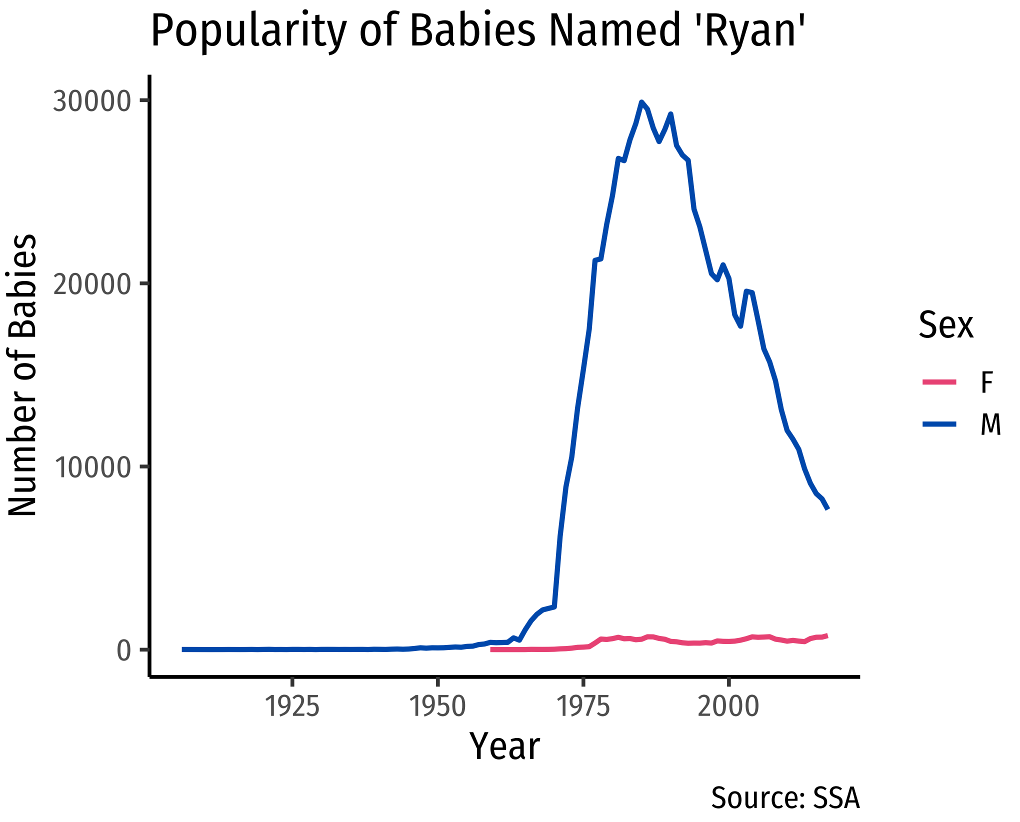
```
babynames %>%
  filter(name == "Ryan") %>%
  count(sex, wt=n) %>%
  mutate(percent = round((n/sum(n)*100),2))
```

```
## # A tibble: 2 x 3
##   sex         n percent
##   <chr>  <int>   <dbl>
## 1 F      22910    2.42
## 2 M     924877   97.6
```

Question 4

Make a line graph of the number of babies with your name over time, colored by sex.

```
# note here I'm going to wrangle the data and then pipe it directly into ggplot
# you can wrangle the data and save it as a different tibble, then use THAT tibble
# for your (data = ...) command in ggplot

# first wrangle data
babynames %>%
  filter(name == "Ryan") %>%

  # now we pipe into ggplot
  ggplot(data = .)+ # the "." is a placeholder for the stuff above!
  aes(x = year,
      y = n,
      color = sex)+
  geom_line(size = 1)+
  scale_color_manual(values = c("F" = "#e64173", # make my own colors
                                "M" = "#0047AB"))+
  labs(x = "Year",
       y = "Number of Babies",
       title = "Popularity of Babies Named 'Ryan'",
       color = "Sex",
       caption = "Source: SSA")+
    theme_classic(base_family = "Fira Sans Condensed", base_size=16)
```



Question 5

Part A

Find the most common name for boys by year between 1980-2017. [Hint: you'll want to first group_by(year). Once you've got all the right conditions, you'll get a table with a lot of data. You only want to slice(1) to keep just the 1st row of each year's data.]

```
babynames %>%
  group_by(year) %>% # we want one observation per year
  filter(sex == "M",
         year>1979) %>% # or >==1980
  arrange(desc(n))%>% # start with largest n first
  slice(1) # take first row only
```

```
## # A tibble: 38 x 5
## # Groups:   year [38]
##     year sex   name        n   prop
##    <dbl> <chr> <chr>   <int>  <dbl>
##  1  1980 M     Michael 68693 0.0370
##  2  1981 M     Michael 68765 0.0369
##  3  1982 M     Michael 68228 0.0362
##  4  1983 M     Michael 67995 0.0365
##  5  1984 M     Michael 67736 0.0361
##  6  1985 M     Michael 64906 0.0337
##  7  1986 M     Michael 64205 0.0334
##  8  1987 M     Michael 63647 0.0326
##  9  1988 M     Michael 64133 0.0320
## 10  1989 M     Michael 65382 0.0312
## # ... with 28 more rows
```

Part B

Now do the same for girls.

```
babynames %>%
  group_by(year) %>% # we want one observation per year
  filter(sex == "F",
         year>1979) %>% # or >==1980
  arrange(desc(n))%>% # start with largest n first
  slice(1) # take first row only
```

```
## # A tibble: 38 x 5
## # Groups:   year [38]
##     year sex   name         n   prop
##    <dbl> <chr> <chr>    <int>  <dbl>
##  1  1980 F     Jennifer 58376 0.0328
##  2  1981 F     Jennifer 57049 0.0319
##  3  1982 F     Jennifer 57115 0.0315
##  4  1983 F     Jennifer 54342 0.0304
##  5  1984 F     Jennifer 50561 0.0280
##  6  1985 F     Jessica  48346 0.0262
##  7  1986 F     Jessica  52674 0.0285
##  8  1987 F     Jessica  55991 0.0299
##  9  1988 F     Jessica  51538 0.0268
## 10  1989 F     Jessica  47885 0.0240
## # ... with 28 more rows
```

Question 6

Now let's graph the evolution of the most common names since 1880.

Part A

First, find out what are the top 5 *overall* most popular names for boys and for girls in the data. [Hint: first `group_by(name)`.] You may want to create two objects, each with these top 5 names as character elements.

```
babynames %>%
  group_by(name) %>% # we want one row per name
  filter(sex=="M") %>%
  summarize(total=sum(n)) %>% # add upp all of the n's for all years for each name
  arrange(desc(total)) %>% # list largest total first
  slice(1:5)
```

```
## # A tibble: 5 x 2
##   name      total
##   <chr>     <int>
## 1 James   5150472
## 2 John    5115466
## 3 Robert  4814815
## 4 Michael 4350824
## 5 William 4102604
```

```
# make a vector of the names (we'll need this for our graph below)
top_boys_names<-c("James", "John", "Robert", "Michael", "William")
```

```
# you could alternatively add a command,
# %>% pull(name) to the first chunk of code,
# and it would do the same thing, but we'd want to save it,
# for example:

babynames %>%
  group_by(name) %>% # we want one row per name
  filter(sex=="M") %>%
  summarize(total=sum(n)) %>% # add upp all of the n's for all years for each name
  arrange(desc(total)) %>% # list largest total first
  slice(1:5) %>%
  pull(name)
```

```
## [1] "James"   "John"    "Robert"  "Michael" "William"
```

```
babynames %>%
  group_by(name) %>% # we want one row per name
  filter(sex=="F") %>%
  summarize(total=sum(n)) %>% # add upp all of the n's for all years for each name
  arrange(desc(total)) %>% # list largest total first
  slice(1:5)
```

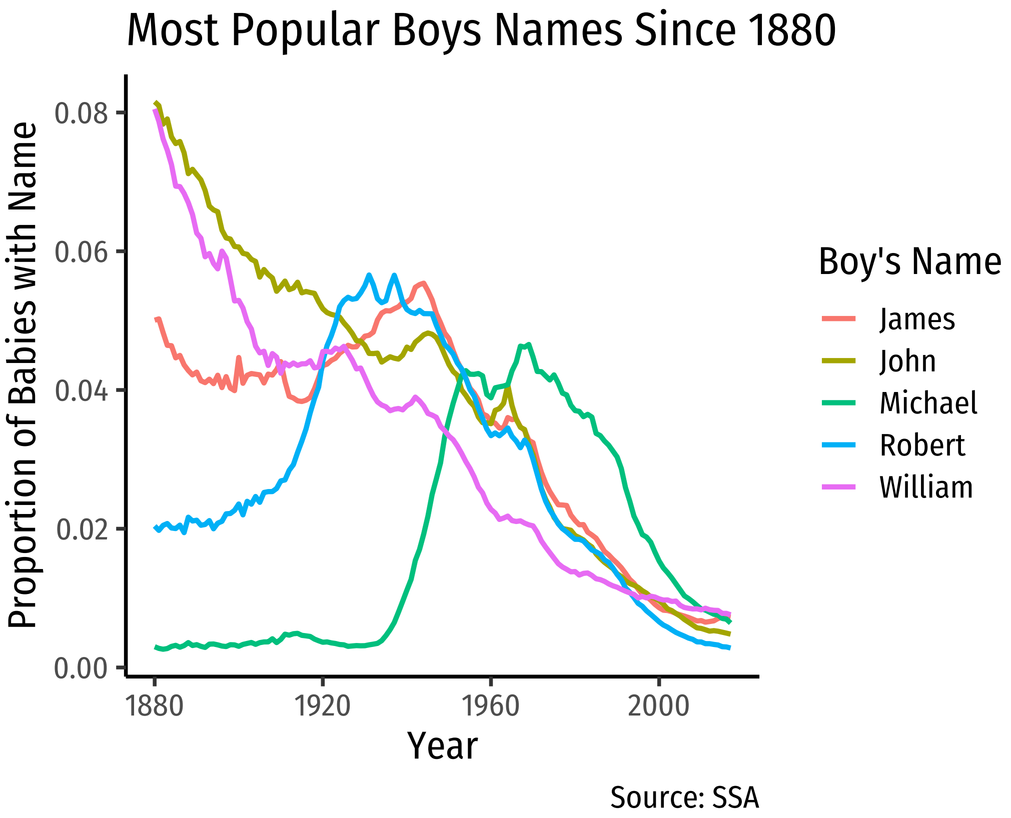
```
## # A tibble: 5 x 2
##   name        total
##   <chr>       <int>
## 1 Mary      4123200
## 2 Elizabeth 1629679
## 3 Patricia  1571692
## 4 Jennifer  1466281
## 5 Linda     1452249
```

```
# make a vector of the names (we'll need this for our graph below)
top_girls_names<-c("Mary", "Elizabeth", "Patricia", "Jennifer", "Linda")
```

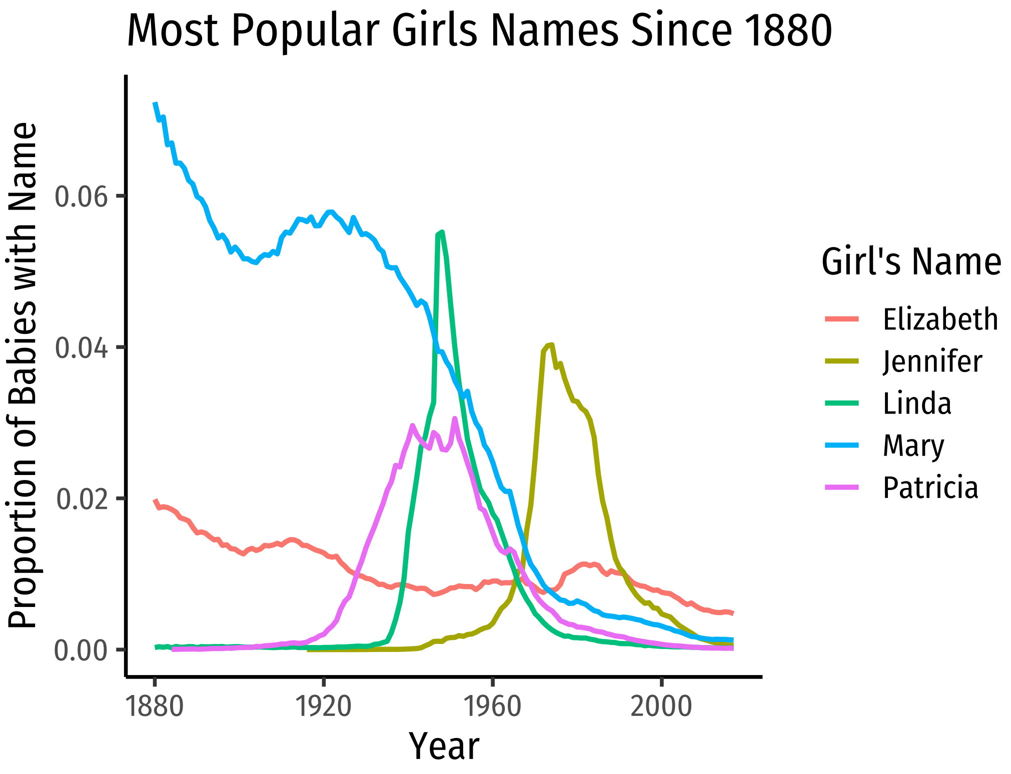
Part B

Now make two linegraphs of these 5 names over time, one for boys, and one for girls. [Hint: you'll first want to subset the data to use for your `data` layer in the plot. First `group_by(year)` and also make sure you only use the names you found in Part A. Try using the `%in%` command to do this.]

```
babynames %>%
  group_by(year) %>%
  filter(sex == "M",
         name %in% top_boys_names) %>%
  ggplot(data = .,
         aes(x = year,
             y = prop,
             color = name))+
  geom_line(size = 1)+
      labs(x = "Year",
         y = "Proportion of Babies with Name",
         title = "Most Popular Boys Names Since 1880",
         color = "Boy's Name",
         caption = "Source: SSA")+
    theme_classic(base_family = "Fira Sans Condensed", base_size=16)
```



```
babynames %>%
  group_by(year) %>%
  filter(sex == "F",
         name %in% top_girls_names) %>%
  ggplot(data = .,
         aes(x = year,
             y = prop,
             color = name))+
  geom_line(size = 1)+
    labs(x = "Year",
         y = "Proportion of Babies with Name",
         title = "Most Popular Girls Names Since 1880",
         color = "Girl's Name",
         caption = "Source: SSA")+
    theme_classic(base_family = "Fira Sans Condensed", base_size=16)
```


Most Popular Girls Names Since 1880

Source: SSA

Question 7

Bonus (hard!): What are the 10 most common "gender-neutral" names? [This is hard to define. For our purposes, let's define this as names where between 48% and 52% of the babies with the name are Male.]

There's a lot to this, so I'll break this up step by step and show you what happens at each major step.

We want to find the names where 48% to 52% of the babies with the name are male, as I defined in the footnote. First let's `mutate` a variable to figure out how many babies with a particular name are male.

To do this, we'll need to make a two variables to count the number of `male`s and `female`s of each name each year. We'll use the `ifelse()` function for each:

1. Make a `male` variable where, for each name in each year, if `sex=="M"`, then count the number of males (`n`) that year, otherwise set it equal to `0`.
2. Make a `female` variable where, for each name in each year, if `sex=="F"`, then count the number of females (`n`) that year, otherwise set it equal to `0`.

```
babynames %>%
  mutate(male = ifelse(sex == "M", n, 0),
         female = ifelse(sex == "F", n, 0))
```

```
## # A tibble: 1,924,665 x 7
##     year sex   name          n   prop  male female
##    <dbl> <chr> <chr>     <int>  <dbl> <dbl>  <dbl>
##  1  1880 F     Mary       7065 0.0724     0   7065
##  2  1880 F     Anna       2604 0.0267     0   2604
##  3  1880 F     Emma       2003 0.0205     0   2003
##  4  1880 F     Elizabeth  1939 0.0199     0   1939
##  5  1880 F     Minnie     1746 0.0179     0   1746
##  6  1880 F     Margaret   1578 0.0162     0   1578
##  7  1880 F     Ida        1472 0.0151     0   1472
##  8  1880 F     Alice      1414 0.0145     0   1414
##  9  1880 F     Bertha     1320 0.0135     0   1320
## 10  1880 F     Sarah      1288 0.0132     0   1288
## # ... with 1,924,655 more rows
```

Now with this variable, we want to count the total number of males and females with each name over the entire dataset. Let's first `group_by(name)` so we'll get one row for every name. We will `summarize()` and take the `sum` of our `male` and of our `female` variables.

```
babynames %>%
  mutate(male = ifelse(sex == "M", n, 0),
         female = ifelse(sex == "F", n, 0)) %>%
  group_by(name) %>%
    summarize(Male = sum(male),
              Female = sum(female))
```

```
## # A tibble: 97,310 x 3
##    name        Male Female
##    <chr>      <dbl>  <dbl>
##  1 Aaban        107      0
##  2 Aabha          0     35
##  3 Aabid         10      0
##  4 Aabir          5      0
##  5 Aabriella      0     32
##  6 Aada           0      5
##  7 Aadam        254      0
##  8 Aadan        130      0
##  9 Aadarsh      199      0
## 10 Aaden       4653      5
## # ... with 97,300 more rows
```

Now, we want to figure out what *fraction* of each name is Male or Female. It doesn't matter which we do here, I'll do Male. `mutate()` a new variable I'll call `perc_male` for the percent of the name being for Male babies. It takes the summed variables we made before, and takes the fraction that are Male, multiplying by 100 to get percents (which isn't necessary, but is easy to read).

```
babynames %>%
  mutate(male = ifelse(sex == "M", n, 0),
         female = ifelse(sex == "F", n, 0)) %>%
  group_by(name) %>%
    summarize(Male = sum(male),
              Female = sum(female))%>%
  mutate(perc_male = (Male/(Male+Female)*100))
```

```
## # A tibble: 97,310 x 4
##    name        Male Female perc_male
##    <chr>      <dbl>  <dbl>     <dbl>
##  1 Aaban        107      0       100
##  2 Aabha          0     35         0
##  3 Aabid         10      0       100
##  4 Aabir          5      0       100
##  5 Aabriella      0     32         0
##  6 Aada           0      5         0
##  7 Aadam        254      0       100
##  8 Aadan       130      0     100
##  9 Aadarsh     199      0     100
## 10 Aaden      4653      5      99.9
## # ... with 97,300 more rows
```

Right now, it's still in alphabetical order. We want to arrange it by `perc_male`, and more importantly, we want `perc_male` to be between 48 and 52, so let's `filter` accordingly:

```
babynames %>%
  mutate(male = ifelse(sex == "M", n, 0),
         female = ifelse(sex == "F", n, 0)) %>%
  group_by(name) %>%
    summarize(Male = sum(male),
              Female = sum(female))%>%
  mutate(perc_male = (Male/(Male+Female)*100)) %>%
  arrange(perc_male) %>%
  filter(perc_male > 48,
         perc_male < 52)
```

```
## # A tibble: 266 x 4
##    name        Male Female perc_male
##    <chr>      <dbl>  <dbl>     <dbl>
##  1 Demetrice   1623   1754      48.1
##  2 Shenan        25     27      48.1
##  3 Yael        3162   3414      48.1
##  4 Harlo        164    177      48.1
##  5 Daylyn       202    218      48.1
##  6 Oluwatosin   139    150      48.1
##  7 Chaning       13     14      48.1
##  8 Kirin        351    378      48.1
##  9 Odera         13     14      48.1
## 10 Jireh        644    693      48.2
## # ... with 256 more rows
```

This gives us a lot of names, all falling between 48% and 52% male. But we want the most popular names that are in this range. So let's finally `mutate` a new variable called `total` that simply adds the number of `Male` and `Female` babies with a name. Then let's `arrange` our results by `desc(total)` to get the largest first, and then `slice(1:10)` to get the top 10 only.

```
babynames %>%
  mutate(male = ifelse(sex == "M", n, 0),
         female = ifelse(sex == "F", n, 0)) %>%
  group_by(name) %>%
    summarize(Male = sum(male),
              Female = sum(female))%>%
  mutate(perc_male = (Male/(Male+Female)*100)) %>%
  arrange(perc_male) %>%
  filter(perc_male > 48,
         perc_male < 52) %>%
  mutate(total = Male+Female) %>%
  arrange(desc(total)) %>%
  slice(1:10)
```

```
## # A tibble: 10 x 5
##    name      Male Female perc_male total
##    <chr>    <dbl>  <dbl>     <dbl> <dbl>
##  1 Kerry    49596  48534      50.5 98130
##  2 Robbie   20863  22264      48.4 43127
##  3 Justice  17080  15782      52.0 32862
##  4 Blair    14470  14195      50.5 28665
##  5 Kris     13982  13490      50.9 27472
##  6 Elisha   13330  13599      49.5 26929
##  7 Unknown   9307   9416      49.7 18723
##  8 Mckinley  9389   8955      51.2 18344
##  9 Baby      6078   5871      50.9 11949
## 10 Santana   4651   4952      48.4  9603
```

Political and Economic Freedom Around the World

For the remaining questions, we'll look at the relationship between Economic Freedom and Political Freedom in countries around the world today. Our data for economic freedom comes from the Fraser Institute, and our data for political freedom comes from Freedom House.

Question 8

Download these two datasets that I've cleaned up a bit: [If you want a challenge, try downloading them from the websites and cleaning them up yourself!]

- `econ_freedom.csv`
- `pol_freedom.csv`

Below is a brief description of the variables I've put in each dataset:

Econ Freedom

Variable	Description
`year`	Year
`ISO`	Three-letter country code
`country`	Name of the country
`ef_index`	Total economic freedom index (0 - least to 100 - most)
`rank`	Rank of the country in terms of economic freedom
`continent`	Continent the country is in

Pol Freedom

Variable	Description
`country`	Name of the country
`C/T`	Whether the location is a country (C) or territory (T)
`year`	Year
`status`	Whether the location is Free (F), Partly Free (F) or Not Free (NF)
`fh_score`	Total political freedom index (0 - least to 100 - most)

Import and save them each as an object using `my_df_name <- read_csv("name_of_the_file.csv")`. I suggest one as `econ` and the other as `pol`, but it's up to you. Look at each object you've created.

```r
# import data with read_csv from readr

# note these file paths assume you have a folder called "data" in your working directory

# if you used an R project and did just that (or downloaded my R Project from the website)
# then you already have this done

econ<-read_csv("data/econ_freedom.csv")
```

```
## Rows: 4050 Columns: 6

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): ISO, country, continent
## dbl (3): year, ef_index, rank

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
pol<-read_csv("data/pol_freedom.csv")
```

```
## Rows: 1885 Columns: 5

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): country, C/T, status
## dbl (2): year, fh_score

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# look at each dataframe
econ
```

```
## # A tibble: 4,050 x 6
##     year ISO   country      ef_index  rank continent
##    <dbl> <chr> <chr>           <dbl> <dbl> <chr>
##  1  2018 ALB   Albania          7.8     26 Europe
##  2  2018 DZA   Algeria          4.97   157 Africa
##  3  2018 AGO   Angola           4.75   159 Africa
##  4  2018 ARG   Argentina        5.78   144 Americas
##  5  2018 ARM   Armenia          7.92    18 Asia
##  6  2018 AUS   Australia        8.23     5 Oceania
##  7  2018 AUT   Austria          7.8     26 Europe
##  8  2018 AZE   Azerbaijan       6.37   112 Asia
##  9  2018 BHS   Bahamas, The     7.62    39 Americas
## 10  2018 BHR   Bahrain          7.16    70 Asia
## # ... with 4,040 more rows
```

```r
pol
```

```
## # A tibble: 1,885 x 5
##    country             `C/T`  year status fh_score
##    <chr>               <chr> <dbl> <chr>     <dbl>
##  1 Abkhazia            t      2021 PF           40
##  2 Afghanistan         c      2021 NF           27
##   3 Albania             c       2021 PF          66
##   4 Algeria             c       2021 NF          32
##   5 Andorra             c       2021 F           93
##   6 Angola              c       2021 NF          31
##   7 Antigua and Barbuda c       2021 F           85
##   8 Argentina           c       2021 F           84
##   9 Armenia             c       2021 PF          55
##  10 Australia           c       2021 F           97
## # ... with 1,875 more rows
```

Question 9

Now let's join them together so that we can have a single dataset to work with. You can learn more about this in the 1.4 slides. Since both datasets have both `country` and `year` (spelled exactly the same in both!), we can use these two variables as a `key` to combine observations. Run the following code (substituting whatever you want to name your objects):

```
freedom <- left_join(econ, pol, by=c("country", "year")
```

Take a look at `freedom` to make sure it appears to have worked.

```
freedom <- left_join(econ, pol, by=c("country", "year"))
freedom
```

```
## # A tibble: 4,050 x 9
##     year ISO   country      ef_index  rank continent `C/T` status fh_score
##    <dbl> <chr> <chr>           <dbl> <dbl> <chr>     <chr> <chr>     <dbl>
##  1  2018 ALB   Albania          7.8     26 Europe    c     PF           68
##  2  2018 DZA   Algeria          4.97   157 Africa    c     NF           35
##  3  2018 AGO   Angola           4.75   159 Africa    c     NF           26
##  4  2018 ARG   Argentina        5.78   144 Americas  c     F            83
##  5  2018 ARM   Armenia          7.92    18 Asia      c     PF           45
##  6  2018 AUS   Australia        8.23     5 Oceania   c     F            98
##  7  2018 AUT   Austria          7.8     26 Europe    c     F            94
##  8  2018 AZE   Azerbaijan       6.37   112 Asia      c     NF           12
##  9  2018 BHS   Bahamas, The     7.62    39 Americas  <NA>  <NA>         NA
## 10  2018 BHR   Bahrain          7.16    70 Asia      c     NF           12
## # ... with 4,040 more rows
```

Question 11

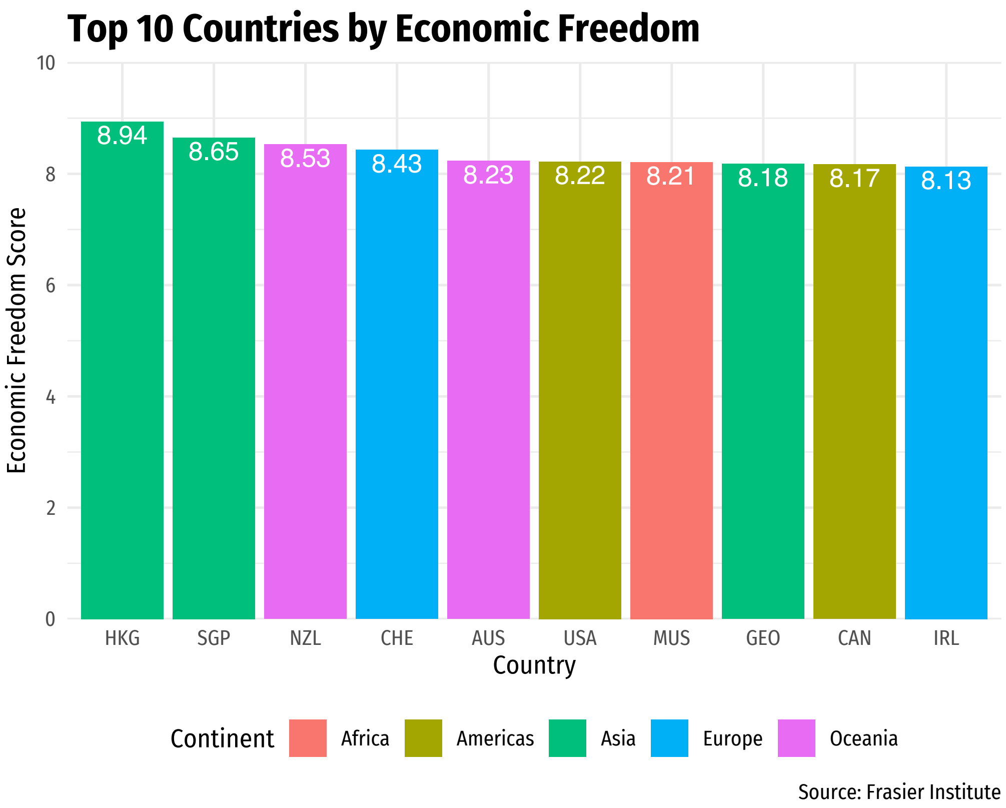
Part A

Make a barplot of the 10 countries with the highest Economic Freedom index score in 2018. You may want to find this first and save it as an object for your plot's `data` layer. Use `geom_col()` since we will map `ef_index` to `y`. If you want to order the bars, set `x = fct_reorder(ISO, desc(ef_index))` to reorder `ISO` (or `country`, if you prefer) by EF score in descending order.

```
# grab the top 10 countries by ef in 2018
ef_10<-freedom %>%
  filter(year == 2018) %>%
  arrange(desc(ef_index)) %>%
  slice(1:10)

# look at it just to check
ef_10
```

```
## # A tibble: 10 x 9
##     year ISO   country              ef_index  rank continent `C/T` status fh_score
##    <dbl> <chr> <chr>                   <dbl> <dbl> <chr>     <chr> <chr>     <dbl>
##  1  2018 HKG   Hong Kong SAR, China     8.94     1 Asia      t     PF           59
##  2  2018 SGP   Singapore                8.65     2 Asia      c     PF           52
##  3  2018 NZL   New Zealand              8.53     3 Oceania   c     F            98
##  4  2018 CHE   Switzerland              8.43     4 Europe    c     F            96
##  5  2018 AUS   Australia                8.23     5 Oceania   c     F            98
##  6  2018 USA   United States            8.22     6 Americas  c     F            86
##  7  2018 MUS   Mauritius                8.21     7 Africa    c     F            89
##  8  2018 GEO   Georgia                  8.18     8 Asia      c     PF           64
##  9  2018 CAN   Canada                   8.17     9 Americas  c     F            99
## 10  2018 IRL   Ireland                  8.13    10 Europe    c     F            96
```

```
# now plot it
ggplot(data = ef_10)+
  aes(x = fct_reorder(ISO, desc(ef_index)), # reorder ISO by ef in order
      y = ef_index)+
  geom_col(aes(fill = continent))+ # coloring is optional
  # above is sufficient, now let's just make it prettier
  geom_text(aes(label = ef_index), # add the score onto the bar
            vjust = 1.2, # adjust it vertically
            color = "white"
            )+
  scale_y_continuous(breaks = seq(0,10,2),
                     limits = c(0,10),
                     expand = c(0,0)
                     )+
  labs(x = "Country",
       y = "Economic Freedom Score",
       title = "Top 10 Countries by Economic Freedom",
       caption = "Source: Frasier Institute",
       fill = "Continent")+
  theme_minimal(base_family = "Fira Sans Condensed")+
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold", size = rel(1.5))
        )
```


Top 10 Countries by Economic Freedom

Source: Frasier Institute

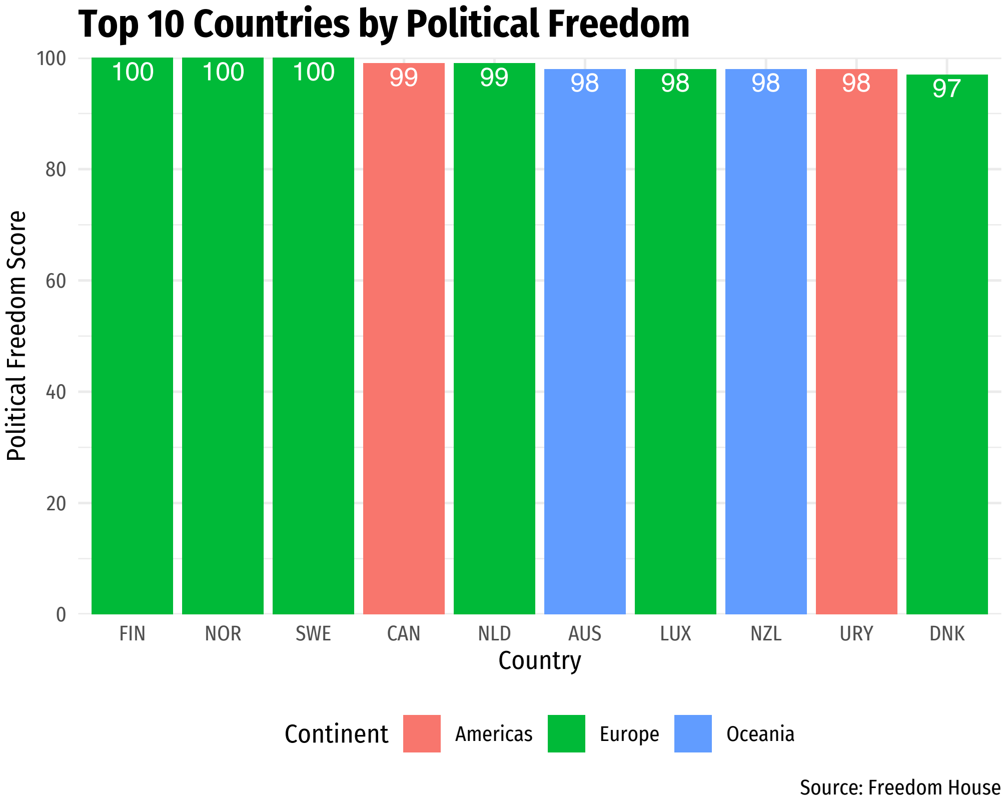
Part B

Make a barplot of the 10 countries with the highest Freedom House index score in 2018, similar to what you did for Part A.

```
# grab the top 10 countries by fh in 2018
pf_10<-freedom %>%
  filter(year == 2018) %>%
  arrange(desc(fh_score)) %>%
  slice(1:10)

# look at it just to check
pf_10
```

```
## # A tibble: 10 x 9
##     year ISO   country     ef_index  rank continent `C/T` status fh_score
##    <dbl> <chr> <chr>          <dbl> <dbl> <chr>     <chr> <chr>     <dbl>
##  1  2018 FIN   Finland         7.76    29 Europe    c     F           100
##  2  2018 NOR   Norway          7.6     43 Europe    c     F           100
##  3  2018 SWE   Sweden          7.58    46 Europe    c     F           100
##  4  2018 CAN   Canada          8.17     9 Americas  c     F            99
##  5  2018 NLD   Netherlands     7.82    24 Europe    c     F            99
##  6  2018 AUS   Australia       8.23     5 Oceania   c     F            98
##  7  2018 LUX   Luxembourg      7.75    31 Europe    c     F            98
##  8  2018 NZL   New Zealand     8.53     3 Oceania   c     F            98
##  9  2018 URY   Uruguay         7.25    66 Americas  c     F            98
## 10  2018 DNK   Denmark         8.1     11 Europe    c     F            97
```

```r
# now plot it
ggplot(data = pf_10)+
  aes(x = fct_reorder(ISO, desc(fh_score)),
      y = fh_score)+
  geom_col(aes(fill = continent))+ # coloring is optional
  # above is sufficient, now let's just make it prettier
  geom_text(aes(label = fh_score), # add the score onto the bar
            vjust = 1.2, # adjust it vertically
            color = "white")+
  scale_y_continuous(breaks = seq(0,100,20),
                     limits = c(0,100),
                     expand = c(0,0))+
  labs(x = "Country",
       y = "Political Freedom Score",
       title = "Top 10 Countries by Political Freedom",
       caption = "Source: Freedom House",
       fill = "Continent")+
  theme_minimal(base_family = "Fira Sans Condensed")+
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold", size = rel(1.5))
        )
```

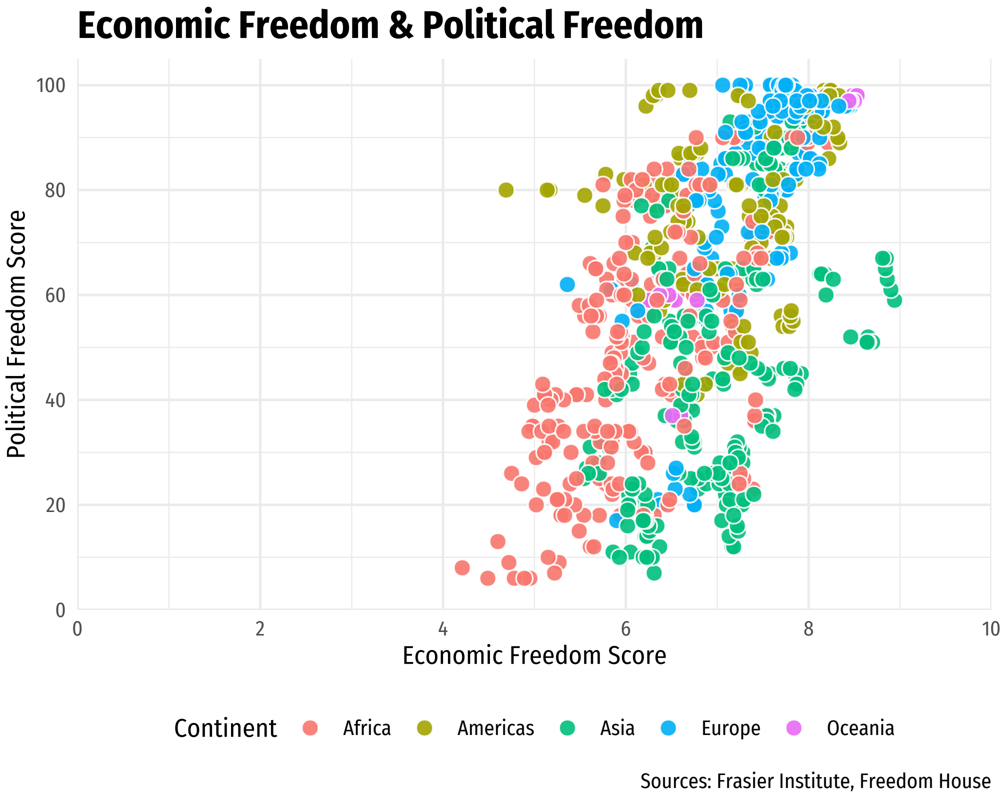


Question 11

Now make a scatterplot of Political freedom (`fh_score` as `y`) on Economic Freedom (`ef_index` as `x`) and color by `continent`.

```
ggplot(data = freedom)+
  aes(x = ef_index,
      y = fh_score)+
  # doing just geom_point() is fine, but since there's a lot of overlap, here are some things I like to
  geom_point(aes(fill = continent), # fill the points with color by continent
             alpha = 0.9, # make points slightly transparent
             color = "white", # outline the points with a white border
             pch = 21, # this shape has an outline and a fill color
             size = 3)+
  scale_x_continuous(breaks = seq(0,10,2),
                     limits = c(0,10),
                     expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,100,20),
                     limits = c(0,105),
                     expand = c(0,0))+
  labs(x = "Economic Freedom Score",
       y = "Political Freedom Score",
       caption = "Sources: Frasier Institute, Freedom House",
       title = "Economic Freedom & Political Freedom",
       fill = "Continent")+
  theme_minimal(base_family = "Fira Sans Condensed")+
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold", size = rel(1.5))
        )
```

```
## Warning: Removed 3166 rows containing missing values (geom_point).
```

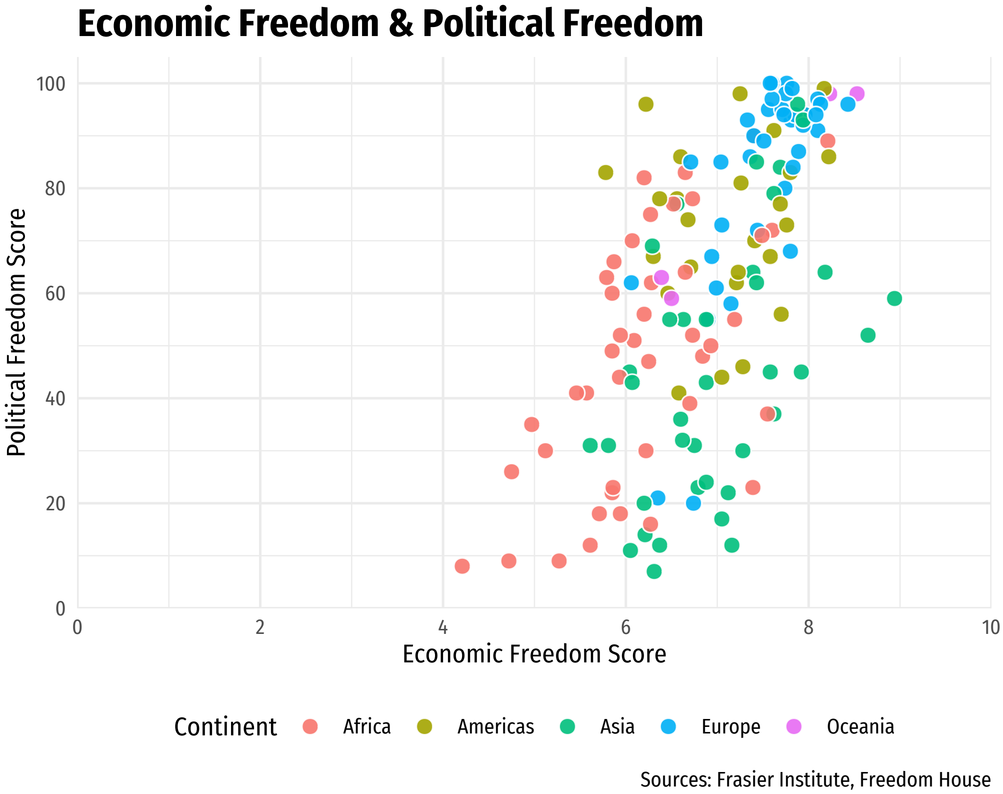

Economic Freedom & Political Freedom

Sources: Frasier Institute, Freedom House

Note, I meant to ask you to look at one year only, e.g. 2018. We would just have to filter first:

```
# save as p
p <- freedom %>%
  filter(year == "2018") %>%
ggplot(data = .)+
  aes(x = ef_index,
      y = fh_score)+
  geom_point(aes(fill = continent),
             alpha = 0.9,
             color = "white",
             pch = 21,
             size = 3)+
  scale_x_continuous(breaks = seq(0,10,2),
                     limits = c(0,10),
                     expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,100,20),
                     limits = c(0,105),
                     expand = c(0,0))+
  labs(x = "Economic Freedom Score",
       y = "Political Freedom Score",
       caption = "Sources: Frasier Institute, Freedom House",
       title = "Economic Freedom & Political Freedom",
       fill = "Continent")+
  theme_minimal(base_family = "Fira Sans Condensed")+
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold", size = rel(1.5))
        )

# look at it
p
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```



Question 12

Save your plot from Question 11 as an object, and add a new layer where we will highlight a few countries. Pick a few countries (I suggest using the `ISO` code) and create a new object `filter`ing the data to only include these countries (again the `%in%` command will be most helpful here).

Additionally, *install* and *load* a package called `"ggrepel"`, which will adjust labels so they do not overlap on a plot.

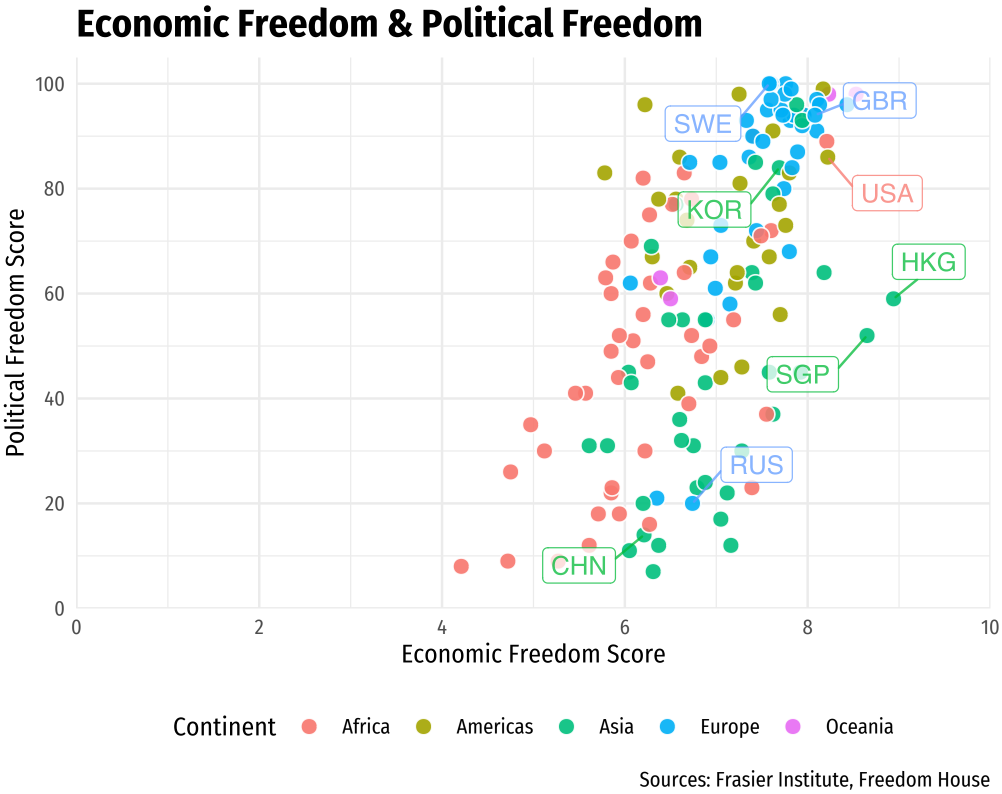
Then, add the following layer to your plot:

```
geom_label_repel(data = countries, # or whatever object name you created
                     aes(x = ef_index,
                         y = fh_score,
                         label = ISO, # show ISO as label (you could do country instead)
                         color = continent),
                     alpha = 0.5, # make it a bit transparent
                     box.padding = 0.75, # control how far labels are from points
                     show.legend = F) # don't want this to add to the legend
```

This should highlight these countries on your plot.

```
library(ggrepel)

# pick some countries
some_countries <- freedom %>%
  filter(year==2018,
         country %in% c("United States",
                        "United Kingdom",
                        "Sweden",
                        "China",
                        "Singapore",
                        "Russian Federation",
                        "Korea, Rep.",
                        "Hong Kong SAR, China"))
# add layer
p + geom_label_repel(data = some_countries, # or whatever object name you created
                     aes(x = ef_index,
                         y = fh_score,
                         label = ISO, # show ISO as label (you could do country instead)
                         color = continent),
                     alpha = 0.75, # make it a bit transparent
                     box.padding = 0.75, # control how far labels are from points
                     show.legend = F) # don't want this to add to the legend
```

```
## Warning: Removed 13 rows containing missing values (geom_point).
```

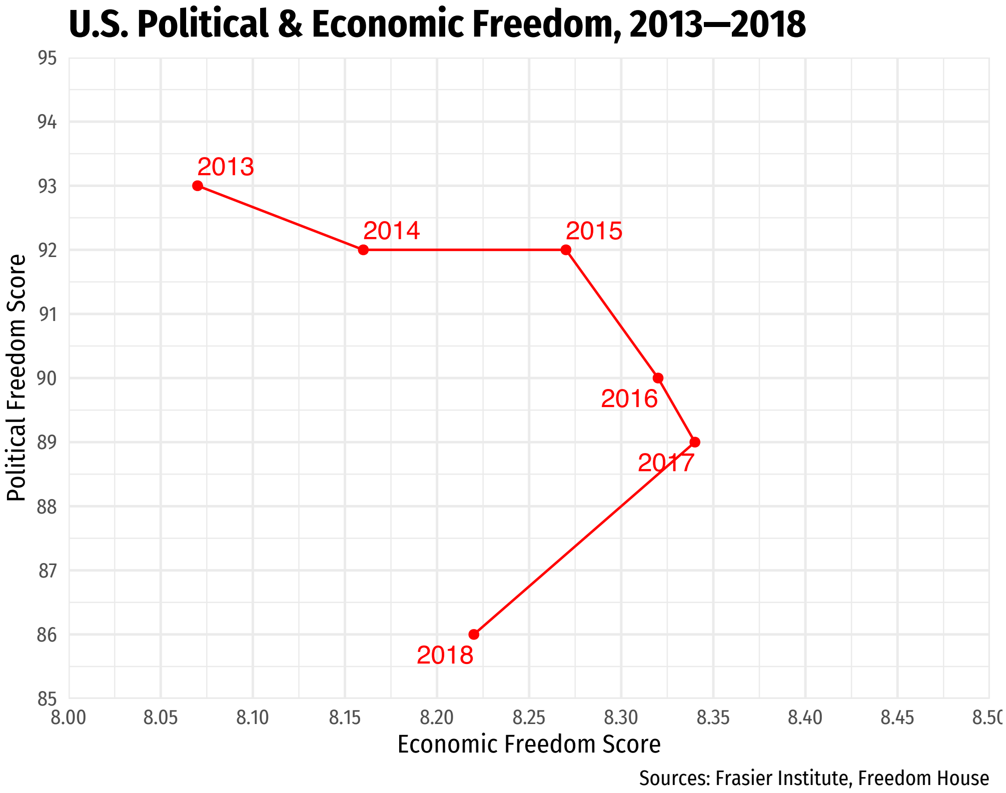


Question 13

Let's just look only at the United States and see how it has fared in both measures of freedom over time. `filter()` the data to look only at `ISO == "USA"`. Use both a `geom_point()` layer and a `geom_path()` layer, which will connect the dots over time. Let's also see this by labeling the years with an additional layer `geom_text_repel(aes(label = year))`.

```
# save plot as us
us<-freedom %>%
  filter(ISO == "USA") %>%
ggplot(data = .)+
  aes(x = ef_index,
      y = fh_score)+
  geom_point(color = "red")+
  geom_path(color = "red")+
  geom_text_repel(aes(label = year),
                  color = "red")+
  scale_x_continuous(breaks = seq(8,8.5,0.05),
                     limits = c(8,8.5),
                     expand = c(0,0))+
  scale_y_continuous(breaks = seq(85,95,1),
                     limits = c(85,95),
                     expand = c(0,0))+
  labs(x = "Economic Freedom Score",
       y = "Political Freedom Score",
       caption = "Sources: Frasier Institute, Freedom House",
       title = "U.S. Political & Economic Freedom, 2013-2018",
       fill = "Continent")+
  theme_minimal(base_family = "Fira Sans Condensed")+
  theme(legend.position = "bottom",
        plot.title = element_text(face = "bold", size = rel(1.5))
        )
# look at it
us
```

```
## Warning: Removed 19 rows containing missing values (geom_point).

## Warning: Removed 19 row(s) containing missing values (geom_path).

## Warning: Removed 19 rows containing missing values (geom_text_repel).
```

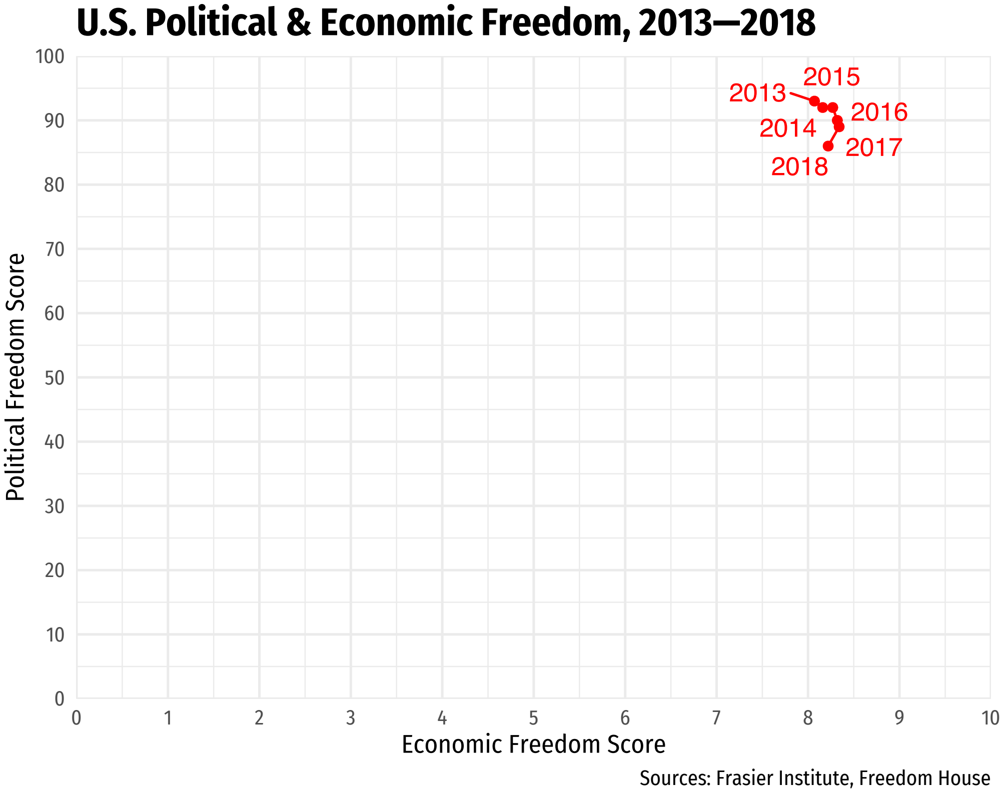


Note that the way I zoomed in on the scales, these look like pretty dramatic changes!

If we maintain the full perspective, the change appears minor. Be very careful how you present your analysis!

```
us+
  scale_x_continuous(breaks = seq(0,10,1),
                     limits = c(0,10),
                     expand = c(0,0)
                     )+
  scale_y_continuous(breaks = seq(0,100,10),
                     limits = c(0,100),
                     expand = c(0,0)
                     )
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.

## Warning: Removed 19 rows containing missing values (geom_point).

## Warning: Removed 19 row(s) containing missing values (geom_path).

## Warning: Removed 19 rows containing missing values (geom_text_repel).
```


U.S. Political & Economic Freedom, 2013—2018
Political Freedom Score
Economic Freedom Score
Sources: Frasier Institute, Freedom House