Assignment 2

Josh Ellis

3/5/2022

Packages used throughout this assignment:

```
library(tidyverse)
library(reshape2)
library(knitr)
library(babynames)
```

1. The data set `tips` contains tip amounts for different party sizes as well as total bill amounts per payment. We can get the data from the reshape2 package as follows:

```
tips.dat <- tips
```

a. [2 pts] Compute the tip rate, dividing tip by total bill, and create a new column called `tip.rate` in the dataframe `tips.dat`. Demonstrate your results by showing the head of `tips.dat`.

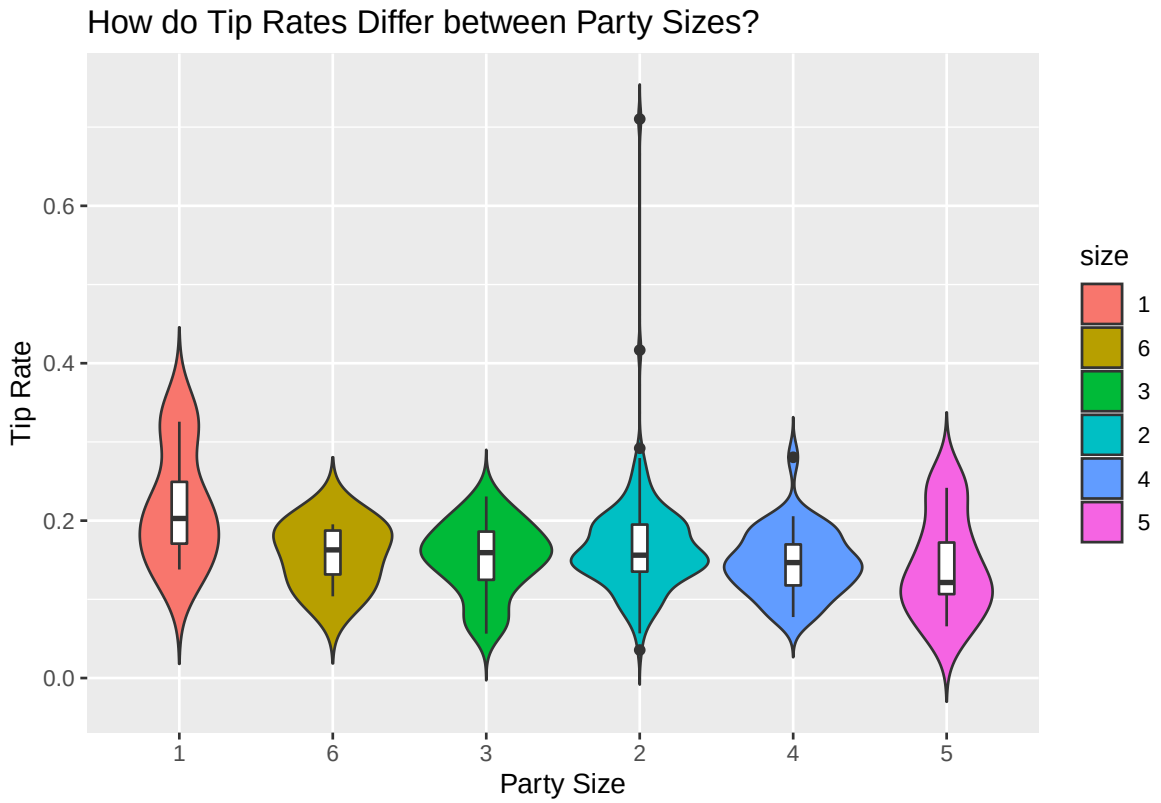
```
tips.dat$tip.rate <- round(tips.dat$tip / tips.dat$total_bill, digits=4)
head(tips.dat)
```

```
##   total_bill  tip    sex smoker day   time size tip.rate
## 1      16.99 1.01 Female     No Sun Dinner    2   0.0594
## 2      10.34 1.66   Male     No Sun Dinner    3   0.1605
## 3      21.01 3.50   Male     No Sun Dinner    3   0.1666
## 4      23.68 3.31   Male     No Sun Dinner    2   0.1398
## 5      24.59 3.61 Female     No Sun Dinner    4   0.1468
## 6      25.29 4.71   Male     No Sun Dinner    4   0.1862
```

b. [3 pts] Draw a side-by-side violin plot of the tip rate for each party size. Order the party sizes by the median tip rate. Provide your code as well as your plot. Which party size is responsible for the highest median tip rate?

```
# Calculate the Median tip rate for each table size and sort values in desc Order
median_size <- tips.dat %>%
  group_by(as.factor(size)) %>%
  summarise(median(tip.rate)) %>%
  rename(
    median_tip_rate = `median(tip.rate)`,
    size = `as.factor(size)`) %>%
```

```
  arrange(desc(median_tip_rate)) %>%
  select(size)

# Assign factor levels to the size variable based on median tip rate
tips.dat$size <- factor(
  tips.dat$size,
  levels = as.character(median_size$size))


p <- ggplot(data=tips.dat, aes(x=size, y=tip.rate)) +
  geom_violin(trim=FALSE, aes(fill=size)) +
  geom_boxplot(width=0.1) +
  labs(
    x='Party Size',
    y='Tip Rate',
    title='How do Tip Rates Differ between Party Sizes?')
p
```

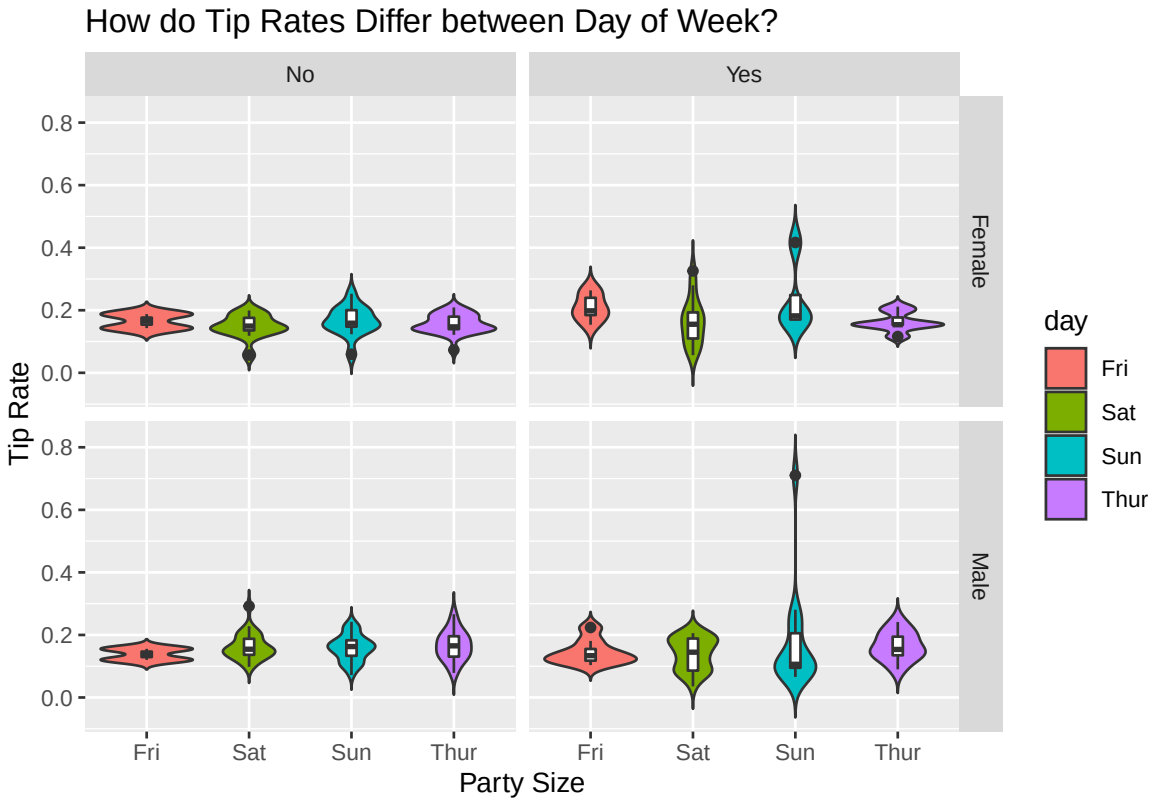


From this graph, we can see that Customers with table size of 1 tip the highest rates

c. [2 pts] Generate a similar plot to the one you created in question (b) for each day (instead of party size) and facet by sex and smoker. Is the shape of the violin plot similar for each faceted condition?

```
p <- ggplot(data=tips.dat, aes(x=day, y=tip.rate)) +
  geom_violin(trim=FALSE, aes(fill=day)) +
  geom_boxplot(width=0.1) +
  labs(
    x='Party Size',
```

```
    y='Tip Rate',
    title='How do Tip Rates Differ between Day of Week?') +
  facet_grid(sex~smoker)
p
```



It appears that for the most part, there is not any drastic differences in the violin plots for each facet. However, it does seem that non-smoking Male and Female customers tip in similar patters, and smoking male and females customers tip in similar patters, but customers who smoke tip differently than customers who do not smoke.

2. We can generate an $n \times k$ matrix M and a vector V of length k for some specific values of n and k as follows:

```
set.seed(321)
n = 9
k = 5
V = sample(50, size = k, replace = TRUE)
M = matrix(rnorm(n * k), ncol = k)
```

a. [3 pts] Now, carefully review the following for-loop. Rewrite the code so that you perform the same job without a loop.

```
X = M
for(i in 1:n) {
  X[i, ] = round(M[i, ] / V, digits = 4)
}
```

```
# Code with no Loop
X_no_loop = round(t(t(M)/V), digits=4)
```

b. [2 pts] Now do the same experiment for $n = 900$ and $k = 500$. Which runs faster, your code or the for-loop? Demonstrate this using the function `system.time()`.

```
set.seed(321)
n = 900
k = 500
V = sample(50, size = k, replace = TRUE)
M = matrix(rnorm(n * k), ncol = k)

X = M
time_loop <- system.time(
  for(i in 1:n) {
    X[i, ] = round(M[i, ] / V, digits = 4)
  }
)

time_no_loop <- system.time(round(t(t(M)/V), digits=4))

time_loop
```

```
##    user  system elapsed
##    0.03    0.00    0.03
```

```
time_no_loop
```

```
##    user  system elapsed
##    0.04    0.00    0.03
```

On my system for this matrix size, there doesn't appear to be any difference in performance between the for-loop and the vectorized operation.

3. We want to generate a plot of US arrest data (USArrests). Please provide the detailed codes to answer the following questions.

a. [3 pts] Obtain USA state boundary coordinates data for generating a USA map using function `map_data()` and store the data in `mdat`. Display the first few rows of data from `mdat`, noticing that there is a column called `order` that contains the true order of the coordinates.

```
mdat <- map_data('state')
head(mdat)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

b. [3 pts] You will find USA crime data in the data frame called `USArrests`. Standardize the crime rates and create a new column called `state` so that all state names are in lower case. Store this new data in an object called `arrest` and report the first few rows of `arrest`.

```
arrest <- USArrests
arrest[1:4] <- lapply(arrest[1:4], function(x) scale(x))
arrest$state <- tolower(rownames(arrest))
head(arrest)
```

```
##                Murder   Assault   UrbanPop         Rape      state
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473    alabama
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941     alaska
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388    arizona
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602   arkansas
## California 0.27826823 1.2628144  1.7589234  2.067820292 california
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207   colorado
```

c. [3 pts] Merge the two data sets `mdat` and `arrest` by state name. Note: merging will change the order of the coordinates data. So, order the data back to the original order and store the merged-ordered data in `odat`. Report the first few rows of data from `odat`.

```
odat <- mdat %>%
  inner_join(
  arrest,
  by = c('region' = 'state')) %>%
  arrange(order)

head(odat)
```

```
##        long      lat group order  region subregion   Murder   Assault
## 1 -87.46201 30.38968     1     1 alabama      <NA> 1.242564 0.7828393
## 2 -87.48493 30.37249     1     2 alabama      <NA> 1.242564 0.7828393
## 3 -87.52503 30.37249     1     3 alabama      <NA> 1.242564 0.7828393
## 4 -87.53076 30.33239     1     4 alabama      <NA> 1.242564 0.7828393
## 5 -87.57087 30.32665     1     5 alabama      <NA> 1.242564 0.7828393
## 6 -87.58806 30.32665     1     6 alabama      <NA> 1.242564 0.7828393
##     UrbanPop         Rape
## 1 -0.5209066 -0.003416473
## 2 -0.5209066 -0.003416473
## 3 -0.5209066 -0.003416473
## 4 -0.5209066 -0.003416473
## 5 -0.5209066 -0.003416473
## 6 -0.5209066 -0.003416473
```

d. [2 pts] All the columns of `odat` are not necessary for our analysis. So, obtain a subset by selecting only the columns `long`, `lat`, `group`, `region`, `Murder`, `Assault`, `UrbanPop`, and `Rape`. Store the data in `sdat` and report the first few rows.

```
sdat <- odat %>% select(-c('order', 'subregion'))
head(sdat)
```

```
##        long      lat group  region   Murder   Assault   UrbanPop         Rape
## 1 -87.46201 30.38968     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
```

```
## 2 -87.48493 30.37249     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 3 -87.52503 30.37249     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 4 -87.53076 30.33239     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 5 -87.57087 30.32665     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 6 -87.58806 30.32665     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
```

e. [3 pts] Melt the data frame `sdat` with id variables `long`, `lat`, `group`, `region`. Store the molten data in `msdat` and report the first few rows of data. Use two ways to do this (`reshape2` and `tidyr`).

```
msdat = tidyr::pivot_longer(sdat, cols = -c('long', 'lat', 'group', 'region'))
msdat = reshape2::melt(sdat, id = c('long', 'lat', 'group', 'region'))
```

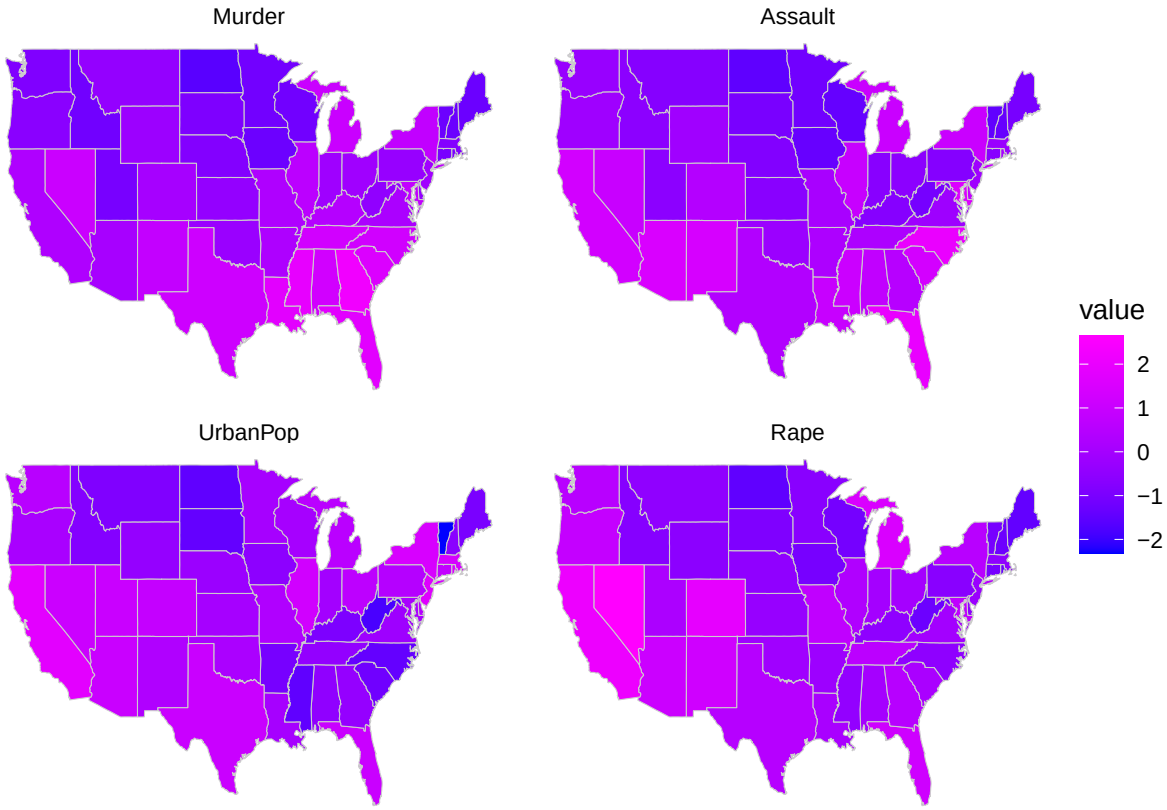
```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
head(msdat)
```

```
##        long      lat group  region variable    value
## 1 -87.46201 30.38968     1 alabama   Murder 1.242564
## 2 -87.48493 30.37249     1 alabama   Murder 1.242564
## 3 -87.52503 30.37249     1 alabama   Murder 1.242564
## 4 -87.53076 30.33239     1 alabama   Murder 1.242564
## 5 -87.57087 30.32665     1 alabama   Murder 1.242564
## 6 -87.58806 30.32665     1 alabama   Murder 1.242564
```

f. [2 pts] The molten data frame `msdat` is now ready to be plotted. Let's use `msdat` from the `reshape2` output. Create a plot showing the USA state map, fill the color by `value`, and `facet_wrap` with `variable`. Please don't add any legend and make sure that facetting labels are identified so that we can compare the facetted plots.

```
ggplot(data=msdat, aes(x=long, y=lat, group=group, fill=value)) +
  geom_polygon(color='grey80', size=0.2) +
  theme_void() +
  facet_wrap(~variable) +
  scale_fill_continuous(low='blue', high='magenta')
```



g. Now examine the plot you have generated in question (f) and answer the following questions based on what you see in the plot.

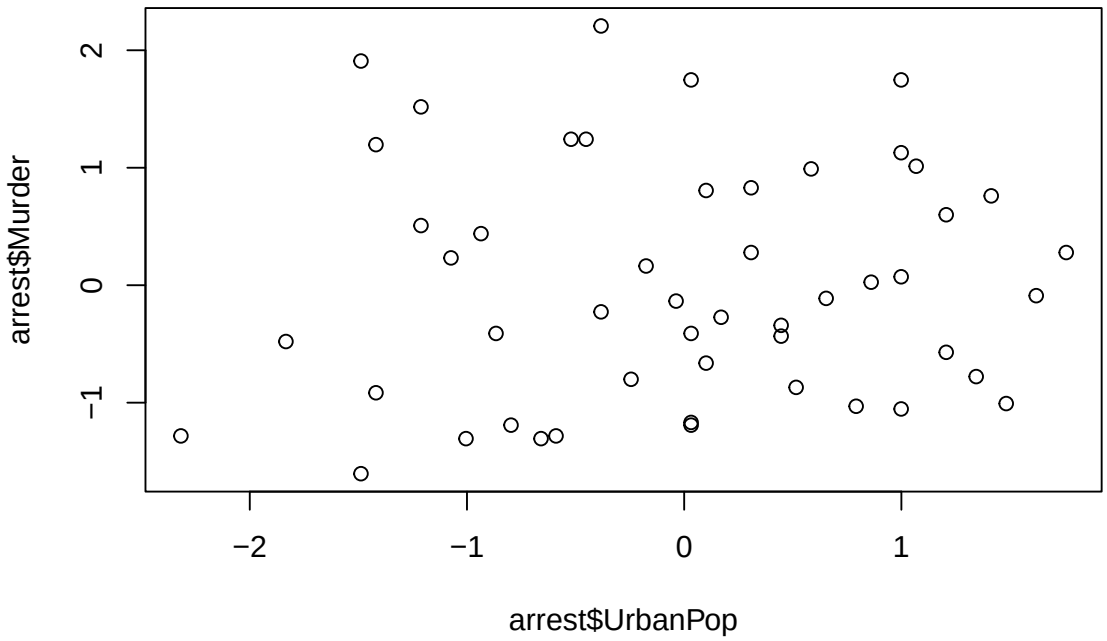
i. [1 pt] For each crime, name two states with its highest rate.

Murder: Georgia and Mississippi; **Assault:** North Carolina and Florida; **UrbanPop:** California and New Jersey; **Rape:** Nevada and California;

ii. [1 pt] Do you think a larger urban population is indicative of a higher murder rate? Why or why not?

My intuition would say yes, because areas with higher urban populations are more likely to have organized crime groups, such as gang violence, as opposed to more rural areas

```
# Testing my Intuition
temp_data <- msdat %>% distinct(region, variable, value)
plot(arrest$UrbanPop, arrest$Murder)
```




```
cor.test(arrest$UrbanPop, arrest$Murder)
```

```
##
##  Pearson's product-moment correlation
##
## data:  arrest$UrbanPop and arrest$Murder
## t = 0.48318, df = 48, p-value = 0.6312
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2128979  0.3413107
## sample estimates:
##       cor
## 0.06957262
```

my intuition is wrong, after plotting urban population and murder rates, there does not seem to be any significant relationship between the two variables. Correlation of only 0.0696

h. [1 pt] In question (b) we standardized the crime rates. Why do you think we did this? Explain what would happen if we did not standardize the data.

Scaling the features allow each feature value to have a zero-mean, with the data being scaled and distributed around this zero-mean. This allows each feature to be comparable on the same scale. In USArrests, the Assault variable is significantly higher than any of the other crime features. Because of this, Assault can not be compared to, for instance, Murder, since they have different scales.

i. [1 pt] In question (c) we ordered the data after merging. Why do you think we had to do this? Explain what would happen if we did not.

If there order was not correct, then we would not be able to properly map the coordinates of each observation in the dataset. The order is required to be in sequential order to be able to properly map the coordinates on a graph. Without this sequential order, the coordinate points would not connect in the correct order, therefore the result of a graph would not be in the shape of the US States.

4. Life expectancy data for four countries can be obtained from the world bank database found at github. It contains life expectancy in years for different genders. Now answer the following questions.

a. [1 pt] Read the data from the above link and display the first few rows of data.

```
life_expectancy = read_csv(
  'http://mamajumder.github.io/data-science/data/life-expectancy.csv')
```

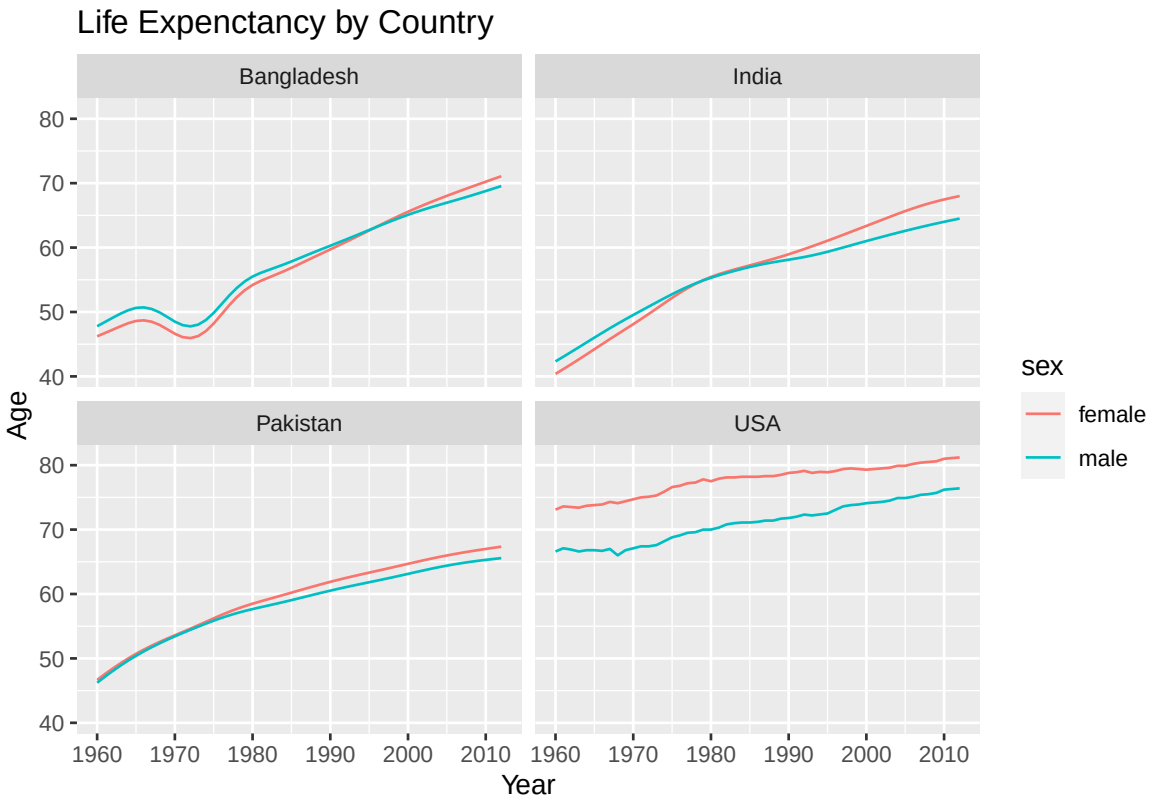
```
## Rows: 106 Columns: 6

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): sex
## dbl (5): year, Bangladesh, India, Pakistan, USA

##
## 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.
```

```
head(life_expectancy)
```

```
## # A tibble: 6 x 6
##    year sex    Bangladesh India Pakistan   USA
##   <dbl> <chr>       <dbl> <dbl>    <dbl> <dbl>
## 1  1960 female       46.2  40.4     46.7  73.1
## 2  1960 male         47.8  42.3     46.2  66.6
## 3  1961 female       46.7  41.1     47.6  73.6
## 4  1961 male         48.4  43.1     47.2  67.1
## 5  1962 female       47.3  41.9     48.4  73.5
## 6  1962 male         49.1  43.8     48.0  66.9
```

b. [2 pts] Generate a plot showing trend lines of life expectancy by year. Color them by sex and facet by country. Include your code with the plot.

```
life_expectancy %>%
  pivot_longer(
    cols = -c('year', 'sex'),
    names_to = 'country',
    values_to = 'life_expectancy') %>%
  ggplot(aes(x=year, y=life_expectancy, color=sex)) +
  geom_line() +
  facet_wrap(~country) +
  labs(title='Life Expenctancy by Country', x='Year', y='Age')
```



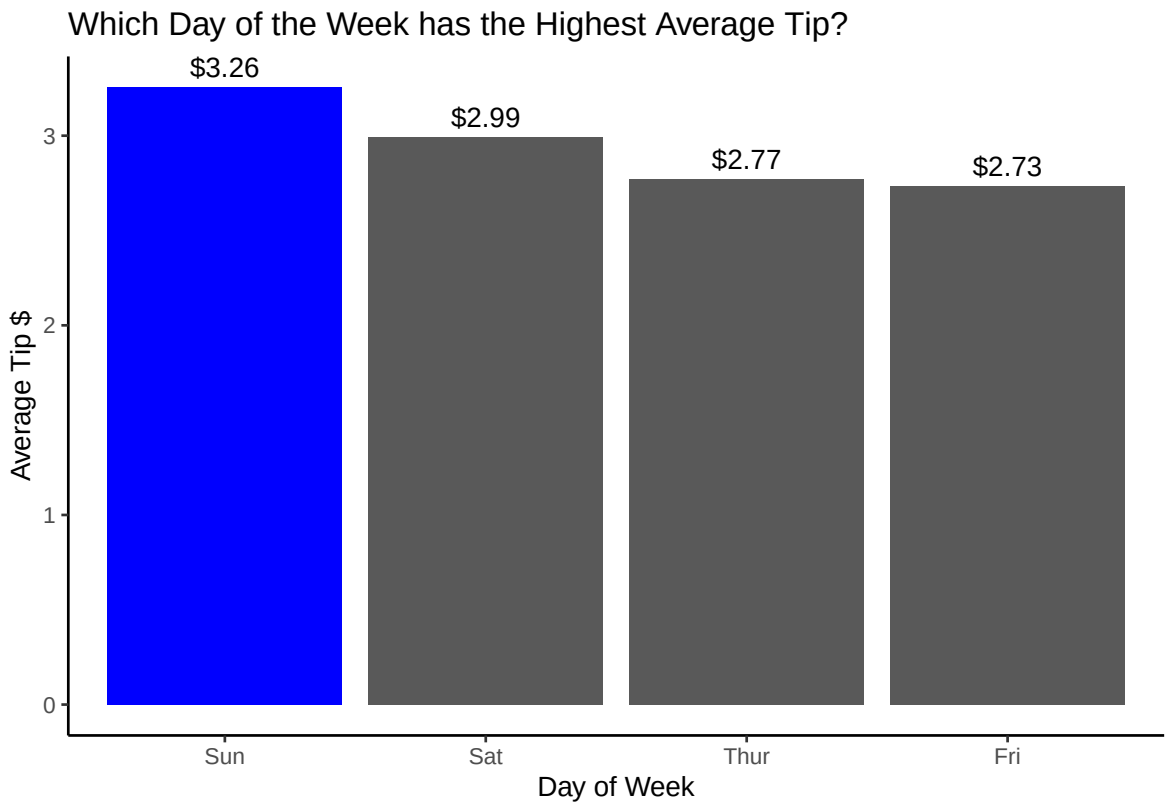
c. [1 pt] Explain what interesting features you noticed in the plot you made in question (b).

(1) The US has a large difference in life expenctancy between male and female, much more than what is seen in other countries. (2) Pakistan life expectancy is platueing much sooner than any of the other countires. (3) The US has consistantly had a higher life expectancy for both male and females compared to other countries.

5. For the following questions please use the data frame `tips`.

a. [2 pts] Create a bar chart that shows the average tip by day.

```
df = tips

avg_by_day <- df %>%
group_by(day) %>%
summarise(avg_tip = mean(tip))

ggplot(data = avg_by_day, aes(x=reorder(day, -avg_tip), y = avg_tip)) +
  geom_col() +
  geom_col(data=avg_by_day[which.max(avg_by_day$avg_tip), ], fill='blue') +
  geom_text(aes(label = paste0('$', round(avg_tip, 2)), vjust = -0.5)) +
  labs(
    title = 'Which Day of the Week has the Highest Average Tip?',
    x = 'Day of Week',
    y = 'Average Tip $') +
  theme_classic()
```



b. [2 pts] Compute the average tip, total tip, and average size grouped by smoker and day. i.e., For each combination of smoker and day you should have a row of these summaries. Report these results in a nice table.

```
tips_smoker_day = df %>%
  group_by(smoker, day) %>%
  summarise(
    avg_tip = mean(tip),
    total_tip = sum(tip),
    avg_size = mean(size)
  )
```

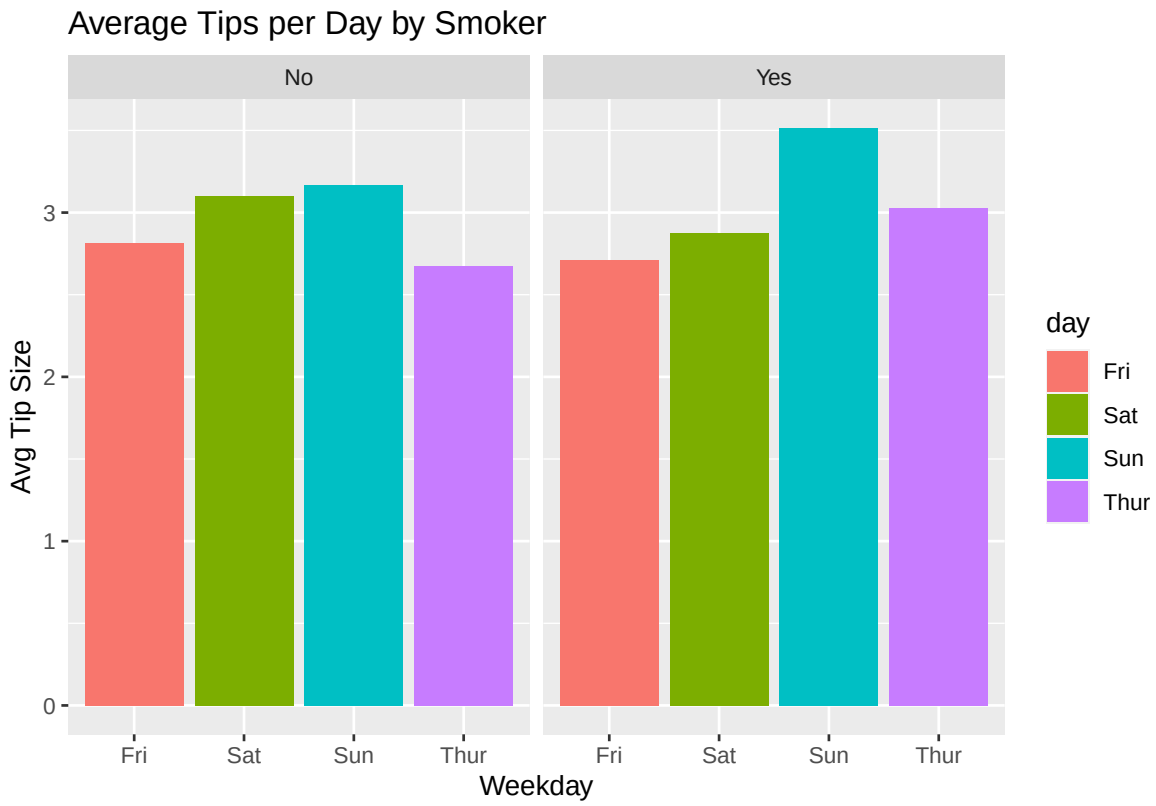
```
## `summarise()` has grouped output by 'smoker'. You can override using the `.groups` argument.
```

```
knitr::kable(tips_smoker_day)
```

smoker	day	avg_tip	total_tip	avg_size
No	Fri	2.812500	11.25	2.250000
No	Sat	3.102889	139.63	2.555556
No	Sun	3.167895	180.57	2.929825
No	Thur	2.673778	120.32	2.488889
Yes	Fri	2.714000	40.71	2.066667
Yes	Sat	2.875476	120.77	2.476190
Yes	Sun	3.516842	66.82	2.578947
Yes	Thur	3.030000	51.51	2.352941

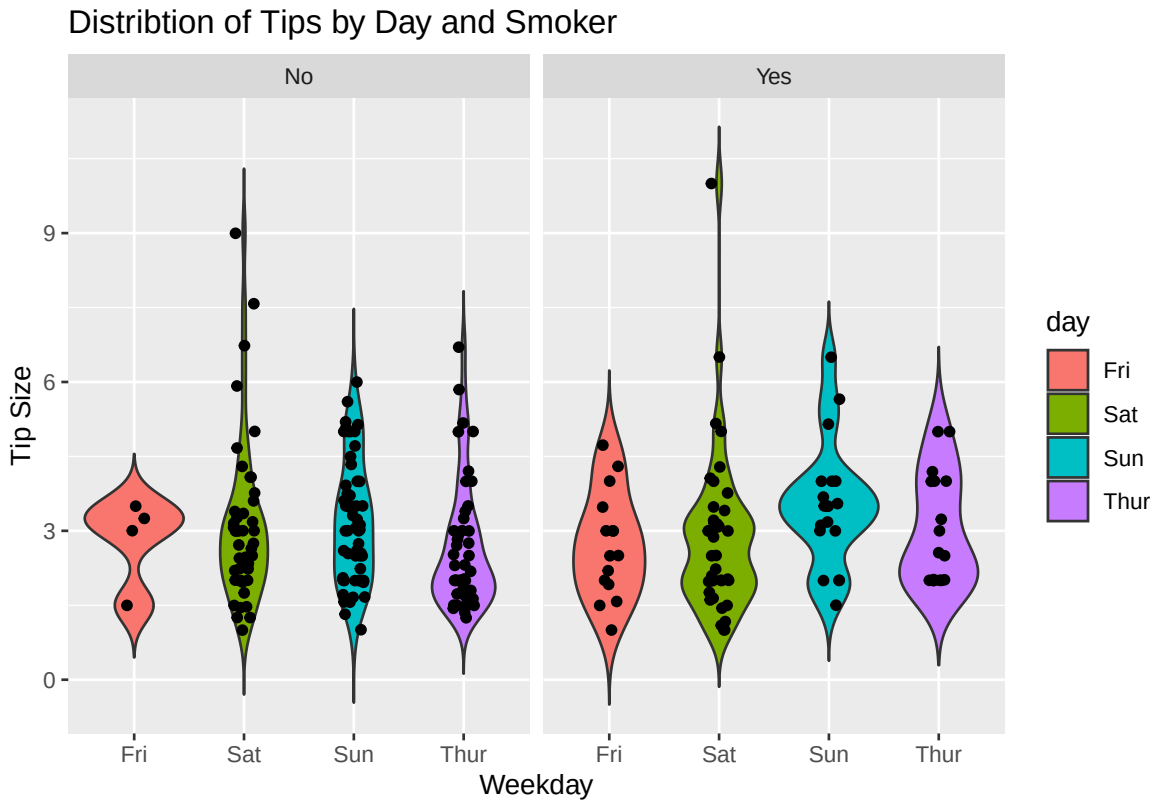
c. [1 pt] Create a bar chart that shows average tip by day, faceted by smoker.

```
ggplot(data=tips_smoker_day, aes(x=day, y=avg_tip)) +
  geom_col(aes(fill=day)) +
  facet_wrap(~smoker) +
  labs(
    title='Average Tips per Day by Smoker',
    x='Weekday',
    y='Avg Tip Size')
```



d. [2 pts] In questions (a) and (c), we plotted a summary of our data which does not show us the whole picture. In practice, we would like to see all of the data. What plot do you suggest would serve a similar purpose to the one in question (c)? In other words, what would be a better plot to show than tips by day, facetted by smoker? Please produce this plot and include your code.

```
df %>%
  ggplot(aes(x=day, y=tip)) +
  geom_violin(trim = FALSE, aes(fill=day)) +
  geom_jitter(width = 0.1) +
  facet_wrap(~smoker) +
  labs(
    title='Distribtion of Tips by Day and Smoker',
    x='Weekday',
    y='Tip Size')
```



I think a violin plot is a better representation because we can see the distribution of the tips for each day faceted by smoker. This gives us a view of all of the data. Note: Adding `geom_jitter()` allows me to see the individual observations under each distribution, indicating the volume of observations.

6. [3 pts] We have the following data set:

```
myDat = read.csv("http://mamajumder.github.io/data-science/data/reshape-source.csv")
knitr::kable(myDat)
```

player	track	walking	cycling
1	A	408	43
1	B	402	31
1	C	386	41
2	A	373	53
2	B	404	41
2	C	422	30
3	A	403	25
3	B	393	46
3	C	422	48

We want to reshape the data and produce the following output:

player	variable	A	B	C
1	walking	408	402	386
1	cycling	43	31	41
2	walking	373	404	422
2	cycling	53	41	30
3	walking	403	393	422
3	cycling	25	46	48

Provide code that will produce this desired output. Demonstrate your answer by displaying the output as well.

```
myDat_new <- myDat %>%
  pivot_longer(cols=c("walking", "cycling")) %>%
  pivot_wider(names_from = track, values_from = value)

knitr::kable(myDat_new)
```

player	name	A	B	C
1	walking	408	402	386
1	cycling	43	31	41
2	walking	373	404	422
2	cycling	53	41	30
3	walking	403	393	422
3	cycling	25	46	48

7. **Ordering the factor** In class, we have seen how to order factors. Suppose we have the following data about a certain value obtained during particular months of the year;

```
month = c("July", "June", "September", "May", "October", "August")
value = c(35, 72, 14, 23, 60, 105)
df = data.frame(month, value)
```

Now please do the following:

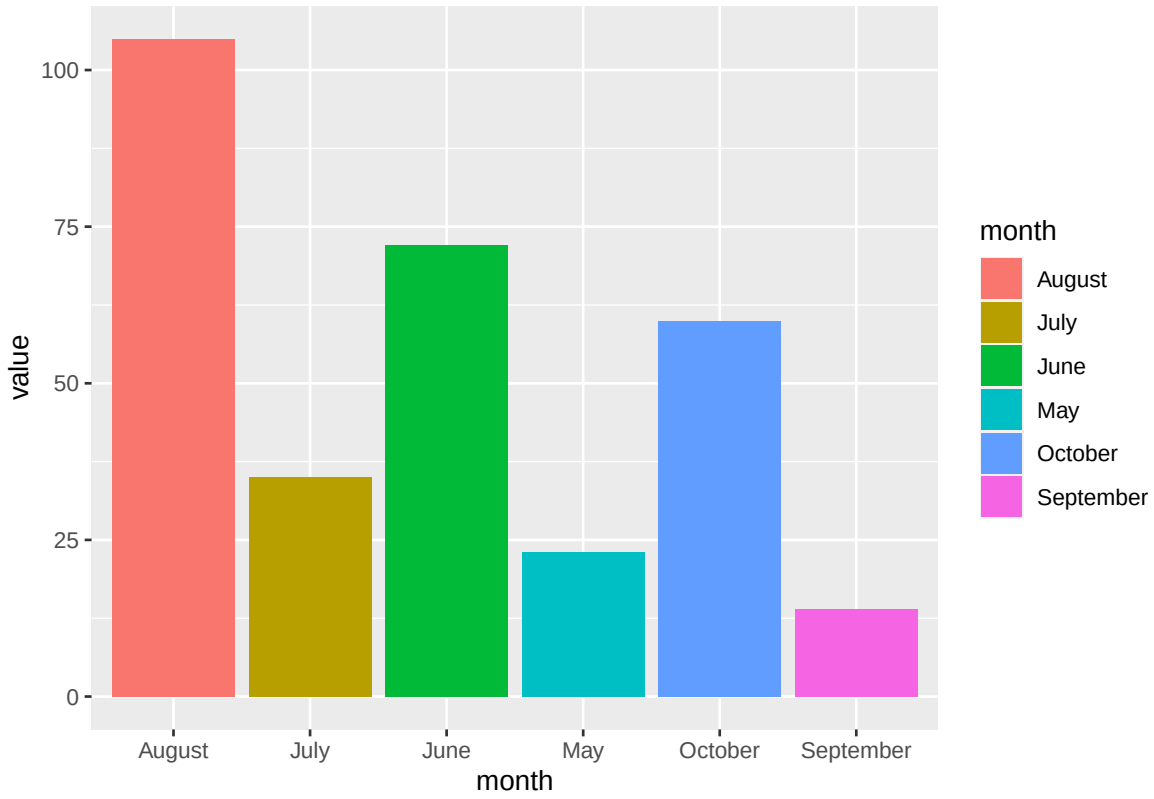
a. [1 pt] Convert the month column of the data frame `df` into a factor column. Demonstrate that it is indeed converted into a factor column.

```
df$month <- as.factor(month)
class(df$month)
```

```
## [1] "factor"
```

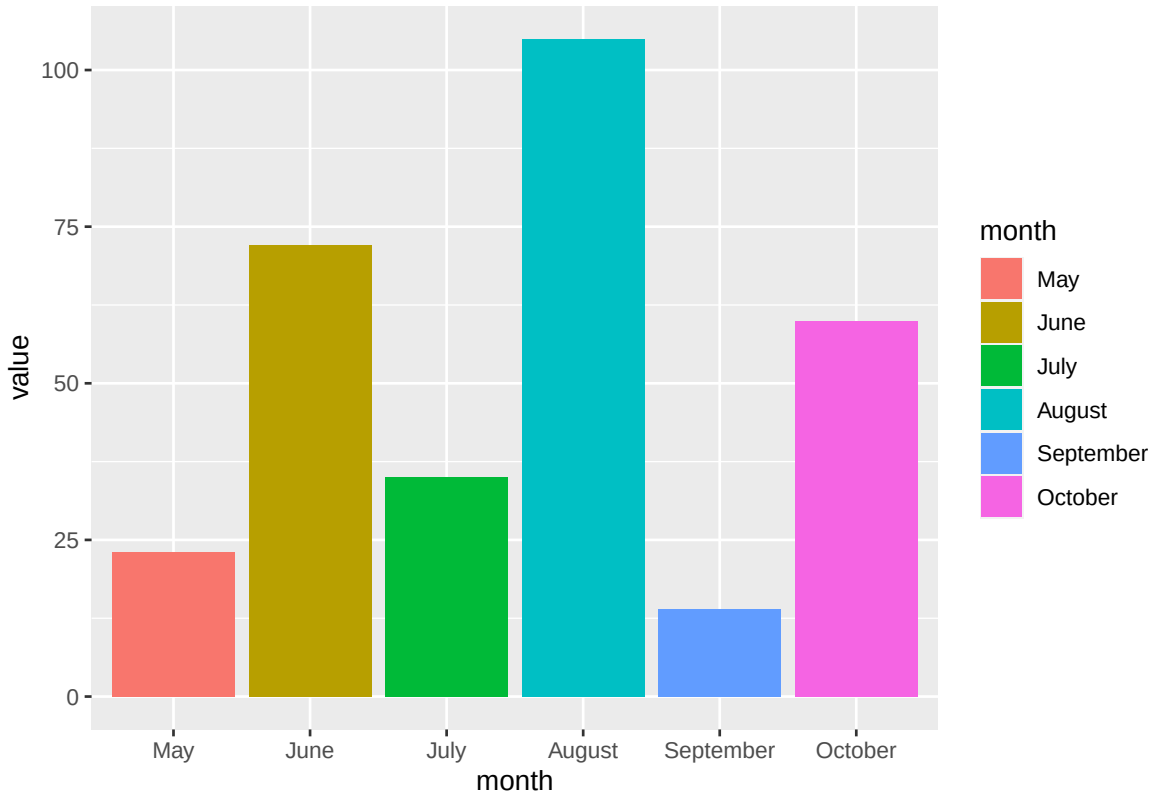
b. [1 pt] Now generate a bar chart showing the value for different months.

```
df %>% ggplot(aes(x=month, y=value)) +
  geom_col(aes(fill=month))
```



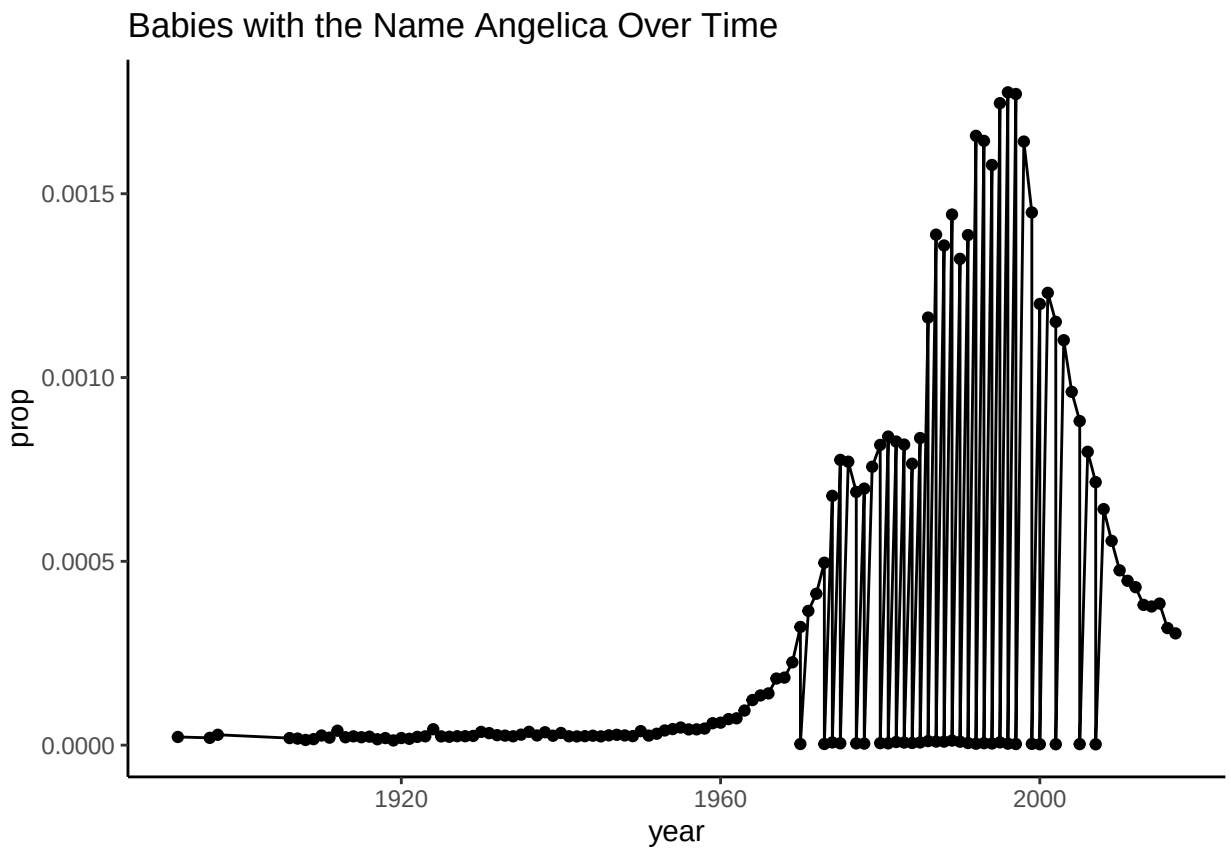
c. [2 pts] Notice the order of the levels of the months is not natural, instead the plot shows the dictionary order. Now, order the bars according to the natural order of the levels of the class (months of the year as they appear in chronological order) and regenerate the bar graph.

```
df$month <- factor(
  df$month,
  levels = c('May', 'June', 'July', 'August', 'September', 'October'))

df %>% ggplot(aes(x=month, y=value)) +
  geom_col(aes(fill=month))
```



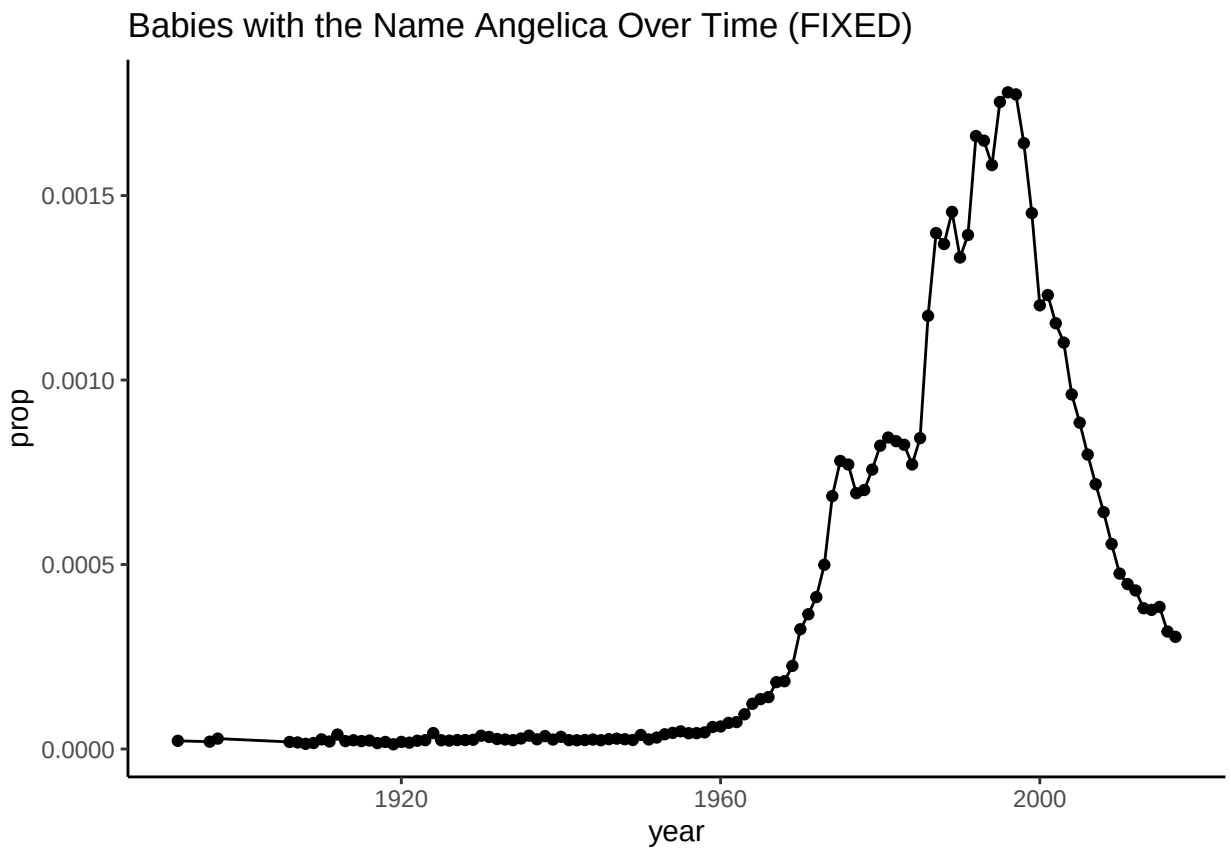
8. [3 pts] Install the `babynames` package with `install.packages()`. This package includes data from the Social Security Administration about American baby names over a wide range of years. Generate a plot of the reported proportion of babies born with the name `Angelica` over time. Do you notice anything odd about the plotted data? (Hint: you should.) If so, describe the issue and generate a new plot that adjusts for this problem. Make sure you show both plots along with all code that was used to generate them.

```
library(babynames)

babynames %>%
  filter(name == 'Angelica') %>%
  ggplot(aes(x=year, y=prop)) +
  geom_line() +
  geom_point() +
  theme_classic() +
  labs(title='Babies with the Name Angelica Over Time')
```



The issue here is that in a given year, the name *"Angelica"* has two observations of sex, M and F. For each year it will connect two points one to F and one to M. This causes a jagged look, as the value for F is much higher than the value for M. To fix this, we need to group by the year to get the total prop for M and F for Angelica.

```
babynames %>%
  filter(name == 'Angelica') %>%
  group_by(year) %>%
  summarise(
    prop = sum(prop)) %>%
  ggplot(aes(x=year, y=prop)) +
  geom_line() +
  geom_point() +
  theme_classic() +
  labs(title='Babies with the Name Angelica Over Time (FIXED)')
```



9. Suppose we have a vector of data as follows:

```
myVector = c(-15, -10, -5, 0, 5, 10, 15, 20)
```

a. [1 pt] Using the function `tapply()`, separately compute the means of the first three values, next two values, and the last three values of `myVector`. Show your code. Your result should be: -10.0, 2.5, 15.0.

```
vectorGroups = c(1,1,1,2,2,3,3,3)
tapply(myVector, vectorGroups, mean)
```

```
##     1     2     3
## -10.0   2.5  15.0
```

b. [1 pt] Now repeat question (a), but instead of computing means, you will compute the sum of squares. Again, show your code. Your result should be: 350, 25, 725.

```
vectorGroups = c(1,1,1,2,2,3,3,3)
tapply(myVector^2, vectorGroups, sum)
```

```
##   1   2   3
## 350  25 725
```