STAT 4410/8416 Homework 2

Danuwar Ramesh

Due on October 7, 2020

```
library(ggplot2)
library(maps)
library(reshape2)
library(scales)
library(knitr)
library(dplyr)
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:

```
library(reshape2)
tips.dat <- tips
```

Now answer the following questions:

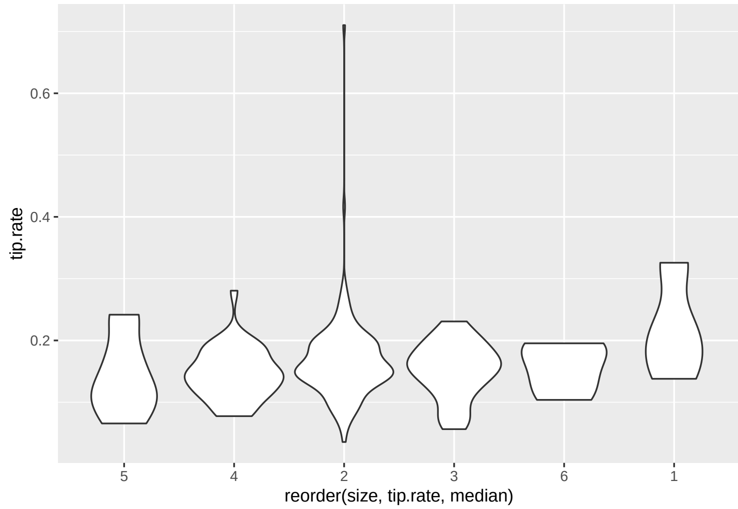
a. Compute the tip rate, dividing tip by total bill, and create a new column called `tip.rate` in the dataframe `tips.dat`. Demonstrate your results by showing the head of `tips.dat`.

```
tips.dat$tip.rate <- with(tips.dat, tip/total_bill)
head(tips.dat)
```

```
##   total_bill  tip    sex smoker day   time size   tip.rate
## 1      16.99 1.01 Female     No Sun Dinner    2 0.05944673
## 2      10.34 1.66   Male     No Sun Dinner    3 0.16054159
## 3      21.01 3.50   Male     No Sun Dinner    3 0.16658734
## 4      23.68 3.31   Male     No Sun Dinner    2 0.13978041
## 5      24.59 3.61 Female     No Sun Dinner    4 0.14680765
## 6      25.29 4.71   Male     No Sun Dinner    4 0.18623962
```

b. 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?

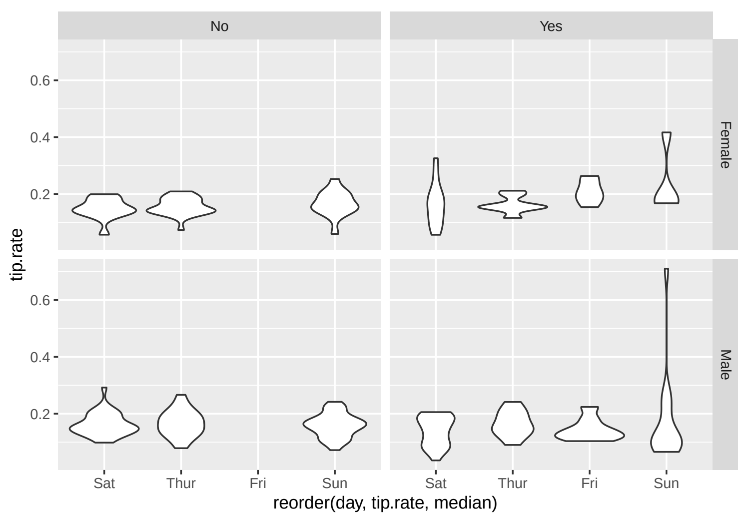
```
library(ggplot2)
ggplot(tips.dat, aes(reorder(size, tip.rate, median), tip.rate)) +
  geom_violin()
```



Which party size is responsible for the highest median tip rate? 2

c. Generate a similar plot to the one you created in question 2b 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?

```
ggplot(tips.dat, aes(reorder(day, tip.rate, median), tip.rate)) +
  geom_violin() +
  facet_grid(sex~smoker)
```



2. We can generate an nxk 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(seq(50), size = k, replace = TRUE)
M <- matrix(rnorm(n * k), ncol = k)
```

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

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

X <- M
for(i in seq(n)) {
  X[i, ] <- round(M[i, ] / V, digits = 4)
}


Z <- round(t(t(M) / V), digits = 4)
head(Z)
```

```
##         [,1]    [,2]    [,3]    [,4]    [,5]
## [1,]  0.0173 -0.0134 -0.0169 -0.0784 -0.0348
## [2,] -0.0070  0.0183  0.0547 -0.0899  0.0633
## [3,] -0.0339  0.0426 -0.0387  0.0766 -0.0154
## [4,] -0.0167  0.1886 -0.0242 -0.0818  0.0254
## [5,] -0.0049 -0.0119 -0.0542 -0.0157  0.0122
## [6,]  0.0072 -0.1500  0.0192  0.0904 -0.0304
```

b. 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()`.

```
n <- 900
k <- 500
V <- sample(seq(50), size = k, replace = TRUE)
M <- matrix(rnorm(n * k), ncol = k)
X <- M

#Using for loop
system.time(for(i in seq(n)){
  X[1,] <- round(M[1,]/ V, digits=4)

})
```

```
##    user  system elapsed
##   0.027   0.000   0.028
```

#Using no loop

```
n <- 900
k <- 500
V <- sample(seq(50), size = k, replace = TRUE)
M <- matrix(rnorm(n * k), ncol = k)
X <- M
```

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

```
##    user  system elapsed
##   0.035   0.000   0.049
```

#Therefore, total run time using no loop is shorter than using "for loop". So, my code runs faster than "for loop".

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

a. 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.

```r
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. 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`.

```r
state<-tolower(row.names(USArrests))
std_crime<-scale(USArrests)
arrest<-data.frame(state,std_crime)
head(arrest)
```

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

c. 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`.

```r
arrestDat <- merge(mdat, arrest, by.x='region', by.y='state', all.x=TRUE)
odat <- arrestDat[order(arrestDat$order),]
head(odat)
```

```
##    region      long      lat group order subregion   Murder   Assault
## 1 alabama -87.46201 30.38968     1     1      <NA> 1.242564 0.7828393
## 2 alabama -87.48493 30.37249     1     2      <NA> 1.242564 0.7828393
## 6 alabama -87.52503 30.37249     1     3      <NA> 1.242564 0.7828393
## 7 alabama -87.53076 30.33239     1     4      <NA> 1.242564 0.7828393
## 8 alabama -87.57087 30.32665     1     5      <NA> 1.242564 0.7828393
## 9 alabama -87.58806 30.32665     1     6      <NA> 1.242564 0.7828393
##     UrbanPop         Rape
## 1 -0.5209066 -0.003416473
## 2 -0.5209066 -0.003416473
## 6 -0.5209066 -0.003416473
## 7 -0.5209066 -0.003416473
## 8 -0.5209066 -0.003416473
## 9 -0.5209066 -0.003416473
```

d. 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.

```r
sdat <- subset(odat, select=c("long", "lat", "group", "region", "Murder",
                              "Assault", "UrbanPop", "Rape" ))
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
## 6 -87.52503 30.37249     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 7 -87.53076 30.33239     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 8 -87.57087 30.32665     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
## 9 -87.58806 30.32665     1 alabama 1.242564 0.7828393 -0.5209066 -0.003416473
```

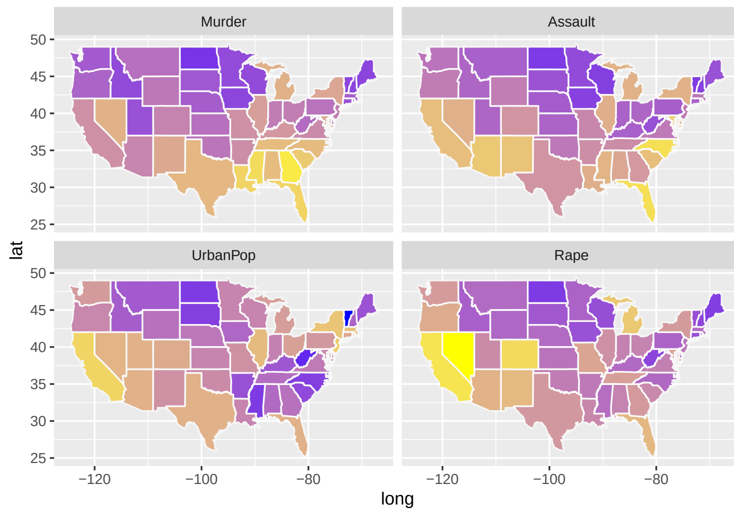
e. Melt the data frame `sdat` with id variables long, lat, group, region. Store the molten data in `msdat` and report the first few rows of data.

```r
msdat <- melt(sdat, id=c("long", "lat", "group", "region"))
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. The molten data frame `msdat` is now ready to be plotted. Create a plot showing the USA state map, fill 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.

```r
ggplot(msdat, aes(x=long, y=lat, group=group)) +
  geom_polygon(aes(fill=value), colour = alpha("white", 1/2), size = 0.5) +
  theme(legend.position = "none") +
  facet_wrap(~variable) +
  scale_fill_continuous(low="blue", high="yellow")
```



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. For each crime, name two states with its highest rate.

```
# Murder:Georgia, Mississippi
# Assault= North Carolina, Florida
# Rape= Nevada,California
```

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

According to the data from the above plot crime, i do not think that larger urban population is indicati
```

h. In question (3b) we standardized the crime rates. Why do you think we did this? Explain what would happen if we did not standardize the data.

We standardized the crime rates in question # 3b because it will make very easy and quick to find the number of each crime rates according to each state.Also, standardizing the crime rates helps to make clear plot of each crime rates.If we did not standardize the data, i think it will be a great mess. It will be very difficult and will take long time to find out the number of crime rates in each state. Also, it will be difficult while making plots.

  i. In question (3c) we ordered the data after merging. Why do you think we had to do this? Explain what would happen if we did not.

We ordered the data after merging .I think we had to do this because we can not do ploting without merging. If we did not do merging, we can still create the map or plot but it will be very confusing and disorder. Also, merging and ordering are correlated which runs one after significantly to make the plot systematic.

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. Read the data from the above link and display the first few rows of data.

```r
f <- read.csv("http://mamajumder.github.io/data-science/data/life-expectancy.csv")
head(f)
```

```
##   year    sex Bangladesh  India Pakistan  USA
## 1 1960 female     46.224 40.391   46.655 73.1
## 2 1960   male     47.787 42.329   46.223 66.6
## 3 1961 female     46.731 41.125   47.564 73.6
## 4 1961   male     48.445 43.052   47.156 67.1
## 5 1962 female     47.254 41.876   48.426 73.5
## 6 1962   male     49.104 43.784   48.044 66.9
```

b. 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.

```r
b <- melt(data=f, id=c("year", "sex"))
ggplot(b, aes(year, value)) +
  geom_line(aes(color=sex)) +
  facet_wrap(~variable)
```

c. Explain what interesting features you noticed in the plot you made in question 4b. #The interesting features which i noticed in the plot made in question 4b are:

i) Bangladesh, India and Pakistan which all belongs to Asian countries and USA which belongs to North America. It shows that all the countries shown in the above plot has increasing tread of Life expectancy.

ii) The asian countries shown in the above plot has almost the same life expectancy(male & female) starting around 40s in 1960 whereas the life expectancy(male & female) of USA was around 65 for male and above 70 for female. It shows that the asian countries shown in the above plot has very low life expectancy in comparison to USA since 1960 to 2010.

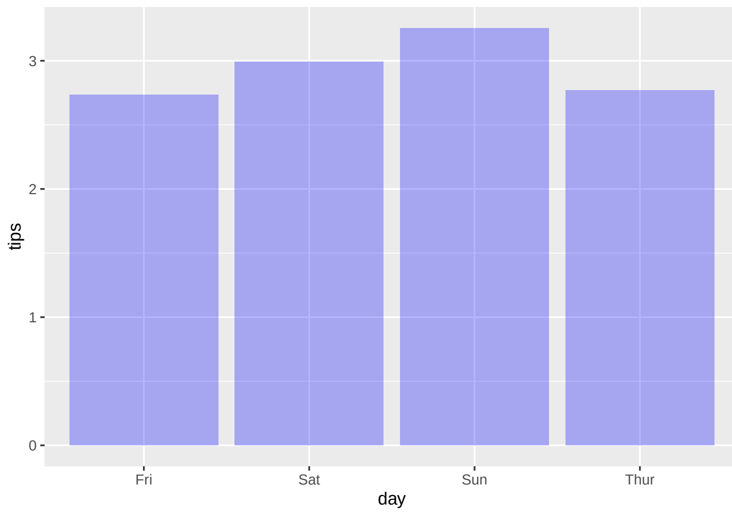
iii) looking at the above plot, it clearly shows that health conditions of USA is literally way better than the other mentioned countries.

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

a. Create a bar chart that shows the average tip by day.

```
average_tip <- tapply(tips$tip, tips$day, mean)
average_dat <- melt(average_tip)
names(average_dat)[1] <- "day"
names(average_dat)[2] <- "tips"
```

```
ggplot(average_dat, aes(day, tips)) +
  geom_bar(stat="identity", fill="blue", alpha=0.3)
```



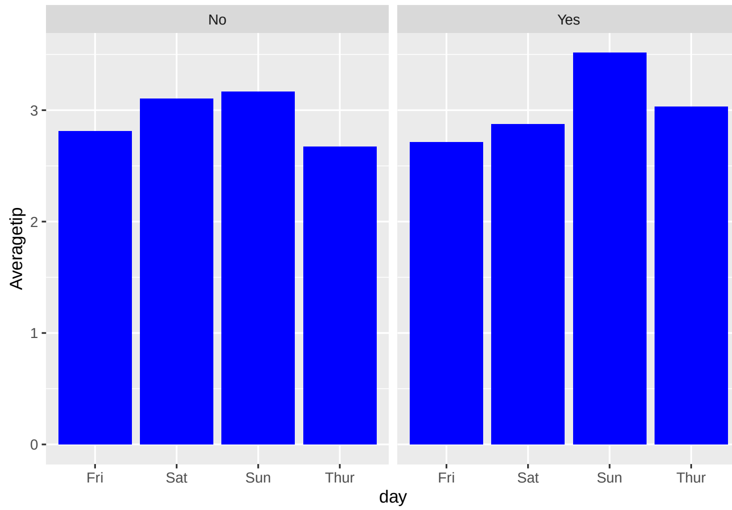
b. Compute the average tip, total tip, and average size grouped by smoker and day. i.e., For each combination of smoker and day you should have a row of these summaries. Report these results in a nice table.

```
da <- tips %>%
  group_by(smoker, day) %>%
  summarize(mean(tip), sum(tip), mean(size))
  names(da)[3] <- "Averagetip"
  names(da)[4] <- "Totaltip"
  names(da)[5] <- "Averagesize"
kable(da)
```

smoker	day	Averagetip	Totaltip	Averagesize
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. Create a bar chart that shows average tip by day, faceted by smoker.

```
ggplot(da, aes(day, Averagetip)) +
  geom_bar(stat="identity", fill="blue")+ facet_wrap(~da$smoker)
```

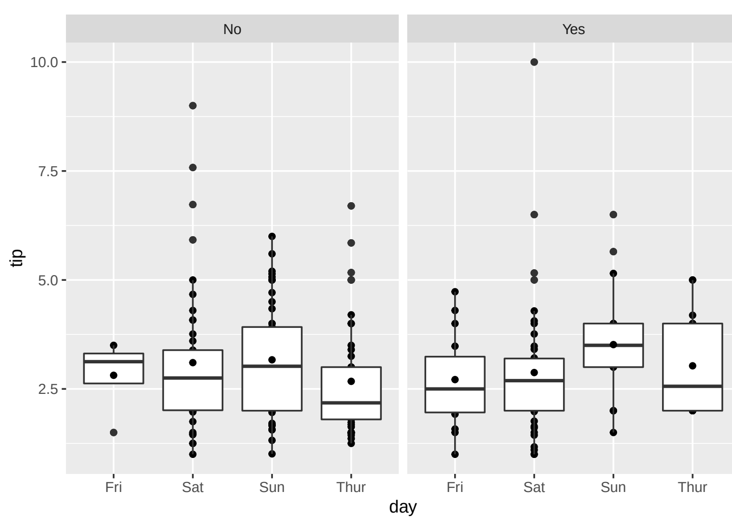


d. In questions 5a and 5c, we plotted a summary of our data which does not show us the whole picture. In practice, we would like to see all of the data. What plot do you suggest would serve a similar purpose to the one in question 5c? In other words, what would be a better plot to show than tips by day, facetted by smoker? Please produce this plot and include your code.

Here, box plot is much better than other plots.

```
ggplot(tips, aes(day, tip)) +
  geom_point() +
  geom_boxplot() +
  stat_summary(fun.y = mean, geom = "point") +
  facet_wrap(~smoker)
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```



6. We have the following data set:

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

```
myDat <- melt(myDat, id = c("player", "track"))
Dcast <- dcast(myDat, player + variable ~ track)
kable(Dcast)
```

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

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. Convert the month column of dataframe `df` into a factor column. Demonstrate that it is indeed converted into a factor column.

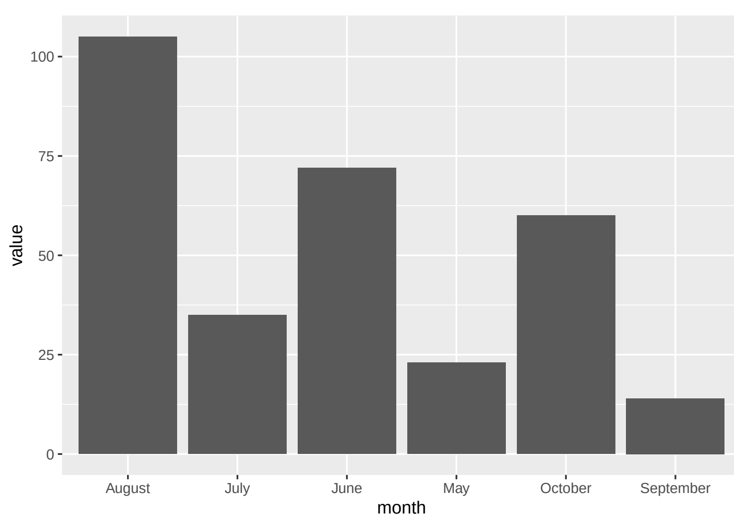
```
df$month <- factor(df$month)
str(df)
```

```
## 'data.frame':    6 obs. of  2 variables:
##  $ month: Factor w/ 6 levels "August","July",..: 2 3 6 4 5 1
##  $ value: num  35 72 14 23 60 105
```

Here, is.factor() lets us check if the parameter is a factor or not. It returns TRUE if it is factor and returns FALSE if it is not.

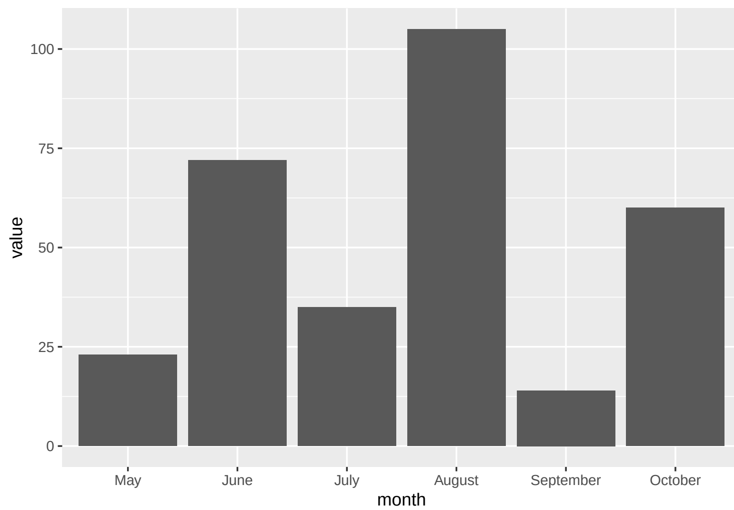
b. Now generate a bar chart showing the value for different months.

```
ggplot(df) +
  geom_bar(aes(month, value), stat="identity")
```



c. 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(month, levels = c("May", "June", "July", "August", "September", "October"))
ggplot(df) +
  geom_bar(aes(month, value), stat="identity")
```



8. 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.

#install the package for the first time install.packages('babynames')

#load the package library(babynames)

#print the variable names in the dataset babynames names(babynames)

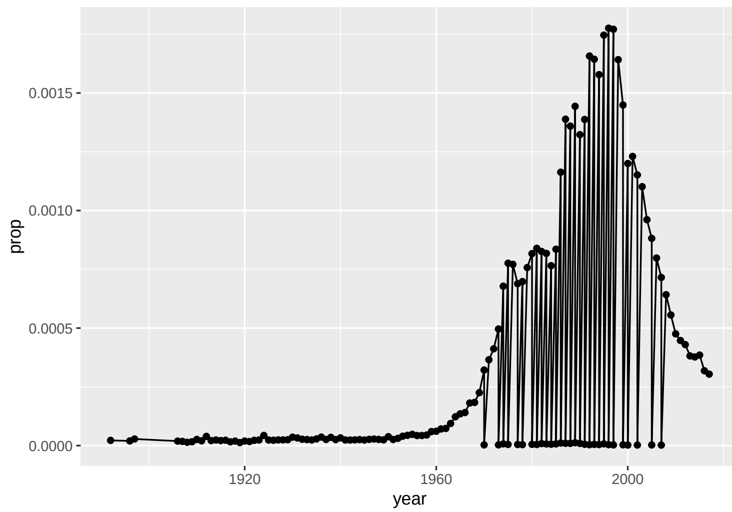
#create a subset of data with names Angelica dat<-subset(babynames,name=="Angelica")

```
library(babynames)
head(babynames)
```

```
## # A tibble: 6 x 5
##    year sex   name          n   prop
##   <dbl> <chr> <chr>     <int>  <dbl>
## 1  1880 F     Mary       7065 0.0724
## 2  1880 F     Anna       2604 0.0267
## 3  1880 F     Emma       2003 0.0205
## 4  1880 F     Elizabeth  1939 0.0199
## 5  1880 F     Minnie     1746 0.0179
## 6  1880 F     Margaret   1578 0.0162
```

```r
angelica <- babynames %>%
  filter(name == "Angelica")

ggplot(angelica, aes(year, prop)) +
  geom_point() +
  geom_line()
```



9. Bonus (2 points) for undergraduates and mandatory for graduate students. Suppose we have a vector of data as follows:

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

a. 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.

```r
vector_indx <- c(1,1,1,2,2,3,3,3)
tapply(myVector, vector_indx, mean)
```

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

b. Now repeat question 9a, but instead of computing means, you will compute the sum of squares. Again, show your code. Your result should be: 350, 25, 725.

```r
sum_squares <- function(x) sum(x^2)
tapply(myVector, vector_indx, sum_squares)
```

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