HW5_Yun_Young

Problem 3 What are you thoughts for what makes a good figure? I think that a good figure should be able to present relationship between variables with clear axis and be easy to recognize.

Problem 4

4.a

```
#create a function that counts proportion of success by summing up the input and dividing by the length.
success <- function(x)
{
  sum(x)/length(x)
}
```

4.b

```
#store data as follows
set.seed(12345)
P4b_data <- matrix(rbinom(10, 1, prob = (30:39)/100), nrow = 10, ncol = 10)
```

4.c

```
#compute the proportion of success in P4b_data by column and then by row
apply(P4b_data, 2, FUN = success)
```

```
##  [1] 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
```

```
apply(P4b_data, 1, FUN = success)
```

```
##  [1] 1 1 1 1 0 0 0 0 1 1
```

```
#The proportion of success is the same across column and row. The random draws were all identical.
```

4.d

```
#create function whose input is a probability and output is a vector whose elements are the outcomes of 10 flips of a coin. Then, use sapply to generate 10 coin flips for each probability.
flipCoin <- function(p)
{
  rbinom(10, 1, p)
}

prob <- (30:39)/100
P4b_data <- sapply(prob, flipCoin)
P4b_data
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    0    0    1    1    1    1    1    1    1     0
##  [2,]    0    0    0    0    1    0    0    0    1     1
##  [3,]    1    1    0    1    0    1    0    0    0     1
##  [4,]    0    1    1    1    0    1    0    0    0     1
##  [5,]    0    0    0    0    1    1    0    0    1     0
##  [6,]    0    0    0    0    0    0    0    0    0     1
##  [7,]    0    1    1    0    1    1    1    1    1     1
##  [8,]    0    0    1    0    0    0    1    0    1     1
##  [9,]    0    0    0    0    0    0    0    1    0     0
## [10,]    1    0    0    0    0    1    0    0    0     0
```

Problem #5

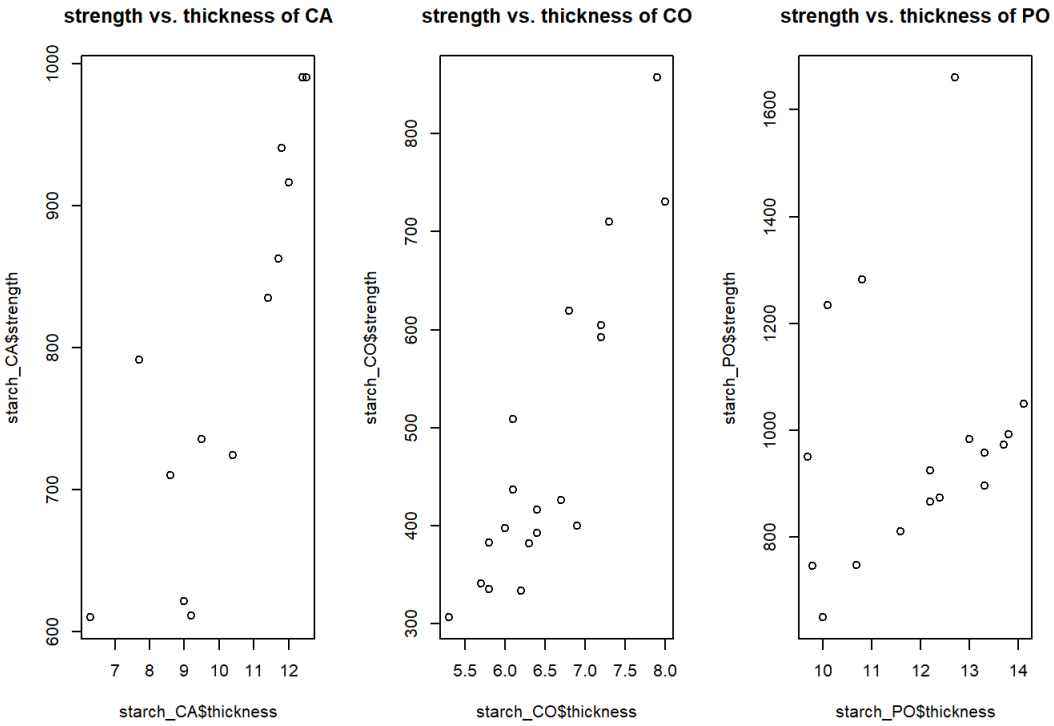
```
###This code creates scatterplot and boxplot of the data.

library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

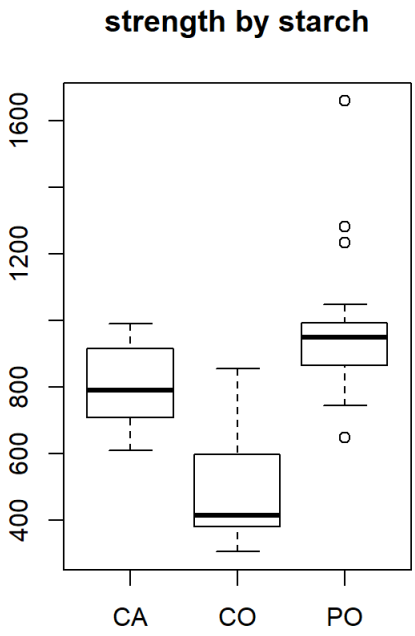
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

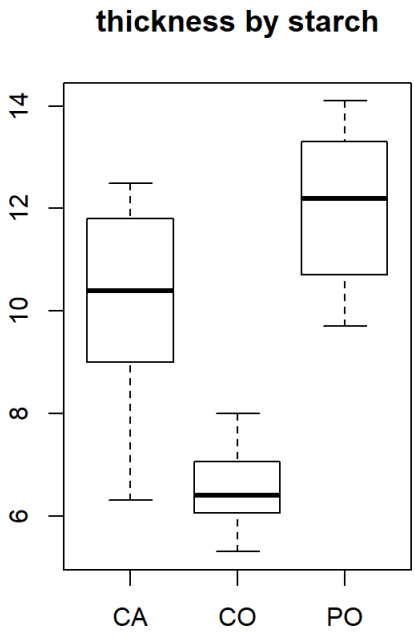
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
url<-"http://www2.isye.gatech.edu/~jeffwu/book/data/starch.dat"
starchData <- read.table(url, header=T, skip=0, fill=T, stringsAsFactors = F)

#filter columns by starch
starch_CA <- filter(starchData, starch == "CA")
starch_CO <- filter(starchData, starch == "CO")
starch_PO <- filter(starchData, starch == "PO")

#create scatterplot of strength vs. thickness by starch
par(mfrow=c(1,3))
plot(starch_CA$thickness, starch_CA$strength, main = "strength vs. thickness of CA")
plot(starch_CO$thickness, starch_CO$strength, main = "strength vs. thickness of CO")
plot(starch_PO$thickness, starch_PO$strength, main = "strength vs. thickness of PO")
```




```
#create boxplots comparing strength and thickness by starch.
par(mfrow=c(1,2))
boxplot(strength~starch, starchData, main = "strength by starch")
boxplot(thickness~starch, starchData, main = "thickness by starch")
```





Problem #6

6.A Get and import a database of US cities and states.

```
#read in data.
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
states <- fread("./us_cities_and_states/states.sql",skip = 23,sep = "'", sep2 = ",", header = F, select = c(2,4))
cities <- fread(input = "./us_cities_and_states/cities.sql",skip = 23,sep = "'", sep2 = ",", header = F, select = c(2,4))

#can you figure out how to limit this to the 50?
###Yes! Add the following: nrows = 50
```

6.B Create a summary table of the number of cities included by state.

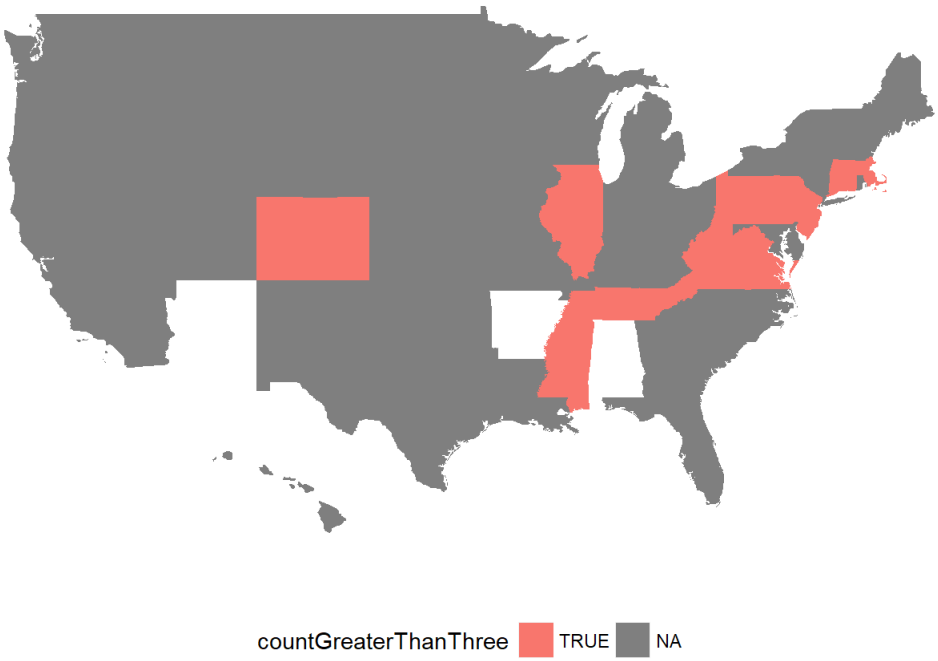
```
###The states in cities column are unlisted, put into table and stored. By doing this, we obtain frequency of the states in the cities column.

citycounts <- as.data.frame(table(unlist(cities$V4)))

colnames(citycounts) <- c("State", "City Count")

citycounts
```

```
##    State City Count
## 1     AK        229
## 2     AL        578
## 3     AR        605
## 4     AZ        264
## 5     CA       1239
## 6     CO        400
## 7     CT        269
## 8     DC          3
## 9     DE         57
## 10    FL        524
## 11    GA        628
## 12    HI         92
## 13    IA        937
## 14    ID        266
## 15    IL       1287
## 16    IN        738
## 17    KS        634
## 18    KY        803
## 19    LA        478
## 20    MA        511
## 21    MD        430
## 22    ME        461
## 23    MI        885
## 24    MN        810
## 25    MO        942
## 26    MS        440
## 27    MT        360
## 28    NC        762
## 29    ND        373
## 30    NE        528
## 31    NH        255
## 32    NJ        579
## 33    NM        346
## 34    NV         99
## 35    NY       1612
## 36    OH       1069
## 37    OK        585
## 38    OR        379
## 39    PA       1801
## 40    PR         99
## 41    RI         70
## 42    SC        377
## 43    SD        364
## 44    TN        548
## 45    TX       1466
## 46    UT        250
## 47    VA        839
## 48    VT        288
## 49    WA        493
## 50    WI        753
## 51    WV        753
## 52    WY        176
```

Part c. Create a function that counts the number of occurances of a letter in a string.

```
###This code creates 2 functions that counts letters and uses 'sapply' to generate counts for each of the states.


#This function counts the number of occurances of a given letter in a given string.
countLetter <- function(letter, state)
{
  state2 <- gsub(letter,"",state)
  return (nchar(state) - nchar(state2))
}

#counts occurrences of letters a~z in a given string using for-loop.
counts <- function(state)
{
  state <- tolower(state)
  Ocurrence <- vector('numeric')
  for(i in 1:26)
  {
    Ocurrence <- combine(Ocurrence, countLetter(letters[i], state))
  }
  return (Ocurrence)
}

#counts the occurrences of letters for each state, and name the column with letters
letter_count <- t(sapply(states$V2, counts))
colnames(letter_count) <- c(letters[1:26])
```

```
####This code creates a graphic of the United States, using ggplot,  in which the states with 3 or more recurring letters are colored.

library(ggplot2)
library(fiftystater)

#creates a data.frame with state names and a blank column that will have 'TRUE' value if the states name has more than or equal to 3 recurring letters.

letter_count_three <- select(states, V2) %>% mutate(countGreaterThanThree = NA)


#for-loops once to go through letters a~z, and once again to go through 50 states, and fills the variable with 'TRUE' if the state had more than or equal to 3 recurring letters
for(i in 1:47)
{
  for(j in 1:26)
  {
   if(letter_count[i, j] >= 3)
   letter_count_three[i, 2] <- TRUE
  }
}

#puts the above variable into data.frame, adding a column with small case state names.
letterCounts <- data.frame(state = tolower(states$V2), letter_count_three)


#uses ggplot to color the maps according to the column 'countGreaterThanTree'
p <- ggplot(letterCounts, aes(map_id = state)) +
  geom_map(aes(fill = countGreaterThanThree), map = fifty_states) +
  expand_limits(x = fifty_states$long, y = fifty_states$lat) +
  coord_map() +
  scale_x_continuous(breaks = NULL) +
  scale_y_continuous(breaks = NULL) +
  labs(x = "", y = "") +
  theme(legend.position = "bottom", panel.background = element_blank())
p
```


countGreaterThanThree
TRUE
NA