HW5_Do_Quyen

Quyen Do

September 21, 2018

Problem 3

I believe a good figure should provide us with an overall impression about the data like its size, structure, relationship between variables. It should also provide insight into the data and inform the next steps in the analysis process. Some important aspects of a good figures are reasonable scaling, interpretability, size and the use of colors.

Problem 4

a. A function computing the proportion of successes in a vector

```
count_success <- function (vect, value = 1) {
  # Compute the proportion of successes in a vector

  # Args:
  # vect: the vector on which the proportion of successes will be computed
  # value: the value presented "success" value in the vector. Default value is 1

  #Return:
  # A real number from 0 to 1

  length(vect[which(vect==value)])/length(vect)
}
```

b. Create a simuluated matrix

```
set.seed(12345)
P4b_data <- matrix(rbinom(10,1,prob =(30:40)/100),nrow = 10, ncol =10)
```

c. Checking the proportion of success

```
# Calculate the proportion of success across matrix row
prop_row <- apply(P4b_data,1,count_success)
prop_col <- apply(P4b_data,2,count_success)

prop_mat <- matrix(c(prop_row,prop_col),nrow=2,ncol=10,byrow = TRUE, dimnames = list(c("By Row","By Col"
prop_mat
```

```
##          1   2   3   4   5   6   7   8   9  10
## By Row 1.0 1.0 1.0 1.0 0.0 0.0 0.0 0.0 1.0 1.0
## By Col 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6 0.6
```

The matrix of the simulated binomial using the code in b didn't produce the data as the intention. Instead of applying different probabilities to draw success among 10 rows of the matrix, the function seems to apply $p = 1$ to all the rows instead.

d.

```
simulate_binom <- function(probability) {
  # Simulate 10 random binomial variables
```

```
  #  of n = 10 and given probability

  # Args:
  # probability: the probability for the binomial distribution

  #Return:
  # a vector containing 10 RVs drawn from binomial distribution
  return(rbinom(10, 1, prob = probability))
}

# A vector of probability
prob_vect <- (31:40)/100

# apply simulate_binom on each element of prob_vect
correct_mat <- sapply(prob_vect,simulate_binom)

# Calculate the proportion of success
# across rows and columns of correct_mat
prop_row2 <- apply(correct_mat,1,count_success)
prop_col2 <- apply(correct_mat,2,count_success)

prop_mat2 <- matrix(c(prob_vect,prop_row2,prop_col2),nrow=3,ncol=10,byrow = TRUE, dimnames = list(c("Tru
prop_mat2
```

```
##                      1    2    3    4    5    6    7    8    9  10
## True probability 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4
## By Row           0.70 0.30 0.50 0.50 0.30 0.10 0.80 0.40 0.10 0.2
## By Col           0.20 0.30 0.40 0.30 0.40 0.60 0.30 0.30 0.50 0.6
```

To fix the code in b, I created a vector of probabilities, then used the sapply on each element of that vector and pasted it onto simulation function. "sapply" on the probability vector return a matrix of data whose columns are 10 random data points drawn from the binomial distribution having corresponding probability from the vector.

The "By col" information tell us the probabilities of success for each true probability. Compared with the true probability, we saw that they are pretty close. Any difference is due to randomization.

Problem 5

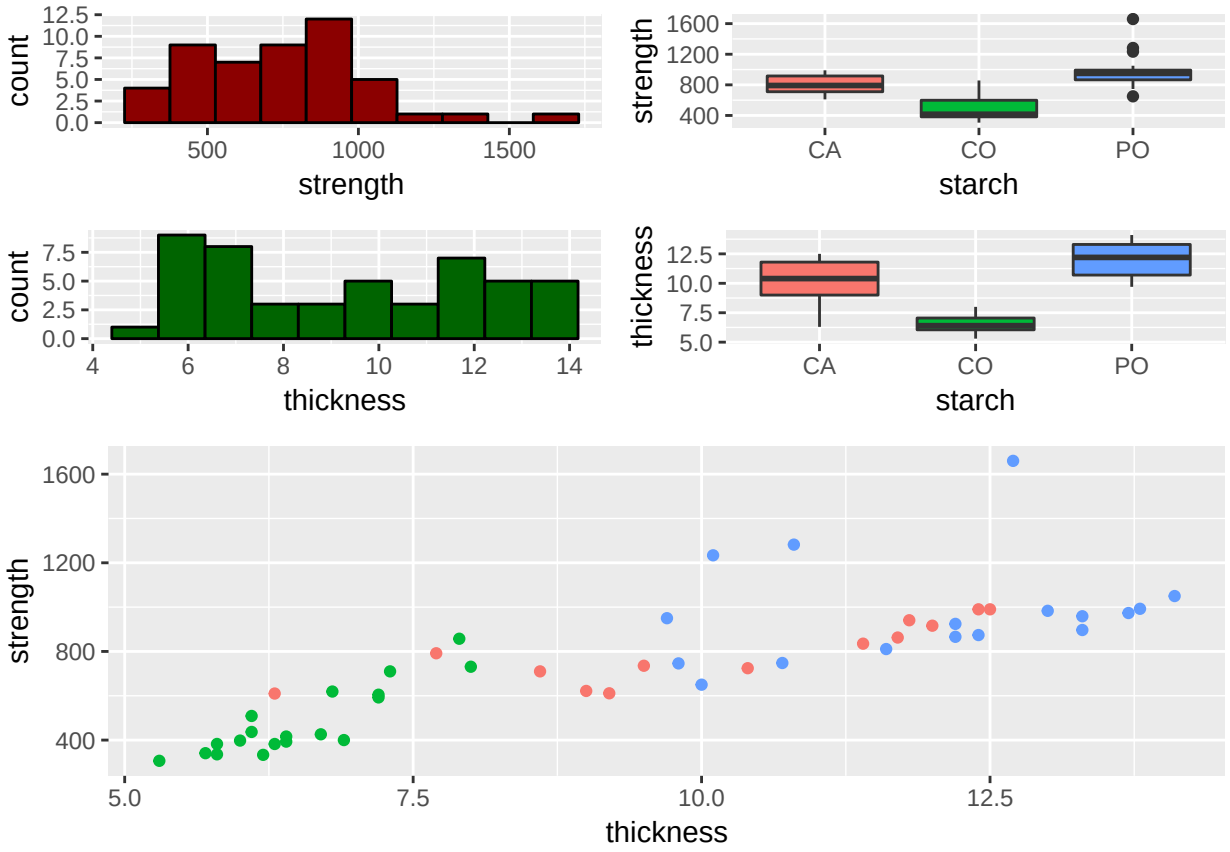
```
#Import raw data from url
url <- "https://www2.isye.gatech.edu/~jeffwu/book/data/starch.dat"
starch.dat <- read.csv(url, header=TRUE,sep="")

#Summary
str(starch.dat)
```

```
## 'data.frame':    49 obs. of  3 variables:
##  $ starch   : Factor w/ 3 levels "CA","CO","PO": 1 1 1 1 1 1 1 1 1 1 ...
##  $ strength : num  792 610 710 941 990 ...
##  $ thickness: num  7.7 6.3 8.6 11.8 12.4 12 11.4 10.4 9.2 9 ...
```

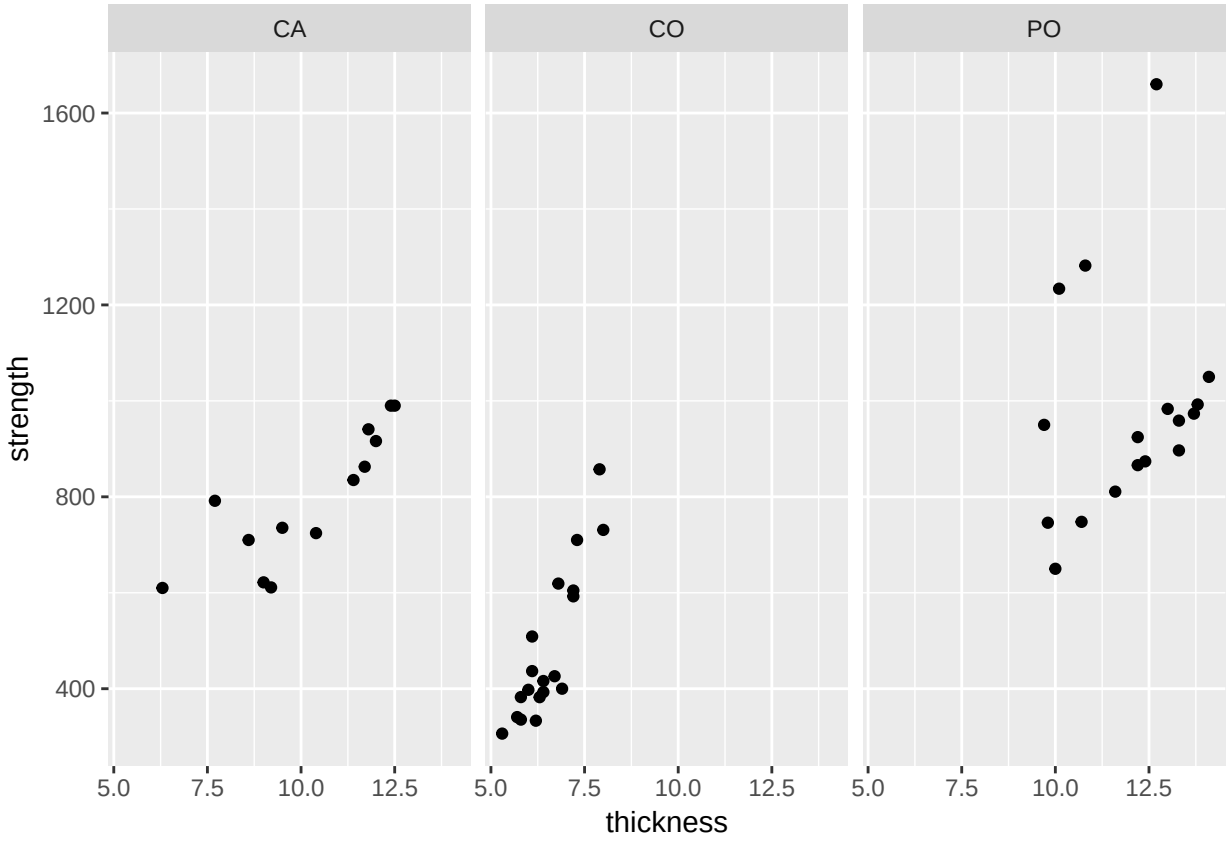
```
starch.dat$starch <- factor(starch.dat$starch)
knitr::kable(summary(starch.dat))
```

starch	strength	thickness
CA:13	Min. : 306.4	Min. : 5.300
CO:19	1st Qu.: 508.8	1st Qu.: 6.700
PO:17	Median : 735.4	Median : 9.500
NA	Mean : 737.0	Mean : 9.388
NA	3rd Qu.: 924.4	3rd Qu.:12.000
NA	Max. :1660.0	Max. :14.100

```
#Multipane plot using ggplot and ggpubr
p1 <- ggplot(starch.dat,aes(x=strength)) + geom_histogram(colour= "black",bins=10,fill="darkred")

p2 <- ggplot(starch.dat,aes(x=starch,y=strength ,group=starch,fill=starch))
p2 <- p2 + geom_boxplot() + guides(fill=FALSE) + labs(x="starch")

p3 <- ggplot(starch.dat,aes(x=thickness)) + geom_histogram(colour= "black",bins=10,fill="darkgreen")

p4 <- ggplot(starch.dat,aes(x=starch,y=thickness ,group=starch,fill=starch))

p4 <- p4 + geom_boxplot() + guides(fill=FALSE) + labs(x="starch")

p5 <- ggplot(starch.dat,aes(thickness,strength,colour=starch)) + geom_point() + labs(x="thickness",y="s

ggarrange(ggarrange(p1,p2,p3,p4,ncol = 2,nrow=2), p5, nrow = 2)
```



```
ggplot(starch.dat,aes(thickness,strength)) + geom_point() + facet_wrap(~starch)
```



Problem 6

```
##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
## The following object is masked from 'package:purrr':
##
##     transpose
```

Part b. A summmary table of the number of cities included by state

```
#The number of cities included by states
#knitr::kable(t(table(cities$state_code)),caption = "Number of cities by state")
t(table(cities$state_code))
```

```
##
##          AK   AL   AR   AZ   CA   CO   CT   DC   DE   FL   GA   HI   IA
##   [1,]  273  838  709  532 2651  659  438  284   98 1487  972  139 1060
##
##          ID   IL   IN   KS   KY   LA   MA   MD   ME   MI   MN   MO   MS
##   [1,]  325 1587  989  756  961  725  703  619  489 1170 1031 1170  533
```

```
##
##          MT   NC   ND   NE   NH   NJ   NM   NV   NY   OH   OK   OR   PA
##    [1,]  405 1090  407  620  284  733  426  253 2207 1446  774  484 2208
##
##          PR   RI   SC   SD   TN   TX   UT   VA   VT   WA   WI   WV   WY
##    [1,]  176   91  539  394  795 2650  344 1238  309  732  898  859  195
```

```
city_count_data <- as.data.frame(table(cities$state_code))
names(city_count_data) <- c("state_code", "city_count")
```

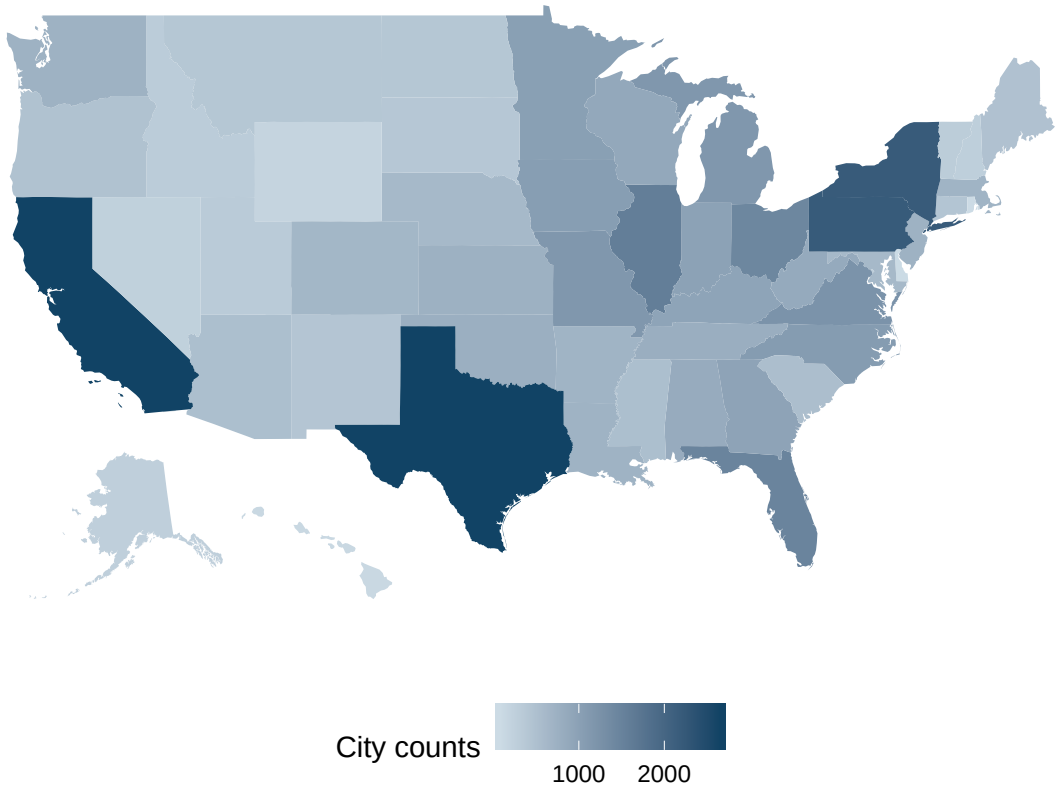
Part c. Function that counts occurances of a letter in a string

```
count.occurances <- function(string, letter){
  #Count the occurance of a letter from a given string

  #Args:
  # string: the string from which the letter will be calculated from
  # letter: the letter whose occurances in the string will be calculated

  #Returns:
  # the number of occurances of the letter in the string

  # Split the string into a vector of characters
  char.vect <- strsplit(string,split = NULL)[[1]]

  # Ensure lower case is across the two variables
  char.vect <- tolower(char.vect)
  letter <- tolower(letter)

  return (sum(char.vect==letter))

}

letter_count <- data.frame(matrix(NA,nrow=51,ncol=26))
for (i in 1:51){
  letter_count[i,] <- sapply(LETTERS,count.occurances,string=states$state_name[i])
}
names(letter_count) <- LETTERS

#Merge the information from states data.frame onto letter_count
letter_count$state_name <- states$state_name
letter_count$state_code <- states$state_code
letter_city_total <- merge(letter_count,city_count_data,by="state_code")

#Exclude "district of columbia"
#letter_city_total <- letter_city_total[letter_city_total$state_code != "DC",]
```

Part d.

```
#https://cran.r-project.org/web/packages/fiftystater/vignettes/fiftystater.html
library(ggplot2)
library(fiftystater)

#Create US map colored by city count
data("fifty_states") # this line is optional due to lazy data loading
```

```r
# Set up dataset to make sure state_names match that of fifty_states
letter_city_total$state_code <- tolower(letter_city_total$state_code)
letter_city_total$state_name <- tolower(letter_city_total$state_name)

# Color for the map
#(reference: https://medium.com/@NickDoesData/visualizing-geographic-data-in-r-fb2e0f5b59c5 )

low_color='#ccdbe5'
high_color="#114365"
legend_title = 'City counts'

# US Maps colored with city counts

# map_id creates the aesthetic mapping to the state name column in your data
p <- ggplot(letter_city_total, aes(map_id = state_name))

# map points to the fifty_states shape data
p <- p + geom_map(aes(fill = city_count), map = fifty_states)

p <- p + expand_limits(x = fifty_states$long, y = fifty_states$lat) + coord_map()
p <- p + scale_x_continuous(breaks = NULL) + scale_y_continuous(breaks = NULL)

#Set gradient color for city counts
p <- p + scale_fill_continuous(low = low_color, high = high_color, guide = guide_colorbar(title = legend


p <- p + labs(x = "", y = "") + theme(legend.position = "bottom", panel.background = element_blank())
p
```




```
#Create a new variable signaled state_name with more than 3 occurances of any letter

letter_city_total$more_than_3 <- c()
for (i in 1:51) {
  letter_city_total$more_than_3[i] <- ifelse(sum(letter_city_total[i,2:27]>3) >=1,1,0)
  if (sum(letter_city_total[i,2:27]>3))
    print(letter_city_total[i,"state_name"])
}
```

```
## [1] "alabama"
## [1] "massachusetts"
## [1] "mississippi"
## [1] "tennessee"
```

```
# US Maps highlighted by states that have more than 3 occurrences of any letter in their name
high_color <- "darkred"

legend_title <- "State with name having 3 or more occurances of some letter"
# map_id creates the aesthetic mapping to the state name column in your data
letter_city_total$id <- letter_city_total$state_name
p <- ggplot(letter_city_total, aes(map_id = state_name))

# map points to the fifty_states shape data
p <- p + geom_map(aes(fill = more_than_3), map = fifty_states,color = "#ffffff")

p <- p + expand_limits(x = fifty_states$long, y = fifty_states$lat) + coord_map()
```

```
# ATTEMPT TO ADD STATE ABBREVIATION - NOT WORKING!
# letter_city_total$id <- letter_city_total$state_name
# p <- p + geom_text(data = fifty_states %>%
#                group_by(id) %>%
#                summarise(lat = mean(c(max(lat), min(lat))),
#                          long = mean(c(max(long), min(long)))) %>%
#                mutate(state = id) %>%
#                left_join(letter_city_total, by = "id"), aes(x = long, y = lat, label = state_code ))

p <- p + scale_x_continuous(breaks = NULL) + scale_y_continuous(breaks = NULL)

#Set gradient color for city counts
p <- p + scale_fill_continuous(high = high_color, guide = guide_colorbar(title = legend_title),labels =
p <- p + labs(x = "", y = "") + theme(legend.position = "bottom", panel.background = element_blank())
p
```