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

```
###This code creates 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

summary of proportion of success by col and row

byColumn0.60.60.60.60.60.60.60.60.60.6
byRows   1.01.01.01.00.00.00.00.01.01.0
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)

colnames(P4b_data)<-c("Draw_1", "Draw_2", "Draw_3", "Draw_4","Draw_5","Draw_6","Draw_7","Draw_8","Draw_9","Draw_10")

knitr::kable(P4b_data, caption="10 randomly generated coin filps of 10")
```

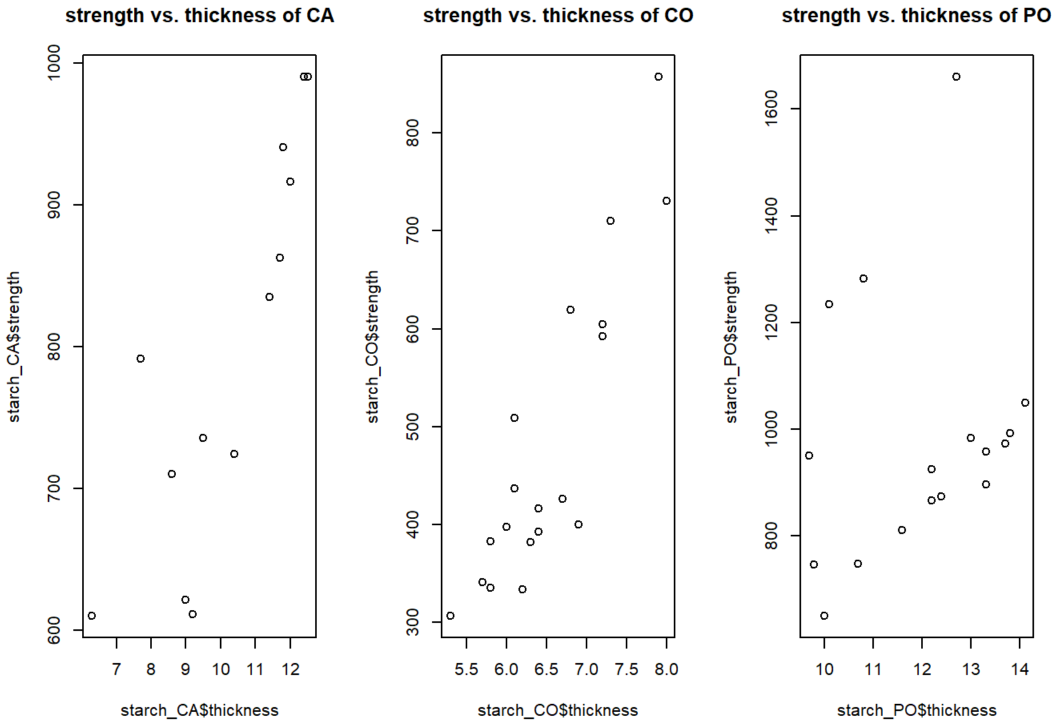
10 randomly generated coin filps of 10

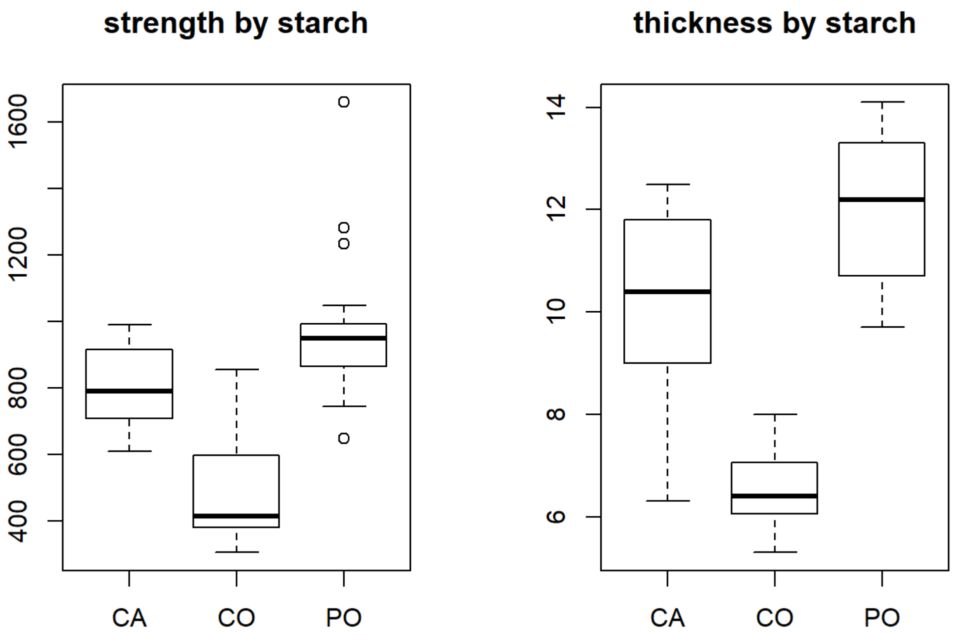
Draw_1	Draw_2	Draw_3	Draw_4	Draw_5	Draw_6	Draw_7	Draw_8	Draw_9	Draw_10
0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	0	0	0	1	1
1	1	0	1	0	1	0	0	0	1
0	1	1	1	0	1	0	0	0	1
0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	0	0	0	0	1
0	1	1	0	1	1	1	1	1	1
0	0	1	0	0	0	1	0	1	1
0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	1	0	0	0	0

Problem #5

```
##
## 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
```





Problem #6

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

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

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##                      V2 V4
## 1:           California CA
## 2:             Colorado CO
## 3:          Connecticut CT
## 4: District of Columbia DC
## 5:             Delaware DE
## 6:              Florida FL
```

```
##             V2 V4
## 1:   Abbeville MS
## 2:   Abbeville SC
## 3:       Abbot ME
## 4:  Abbotsford WI
## 5:      Abbott TX
## 6: Abbottstown PA
```

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

City counts by State

	State	CityCount
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
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

	State	CityCount
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. The first function, countLetter, counts the number of occurances of a given letter in a given string. The second function, counts, counts the occurrences of letters a~z in a given string using for-loop. Lastly, using the 'sapply' function counts the occurrences of letters for each state in state data, and name the column with letters.

countLetter <- function(letter, state)
{
  state2 <- gsub(letter,"",state)
  return (nchar(state) - nchar(state2))
}


counts <- function(state)
{
  state <- tolower(state)
  Ocurrence <- vector('numeric')
  for(i in 1:26)
  {
    Ocurrence <- combine(Ocurrence, countLetter(letters[i], state))
  }
  return (Ocurrence)
}


letter_count <- t(sapply(states$V2, counts))
colnames(letter_count) <- c(letters[1:26])
```

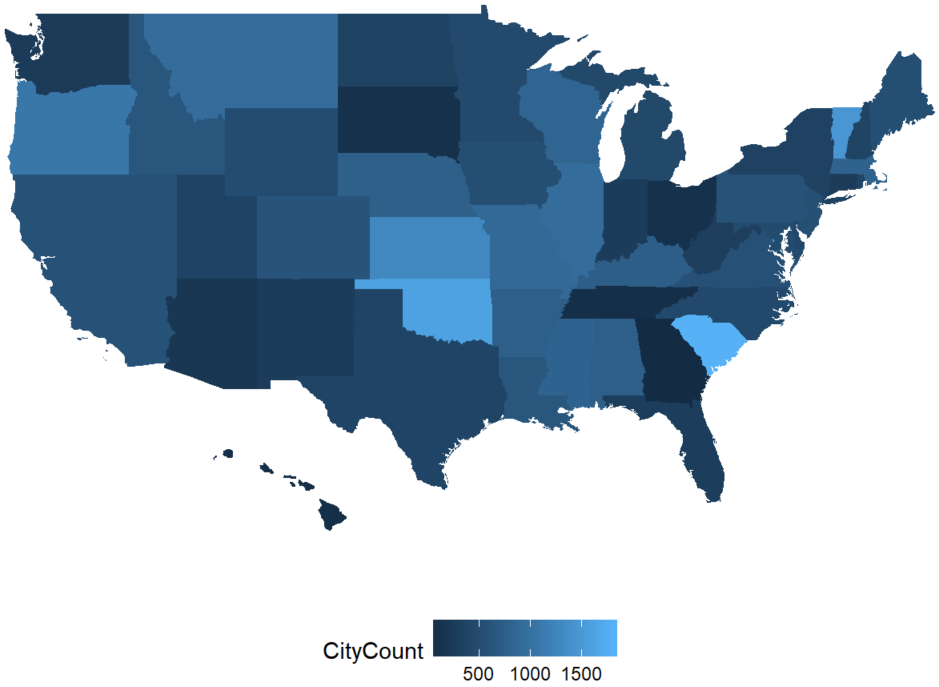
Part d. Create 2 maps to finalize this. Map 1 should be colored by count of cities on our list within the state. Map 2 should highlight only those states that have more than 3 occurances of ANY letter in thier name.

```
####This code creates two graphics of the United States, using ggplot,in which the states with different citycounts are drawn and with 3 or more recurring letters are drawn. The first one counts the citycounts adds an lower case column of state names and put into graph using ggplot. The second map with 3 or more recurring letters is done by creating a variable called letter_count_three that counts if the state name has more than 3 recurring letters in it by for-looping once to go through letters a~z, and once again to go through 50 states, and filling the variable with 'TRUE' if the state had more than or equal to 3 recurring letters

library(ggplot2)
library(fiftystater)

citycounts <- data.frame(states = tolower(states$V2), citycounts)
citycounts <- select(citycounts, states, CityCount)


#Draw first map
f <- ggplot(citycounts, aes(map_id = states)) +
  geom_map(aes(fill = CityCount), 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())
f
```

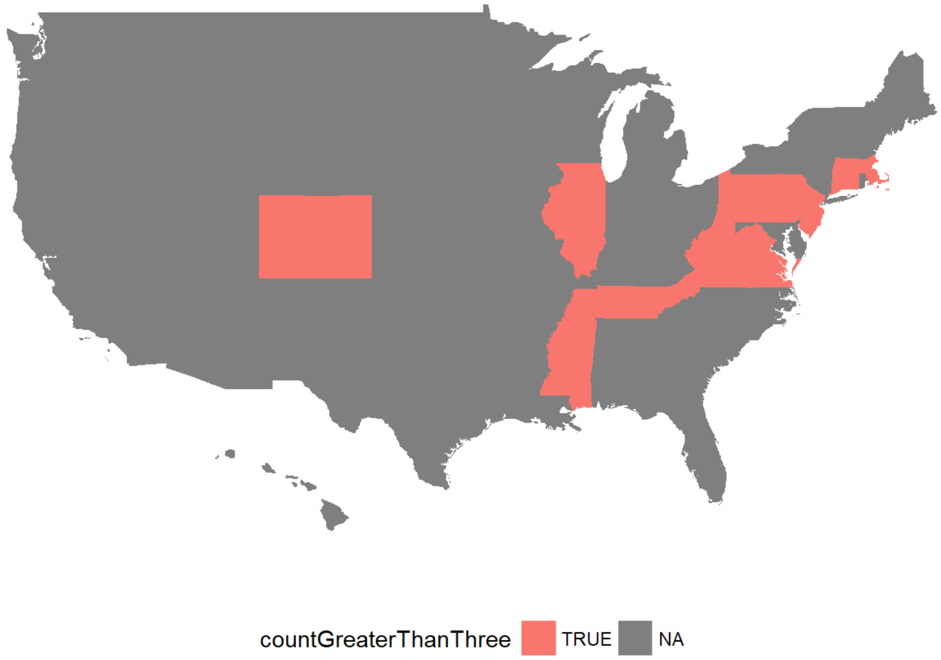


```
letter_count_three <- select(states, V2) %>% mutate(countGreaterThanThree = NA)

for(i in 1:47)
{
  for(j in 1:26)
  {
   if(letter_count[i, j] >= 3)
   letter_count_three[i, 2] <- TRUE
  }
}

letterCounts <- data.frame(state = tolower(states$V2), letter_count_three)


#Draw second map
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