Homework 6

Due Wednesday Oct 9, 9am

Eric Bae

2019-10-14

For each assignment, turn in by the due date/time. Late assignments must be arranged prior to submission. In every case, assignments are to be typed neatly using proper English in Markdown.

This week, we spoke about the apply family of functions. We can use these functions to simplify our code (ie our job) if we can create functions. Ultimately, our goal is to find deficiencies and explore relationships in data and quantify these relationships. Efficiently. So, functions and methods to use these functions could be helpful in some scenarios.

Problem 1

Work through the Swirl "R_programming_E" lesson parts 10 and 11, and perhaps 12 if you need some help with things important to Chris' class (there is also a set of swirl lessons on probability…).

Problem 2

As in the last homework, create a new R Markdown file (file–>new–>R Markdown–>save as.

The filename should be: HWXX_lastname_firstname, i.e. for me it would be HWXX_Settlage_Bob

You will use this new R Markdown file to solve the following problems:

Problem 3

a. Create a function that computes the proportion of successes in a vector. Use good programming practices.

```
prop_success <- function(vec) {
  # Assuming the vector is a boolean/numeric character,
  # where TRUE or 1 incidates success,
  # and FALSE or 0 indicates failure.
  return(sum(vec)/length(vec))
}
```

b. Create a matrix to simulate 10 flips of a coin with varying degrees of "fairness" (columns = probability) as follows:

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

c. Use your function in conjunction with apply to compute the proportion of success in P4b_data by column and then by row. What do you observe? What is going on?

```
sapply(1:ncol(P4b_data), function(k) prop_success(P4b_data[,k]))
```

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

```
sapply(1:nrow(P4b_data), function(k) prop_success(P4b_data[k,]))
```

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

It appears that the proportion of success is exactly 0.6 for every column, while the proportion fo success is 1 for the first 4 and the last 2 rows and 0 for the rest.

What appears to be happening is the "rbinom(10, 1, prob = (30:40)/100)" is only generating a vector of length 10 because according to the function, it does not take a vector input for the argument "prob," which meant only the first argument of the (30:40)/100 was used.

Even though only a single vector of 10 was generated, we are forcing it into a 10 × 10 matrix, we are simply copying the vector 10 times by column (since we said byrow = F). That's why every column is identical to one another.

d. You are to fix the above matrix by creating a function whose input is a probability and output is a vector whose elements are the outcomes of 10 flips of a coin. Now create a vector of the desired probabilities. Using the appropriate apply family function, create the matrix we really wanted above. Prove this has worked by using the function created in part a to compute and tabulate the appropriate marginal successes.

```
coin_outcome <- function(prob) {
  # probability has to be a single value
  outcome <- rbinom(10, 1, prob)
  return(outcome)
}

prob <- 30:40/100
outcome_matrix <- sapply(prob, function(k) coin_outcome(k))
outcome_matrix
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    0    0    1    1    1    1    1    1    1     0     1
##  [2,]    0    0    0    0    1    0    0    0    1     1     0
##  [3,]    1    1    0    1    0    1    0    0    0     1     1
##  [4,]    0    1    1    1    0    1    0    0    0     1     0
##  [5,]    0    0    0    0    1    1    0    0    1     0     1
##  [6,]    0    0    0    0    0    0    0    0    0     1     1
##  [7,]    0    1    1    0    1    1    1    1    1     1     1
##  [8,]    0    0    1    0    0    0    1    0    1     1     0
##  [9,]    0    0    0    0    0    0    0    1    0     0     0
## [10,]    1    0    0    0    0    1    0    0    0     0     0
```

```
sapply(1:10, function(k) prop_success(outcome_matrix[,k]))
```

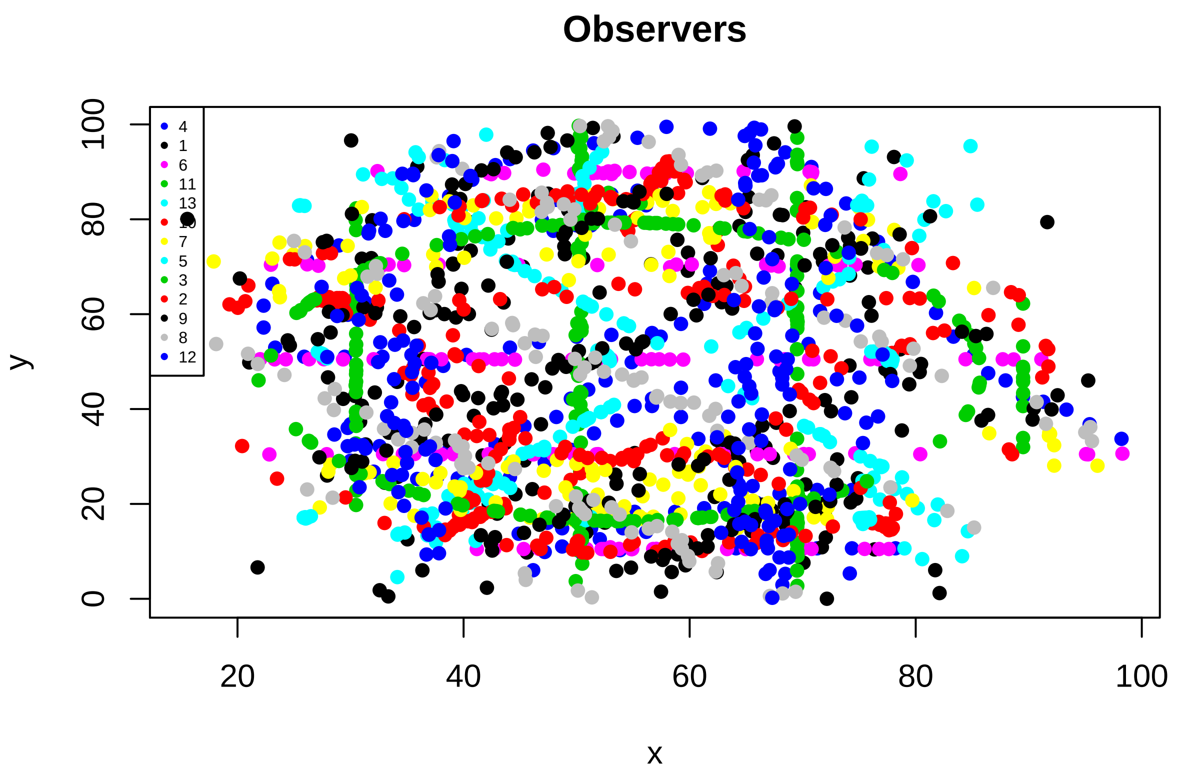
```
##  [1] 0.2 0.3 0.4 0.3 0.4 0.6 0.3 0.3 0.5 0.6
```

Problem 4

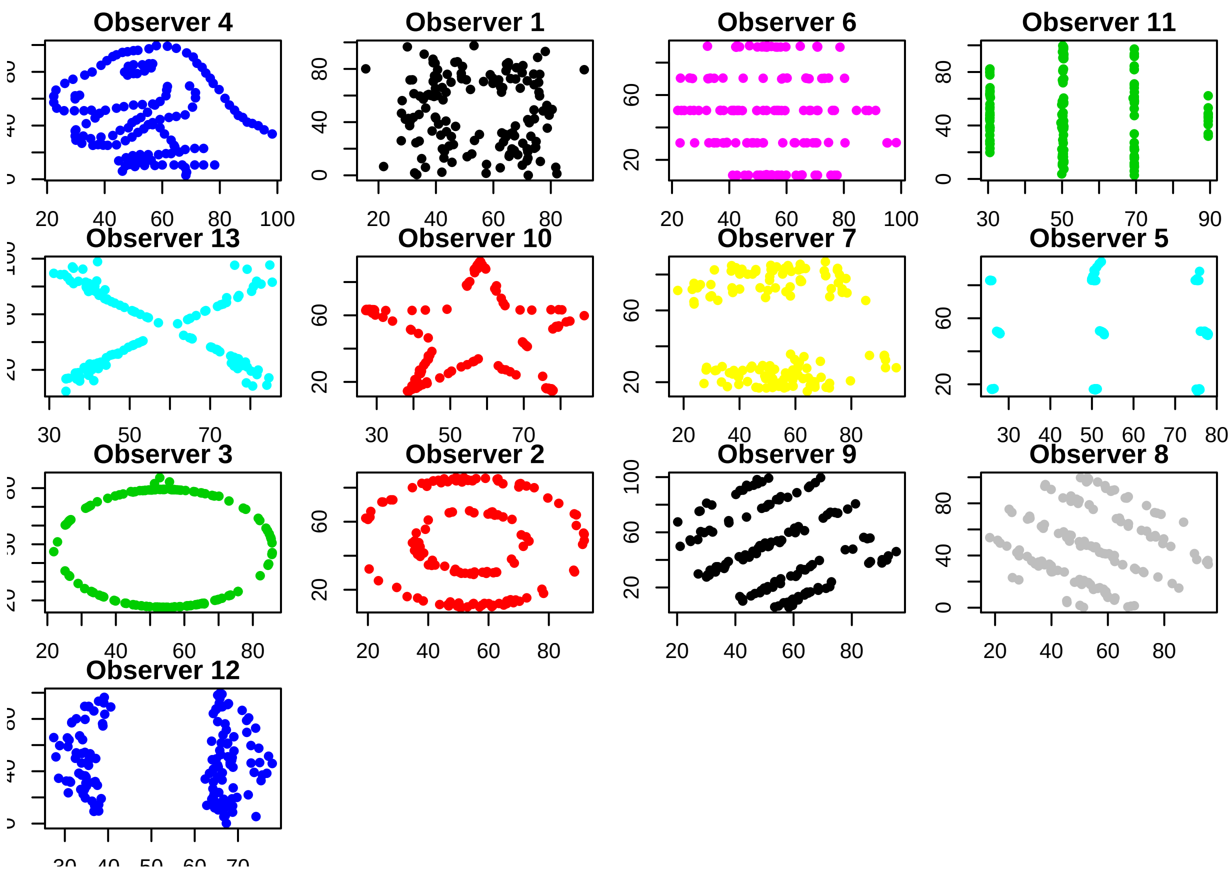
In Homework 4, we had a dataset we were to compute some summary statistics from. The description of the data was given as "a dataset which has multiple repeated measurements from two devices by thirteen Observers". Where the device measurements were in columns "dev1" and "dev2". Reimport that dataset, change the names of "dev1" and "dev2" to x and y and do the following:

1. create a function that accepts a dataframe of values, title, and x/y labels and creates a scatter plot
2. use this function to create:
   (a) a single scatter plot of the entire dataset
   (b) a seperate scatter plot for each observer (using the apply function)

```r
dat <- readRDS("/Users/ericb/Desktop/STAT_5014/Hw4_data.rds")
names(dat)[2:3] <- c("x", "y")

#png("C:/Users/ericb/Desktop/STAT_5014/temp.png")
plot_data <- function(dat, obs = unique(dat$Observer)) {
  dat_new <- dat[which(dat$Observer %in% obs),]
  observer <- dat_new$Observer
  plot(dat_new$x, dat_new$y, xlab = "x", ylab = "y",
       col = observer, pch = 16)
  if (length(obs) == 1) {
    title(paste("Observer", obs), add = T)
  }
  else if (length(obs) != 1) {
    title("Observers", add = T)
  }
}

# Single scatter plot of the entire dataset
obs <- unique(dat$Observer)
plot_data(dat)
legend("topleft", legend = paste(obs), col = obs, pch = 16, cex = 0.5)
```



```r
# Multiple scatter plots
par(mfrow = c(4, 4))
par(mar = c(1.5, 1.5, 1.5, 1.5))
p0 <- sapply(obs, function(k) plot_data(dat, k))

#vapply(dat, plot_data, observer = obs)
#apply(dat, 1, plot_data)
```



```
#knitr::include_graphics("C:/Users/ericb/Desktop/STAT_5014/temp.png")
```

Problem 5

Our ultimate goal in this problem is to create an annotated map of the US. I am giving you the code to create said map, you will need to customize it to include the annotations.

Part a. Get and import a database of US cities and states. Here is some R code to help:

```
    #we are grabbing a SQL set from here
    # http://www.farinspace.com/wp-content/uploads/us_cities_and_states.zip
    #download the files, looks like it is a .zip
    library(downloader)
    download("http://www.farinspace.com/wp-content/uploads/us_cities_and_states.zip",dest="us_cities_sta
    unzip("us_cities_states.zip", exdir = ".")

    #read in data, looks like sql dump, blah
    library(data.table)
    states <- fread(input = "./us_cities_and_states/states.sql",
                    skip = 23, sep = "'", sep2 = ",",
                    header = F, select = c(2,4))
    ### YOU do the CITIES
    ### I suggest the cities_extended.sql may have everything you need
    ### can you figure out how to limit this to the 50?
    cities <- fread(input = "./us_cities_and_states/cities_extended.sql",
                    sep = "'", sep2 = ",",
```

```r
                    header = F, select = 2*(1:6))
    names(cities) <- c("name", "state", "zip", "N", "W", "county")
    cities <- cities[-which(cities$state %in% c("DC", "PR")),]
```

Part b. Create a summary table of the number of cities included by state.

```r
library(pander)
cities_table <- table(cities$state, dnn = "States")
pander(cities_table)
```

Table 1: Table continues below

AK	AL	AR	AZ	CA	CO	CT	DE	FL	GA	HI	IA	ID
273	838	709	532	2651	659	438	98	1487	972	139	1060	325

Table 2: Table continues below

IL	IN	KS	KY	LA	MA	MD	ME	MI	MN	MO	MS
1587	989	756	961	725	703	619	489	1170	1031	1170	533

Table 3: Table continues below

MT	NC	ND	NE	NH	NJ	NM	NV	NY	OH	OK	OR
405	1090	407	620	284	733	426	253	2207	1446	774	484

PA	RI	SC	SD	TN	TX	UT	VA	VT	WA	WI	WV	WY
2208	91	539	394	795	2650	344	1238	309	732	898	859	195

Part c. Create a function that counts the number of occurances of a letter in a string. The input to the function should be "letter" and "state_name". The output should be a scalar with the count for that letter.

Create a for loop to loop through the state names imported in part a. Inside the for loop, use an apply family function to iterate across a vector of letters and collect the occurance count as a vector.

```r
letter_count <- data.frame(matrix(NA, nrow = 50, ncol = 26))
colnames(letter_count) <- letters
rownames(letter_count) <- state.name
getCount <- function(letter, state_name) {
  state_name_lower <- tolower(state_name)
  state_name_spl <- strsplit(state_name_lower, "")[[1]]
  count <- length(which(state_name_spl == letter))
return(count)
}
for (i in 1:50){
  letter_count[i,] <- sapply(letters, function(k) getCount(k, state.name[i]))
}
pander(letter_count)
```

Table 5: Table continues below

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Alabama	4	1	0	0	0	0	0	0	0	0	0	1	1	0	0
Alaska	3	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Arizona	2	0	0	0	0	0	0	0	1	0	0	0	0	1	1
Arkansas	3	0	0	0	0	0	0	0	0	0	1	0	0	1	0
California	2	0	1	0	0	1	0	0	2	0	0	1	0	1	1
Colorado	1	0	1	1	0	0	0	0	0	0	0	1	0	0	3
Connecticut	0	0	3	0	1	0	0	0	1	0	0	0	0	2	1
Delaware	2	0	0	1	2	0	0	0	0	0	0	1	0	0	0
Florida	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1
Georgia	1	0	0	0	1	0	2	0	1	0	0	0	0	0	1
Hawaii	2	0	0	0	0	0	0	1	2	0	0	0	0	0	0
Idaho	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1
Illinois	0	0	0	0	0	0	0	0	3	0	0	2	0	1	1
Indiana	2	0	0	1	0	0	0	0	2	0	0	0	0	2	0
Iowa	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1
Kansas	2	0	0	0	0	0	0	0	0	0	1	0	0	1	0
Kentucky	0	0	1	0	1	0	0	0	0	0	2	0	0	1	0
Louisiana	2	0	0	0	0	0	0	0	2	0	0	1	0	1	1
Maine	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0
Maryland	2	0	0	1	0	0	0	0	0	0	0	1	1	1	0
Massachusetts	2	0	1	0	1	0	0	1	0	0	0	0	1	0	0
Michigan	1	0	1	0	0	0	1	1	2	0	0	0	1	1	0
Minnesota	1	0	0	0	1	0	0	0	1	0	0	0	1	2	1
Mississippi	0	0	0	0	0	0	0	0	4	0	0	0	1	0	0
Missouri	0	0	0	0	0	0	0	0	2	0	0	0	1	0	1
Montana	2	0	0	0	0	0	0	0	0	0	0	0	1	2	1
Nebraska	2	1	0	0	1	0	0	0	0	0	1	0	0	1	0
Nevada	2	0	0	1	1	0	0	0	0	0	0	0	0	1	0
New Hampshire	1	0	0	0	2	0	0	2	1	0	0	0	1	1	0
New Jersey	0	0	0	0	3	0	0	0	0	1	0	0	0	1	0
New Mexico	0	0	1	0	2	0	0	0	1	0	0	0	1	1	1
New York	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1
North Carolina	2	0	1	0	0	0	0	1	1	0	0	1	0	2	2
North Dakota	2	0	0	1	0	0	0	1	0	0	1	0	0	1	2
Ohio	0	0	0	0	0	0	0	1	1	0	0	0	0	0	2
Oklahoma	2	0	0	0	0	0	0	1	0	0	1	1	1	0	2
Oregon	0	0	0	0	1	0	1	0	0	0	0	0	0	1	2
Pennsylvania	2	0	0	0	1	0	0	0	1	0	0	1	0	3	0
Rhode Island	1	0	0	2	1	0	0	1	1	0	0	1	0	1	1
South Carolina	2	0	1	0	0	0	0	1	1	0	0	1	0	1	2
South Dakota	2	0	0	1	0	0	0	1	0	0	1	0	0	0	2
Tennessee	0	0	0	0	4	0	0	0	0	0	0	0	0	2	0
Texas	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Utah	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Vermont	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1
Virginia	1	0	0	0	0	0	1	0	3	0	0	0	0	1	0
Washington	1	0	0	0	0	0	1	1	1	0	0	0	0	2	1
West Virginia	1	0	0	0	1	0	1	0	3	0	0	0	0	1	0

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Wisconsin	0	0	1	0	0	0	0	0	2	0	0	0	0	2	1
Wyoming	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1

	p	q	r	s	t	u	v	w	x	y	z
Alabama	0	0	0	0	0	0	0	0	0	0	0
Alaska	0	0	0	1	0	0	0	0	0	0	0
Arizona	0	0	1	0	0	0	0	0	0	0	1
Arkansas	0	0	1	2	0	0	0	0	0	0	0
California	0	0	1	0	0	0	0	0	0	0	0
Colorado	0	0	1	0	0	0	0	0	0	0	0
Connecticut	0	0	0	0	2	1	0	0	0	0	0
Delaware	0	0	1	0	0	0	0	1	0	0	0
Florida	0	0	1	0	0	0	0	0	0	0	0
Georgia	0	0	1	0	0	0	0	0	0	0	0
Hawaii	0	0	0	0	0	0	0	1	0	0	0
Idaho	0	0	0	0	0	0	0	0	0	0	0
Illinois	0	0	0	1	0	0	0	0	0	0	0
Indiana	0	0	0	0	0	0	0	0	0	0	0
Iowa	0	0	0	0	0	0	0	1	0	0	0
Kansas	0	0	0	2	0	0	0	0	0	0	0
Kentucky	0	0	0	0	1	1	0	0	0	1	0
Louisiana	0	0	0	1	0	1	0	0	0	0	0
Maine	0	0	0	0	0	0	0	0	0	0	0
Maryland	0	0	1	0	0	0	0	0	0	1	0
Massachusetts	0	0	0	4	2	1	0	0	0	0	0
Michigan	0	0	0	0	0	0	0	0	0	0	0
Minnesota	0	0	0	1	1	0	0	0	0	0	0
Mississippi	2	0	0	4	0	0	0	0	0	0	0
Missouri	0	0	1	2	0	1	0	0	0	0	0
Montana	0	0	0	0	1	0	0	0	0	0	0
Nebraska	0	0	1	1	0	0	0	0	0	0	0
Nevada	0	0	0	0	0	0	1	0	0	0	0
New Hampshire	1	0	1	1	0	0	0	1	0	0	0
New Jersey	0	0	1	1	0	0	0	1	0	1	0
New Mexico	0	0	0	0	0	0	0	1	1	0	0
New York	0	0	1	0	0	0	0	1	0	1	0
North Carolina	0	0	2	0	1	0	0	0	0	0	0
North Dakota	0	0	1	0	2	0	0	0	0	0	0
Ohio	0	0	0	0	0	0	0	0	0	0	0
Oklahoma	0	0	0	0	0	0	0	0	0	0	0
Oregon	0	0	1	0	0	0	0	0	0	0	0
Pennsylvania	1	0	0	1	0	0	1	0	0	1	0
Rhode Island	0	0	1	1	0	0	0	0	0	0	0
South Carolina	0	0	1	1	1	1	0	0	0	0	0
South Dakota	0	0	0	1	2	1	0	0	0	0	0
Tennessee	0	0	0	2	1	0	0	0	0	0	0
Texas	0	0	0	1	1	0	0	0	1	0	0
Utah	0	0	0	0	1	1	0	0	0	0	0
Vermont	0	0	1	0	1	0	1	0	0	0	0
Virginia	0	0	1	0	0	0	1	0	0	0	0
Washington	0	0	0	1	1	0	0	1	0	0	0

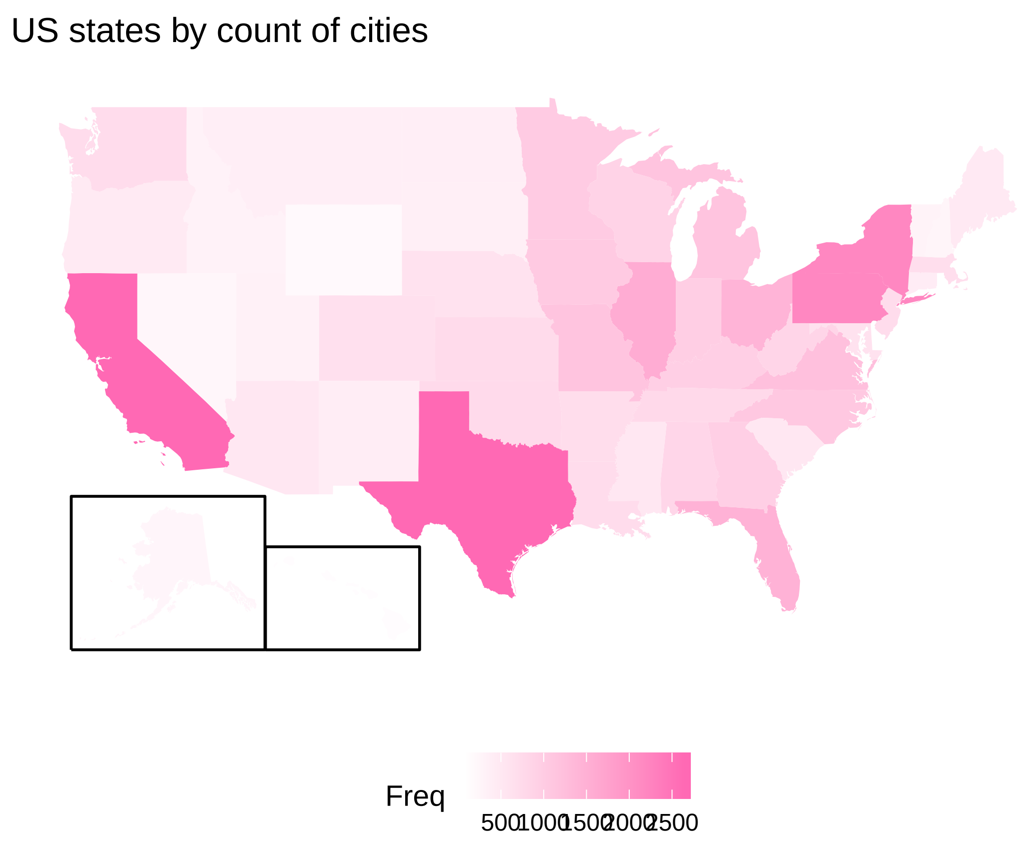
	p	q	r	s	t	u	v	w	x	y	z
West Virginia	0	0	1	1	1	0	1	1	0	0	0
Wisconsin	0	0	0	2	0	0	0	1	0	0	0
Wyoming	0	0	0	0	0	0	0	1	0	1	0

Part d.

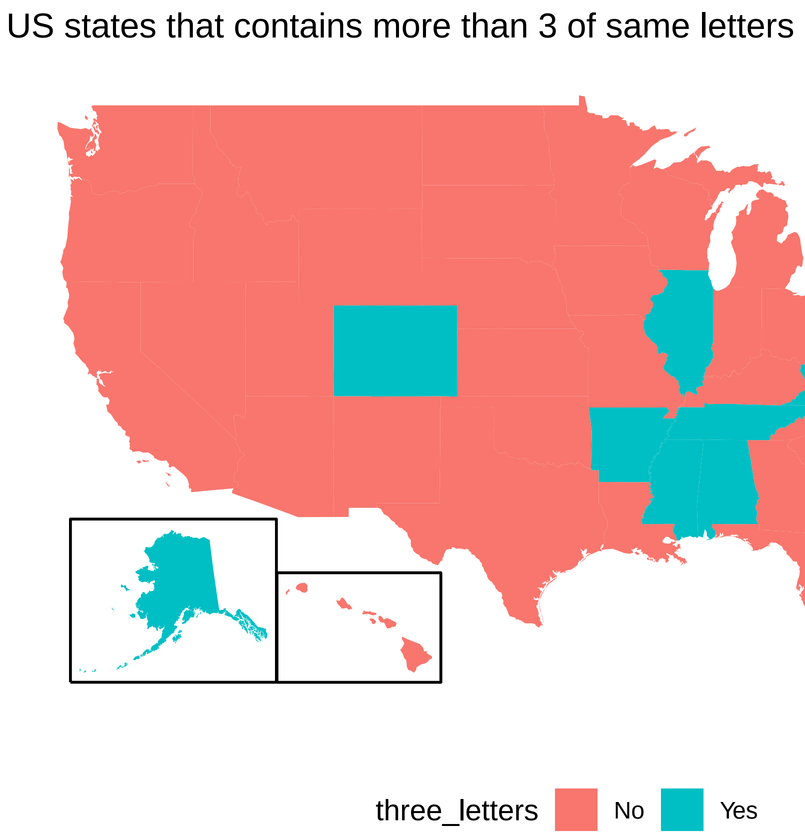
Create 2 maps to finalize this. Map 1 should be colored by count of cities on our list within the state. Map 2 should highlight only those states that have more than 3 occurances of ANY letter in their name.

Quick and not so dirty map:

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

cities_states <- cbind.data.frame(
  state = tolower(states$V2[which(states$V2 != "District of Columbia")]),
  cities_table)
# map_id creates the aesthetic mapping to the state name column in your data
p1 <- ggplot(cities_states, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = Freq), 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) +
  scale_fill_gradient(low = "white", high = "hotpink") +
  labs(x = "", y = "") +
  ggtitle("US states by count of cities") +
  fifty_states_inset_boxes() +
  theme(legend.position = "bottom",
        panel.background = element_blank())
p1
```



```r
letter_count_max <- apply(letter_count, 1, max)
letter_count_max <- ifelse(letter_count_max >= 3, "Yes", "No")
letter_count_max <- cbind.data.frame(state = tolower(names(letter_count_max)),
                                     three_letters = letter_count_max)

p2 <- ggplot(letter_count_max, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = three_letters), 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 = "") +
  ggtitle("US states that contains more than 3 of same letters") +
  fifty_states_inset_boxes() +
  theme(legend.position = "bottom",
        panel.background = element_blank())
p2
```



```r
ggsave(plot = p1, file = "HW6_Problem5_Plot1_Bae.pdf")
ggsave(plot = p2, file = "HW6_Problem5_Plot2_Bae.pdf")
```

Problem 6

Push your homework to submit.

Preperation for next class:

Next week we will talk about parallelizing in R. No swirl. :)

To make sure this experience is more reproducible across the class, please get an account in ARC (arc.vt.edu, requests, account request). When you have done this, please go to ondemand.arc.vt.edu, choose “interactive apps” and the Rstudio under Cascades. Please set Rpackage set = “basic tidyverse”, account = “arc-train”, partition to “normal_q”, hours to 1, nodes to 1, and cores to 10. Hit launch.

After about 10 min, you should get a green Rstudio button. After this first time, you should see the startup takes seconds.