Homework 6

Due Wednesday Oct 9, 9am

2019-10-15

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...).

swirl()

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.

Answer

```
ps <- function(x){
  sum(x) / length(x)
}
```

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?

Answer

The proportions of success by row are either 1 or 0, and those by column are all 0.6. Neither of them is correct values.

```r
# by row
apply(P4b_data, MARGIN = 1, ps)
```

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

```r
# by column
apply(P4b_data, MARGIN = 2, ps)
```

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

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.

Answer

```r
set.seed(704)
# fc: flip coin
fc <- function(probability){
  rbinom(10, size = 1, prob = probability)
}

p4d_data <- sapply((30:40)/100, fc)

knitr::kable(apply(p4d_data, 2, ps),
             caption = "Proportions of Success")
```

Table 1: Proportions of Success

x
0.5
0.6
0.2
0.4
0.4
0.5
0.2
0.2
0.2
0.5
0.3

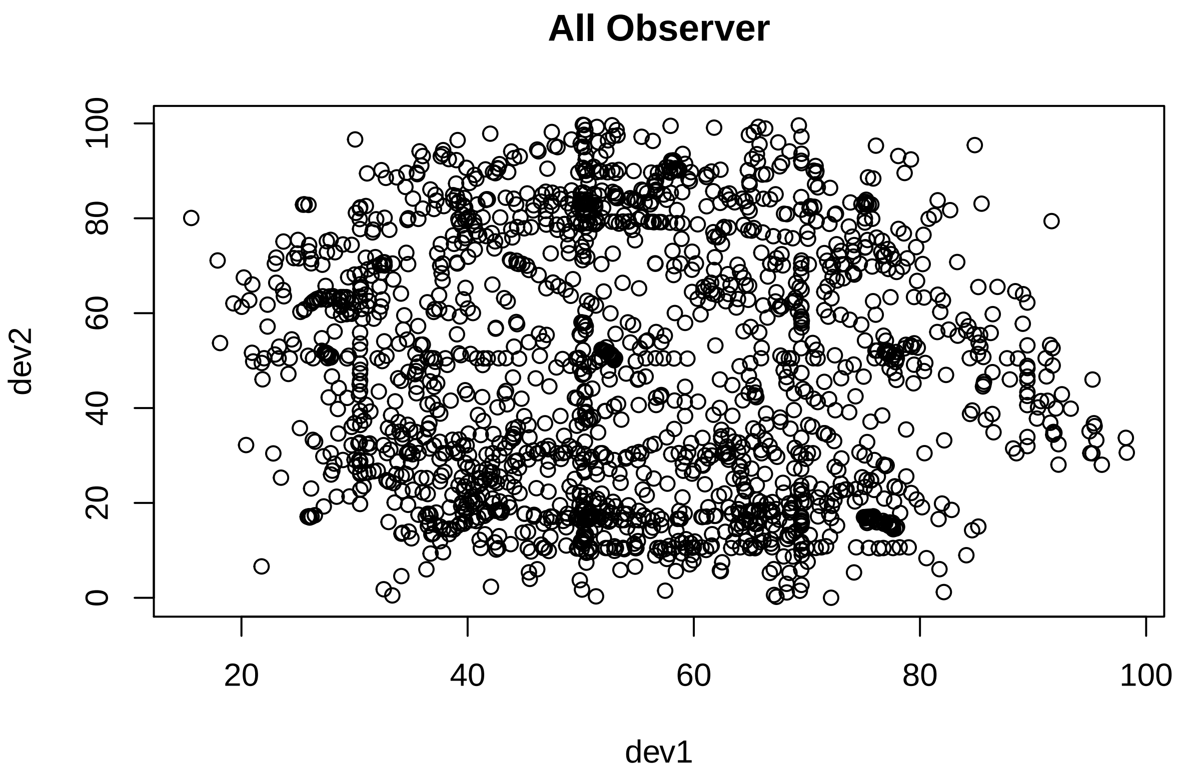
Problem 4

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:

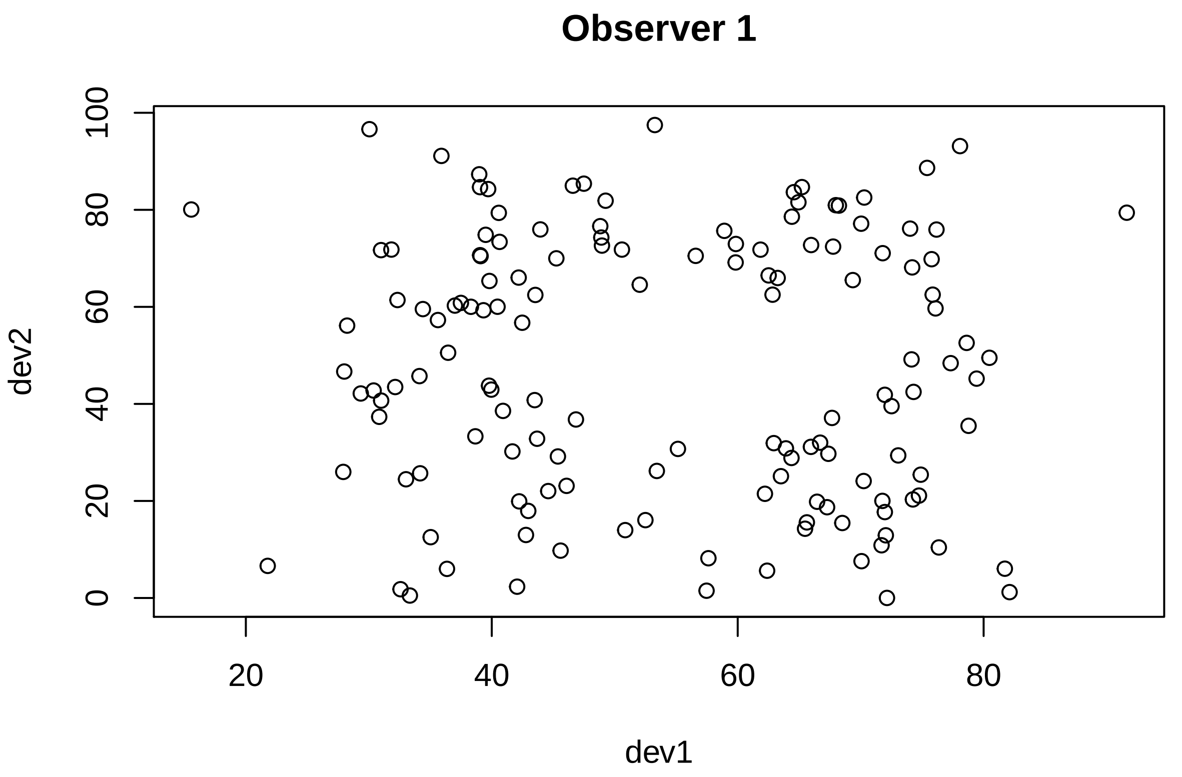
1. create a function that accepts a dataframe of values, title, and x/y labels and creates a scatter plot
2. use this function to create:
   (a) a single scatter plot of the entire dataset
   (b) a seperate scatter plot for each observer (using the apply function)

Answer

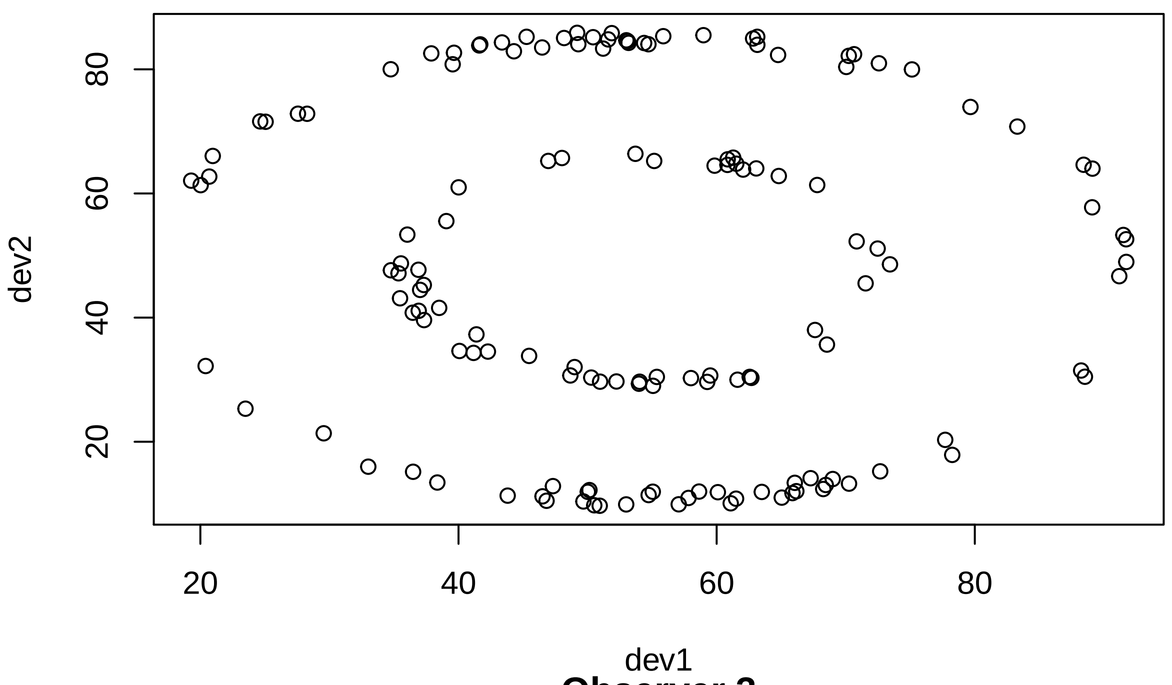
```r
# import dataset
library(tidyverse)
p4df <- readRDS("HW4_data.rds")

# create scatter plot function
sp <- function(df){
  if(length(unique(df[[1]])) > 1){
    t = "All Observer"
  } else{
    t = paste("Observer", unique(df[[1]]))
  }
  plot(df[,2], df[,3],
       xlab = colnames(df)[2],
       ylab = colnames(df)[3],
       main = t
       )
}

# scatter plot for all observers
sp(p4df)
```



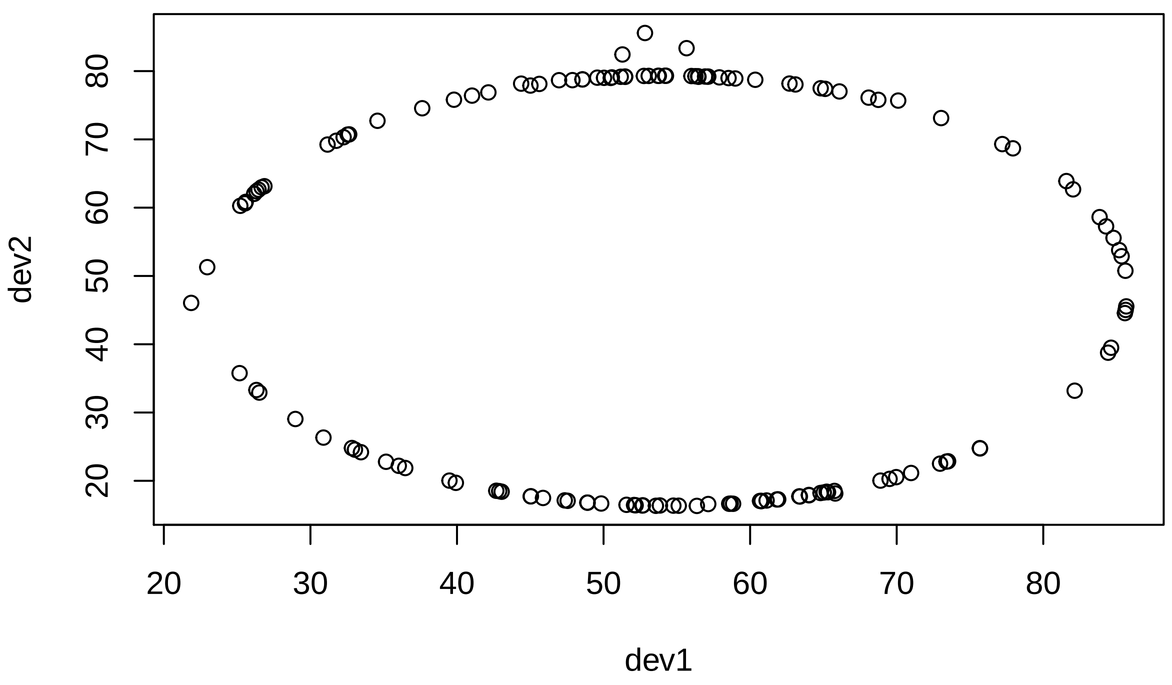
```
# scatter plot for each observer
by(p4df, p4df$Observer, sp)
```

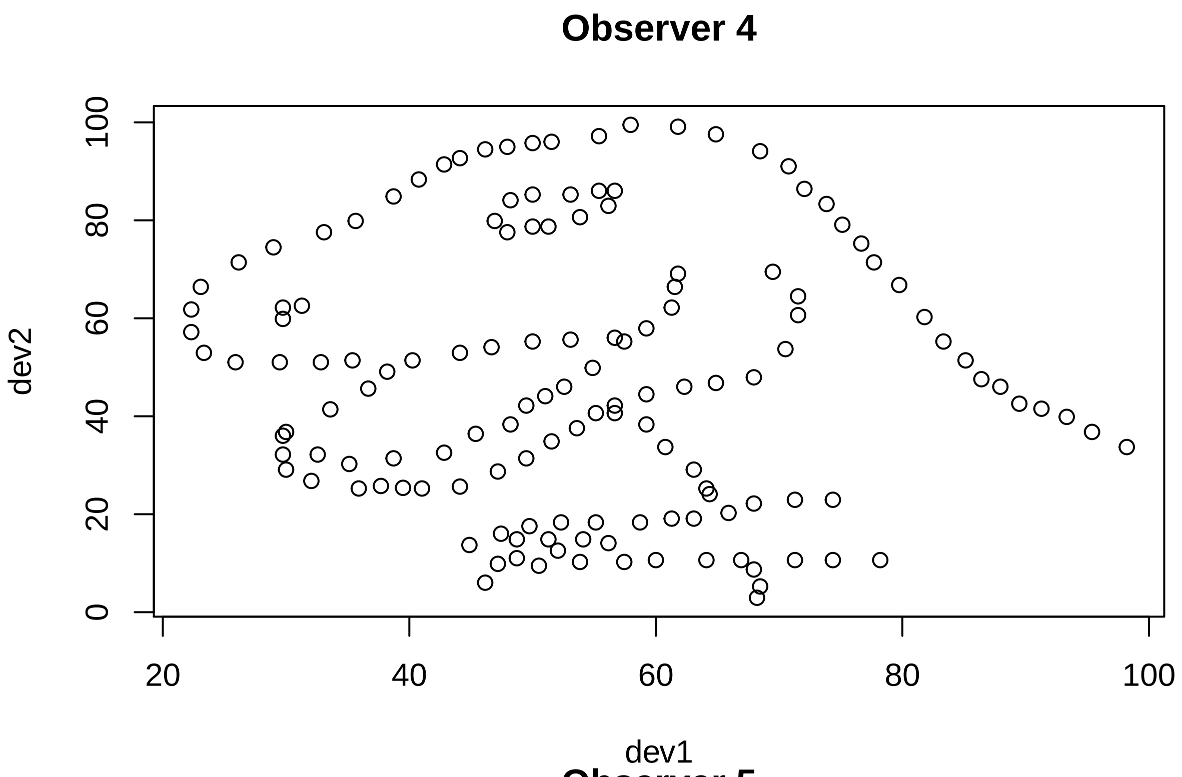


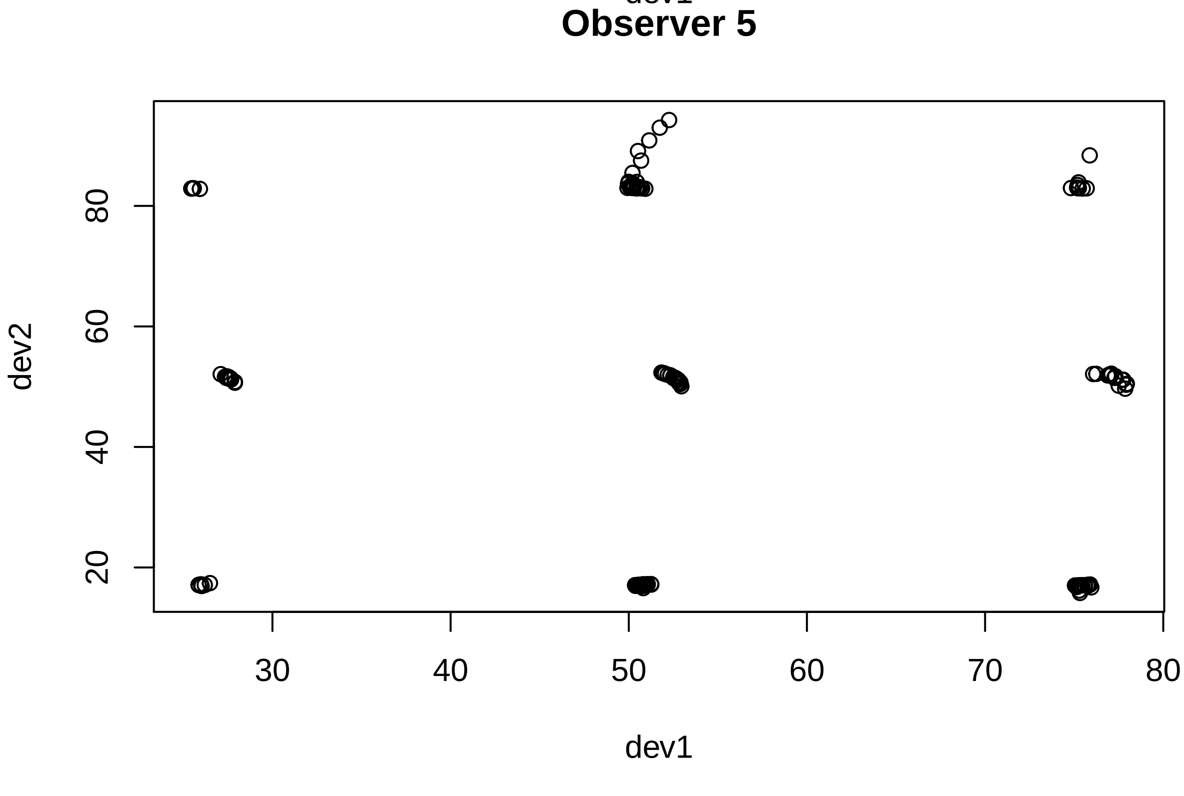
Observer 2



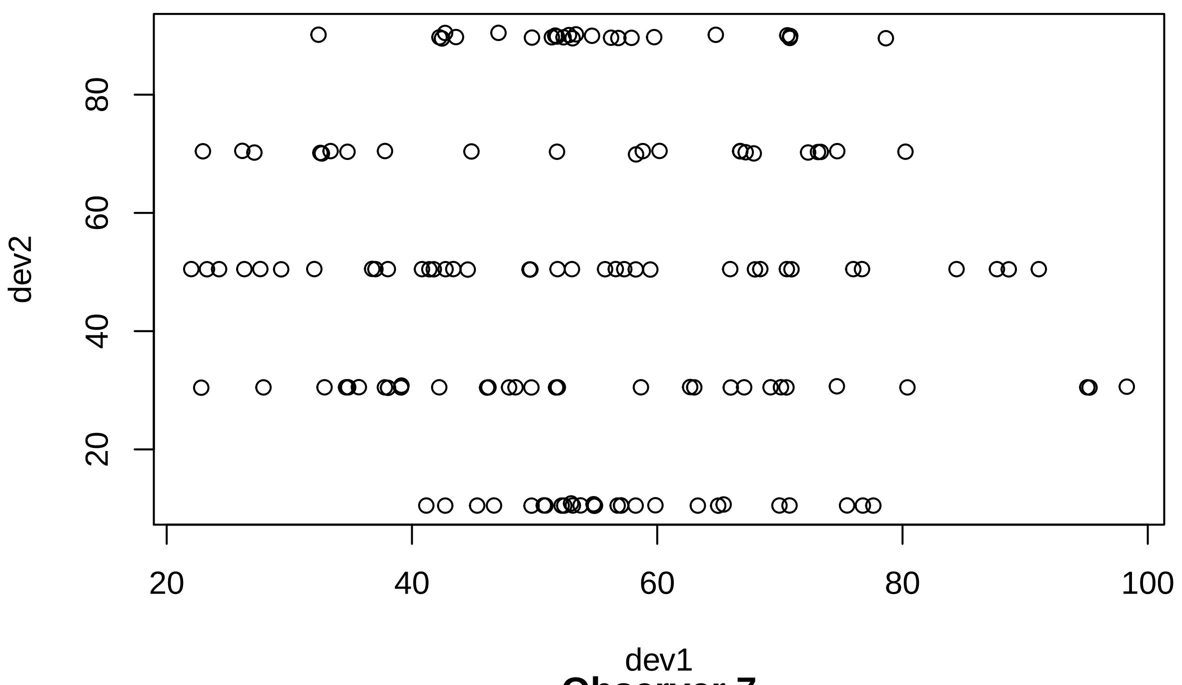
Observer 3



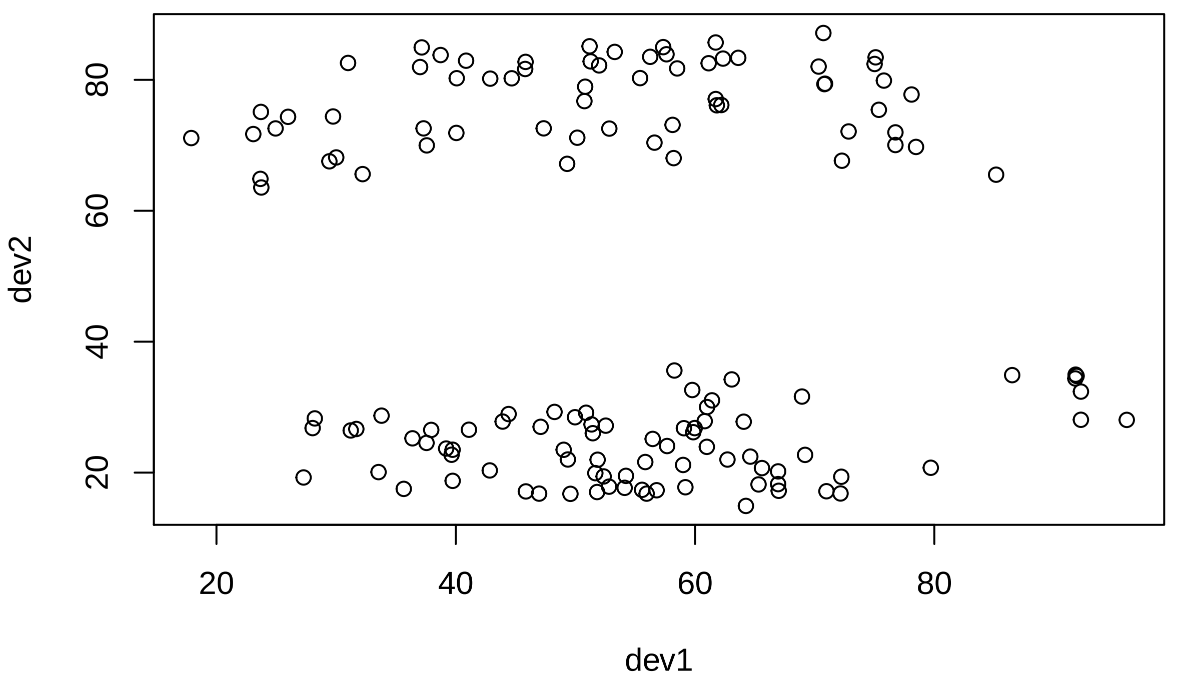

Observer 4
dev2
dev1
0
20
40
60
80
100
20
40
60
80
100


Observer 5
dev2
dev1
20
40
60
80
30
40
50
60
70
80

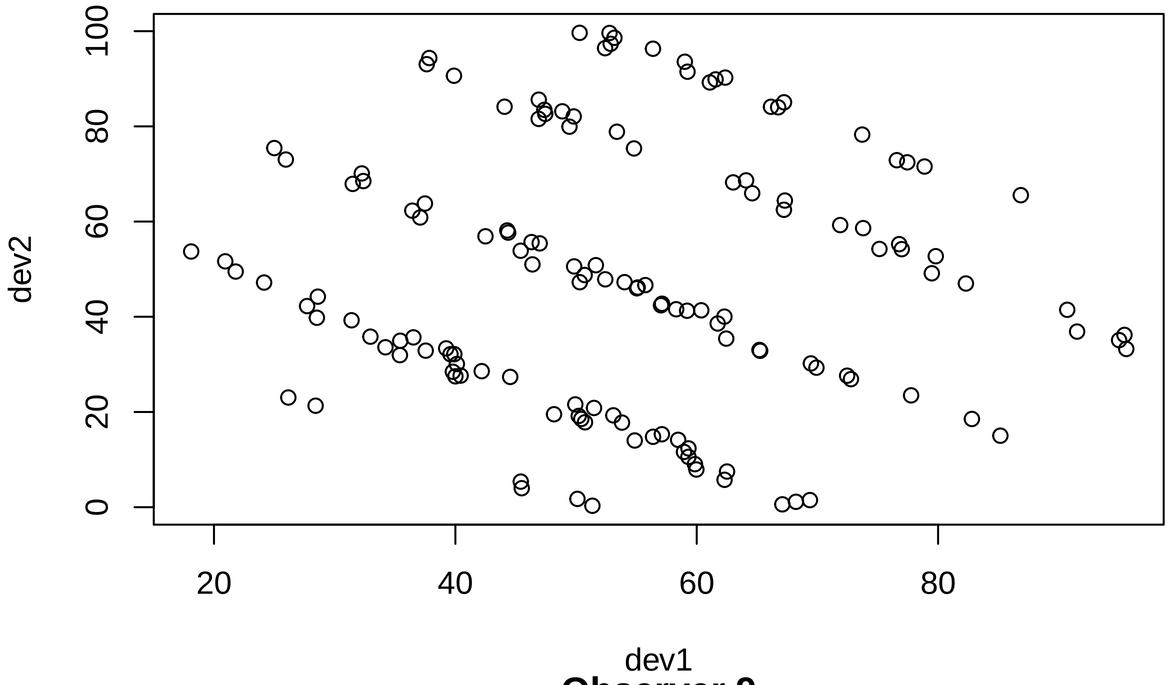
Observer 6



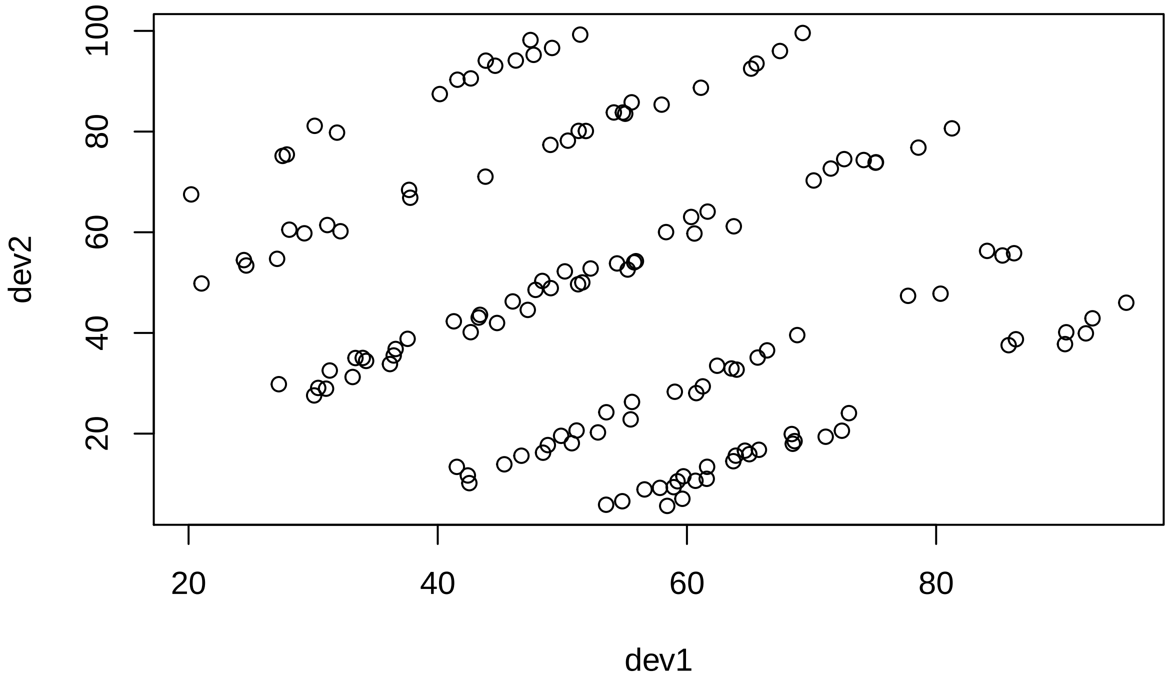
Observer 7

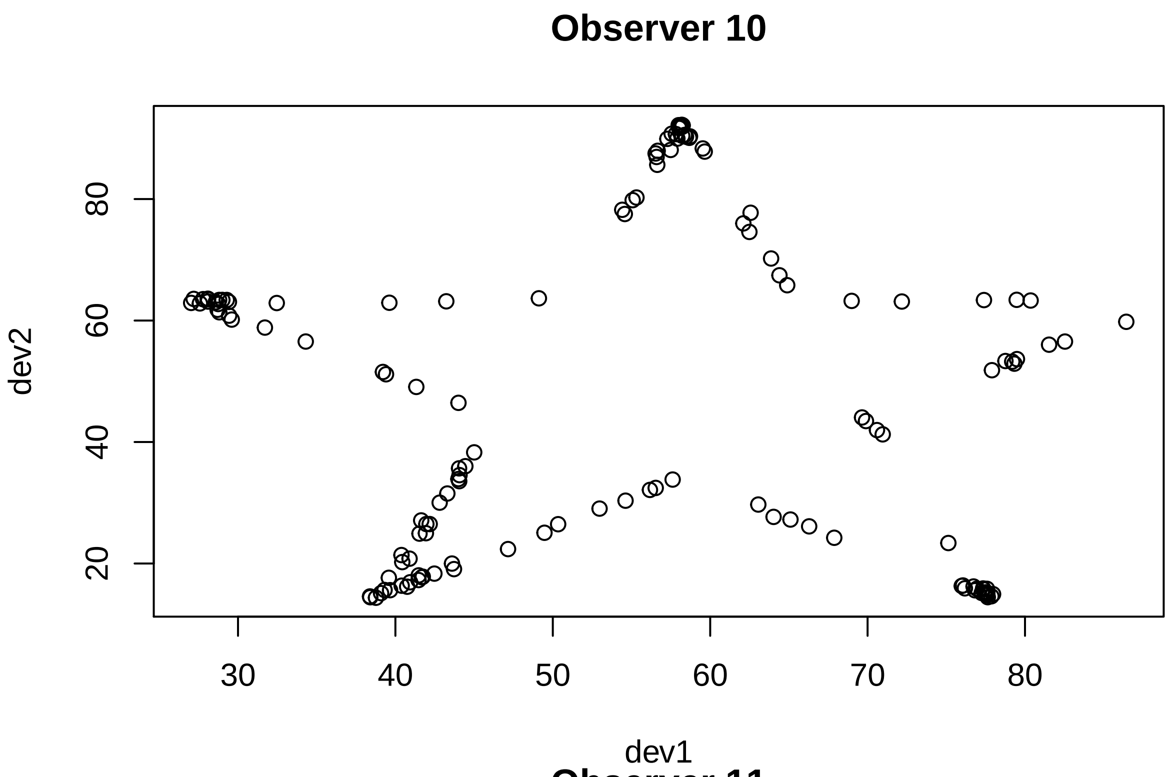


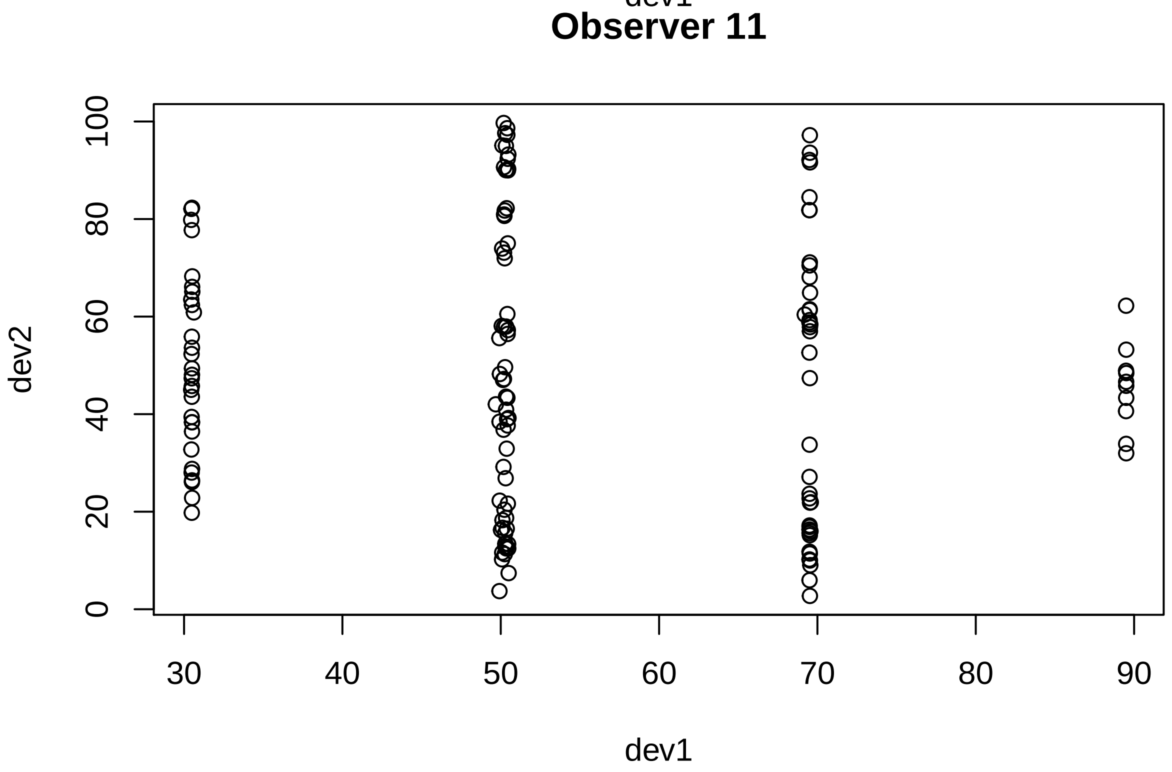
Observer 8

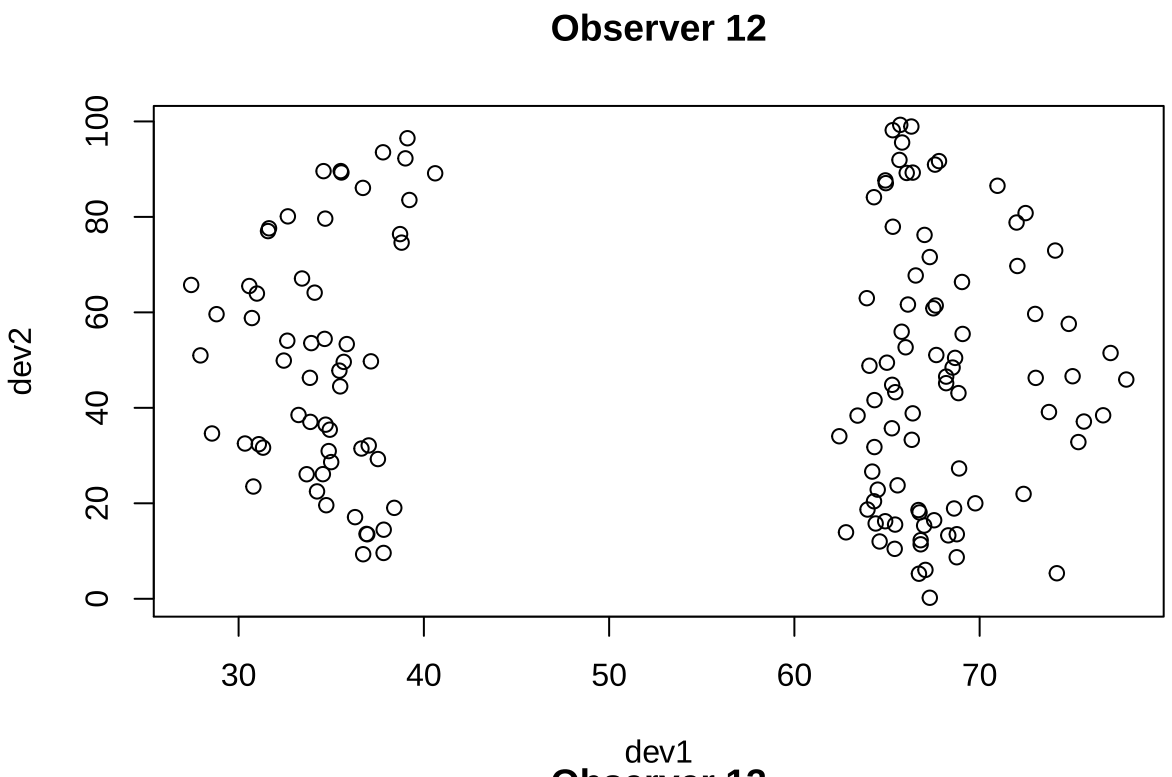


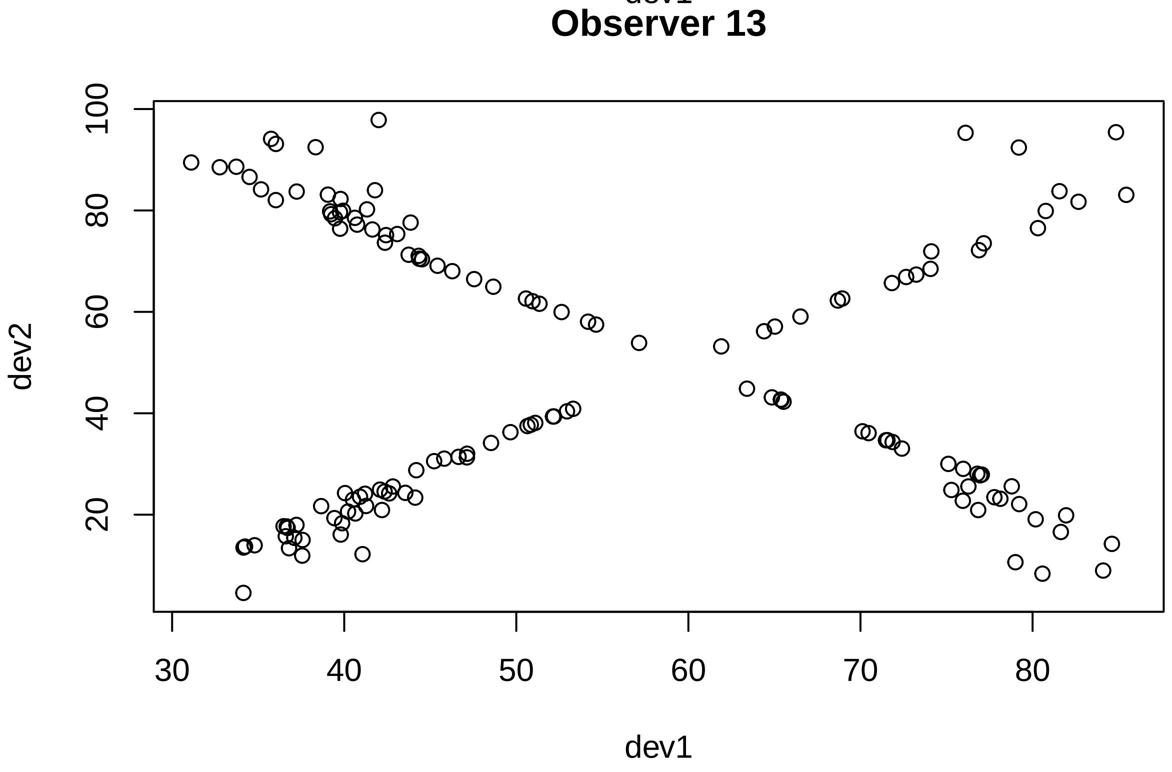
Observer 9




Observer 10
dev2
80
60
40
20
30
40
50
60
70
80
dev1


Observer 11
dev2
100
80
60
40
20
0
30
40
50
60
70
80
90
dev1





```
## p4df$Observer: 1
## NULL
## --------------------------------------------------------
## p4df$Observer: 2
## NULL
## --------------------------------------------------------
## p4df$Observer: 3
## NULL
## --------------------------------------------------------
## p4df$Observer: 4
## NULL
## --------------------------------------------------------
## p4df$Observer: 5
## NULL
## --------------------------------------------------------
## p4df$Observer: 6
## NULL
## --------------------------------------------------------
## p4df$Observer: 7
## NULL
## --------------------------------------------------------
## p4df$Observer: 8
## NULL
## --------------------------------------------------------
## p4df$Observer: 9
## NULL
## --------------------------------------------------------
## p4df$Observer: 10
## NULL
## --------------------------------------------------------
## p4df$Observer: 11
## NULL
## --------------------------------------------------------
## p4df$Observer: 12
## NULL
## --------------------------------------------------------
## p4df$Observer: 13
## NULL
```

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
    ### YOU do the CITIES
    ### I suggest the cities_extended.sql may have everything you need
    cities <- fread(input = "./us_cities_and_states/cities_extended.sql",skip = 23,sep = "'", sep2 = ",
```

```r
    ### can you figure out how to limit this to the 50?
    states <- states[ states$V4 != "DC", ]
    cities <- cities[ cities$V4 != "DC" & cities$V4 != "PR", ]

    # merge two data frame
    sc <- merge(states, cities, by = "V4")
    colnames(sc) <- c("state abbr", "state", "city")
```

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

b. Answer

```r
library(plyr)
n_cities <- ddply(sc, .(state), nrow)
colnames(n_cities) <- c("state", "number of cities")
knitr::kable(n_cities)
```

state	number of cities
Alabama	838
Alaska	273
Arizona	532
Arkansas	709
California	2651
Colorado	659
Connecticut	438
Delaware	98
Florida	1487
Georgia	972
Hawaii	139
Idaho	325
Illinois	1587
Indiana	989
Iowa	1060
Kansas	756
Kentucky	961
Louisiana	725
Maine	489
Maryland	619
Massachusetts	703
Michigan	1170
Minnesota	1031
Mississippi	533
Missouri	1170
Montana	405
Nebraska	620
Nevada	253
New Hampshire	284
New Jersey	733
New Mexico	426
New York	2207
North Carolina	1090
North Dakota	407
Ohio	1446

state	number of cities
Oklahoma	774
Oregon	484
Pennsylvania	2208
Rhode Island	91
South Carolina	539
South Dakota	394
Tennessee	795
Texas	2650
Utah	344
Vermont	309
Virginia	1238
Washington	732
West Virginia	859
Wisconsin	898
Wyoming	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.

c. Answer

```r
##pseudo code
letter_count <- data.frame(matrix(NA,nrow=50, ncol=26))
getCount <- function(letter, state_name){
    temp <- strsplit(state_name, split = "")
    # transform all letters into lower case
    temp <- tolower(unlist(temp))
    # count
    c <- function(x){
      sum(temp == x)
    }
    count <- sapply(letter, c)
    return(count)
}

for(i in 1:50){
    letter_count[i,] <- sapply(states$V2[i], getCount, letter = letters)
}

colnames(letter_count) <- letters
rownames(letter_count) <- states$V2
knitr::kable(letter_count)
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alaska	3	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Alabama	4	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Arkansas	3	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	2	0	0	0	0	0	0	0
Arizona	2	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
California	2	0	1	0	0	1	0	0	2	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0
Colorado	1	0	1	1	0	0	0	0	0	0	0	1	0	0	3	0	0	1	0	0	0	0	0	0	0	0
Connecticut	0	0	3	0	1	0	0	0	1	0	0	0	0	2	1	0	0	0	0	2	1	0	0	0	0	0
Delaware	2	0	0	1	2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0
Florida	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Georgia	1	0	0	0	1	0	2	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Hawaii	2	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Iowa	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
Idaho	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Illinois	0	0	0	0	0	0	0	0	3	0	0	2	0	1	1	0	0	0	1	0	0	0	0	0	0	0
Indiana	2	0	0	1	0	0	0	0	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
Kansas	2	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0
Kentucky	0	0	1	0	1	0	0	0	0	0	2	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0
Louisiana	2	0	0	0	0	0	0	0	2	0	0	1	0	1	1	0	0	0	1	0	1	0	0	0	0	0
Massachusetts	2	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	4	2	1	0	0	0	0	0
Maryland	2	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0
Maine	1	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Michigan	1	0	1	0	0	0	1	1	2	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Minnesota	1	0	0	0	1	0	0	0	1	0	0	0	1	2	1	0	0	0	1	1	0	0	0	0	0	0
Missouri	0	0	0	0	0	0	0	0	2	0	0	0	1	0	1	0	0	1	2	0	1	0	0	0	0	0
Mississippi	0	0	0	0	0	0	0	0	4	0	0	0	1	0	0	2	0	0	4	0	0	0	0	0	0	0
Montana	2	0	0	0	0	0	0	0	0	0	0	0	1	2	1	0	0	0	0	1	0	0	0	0	0	0
North Carolina	2	0	1	0	0	0	0	1	1	0	0	1	0	2	2	0	0	2	0	1	0	0	0	0	0	0
North Dakota	2	0	0	1	0	0	0	1	0	0	1	0	0	1	2	0	0	1	0	2	0	0	0	0	0	0
Nebraska	2	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0
New Hampshire	1	0	0	0	2	0	0	2	1	0	0	0	1	1	0	1	0	1	1	0	0	0	1	0	0	0
New Jersey	0	0	0	0	3	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0
New Mexico	0	0	1	0	2	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	0	0
Nevada	2	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
New York	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0
Ohio	0	0	0	0	0	0	0	1	1	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
Oklahoma	2	0	0	0	0	0	0	1	0	0	1	1	1	0	2	0	0	0	0	0	0	0	0	0	0	0
Oregon	0	0	0	0	1	0	1	0	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0	0	0	0
Pennsylvania	2	0	0	0	1	0	0	0	1	0	0	1	0	3	0	1	0	0	1	0	0	1	0	0	1	0
Rhode Island	1	0	0	2	1	0	0	1	1	0	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0
South Carolina	2	0	1	0	0	0	0	1	1	0	0	1	0	1	2	0	0	1	1	1	1	0	0	0	0	0
South Dakota	2	0	0	1	0	0	0	1	0	0	1	0	0	0	2	0	0	0	1	2	1	0	0	0	0	0
Tennessee	0	0	0	0	4	0	0	0	0	0	0	0	0	2	0	0	0	0	2	1	0	0	0	0	0	0
Texas	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0
Utah	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Virginia	1	0	0	0	0	0	1	0	3	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0
Vermont	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	0	0	0
Washington	1	0	0	0	0	0	1	1	1	0	0	0	0	2	1	0	0	0	1	1	0	0	1	0	0	0
Wisconsin	0	0	1	0	0	0	0	0	2	0	0	0	0	2	1	0	0	0	2	0	0	0	1	0	0	0
West Virginia	1	0	0	0	1	0	1	0	3	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	0	0
Wyoming	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1	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 thier name.

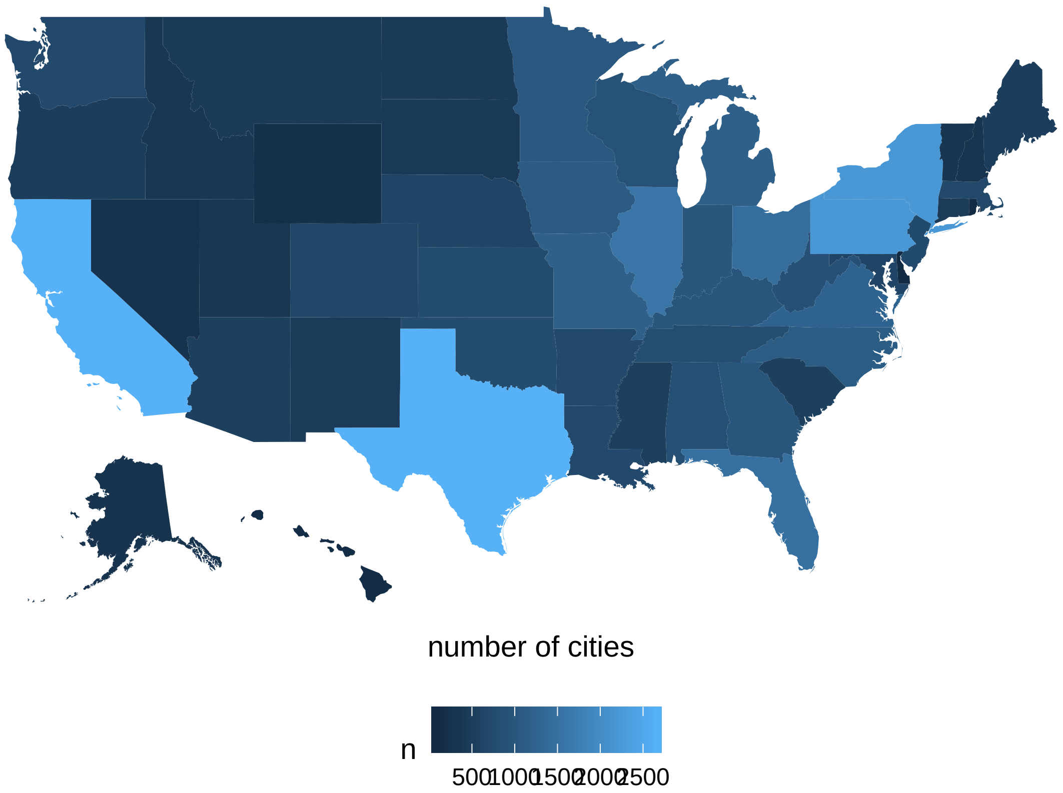
Quick and not so dirty map:

Answer

map 1

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

data("fifty_states") # this line is optional due to lazy data loading
df_mp2 <- data.frame(state = as.factor(tolower(n_cities$state)), n = n_cities$`number of cities`)
# map_id creates the aesthetic mapping to the state name column in your data
p2 <- ggplot(df_mp2, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = n), 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 = "number of cities", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())

#ggsave(plot = p, file = "HW6_Problem6_Plot_Settlage.pdf")
```

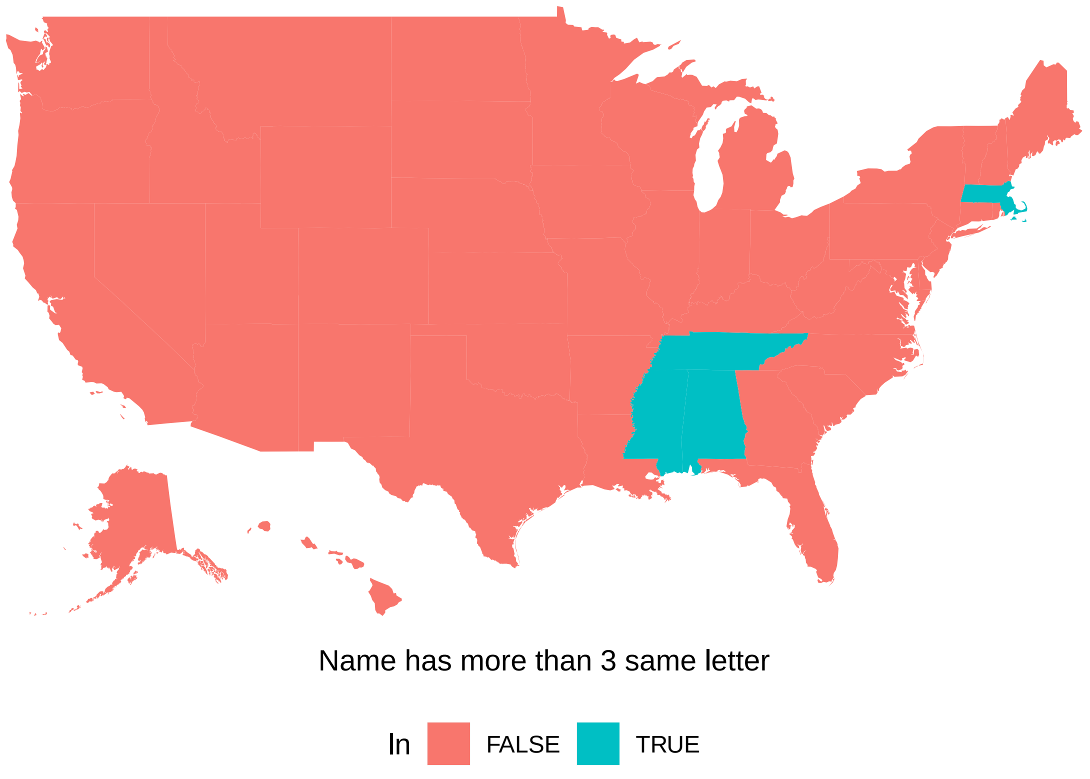
```
p2
```



map 2

```r
data("fifty_states") # this line is optional due to lazy data loading


# select state names that has more than 3 same letters
ln <- c()
for (i in 1:nrow(letter_count)){
  ln[i] <- sum(letter_count[i, ] > 3) >= 1
}
# create dataset
df_mp3 <- data.frame(state = as.factor(tolower(rownames(letter_count))), n = ln)
# map_id creates the aesthetic mapping to the state name column in your data
p3 <- ggplot(df_mp3, aes(map_id = state)) +
  # map points to the fifty_states shape data
  geom_map(aes(fill = ln), 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 = "Name has more than 3 same letter", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())
```

```r
p3
```



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.