Homework 5

Tessa Anwyll

September 21, 2018

Problem 3

A good figure is dependent on your medium, audience and itent, but in general I think a good figure is one that shows information that would be otherwise hard to communicate and communicates it efficiently. If you are making a figure to use for yourself, the figure just has to show the relationship you are looking for, wheras if you are making it for a more general audience, it needs to be colored nicely (in a way that makes the figure easier to understand), have titles, captions and labels and be simple enough that people who may not know a lot about the topic can make use of it. Additonally, if the figure is in a paper or online, it can be more complex than if you are putting it in a presentation where your audience sees it for less time and therefore it has to be simpler so it can be understood quickly. The figure should attempt to convey information clearly and objectively

Problem 4

```
#Initialize function that accepts arguments
#for your data and the ability to define what counts as a success

propSuccess <- function(data, success = 1){
  counter <- length(data) #determine total number of elements in data set
  num <- 0 #counter to record the number of successes
  for(i in 1:counter){ #checks each element in the data set
    #to see if it is success and updates num if it is
    if(data[i] == success){
     num <- num+1
    }
  }
  prop <- num/counter #calculate proportion of
                      #successes by dividing successes by total
  return(prop)
}

# Set seed for generating random flip values
    set.seed(12345)
    # make a matrix of 10 observations for 10
    # different degrees of fairness between .3 and .4
    P4b_data <- matrix(rbinom(10, 1, prob = (30:40)/100), nrow = 10, ncol = 10)
# Use apply to find proportions of success for each column and each row
resultsCols <- apply(P4b_data, 2, propSuccess)
resultsRows <- apply(P4b_data, 1, propSuccess)
resultsCols
```

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

```
resultsRows
```

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

Each column is exactly identical with 6/10 entries being a 1, while each row is either all 1s or all 0s. The problem is that it is only generating one set of 10 random numbers and then copying that set of numbers for each of the columns in the matrix, or maybe it is generating each time from the same seed start point so its generating the same random numbers each time

```
makeTenFlips <- function(probability){
  output <- rbinom(10, 1, prob = probability)
}

flipProbs <- (30:40)/100
flipMat <- matrix(rep(0, 100), nrow = 10, ncol = 10)
flipMat <- replicate(10, makeTenFlips(flipProbs))
resultsCols <- apply(flipMat, 2, propSuccess)
resultsRows <- apply(flipMat, 1, propSuccess)
outtable <- data.frame(rbind(resultsCols, resultsRows))
rownames(outtable) <- c("Column Proportions", "Row Proportions")
colnames(outtable) <- c(1:10)
outtable
```

```
##                       1   2   3   4   5   6   7   8   9  10
## Column Proportions 0.2 0.3 0.4 0.4 0.4 0.6 0.2 0.3 0.5 0.3
## Row Proportions    0.6 0.2 0.5 0.4 0.3 0.1 0.7 0.4 0.1 0.3
```

Problem 5

```
p5dat <- readLines("https://www2.isye.gatech.edu/~jeffwu/book/data/starch.dat")
p5elements <- strsplit(p5dat, split = " ")
starch <- vector()
strength <- vector()
thickness <- vector()
position <- 1
for(i in 2:length(p5elements)){
  if(length(p5elements[[i]])<=3){
  starch[position] <- p5elements[[i]][1]
  strength[position] <- p5elements[[i]][2]
  thickness[position] <- p5elements[[i]][3]
  }
  else{
  starch[position] <- p5elements[[i]][1]
  strength[position] <- p5elements[[i]][length(p5elements[[i]])-1]
  thickness[position] <- p5elements[[i]][length(p5elements[[i]])]
  }
  position <- position+1
}


strength <- as.numeric(strength)
thickness <- as.numeric(thickness)
p5table <- data.frame(starch, strength, thickness)
colnames(p5table) <- c(p5elements[[1]][1:3])
p5table
```

```
##    starch strength thickness
## 1      CA    791.7       7.7
```

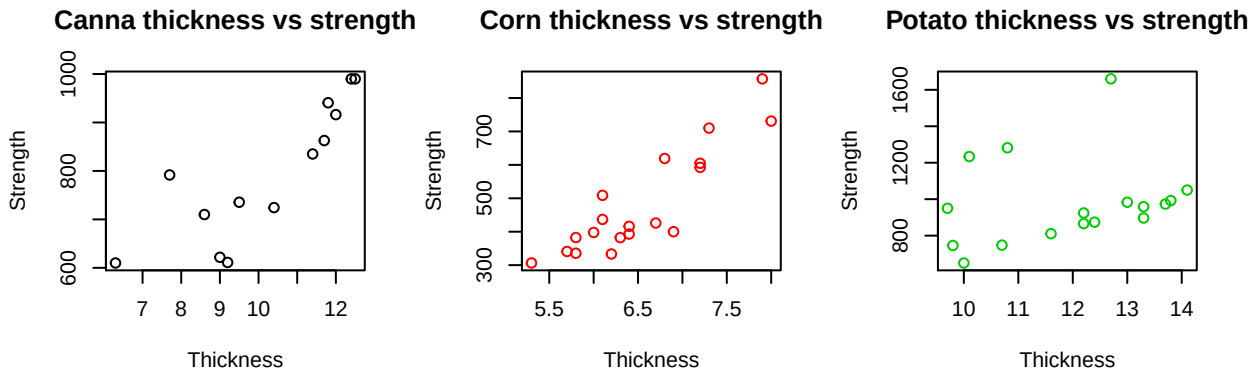
```
## 2      CA    610.0       6.3
## 3      CA    710.0       8.6
## 4      CA    940.7      11.8
## 5      CA    990.0      12.4
## 6      CA    916.2      12.0
## 7      CA    835.0      11.4
## 8      CA    724.3      10.4
## 9      CA    611.1       9.2
## 10     CA    621.7       9.0
## 11     CA    735.4       9.5
## 12     CA    990.0      12.5
## 13     CA    862.7      11.7
## 14     CO    731.0       8.0
## 15     CO    710.0       7.3
## 16     CO    604.7       7.2
## 17     CO    508.8       6.1
## 18     CO    393.0       6.4
## 19     CO    416.0       6.4
## 20     CO    400.0       6.9
## 21     CO    335.6       5.8
## 22     CO    306.4       5.3
## 23     CO    426.0       6.7
## 24     CO    382.5       5.8
## 25     CO    340.8       5.7
## 26     CO    436.7       6.1
## 27     CO    333.3       6.2
## 28     CO    382.3       6.3
## 29     CO    397.7       6.0
## 30     CO    619.1       6.8
## 31     CO    857.3       7.9
## 32     CO    592.5       7.2
## 33     PO    983.3      13.0
## 34     PO    958.8      13.3
## 35     PO    747.8      10.7
## 36     PO    866.0      12.2
## 37     PO    810.8      11.6
## 38     PO    950.0       9.7
## 39     PO   1282.0      10.8
## 40     PO   1233.8      10.1
## 41     PO   1660.0      12.7
## 42     PO    746.0       9.8
## 43     PO    650.0      10.0
## 44     PO    992.5      13.8
## 45     PO    896.7      13.3
## 46     PO    873.9      12.4
## 47     PO    924.4      12.2
## 48     PO   1050.0      14.1
## 49     PO    973.3      13.7
```

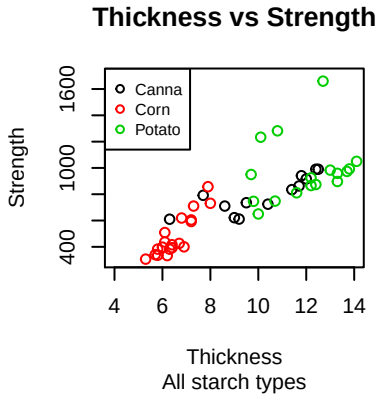
```
# create mat for scatter plots
layMat <- matrix(c(1:3, 0, 4, 0), ncol = 3, nrow = 2, byrow = TRUE)
layout(layMat)

# make vectors with just the data from one of each starch type
```

```r
CA <- p5table[which(p5table[,1]== "CA"), 2:3]
CO <- p5table[which(p5table[,1]== "CO"), 2:3]
PO <- p5table[which(p5table[,1]== "PO"), 2:3]

# Make scatter plots of each factor and one graph with
# them all combined but color coded with legend
plot(CA$thickness, CA$strength, main = "Canna thickness vs strength",
       xlab = "Thickness", ylab = "Strength", col = 1)
plot(CO$thickness, CO$strength, main = "Corn thickness vs strength",
       xlab = "Thickness", ylab = "Strength", col = 2)
plot(PO$thickness, PO$strength, main = "Potato thickness vs strength",
       xlab = "Thickness", ylab = "Strength", col = 3)
plot(thickness, strength, col = p5table$starch, main = "Thickness vs Strength",
       sub = "All starch types", xlab = "Thickness", ylab = "Strength",
       xlim = c(4,14), ylim = c(300,1700))
legend("topleft", legend = c("Canna", "Corn", "Potato"),
         col = 1:3, pch = 1, cex = .8)
```



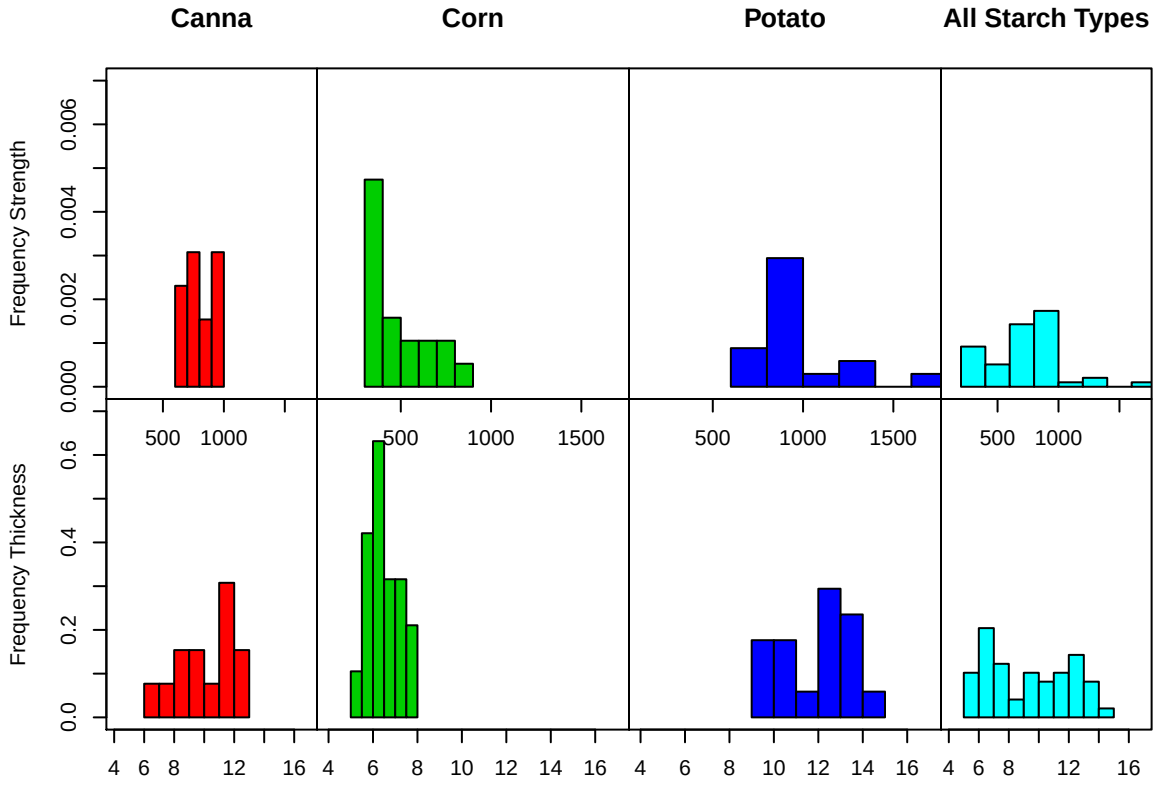



```r
# make single vectors to feed to the for loop to make histograms for data
# labels, margin arguments
data <- c(CA[,1], CO[,1], PO[,1], strength, CA[,2], CO[,2], PO[,2], thickness)
index <- c(length(CA[,1]), length(CO[,1]), length(PO[,1]), length(strength),
            length(CA[,2]), length(CO[,2]), length(PO[,2]), length(thickness))
datax <- c("Frequency Strength", NULL, NULL, NULL, "Thickness", NULL, NULL, NULL)
dataTitle <- c("Canna", "Corn", "Potato", "All Starch Types")
margins <- cbind(c(0,4,4,0), c(0,0,4,0), c(0,0,4,0), c(0,0,4,4), c(4,4,0,0),
```

```
                    c(4,0,0,0), c(4,0,0,0), c(4,0,0,4))
mylist <- list(CA[,1], CO[,1], PO[,1], strength, CA[,2], CO[,2], PO[,2], thickness)
# Make layout mat
layout((matrix(c(1:8), ncol=4, nrow = 2, byrow = TRUE)), widths = c(rep(2.5/10, 4)),
         heights = c(5/10, 5/10))

# For loops make each histogram and add to layout mat
counter <- 1
for(i in 1:4){ #top row of mat
  par(mar = margins[,i])
  if(i == 1){
  hist(mylist[[i]], main = dataTitle[i], ylab = datax[i], col = i+1, freq = FALSE,
         xlab = NULL, ylim = c(0,.007), xlim = c(100,1700))
  }
  else{
    hist(mylist[[i]], main = dataTitle[i], ylab = datax[i], col = i+1, freq = FALSE,
           xlim = c(100,1700), xlab = NULL, yaxt = "n", ylim = c(0,.007))
  }
  box()
  counter <<- counter + index[i]
}

for(i in 5:8){ # bottom row of mat
  par(mar = margins[,i])
  if(i == 5){
  hist(mylist[[i]], main = NULL, col = i-3, ylab = "Frequency Thickness", freq = FALSE,
         ylim = c(0,.7), xlab = NULL, xlim = c(4,17))
  }
  else{
    hist(mylist[[i]], yaxt = "n", ann = FALSE, ylab = datax[i], col = (i-3),
           xlim = c(4,17), freq = FALSE, main = NULL, ylim = c(0,.7))
  }
  box()
  counter <- counter + index[i]
}
```




```
# Function that takes a data set and outputs a table of
# summary statistics for exploratory data analysis
dataSummary <- function(dataset, v1, v2, ...){
dataOut <- rbind.data.frame(summary(dataset[,2]), summary(dataset[,3]))
sumNames <- c("Min", "Q1", "Med", "Mean", "Q3", "Max")
rown <- c(paste("Overall", v1), paste("Overall", v2))
factors <- unique(dataset[,1])
headers <- colnames(dataset)


# Find summary information by factor and add
# to data frame
for(i in 1:length(factors)){
  dataOut <- rbind.data.frame(dataOut, summary
                (dataset[which(dataset[,1] == factors[i]), v1]),
                 summary(dataset[which(dataset[,1] == factors[i]), v2]))
  rown <- c(rown, paste(factors[i], v1), paste(factors[i], v2))
}

# Add names to data table
colnames(dataOut) <- sumNames
rownames(dataOut) <- rown

# Calculate IQRs and Ranges and add  to data summary table
dataIQR <- dataOut[,"Q3"]-dataOut[,"Q1"]
dataRange <- dataOut[,"Max"]-dataOut[,"Min"]
```

```
dataOut <- cbind.data.frame(dataOut, dataIQR, dataRange)

# Initialize sd vector and calculate standard deviation within each factor for v1 and v2
dataSD <- c(sd(dataset[,2]), sd(dataset[,3]))
for(i in 1:length(factors)){
  dataSD <- c(dataSD, sd(dataset[which(dataset[,1] == factors[i]), v1]),
              sd(dataset[which(dataset[,1] == factors[i]), v2]))
}
# Initialize variance vector and calculate variance within each factor for v1 and v2
dataVar <- c(var(dataset[,2]), var(dataset[,3]))
for(i in 1:length(factors)){
  dataVar <- c(dataVar, var(dataset[which(dataset[,1] == factors[i]), v1]),
               var(dataset[which(dataset[,1] == factors[i]), v2]))
}

# add remaining columns and column names and print summary data
dataOut <- cbind.data.frame(dataOut, dataVar, dataSD)
sumNames <- c(sumNames, "IQR", "Range", "Variance", "Standard Deviation")
colnames(dataOut) <- sumNames
dataOut

}
# Create data summary of starch data
dataSummary(p5table, "strength", "thickness")
```

```
##                      Min     Q1   Med       Mean     Q3    Max   IQR  Range
## Overall strength  306.4 508.80 735.4 736.975510 924.40 1660.0 415.6 1353.6
## Overall thickness   5.3   6.70   9.5   9.387755  12.00   14.1   5.3    8.8
## CA strength       610.0 710.00 791.7 795.292308 916.20  990.0 206.2  380.0
## CA thickness        6.3   9.00  10.4  10.192308  11.80   12.5   2.8    6.2
## CO strength       306.4 382.40 416.0 482.826316 598.60  857.3 216.2  550.9
## CO thickness        5.3   6.05   6.4   6.531579   7.05    8.0   1.0    2.7
## PO strength       650.0 866.00 950.0 976.429412 992.50 1660.0 126.5 1010.0
## PO thickness        9.7  10.70  12.2  11.964706  13.30   14.1   2.6    4.4
##                       Variance Standard Deviation
## Overall strength  7.979212e+04        282.4749963
## Overall thickness 7.696930e+00          2.7743342
## CA strength       1.933448e+04        139.0484775
## CA thickness      3.884103e+00          1.9708127
## CO strength       2.483643e+04        157.5957693
## CO thickness      5.489474e-01          0.7409098
## PO strength       5.654676e+04        237.7956207
## PO thickness      2.291176e+00          1.5136633
```

Problem 6

```
   #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), data.table = FALSE)

 states <- states[,1]
 states <- states[-8]
 states <- tolower(states)


    #PART B
    # Read in cities_extended and shorten by filtering out repeated city names
    cities <- fread(input = "./us_cities_and_states/cities_extended.sql",sep = "'",
                    sep2 = ",", header = F, select = c(2,4))
    citiesshort <- unique(cities)

    # Give some headers
    colnames(citiesshort) <- c("City", "State")

    #Create summary table of the number of cities included by state
    numbystate <- data.frame(table(citiesshort$State))
    numbystate <- numbystate[c(-8,-40),]
```

PART C

```
letter_count <- data.frame(matrix(NA,nrow=50, ncol=26))
letterCounter <- function(letter, stateName){
  temp <- strsplit(stateName, split = "")
  tf <- temp[[1]] == letter
  counter <- sum(tf)

  return(counter)
}
alphabet <- c("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")
for(i in 1:length(states)){

  letter_count[i,] <- sapply(alphabet, letterCounter, stateName = states[i])

}
colnames(letter_count) <- alphabet
rownames(letter_count) <- states
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
## 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 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 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 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 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 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 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 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 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 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 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 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 0 1 1 1 0 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 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
## new york       0 0 0 0 1 0 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 0 1 1 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0
## oklahoma       2 0 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 0 0
## oregon         0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 2 0 0 1 0 0 0 0 0 0 0 0 0
## pennsylvania   2 0 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 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 0 1 1 0 0 0 1 0 0
## utah           1 0 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 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 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 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 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 0 1 0 0 0 1 1 1 0 1 1 0 0 0
## wyoming        0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0
```

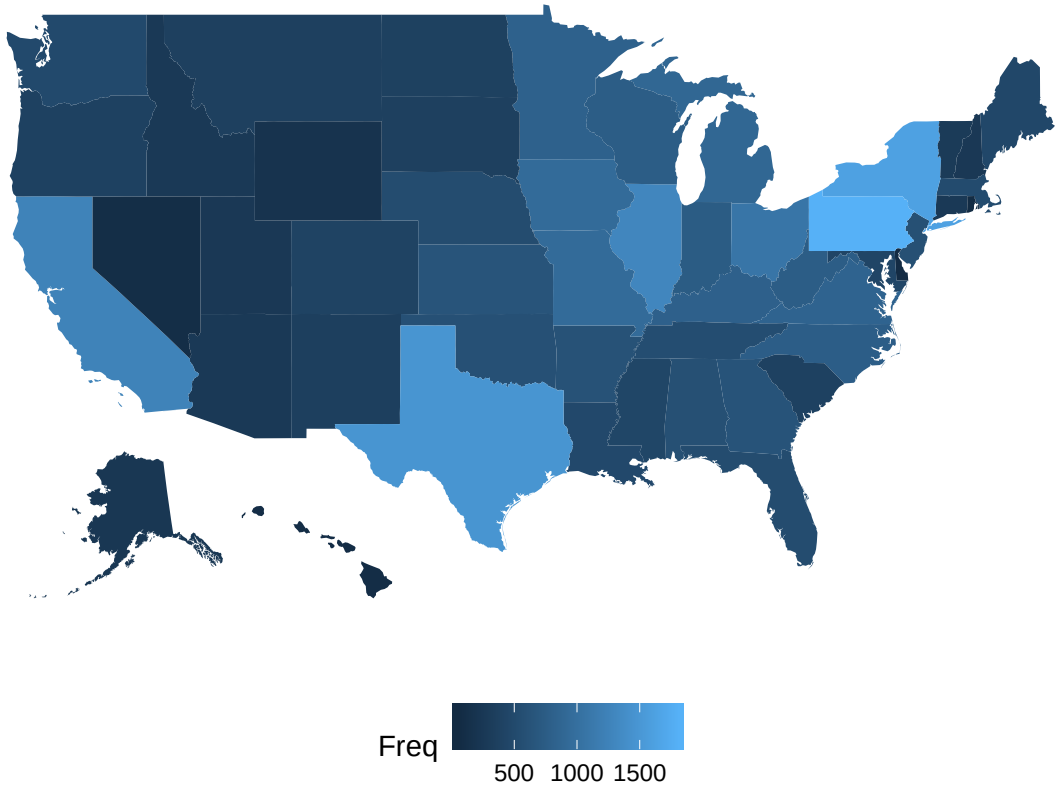
```
tflc <- letter_count >= 3
Three.Or.More <- apply(tflc, 1, sum)
statesum <- cbind(states, Three.Or.More)
```

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
mystates <- data.frame(state = states, numbystate)
# map_id creates the aesthetic mapping to the state name column in your data
p <- ggplot(mystates, 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) +
  labs(x = "", y = "") +
  theme(legend.position = "bottom",
        panel.background = element_blank())

p
```



Map 2

```
mystates <- data.frame(state = states, statesum)
  # map_id creates the aesthetic mapping to the state name column in your data
  q <- ggplot(mystates, aes(map_id = state)) +
    # map points to the fifty_states shape data
    geom_map(aes(fill = Three.Or.More), 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())

q
```