Final Project - Twitter Geolocator

CJ Xiang

12/12/2016

Introduction

Hi, my name is CJ Xiang. The goal of this presentation is to walk you guys through how I set up twitter API, got geolocated tweets from which to perform data analyses, constructed statistical models, and built interactive apps and visualizations to make data-driven explorations. As you will discover soon a handful of quantitative methods are covered in this project, they are included but not limited to Shiny, Bootstrap, text analysis, sentimental analysis, word clouds, and mapping. Now, let us rock.

Set up Twitter API

We set up Twitter API by linking URL and consumer Key to our server. The process is a little of tedious and it cannot be knitted in PDF. Instead, we will attach the resulted my_outh.Rdada file.

Tweets Collecting and Mapping

Our interest in Starbucks. It may take up to 6000s to collect data, and therefore filterStream("tweetsUS.json",track=c("Starbucks
oauth = my_oauth) is optional here as we will attach Json file for you.

```
library(streamR)
# Load back the file
load("my_oauth.Rdata")

# filterStream("tweetsUS.json",track=c("Starbucks"),timeout=6000, oauth = my_oauth)
tweets.df <- parseTweets("tweetsUS.json",simplify = FALSE)
```

```
## 3160 tweets have been parsed.
```

```
# Again, save the file for further use
saveRDS(tweets.df, "StarbucksOriginalData.RDS")
```

Before going any further, let us plot the map data to decide if we need do some data cleaning. It can get really messy. Building visualization is a great tool to give me a sense where I am currently at the project, and the rest is all about curiosity.

```
library(ggplot2)
library(grid)

map.data <- map_data("state")
points <- data.frame(x=as.numeric(tweets.df$place_lon), y=as.numeric(tweets.df$place_lat))

points <- points[points$y>25,]
ggplot(map.data)+
  geom_map(aes(map_id = region),
           map=map.data,
```

```
          fill="white",
          color="grey20",size=0.25)+
  expand_limits(x = map.data$long, y = map.data$lat)+
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        plot.background = element_blank(),
        plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines"))+
        geom_point(data = points,
        aes(x = x, y = y), size = 1,
        alpha = 1/5, color = "darkgreen")+
  ggtitle("                                Tweets mentioning Starbucks in the U.S.")
```

Now as we can see some points are located outside the U.S. territory. Since my priori interest is to focus on tweets within the U.S., we should filter out tweets outside the U.S.

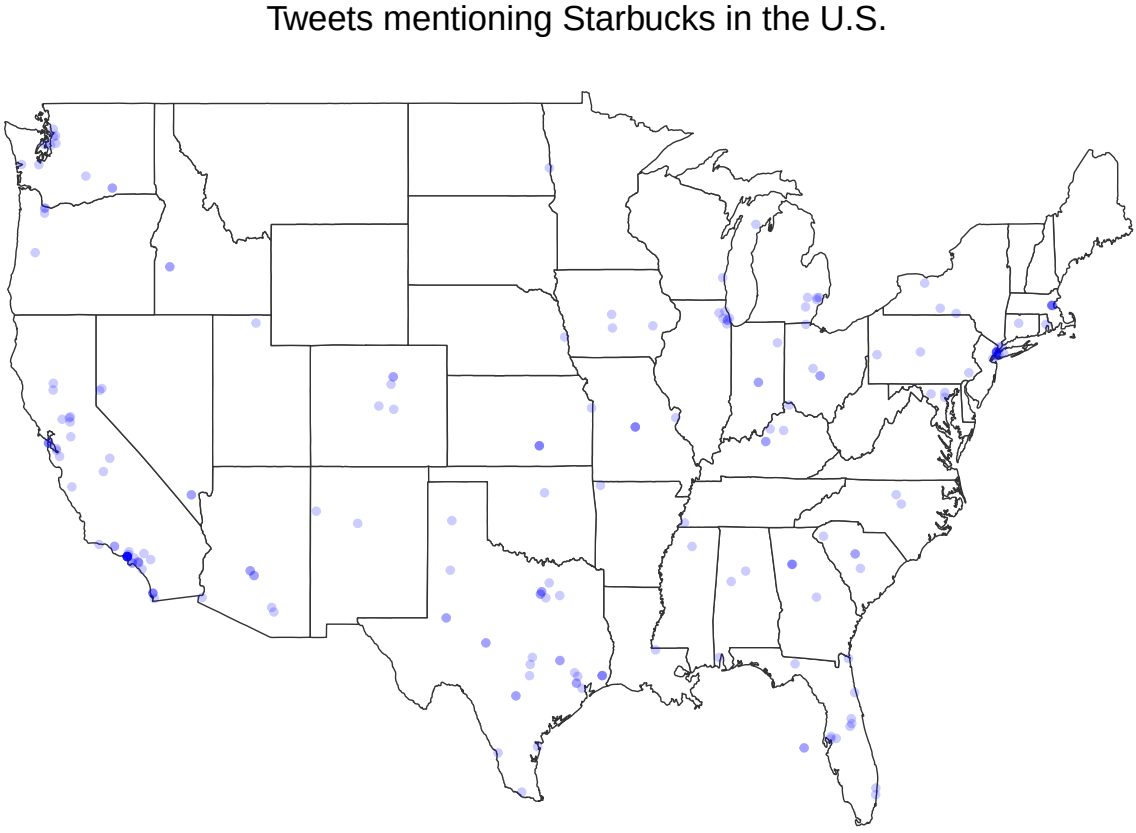
```
index <- which(tweets.df$country_code == "US")
tweetsClean.df <- tweets.df[index,]

map.data <- map_data("state")
points <- data.frame(x=as.numeric(tweetsClean.df$place_lon), y=as.numeric(tweetsClean.df$place_lat))

points <- points[points$y>25,]
```

```
ggplot(map.data)+
  geom_map(aes(map_id = region),
           map=map.data,
           fill="white",
           color="grey20",size=0.25)+
  expand_limits(x = map.data$long, y = map.data$lat)+
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        plot.background = element_blank(),
        plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines"))+
  geom_point(data = points,
             aes(x = x, y = y), size = 1,
             alpha = 1/5, color = "darkgreen")+
  ggtitle("                                    Tweets mentioning Starbucks in the U.S.")
```

We still have an outlier! But why? We already ensured country code is U.S though. It seems the only possibility is that such tweet has a country code U.S, but with bizzare geolocations. I decide to remove it in the interest of consistency. And here we go!

```
newindex <- which(tweetsClean.df$country == "Vereinigte Staaten")
tweetsFinalclean.df <- tweetsClean.df[-newindex,]
saveRDS(tweetsFinalclean.df, "StarbucksUSData.RDS")
```

```
map.data <- map_data("state")
points <- data.frame(x=as.numeric(tweetsFinalclean.df$place_lon), y=as.numeric(tweetsFinalclean.df$plac

points <- points[points$y>25,]
ggplot(map.data)+
  geom_map(aes(map_id = region),
           map=map.data,
           fill="white",
           color="grey20",size=0.25)+
  expand_limits(x = map.data$long, y = map.data$lat)+
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        plot.background = element_blank(),
        plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines"))+
  geom_point(data = points,
             aes(x = x, y = y), size = 1,
             alpha = 1/5, color = "blue") +
  ggtitle("                                    Tweets mentioning Starbucks in the U.S.")
```



It may take a couple of steps to get there, but we finally did!

Word clouds Generating

What is are word clouds? Word cloud is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance. In the following, we will find out most frequently mentioned hashtags when tracking Starbucks.

```
# Let's import necessary packages needed for generating a wordclound
library(twitteR)
library(tm)
library(wordcloud)
library(RColorBrewer)
library(stringr)


# We try to get rid of Emoji and weried characters
Star_text <- sapply(tweetsFinalclean.df$text, function(row) iconv(row, "latin1", "ASCII", sub=""))
# create a corpus
Star_corpus = Corpus(VectorSource(Star_text))
# create document term matrix applying some transformations
tdm = TermDocumentMatrix(Star_corpus,
       control = list(removePunctuation = TRUE,
       stopwords = c("Starbucks", stopwords("english")),
       removeNumbers = TRUE, tolower = TRUE))

# define tdm as matrix
m = as.matrix(tdm)
# get word counts in decreasing order
word_freqs = sort(rowSums(m), decreasing=TRUE)
# create a data frame with words and their frequencies
dm = data.frame(word=names(word_freqs), freq=word_freqs)
# plot wordcloud
wordcloud(dm$word, dm$freq, random.order=FALSE, colors=brewer.pal(8, "Dark2"))
```


```
# Now, let's build a world coulds of just twitter handles from Starbucks tweets

# We first focus on @
pal <- brewer.pal(9,"YlGnBu")
pal <- pal[-(1:4)]
foo <- str_extract_all(Star_text, "@\\w+")
```

```
namesCorpus1 <- Corpus(VectorSource(foo))
wordcloud(words = namesCorpus1, scale=c(3,0.5), max.words=40, random.order=FALSE,
          rot.per=0.10, use.r.layout=FALSE, colors=pal)
```


```
# When focus on #
set.seed(146)
hoo <- str_extract_all(Star_text, "#\\w+")
namesCorpus2 <- Corpus(VectorSource(hoo))
wordcloud(words = namesCorpus2, scale=c(3,0.5), max.words=40, random.order=FALSE,
          rot.per=0.10, use.r.layout=FALSE, colors=pal)
```

Text Analysis and Sentiment Analysis

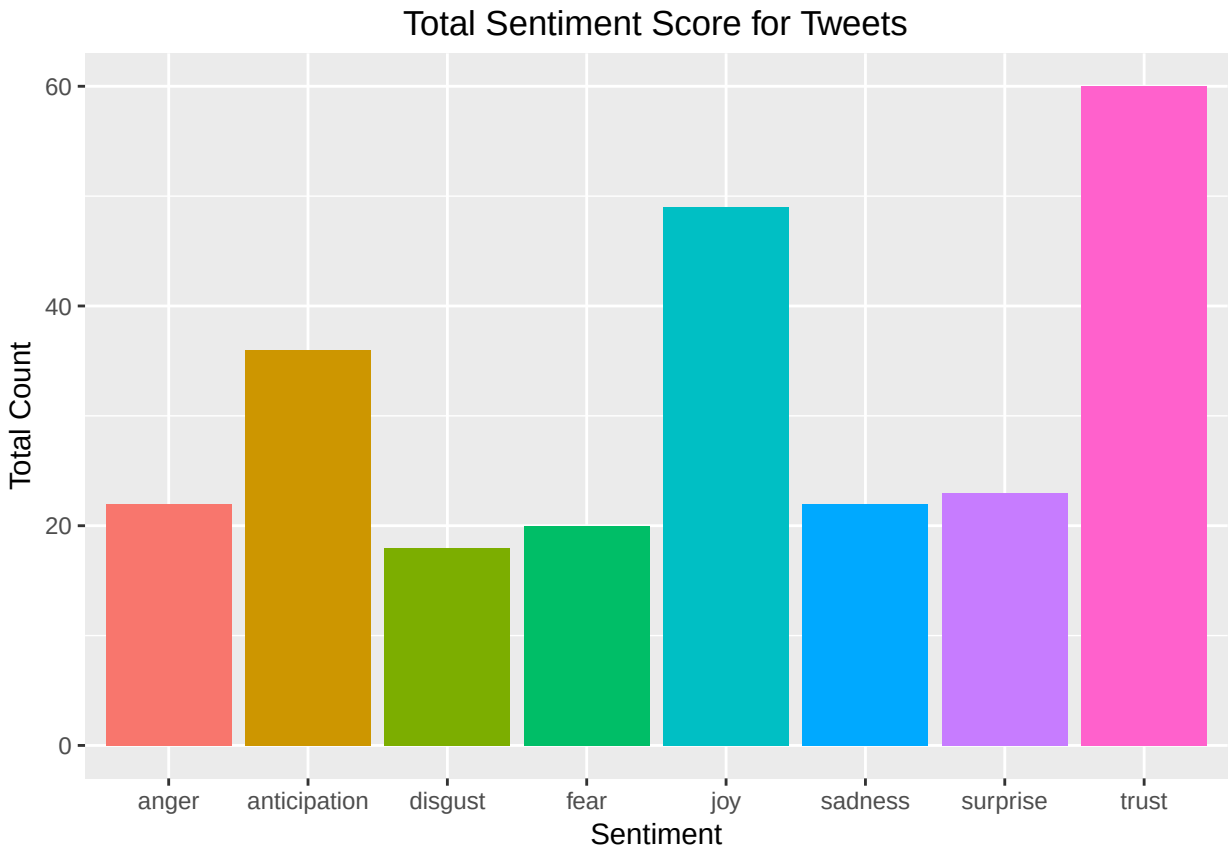
We perform text analysis to remove entitles, @, punctuation, numbers, html links, and unnecessary spaces in text to make sure data is ready to go before I plot a sentiment bar graph and a pie chart. Sentiment analysis helps us identify and categorize opinions expressed tweets, especially in order to determinen whether the tweeter users' attitude towards Starbucks is postive, negative, or neutral.

```
# required pakacges
library(twitteR)
library(sentimentr)
library(plyr)
library(ggplot2)
library(wordcloud)
library(RColorBrewer)
library(syuzhet)
library(plotrix)
```

```
# Let's prepare for sentiment analysis

some_txt <- Star_text

# remove retweet entities
some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
# remove at people
some_txt = gsub("@\\w+", "", some_txt)
# remove punctuation
some_txt = gsub("[[:punct:]]", "", some_txt)
# remove numbers
some_txt = gsub("[[:digit:]]", "", some_txt)
# remove html links
some_txt = gsub("http\\w+", "", some_txt)
# remove unnecessary spaces
some_txt = gsub("[ \t]{2,}", "", some_txt)
some_txt = gsub("^\\s+|\\s+$", "", some_txt)

# define "tolower error handling" function
try.error = function(x)
{
  # create missing value
  y = NA
  # tryCatch error
  try_error = tryCatch(tolower(x), error=function(e) e)
  # if not an error
  if (!inherits(try_error, "error"))
    y = tolower(x)
  # result
  return(y)
}

# lower case using try.error with sapply
some_txt = sapply(some_txt, try.error)
# remove NAs in some_txt
some_txt = some_txt[!is.na(some_txt)]
names(some_txt) = NULL

mySentiment <- get_nrc_sentiment(some_txt)

sentimentTotals <- data.frame(colSums(mySentiment[,c(1:8)]))
names(sentimentTotals) <- "count"
sentimentTotals <- cbind("sentiment" = rownames(sentimentTotals), sentimentTotals)
rownames(sentimentTotals) <- NULL
```

Now we can plot a sentiment histogram

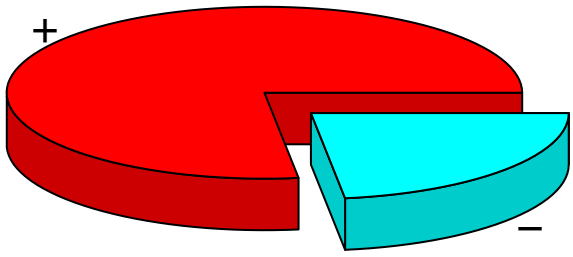
```
ggplot(data = sentimentTotals, aes(x = sentiment, y = count)) +
  geom_bar(aes(fill = sentiment), stat = "identity") +
  theme(legend.position = "none") +
  xlab("Sentiment") + ylab("Total Count") +
  ggtitle("                                Total Sentiment Score for Tweets")
```



We can also plot a pie chart to see the sentiment make up

```
pos <- sum(mySentiment$positive)
neg <- sum(mySentiment$negative)
slices <- c(pos, neg)
lbls <- c("+", "-" )
pie3D(slices,labels=lbls,explode=0.12,main="Pie Chart of Postive and Negative Tweets")
```

Pie Chart of Postive and Negative Tweets



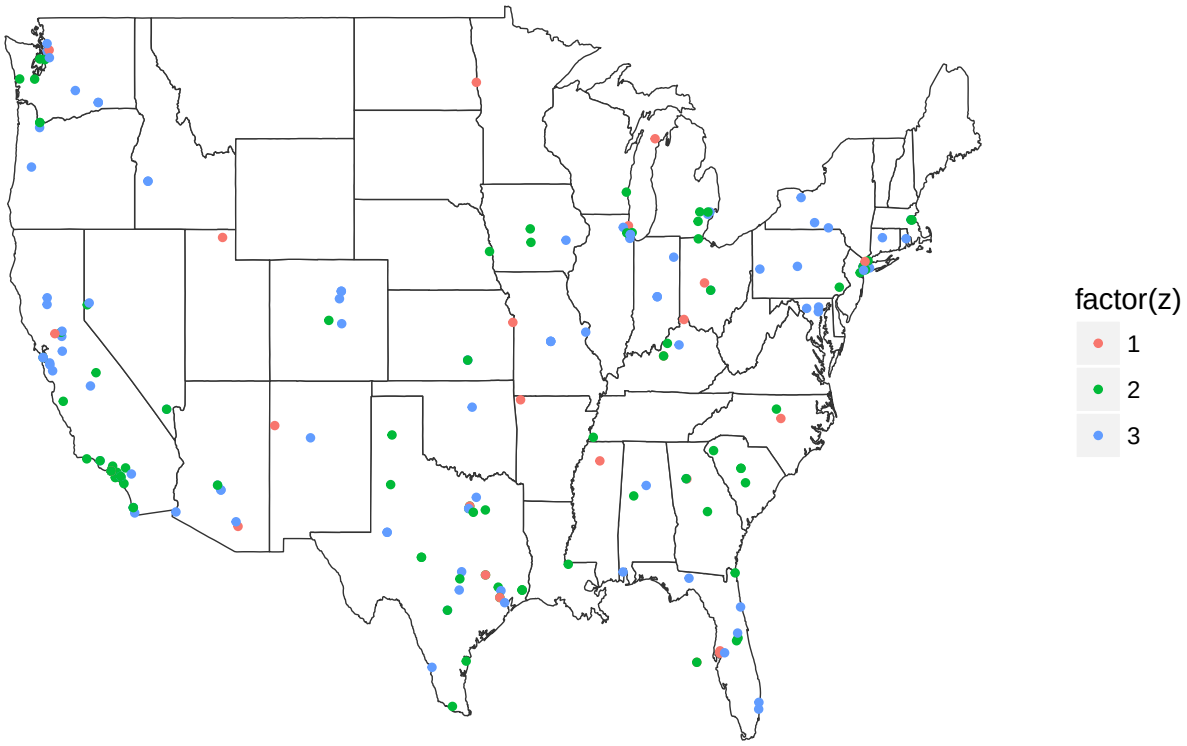
In light of pie chart, we able to discover that more than 3/4 of collected tweets have a positive attitude towards Starbucks while the rest not so friendly.

Tweets Clustering and Spacial Relationships Exploration

We already generated maps at the beginning of exploration, but this time we will take a step further. I am interested in whether tweets sentiments are affected by geolocations. In other words, are there particular regions in the U.S. that favor/dislike Starbucks so much that their tweets sentiments are generally positive/negative. We will first build a visualization to make an hypothesis before we officially to test it.

```
# load in Raw data
Data5 <- readRDS("StarbucksUSData.RDS")
# Data Cleaning
Data4 <- Data5[,c(1,13,20,21,22,29,34,37,38)]
# Combine Sentiment dataset with Data4
Data3 <- cbind(Data4,mySentiment)
# More Data Cleaning
Data2 <- Data3[,c(1:9,18:19)]
# Add categorical variables into table based on the comparison of postive and negative scores
Data1 <- data.frame(Data2$positive - Data2$negative)
beautiful <- function(dk){
  if(dk > 0) {
    return(3)
  } else if (dk < 0) {
    return(1)
  } else {
    return(2)
  }
}
Data <- data.frame(Data2[1:11], apply(Data1, 1, beautiful))
colnames(Data)[12] <- "sentclass"
# Save this file for later use
saveRDS(Data, "Data.RDS")
```

```
map.data <- map_data("state")
points <- data.frame(x=as.numeric(Data$place_lon), y=as.numeric(Data$place_lat), z=Data$sentclass)

points <- points[points$y>25,]
ggplot(map.data)+
  geom_map(aes(map_id = region),
           map=map.data,
           fill="white",
           color="grey20",size=0.25)+
  expand_limits(x = map.data$long, y = map.data$lat)+
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.border = element_blank(),
        panel.grid.major = element_blank(),
        plot.background = element_blank(),
        plot.margin = unit(0 * c(-1.5, -1.5, -1.5, -1.5), "lines")) +
  geom_point(data = points,
             aes(x = x, y = y, colour = factor(z)), size = 1,alpha = 1) +
  ggtitle("                                Starbucks Tweets Clustering in the U.S.")
```



It seems that most of colored points are scattered randomly, and few of them cluster together. Based on this visual, out assumption would be tweets sentiments are not significantly affected by geolocations. We will now build a Statistical Model to test our hypothesis.

Statistical Model Building

Our focus in this setion is to build an Ordinal Logistic Resgression Model to test our null hypothesis that sentiments are not significantly affected by geolocations. Following the same line of logic, some data cleaning is needed to drive conclusion. To begin with, we will categorize all the states into 4 regional divisions used by the United States Census Bureau.

```
# write.csv(Data, "Data.csv")
# We load Data into EXCEl and make sure the full_name column follow the same formart(City, States)
fullnames_update <- read.csv("Data.csv")
# Our ideas is to divide all states into 4 Areas(West, Mid_west, South, North East)
fullnames_update$State_names <- gsub(".*,","",fullnames_update$full_name)
fullnames_update$State_names <- gsub(".* ","",fullnames_update$State_names)
# Now we categorize 50 states into 4 regions: East, West, South, and Midwest

# Regional divisions used by the United States Census Bureau
#
# Region 4: Northeast
# Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, Vermont, New Jersey, New York, and Pe
#
# Region 3: Midwest
# Illinois, Indiana, Michigan, Ohio, Wisconsin, Iowa, Kansas, Minnesota, Missouri, Nebraska, North Dako
#
```

```r
# Region 2: South
# Delaware, Florida, Georgia, Maryland, North Carolina, South Carolina, Virginia, District of Columbia,
# Alabama, Kentucky, Mississippi, Tennessee, Arkansas, Louisiana, Oklahoma, and Texas
#
# Region 1: West
# Arizona, Colorado, Idaho, Montana, Nevada, New Mexico, Utah, Wyomingm, Alaska, California, Hawaii, Or

Handsome <- function(input){
  Reg1 = c("WA", "MT", "OR", "ID", "WY", "CA", "NV", "UT", "CO", "AZ", "NM") # West
  Reg2 = c("OK", "TX", "AR", "LA", "MS", "AL", "TN", "KY", "WV", "MD", "DE", "DC", "VA", "NC", "SC", "G
  Reg3 = c("ND", "SD", "NE", "KS", "MO", "IA", "MN", "WI", "IL", "IN", "OH", "MI") # Midwest
  Reg4 = c("NY", "PA", "NJ", "CT", "RI", "MA", "VT", "NH", "ME") # Northeast
  if (input %in% Reg1)
    return(1)
  else if (input %in% Reg2)
    return(2)
  else if (input %in% Reg3)
    return(3)
  else
    return(4)
}
fullnames_update$regions <- apply(data.frame(fullnames_update$State_names), 1, Handsome)

# Let's categorize sentiments in terms of categorical levels. 3 represents positive; 2 represents neutr
sentiment <- function(x){
  if(x==3)
    return("Positive")
  else if (x==2)
    return("Neutral")
  else
    return("Negative")
}
fullnames_update$sentiment <- apply(data.frame(fullnames_update$sentclass), 1, sentiment)
```

Now data is clean and ready to go. We use 95 % confidence interval to make a judgement. In this example, if 0 is inside the confidence interval, we fail to reject the null hypothesis and conclude that tweets sentiments are not significantly affected by regions, which basically justified our inital observation. However, if 0 is not covered inside the confidence interval, we reject the null hypothesis and conclude that tweets sentiments are significantly affected by regions.

```r
# Ordinal Logistic Regression
# extract data that will be used in the Ordinal Logistic Resgression Model
logdata <- fullnames_update[,c("sentiment","regions")]
attach(logdata)

# load packages
library(foreign)
library(ggplot2)
library(MASS)
library(Hmisc)
library(reshape2)

# description of data
```

```r
# categorical data distribution
lapply(logdata[, c("sentiment","regions")], table)
```

```
## $sentiment
##
## Negative  Neutral Positive
##       26       89       95
##
## $regions
##
##  1  2  3  4
## 82 67 34 27
```

```r
ftable(xtabs(~ regions + sentiment, data = logdata))
```

```
##         sentiment Negative Neutral Positive
## regions
## 1                       6      35       41
## 2                      12      31       24
## 3                       5      12       17
## 4                       3      11       13
```

```r
# Ordinal Logistic Regression
## fit ordered logit model and store results 'm'
logdata$sentiment <- as.factor(logdata$sentiment)
m <- polr(sentiment ~ regions, data = logdata, Hess=TRUE)
## view a summary of the model
summary(m)
```

```
## Call:
## polr(formula = sentiment ~ regions, data = logdata, Hess = TRUE)
##
## Coefficients:
##            Value Std. Error t value
## regions -0.04747     0.1268 -0.3745
##
## Intercepts:
##                   Value   Std. Error t value
## Negative|Neutral -2.0531  0.3327    -6.1715
## Neutral|Positive  0.0957  0.2899     0.3299
##
## Residual Deviance: 412.0096
## AIC: 418.0096
```

```r
## coefficient
coefficient <- coef(summary(m))
## calculate and store p values
p <- pnorm(abs(coefficient[, "t value"]), lower.tail = FALSE) * 2
## combined coefficient and p values
(mresult <- cbind(coefficient, "p value" = p))
```

```
##                         Value Std. Error    t value      p value
## regions          -0.04747184  0.1267535 -0.3745209 7.080168e-01
## Negative|Neutral -2.05313086  0.3326770 -6.1715449 6.762590e-10
## Neutral|Positive  0.09565139  0.2899385  0.3299024 7.414737e-01
```

```
## compute confidence interval
(ci <- confint(m)) # If the 95% CI does not cross 0, the parameter estimate is statistically significan
```

```
##      2.5 %     97.5 %
## -0.2964545  0.2015896
```

We are 95% confident that the value of parameter will fall into a range of (-0.2964545, 0.2015896) when a is at 5%. Therefore, we fail to reject the null hypothesis and conclude that tweets sentiments are not significantly affected by regions.

Now let us reshape the data to make columns regions, and rows sentiments. At this point you may ask we would I do this? It is not related to statistical modeling. Well, you are right. It is not related to statistical modeling but Shiny! We just got our data ready for building a shiny app that would be discussed at the end of presentation. Before Shiny, let us do something associated with bootstrap!

```
sdata <- ftable(xtabs(~ regions + sentiment, data = logdata))
sdata <- as.data.frame(sdata)
shiny.data <- matrix(sdata$Freq,nrow=3,ncol=4,byrow=TRUE)
colnames(shiny.data) <- c("Midwest","Northeast","South","West")
rownames(shiny.data) <- c("Negative","Neutral","Positive")
shiny.data
```

```
##          Midwest Northeast South West
## Negative       6        12     5    3
## Neutral       35        31    12   11
## Positive      41        24    17   13
```

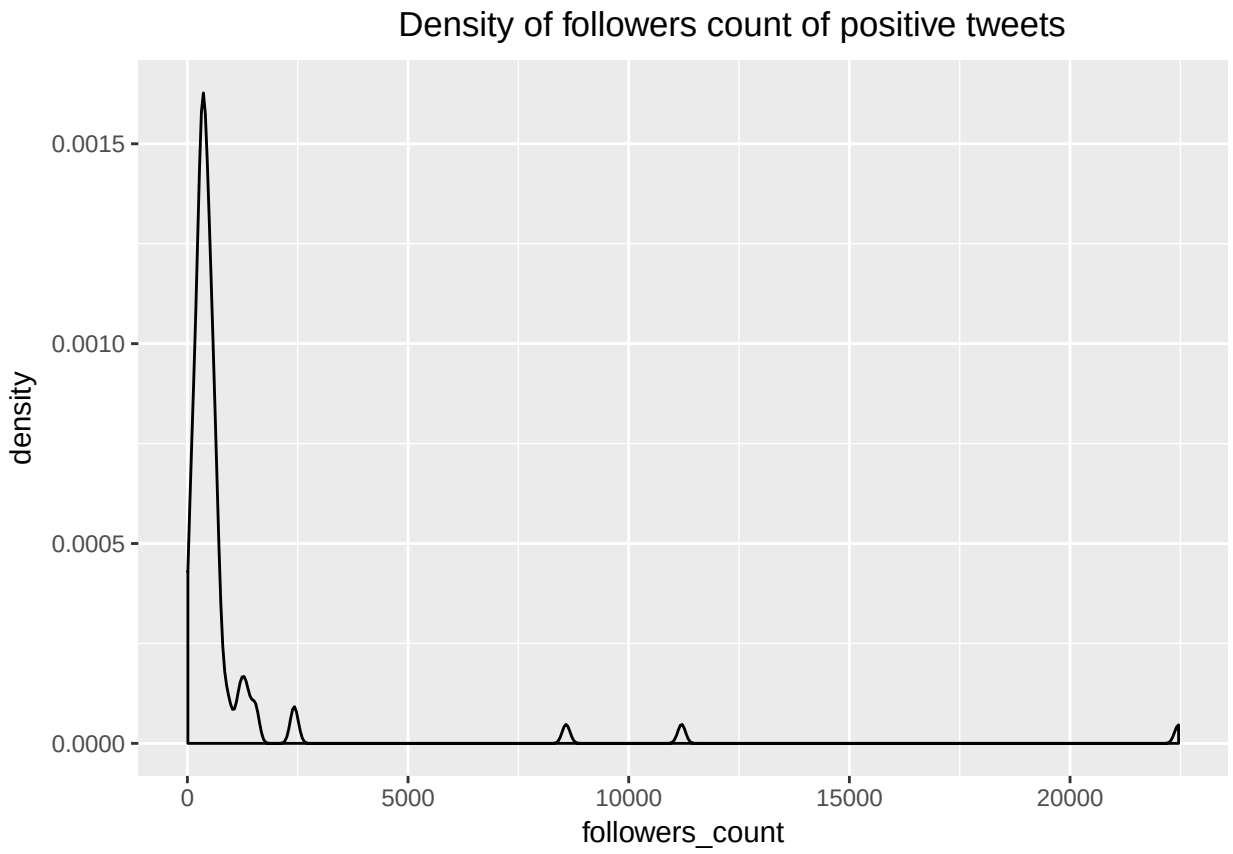
```
saveRDS(shiny.data,"ShinyData.RDS")
```

Bootstrap

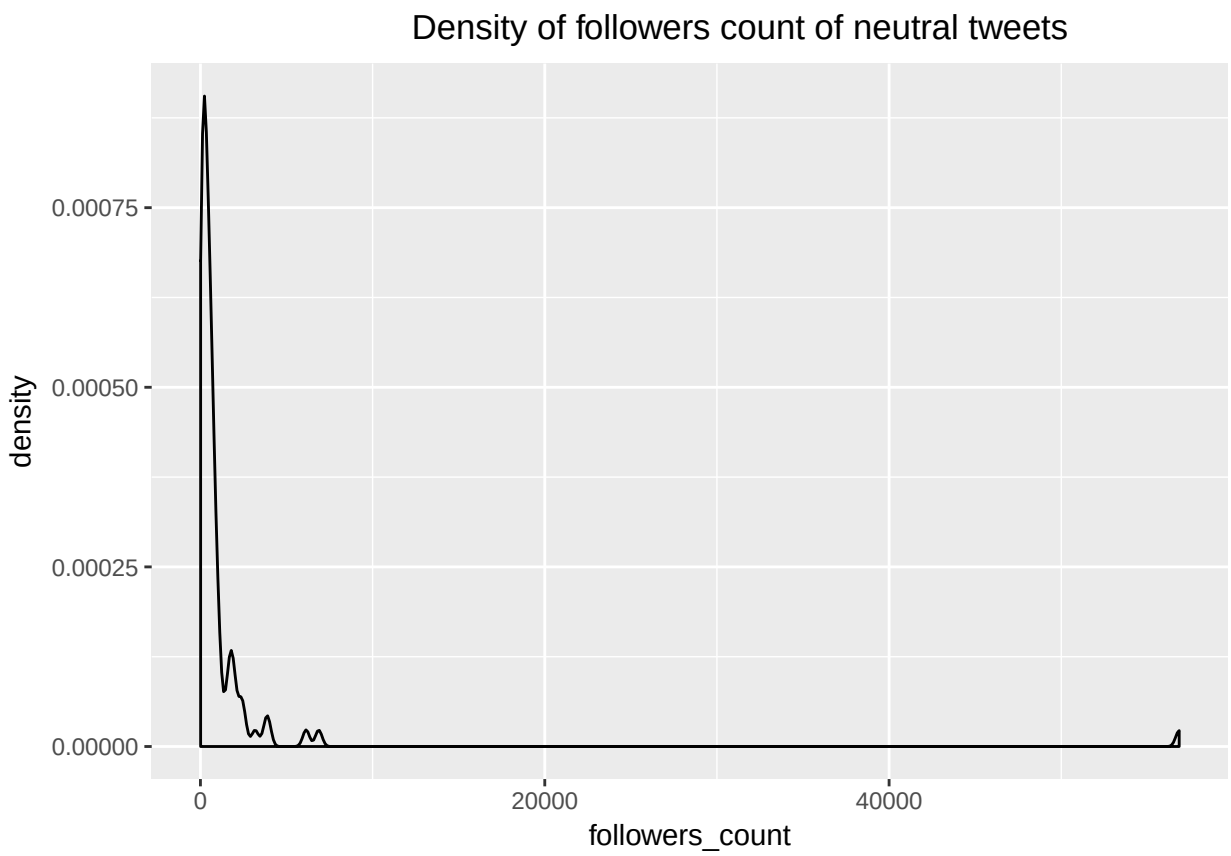
In statistics, bootstrapping can refer to any test or metric that relies on random sampling with replacement. Bootstrapping allows assigning measures of accuracy(defined in terms of bias, variance, confidence intervals, prediction error or some other such measure) to sample estimates. In this example, we will plot a density curve of favourites_count, and use bootstrap to find mean of favourites_count for nagative, neutral and positive tweets. We will also plot a density curve of followers_count, and use bootstrap to find mean of followers_count for nagative, neutral and positive tweets.

```
# load data
starclass <- readRDS("Data.RDS")

index.pos <- which(starclass$sentclass==3)
index.neu <- which(starclass$sentclass==2)
index.neg <- which(starclass$sentclass==1)

pos.fav <- starclass[index.pos,]$favourites_count
```

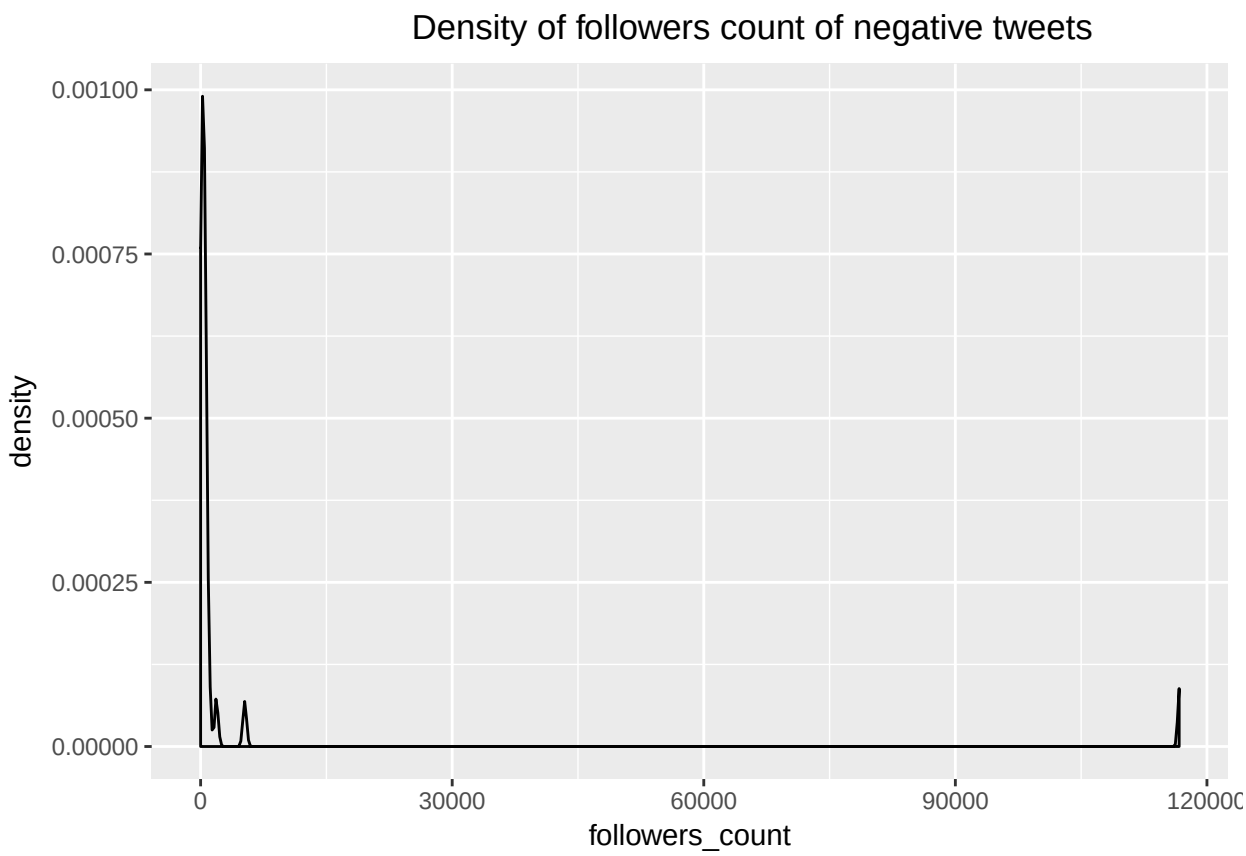
```
neu.fav <- starclass[index.neu,]$favourites_count
neg.fav <- starclass[index.neg,]$favourites_count
pos.fol <- starclass[index.pos,]$followers_count
neu.fol <- starclass[index.neu,]$followers_count
neg.fol <- starclass[index.neg,]$followers_count
pos <- starclass[index.pos,]
neu <- starclass[index.neu,]
neg <- starclass[index.neg,]
attach(pos)
attach(neu)
attach(neg)

# plot density of favourites_count and followers_count

# followers count
ggplot(data=pos, aes(followers_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of followers count of positive tweets")
```



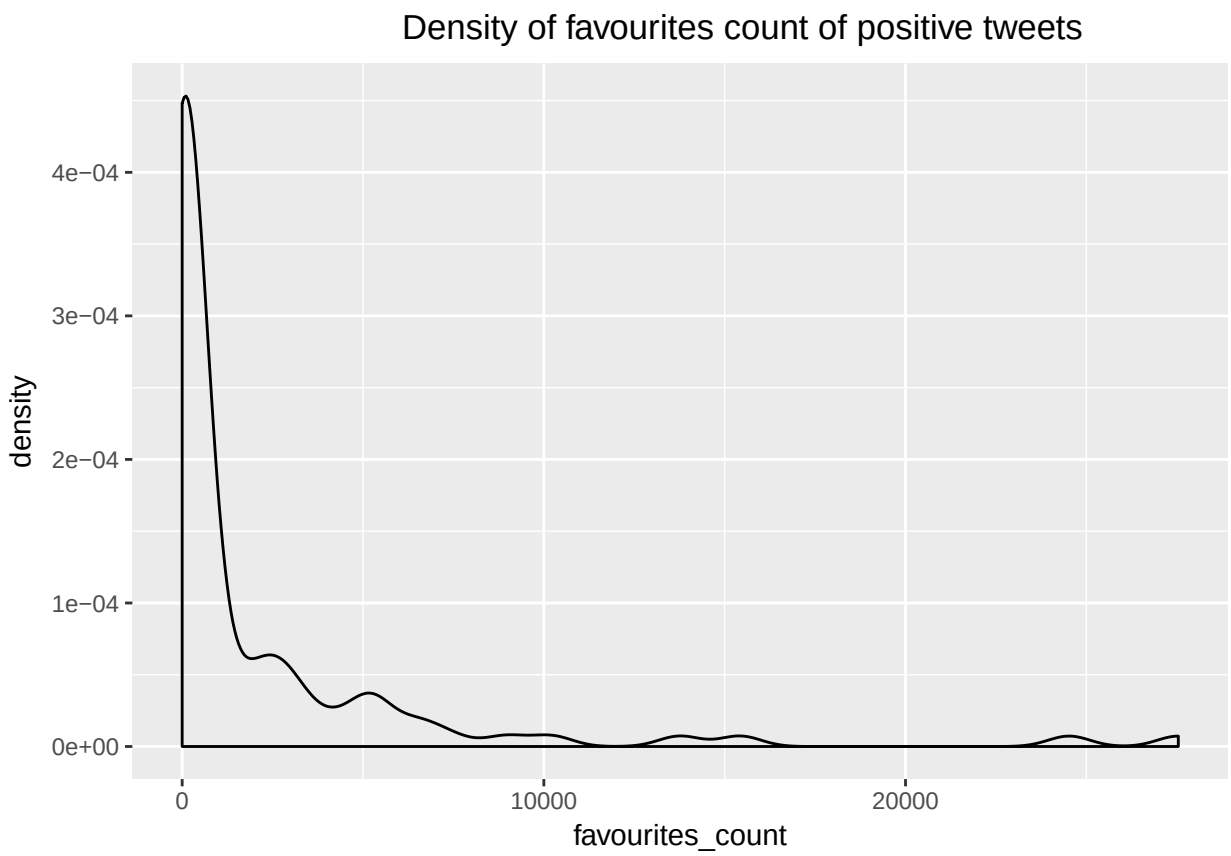

```
ggplot(data=neu, aes(followers_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of followers count of neutral tweets")
```



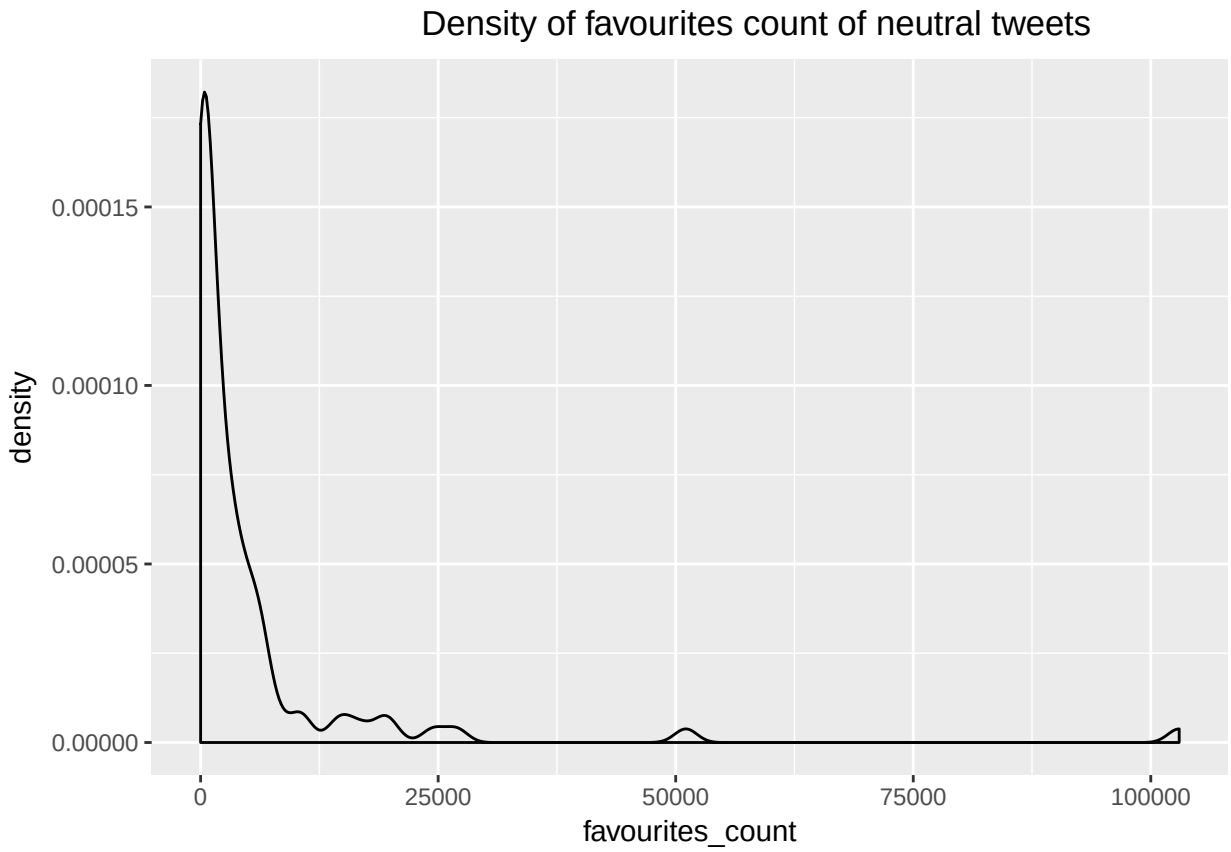

```
ggplot(data=neg, aes(followers_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of followers count of negative tweets")
```



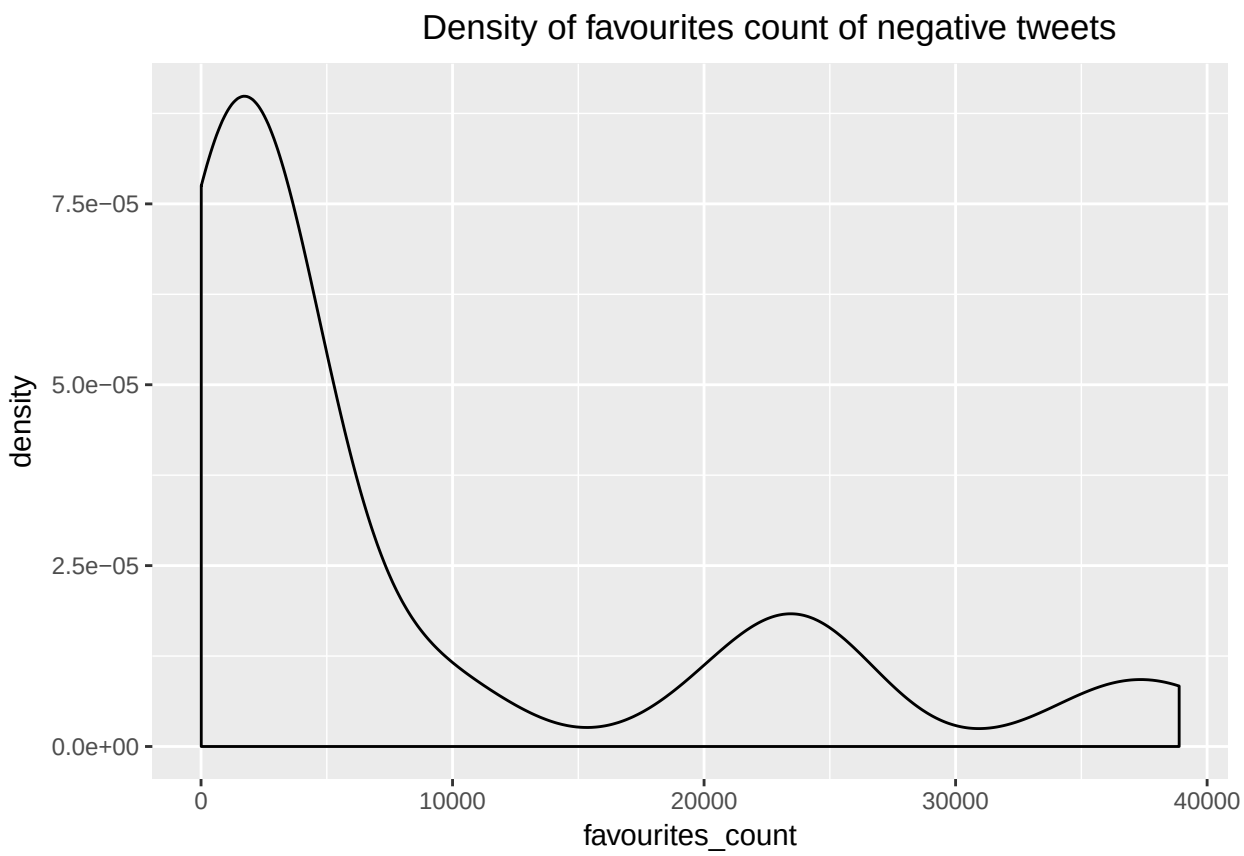

```
# favourites count
ggplot(data=pos, aes(favourites_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of favourites count of positive tweets")
```




```
ggplot(data=neu, aes(favourites_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of favourites count of neutral tweets")
```



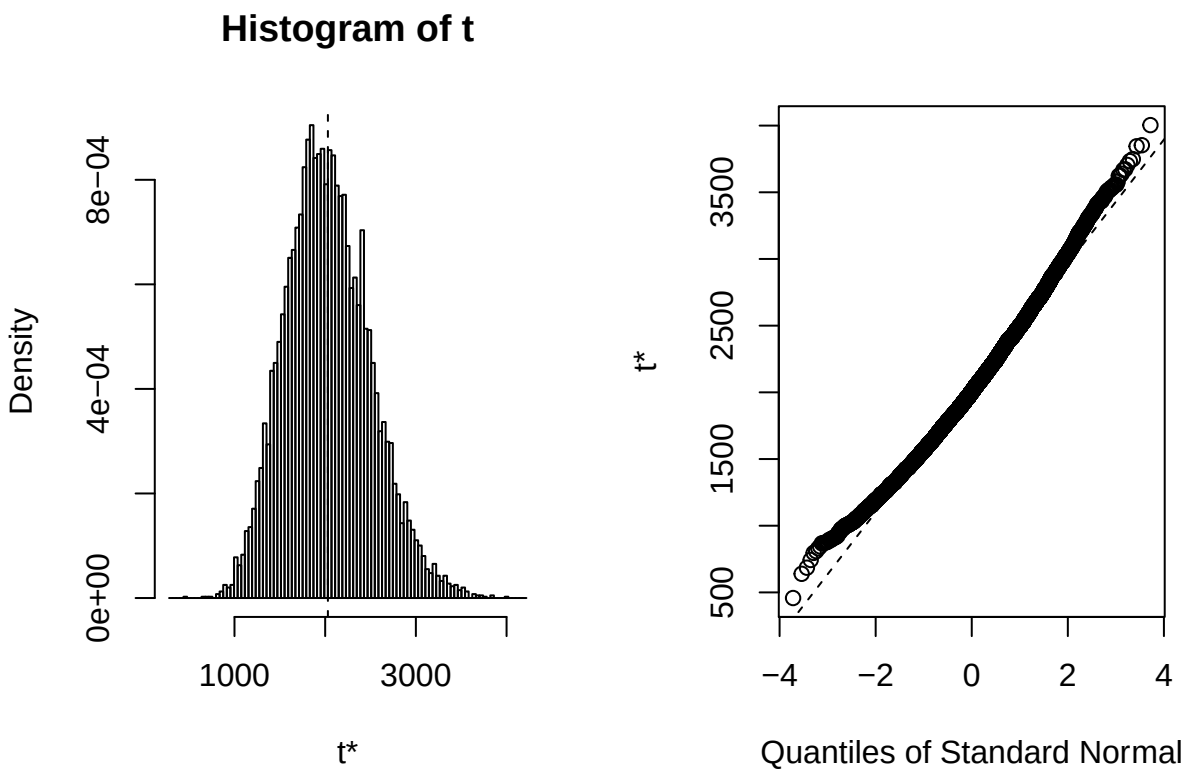

```
ggplot(data=neg, aes(favourites_count))+geom_density(kernel="gaussian") +
ggtitle("                          Density of favourites count of negative tweets")
```




```
# use bootstrap to find mean of favourites_count for nagative, neutral and positive tweets
library(boot)

funmean <- function(data, index)
{
  x <- data[index]
  return(mean(x))
}

# bootstrap for positive, neutral and negtive
bootout.posfav <- boot(pos.fav, funmean, R = 10000)
bootci.posfav <- boot.ci(bootout.posfav, conf = 0.95, type = "all")
bootout.posfav
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = pos.fav, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1* 2029.884 2.681325    465.5788
```

```r
plot(bootout.posfav)
```




```r
bootci.posfav
```

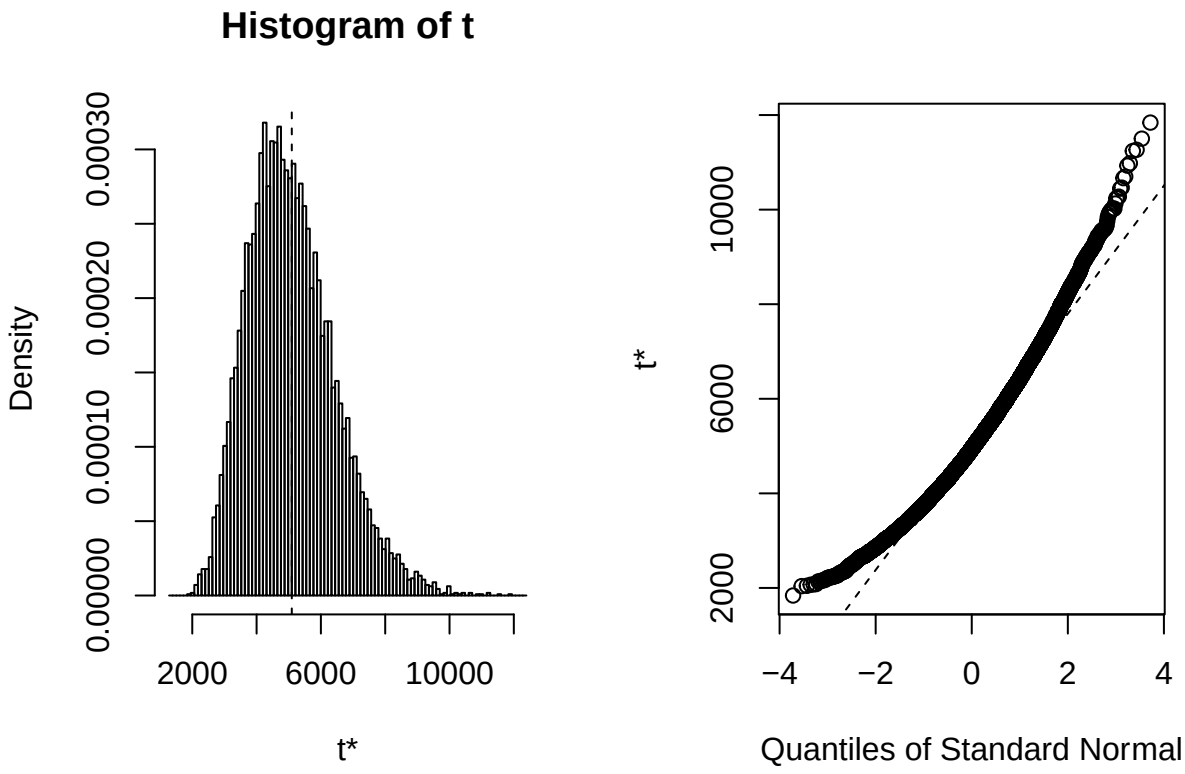
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.posfav, conf = 0.95, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   (1115, 2940 )   (1037, 2858 )
##
## Level     Percentile            BCa
## 95%   (1202, 3023 )   (1314, 3255 )
## Calculations and Intervals on Original Scale
```

```r
bootout.neufav <- boot(neu.fav, funmean, R = 10000)
bootci.neufav <- boot.ci(bootout.neufav, conf = 0.95, type = "all")
bootout.neufav
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
```

```
## boot(data = neu.fav, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 5094.888 -5.944426    1354.588
```

```
plot(bootout.neufav)
```



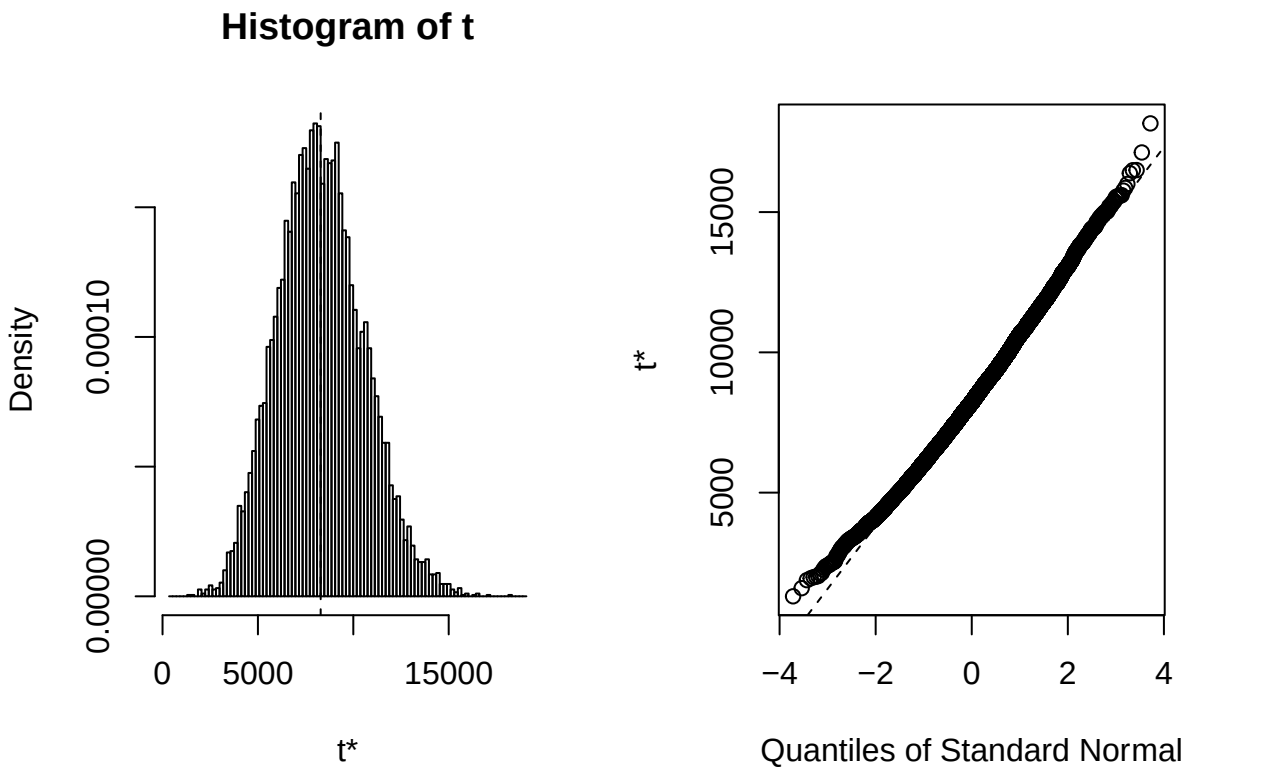

```
bootci.neufav
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.neufav, conf = 0.95, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   (2446, 7756 )   (2030, 7309 )
##
## Level     Percentile            BCa
## 95%   (2881, 8160 )   (3304, 9557 )
## Calculations and Intervals on Original Scale
```

```
bootout.negfav <- boot(neg.fav, funmean, R = 10000)
bootci.negfav <- boot.ci(bootout.negfav, conf = 0.95, type = "all")
bootout.negfav
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = neg.fav, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1* 8288.808 29.03168    2249.205
```

```
plot(bootout.negfav)
```



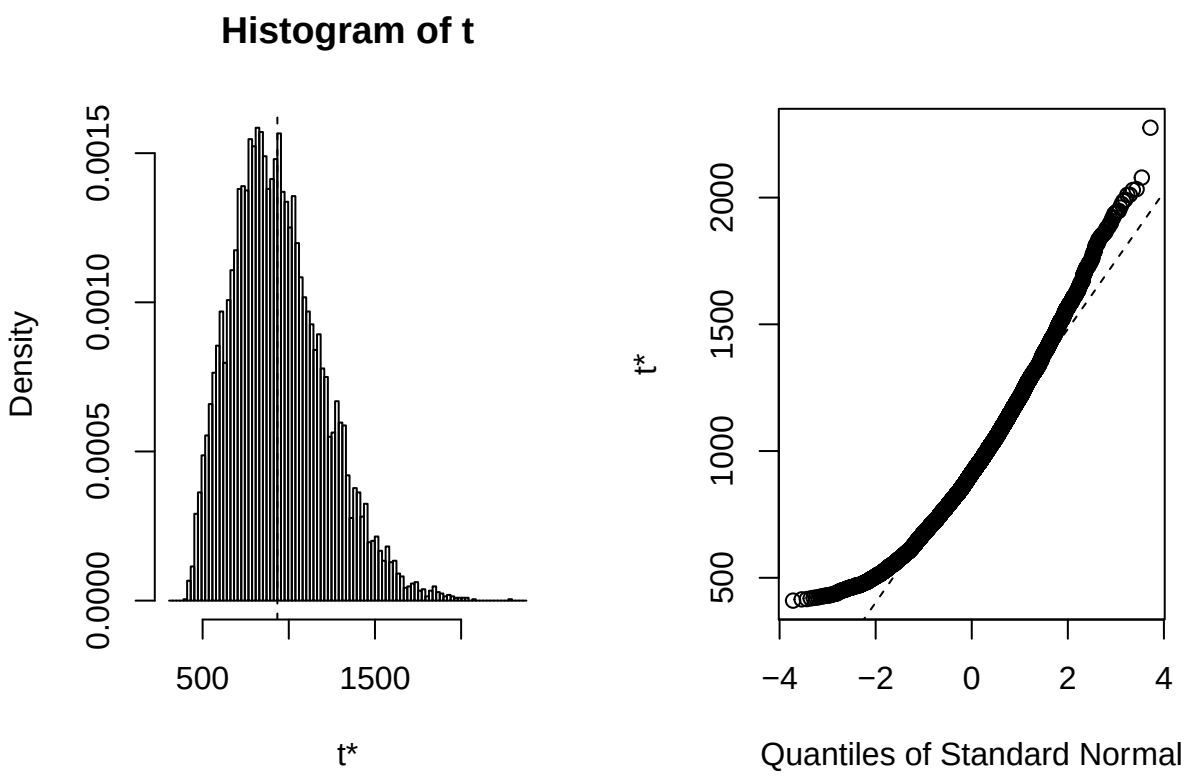

```
bootci.negfav
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.negfav, conf = 0.95, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 3851, 12668 )   ( 3604, 12388 )
##
## Level     Percentile            BCa
## 95%   ( 4189, 12974 )   ( 4608, 13766 )
## Calculations and Intervals on Original Scale
```

```r
# use bootstrap to find mean of followers_count for nagative, neutral and positive tweets

bootout.posfol <- boot(pos.fol, funmean, R = 10000)
bootci.posfol <- boot.ci(bootout.posfol, conf = 0.95, type = "all")
bootout.posfol
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = pos.fol, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1* 933.6947 8.445555    269.8191
```

```r
plot(bootout.posfol)
```




```r
bootci.posfol
```

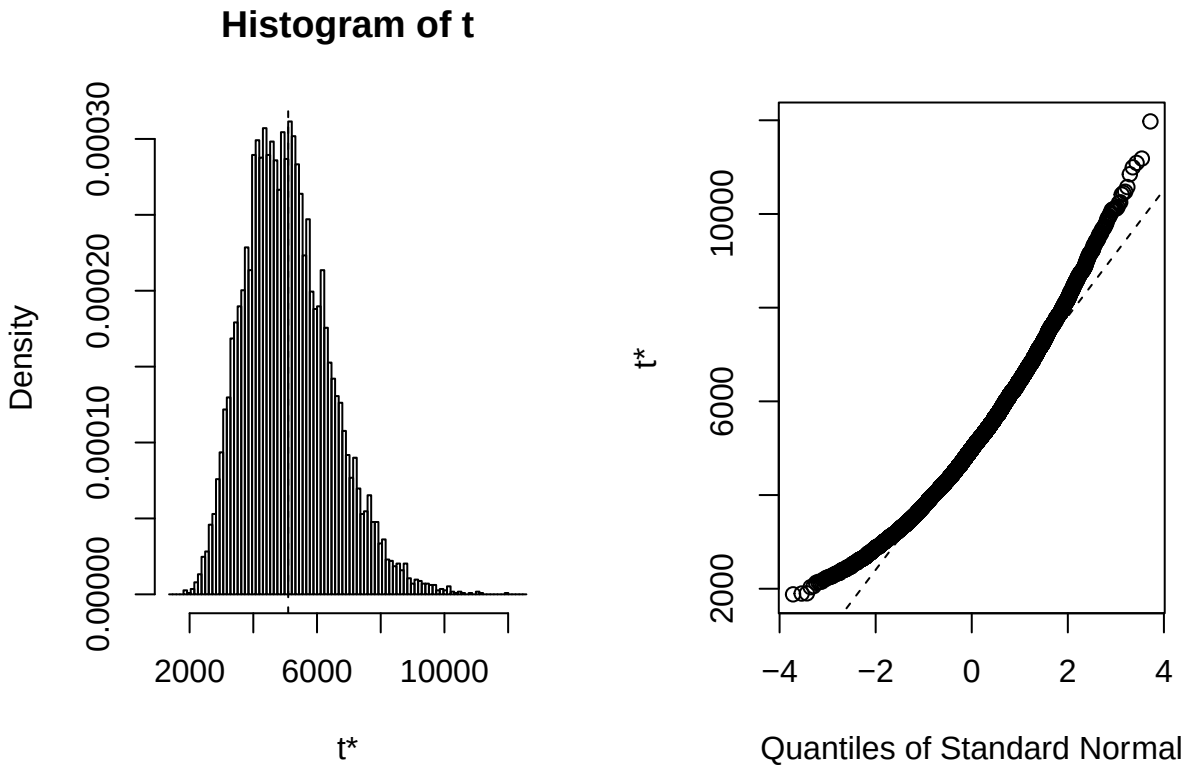
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.posfol, conf = 0.95, type = "all")
```

```
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 396.4, 1454.1 )   ( 312.5, 1356.6 )
##
## Level     Percentile            BCa
## 95%   ( 510.7, 1554.9 )   ( 579.4, 1853.6 )
## Calculations and Intervals on Original Scale
```

```
bootout.neufol <- boot(neu.fav, funmean, R = 10000)
bootci.neufol <- boot.ci(bootout.neufol, conf = 0.95, type = "all")
bootout.neufol
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = neu.fav, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1* 5094.888 13.28601    1352.224
```

```
plot(bootout.neufol)
```



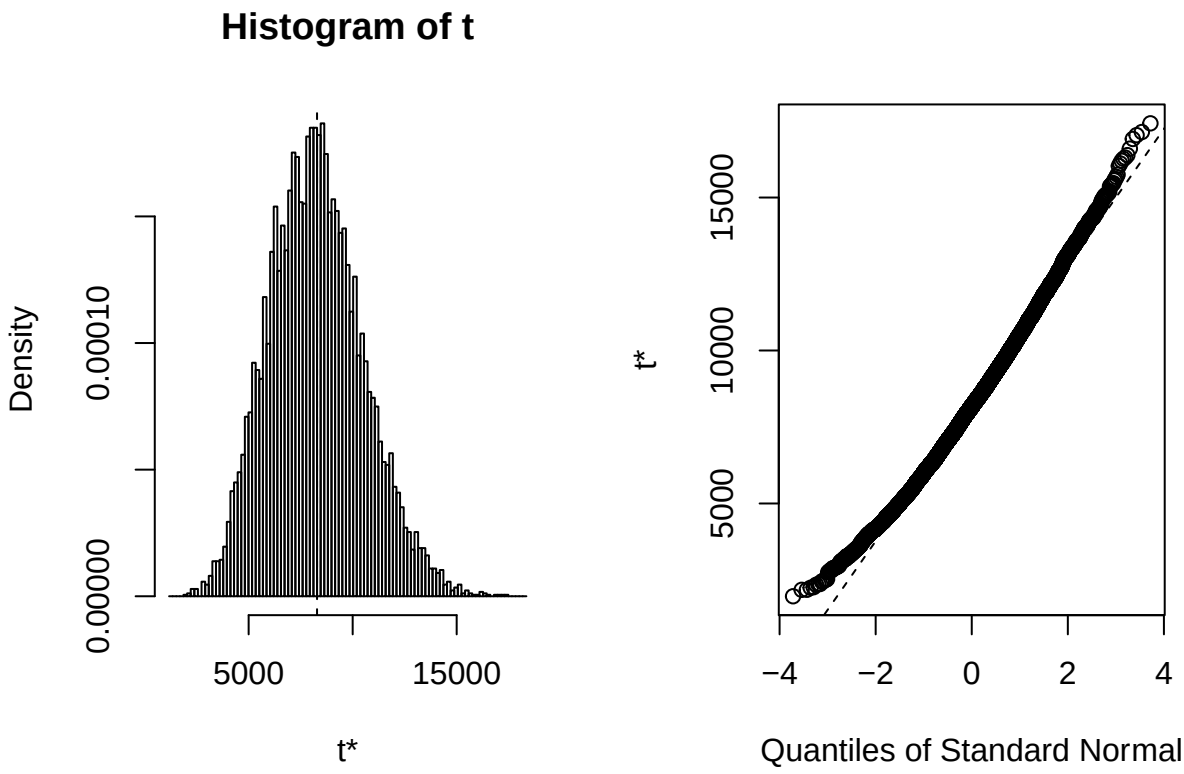
```
bootci.neufol
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.neufol, conf = 0.95, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   (2431, 7732 )   (2072, 7294 )
##
## Level     Percentile            BCa
## 95%   (2896, 8118 )   (3268, 9531 )
## Calculations and Intervals on Original Scale
```

```
bootout.negfol <- boot(neg.fav, funmean, R = 10000)
bootci.negfol <- boot.ci(bootout.negfol, conf = 0.95, type = "all")
bootout.negfol
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = neg.fav, statistic = funmean, R = 10000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 8288.808 -30.03282    2245.668
```

```
plot(bootout.negfol)
```




```
bootci.negfol
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootout.negfol, conf = 0.95, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 3917, 12720 )   ( 3535, 12343 )
##
## Level     Percentile            BCa
## 95%   ( 4234, 13043 )   ( 4666, 13825 )
## Calculations and Intervals on Original Scale
```

Shiny and Web applications

What is Shiny? Shiny is a web application framwork for R that turn your analyses into interactive web application. In the following, I will make a Shiny application that shows what sentiment counts are given different regions. Again, just to refresh what variables we get, we have four different regions: Northeast, South, West, and Midwest; three sentiments include postive, neutral, and negative. Since we have already saved tidy version of data at end of Statistical Model Building session, let me load it in. Shiny has two components: a user-interface script, and a server script.

The app can be located in this link: https://sijiexiang.shinyapps.io/Final_Project/

Conclusion

R is beatiful! Is not it?

Reference

A special-shout to professor Haviland and Zihuan(Vivian) Qiao. Professor Haviland leads me into R. He consistently impressed me with his intelligence, his insights, and his persistence in solving a wide variety of problem. He would be an asset to any undergradue/graduate students. Last but not at least is my teammate Vevian Qiao. Her code is as elegant as herself. It has been a wonderful ride! Thank you MA 615. Thank you Vivian.