Twitter Geolocator Project

Munan Hou, Tian Xia

November 27, 2016

1 Data Description

This project uses Twitter related packages to analyse word frequencies of a mass amount of Twitter, compares them among different locations, tries to find some underlying relations of the data.

We use 29 sample twitter dataset and each sample includes 5 minutes' twitter within an specific hour (from the last hour of 30 Nov to the fourth hour of 1 Dec (Est)).

```
########## Inputs of Secrets (Hidden)
#requestURL <- "https://api.twitter.com/oauth/request_token"
#accessURL <- "https://api.twitter.com/oauth/access_token"
#authURL <- "https://api.twitter.com/oauth/authorize"
#consumerKey <- "vt45JF7fSot9eQ3ocC1yVOCWt"
#consumerSecret <- "6LBLgJR7QvUY3OLpHYzlOyyvuHVoASrYHmMNCE7a7YNsaQ4iGf"
#access_token <- "1003943330-XxiMa42OlEOTvIxdMtQptFSLhNnDUIUEnanahMV"
#access_secret <- "MfV3oHNk4iFUIOcE74JOmDQ2Sx3pArrV9pjktWY4Sd43C"

########## Prepare for streamR
#my_oauth <- OAuthFactory$new(consumerKey = consumerKey, consumerSecret = consumerSecret,
#requestURL = requestURL, accessURL = accessURL, authURL = authURL)
#my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))
#save(my_oauth, file = "my_oauth.Rdata")
#load("my_oauth.Rdata")

########## Prepare for twitteR
#setup_twitter_oauth(consumerKey, consumerSecret, access_token, #access_secret)

########## Grab the data
#filterStream("tweetsUS_1129_23.json",
#             locations = c(-125, 25, -66, 50),
#             timeout = 300,
#             oauth = my_oauth)
#Sys.sleep(3280)
#filterStream("tweetsUS_1130_00.json",
#             locations = c(-125, 25, -66, 50),
#             timeout = 300,
#             oauth = my_oauth)
#Sys.sleep(3280)
#
#...
#
#filterStream("tweetsUS_1201_03.json",
#             locations = c(-125, 25, -66, 50),
#             timeout = 300,
#             oauth = my_oauth)
#
```

Here we grabbed 29 datasets and each contains 300 seconds grabbing of tweets. The filter are the region within United States "(-125, 25, -66, 50)". As long as continuously grabbing 24 hours' Twitter data will definitely overflow my RAM and ruin my computer, so my strategy is capturing a 300 seconds (5 minutes) sample for each hour of the day.

2 Data Cleaning

We merge all the subdatasets and add time_mark (for further adjustment of time-zone error, and geolocation analysis), clean the datasets, filter out sub-datasets for five states (MA, NY, DC, IL, CA).

We imported all the datasets into R, and marked them with different labels of their belonging hours, such as, 0, 1, 2, 3, . . . , 28, each represents 23:00 ~ 23:59 of 29 Nov, 00:00 ~ 00:59 of 30 Nov, etc.

```
tweets_00.df <- parseTweets("./data/tweetsUS_1129_23.json",verbose = FALSE)
tweets_01.df <- parseTweets("./data/tweetsUS_1130_00.json",verbose = FALSE)
tweets_02.df <- parseTweets("./data/tweetsUS_1130_01.json",verbose = FALSE)
tweets_03.df <- parseTweets("./data/tweetsUS_1130_02.json",verbose = FALSE)
tweets_04.df <- parseTweets("./data/tweetsUS_1130_03.json",verbose = FALSE)
tweets_05.df <- parseTweets("./data/tweetsUS_1130_04.json",verbose = FALSE)
tweets_06.df <- parseTweets("./data/tweetsUS_1130_05.json",verbose = FALSE)
tweets_07.df <- parseTweets("./data/tweetsUS_1130_06.json",verbose = FALSE)
tweets_08.df <- parseTweets("./data/tweetsUS_1130_07.json",verbose = FALSE)
tweets_09.df <- parseTweets("./data/tweetsUS_1130_08.json",verbose = FALSE)
tweets_10.df <- parseTweets("./data/tweetsUS_1130_09.json",verbose = FALSE)
tweets_11.df <- parseTweets("./data/tweetsUS_1130_10.json",verbose = FALSE)
tweets_12.df <- parseTweets("./data/tweetsUS_1130_11.json",verbose = FALSE)
tweets_13.df <- parseTweets("./data/tweetsUS_1130_12.json",verbose = FALSE)
tweets_14.df <- parseTweets("./data/tweetsUS_1130_13.json",verbose = FALSE)
tweets_15.df <- parseTweets("./data/tweetsUS_1130_14.json",verbose = FALSE)
tweets_16.df <- parseTweets("./data/tweetsUS_1130_15.json",verbose = FALSE)
tweets_17.df <- parseTweets("./data/tweetsUS_1130_16.json",verbose = FALSE)
tweets_18.df <- parseTweets("./data/tweetsUS_1130_17.json",verbose = FALSE)
tweets_19.df <- parseTweets("./data/tweetsUS_1130_18.json",verbose = FALSE)
tweets_20.df <- parseTweets("./data/tweetsUS_1130_19.json",verbose = FALSE)
tweets_21.df <- parseTweets("./data/tweetsUS_1130_20.json",verbose = FALSE)
tweets_22.df <- parseTweets("./data/tweetsUS_1130_21.json",verbose = FALSE)
tweets_23.df <- parseTweets("./data/tweetsUS_1130_22.json",verbose = FALSE)
tweets_24.df <- parseTweets("./data/tweetsUS_1130_23.json",verbose = FALSE)
tweets_25.df <- parseTweets("./data/tweetsUS_1201_00.json",verbose = FALSE)
tweets_26.df <- parseTweets("./data/tweetsUS_1201_01.json",verbose = FALSE)
tweets_27.df <- parseTweets("./data/tweetsUS_1201_02.json",verbose = FALSE)
tweets_28.df <- parseTweets("./data/tweetsUS_1201_03.json",verbose = FALSE)
```

Then, add "time_mark" to each of the dataframes and keep only needed variables.

```
tweets_00.df[,"time_mark"] <- as.integer(00);tweets_01.df[,"time_mark"] <- as.integer(01)
tweets_02.df[,"time_mark"] <- as.integer(02);tweets_03.df[,"time_mark"] <- as.integer(03)
tweets_04.df[,"time_mark"] <- as.integer(04);tweets_05.df[,"time_mark"] <- as.integer(05)
tweets_06.df[,"time_mark"] <- as.integer(06);tweets_07.df[,"time_mark"] <- as.integer(07)
tweets_08.df[,"time_mark"] <- as.integer(08);tweets_09.df[,"time_mark"] <- as.integer(09)
tweets_10.df[,"time_mark"] <- as.integer(10);tweets_11.df[,"time_mark"] <- as.integer(11)
tweets_12.df[,"time_mark"] <- as.integer(12);tweets_13.df[,"time_mark"] <- as.integer(13)
```

```
tweets_14.df[,"time_mark"] <- as.integer(14);tweets_15.df[,"time_mark"] <- as.integer(15)
tweets_16.df[,"time_mark"] <- as.integer(16);tweets_17.df[,"time_mark"] <- as.integer(17)
tweets_18.df[,"time_mark"] <- as.integer(18);tweets_19.df[,"time_mark"] <- as.integer(19)
tweets_20.df[,"time_mark"] <- as.integer(20);tweets_21.df[,"time_mark"] <- as.integer(21)
tweets_22.df[,"time_mark"] <- as.integer(22);tweets_23.df[,"time_mark"] <- as.integer(23)
tweets_24.df[,"time_mark"] <- as.integer(24);tweets_25.df[,"time_mark"] <- as.integer(25)
tweets_26.df[,"time_mark"] <- as.integer(26);tweets_27.df[,"time_mark"] <- as.integer(27)
tweets_28.df[,"time_mark"] <- as.integer(28)

keep <- c("text","lang","listed_count","geo_enabled","statuses_count","followers_count",
          "favourites_count","friends_count","time_zone","country_code","full_name",
          "place_lat","place_lon","time_mark")
tweets_00.df <- tweets_00.df[,keep];tweets_01.df <- tweets_01.df[,keep]
tweets_02.df <- tweets_02.df[,keep];tweets_03.df <- tweets_03.df[,keep]
tweets_04.df <- tweets_04.df[,keep];tweets_05.df <- tweets_05.df[,keep]
tweets_06.df <- tweets_06.df[,keep];tweets_07.df <- tweets_07.df[,keep]
tweets_08.df <- tweets_08.df[,keep];tweets_09.df <- tweets_09.df[,keep]
tweets_10.df <- tweets_10.df[,keep];tweets_11.df <- tweets_11.df[,keep]
tweets_12.df <- tweets_12.df[,keep];tweets_13.df <- tweets_13.df[,keep]
tweets_14.df <- tweets_14.df[,keep];tweets_15.df <- tweets_15.df[,keep]
tweets_16.df <- tweets_16.df[,keep];tweets_17.df <- tweets_17.df[,keep]
tweets_18.df <- tweets_18.df[,keep];tweets_19.df <- tweets_19.df[,keep]
tweets_20.df <- tweets_20.df[,keep];tweets_21.df <- tweets_21.df[,keep]
tweets_22.df <- tweets_22.df[,keep];tweets_23.df <- tweets_23.df[,keep]
tweets_24.df <- tweets_24.df[,keep];tweets_25.df <- tweets_25.df[,keep]
tweets_26.df <- tweets_26.df[,keep];tweets_27.df <- tweets_27.df[,keep]
tweets_28.df <- tweets_28.df[,keep]

tweets.df <- rbind(tweets_00.df,tweets_01.df,tweets_02.df,tweets_03.df,tweets_04.df,
                   tweets_05.df,tweets_06.df,tweets_07.df,tweets_08.df,tweets_09.df,
                   tweets_10.df,tweets_11.df,tweets_12.df,tweets_13.df,tweets_14.df,
                   tweets_15.df,tweets_16.df,tweets_17.df,tweets_18.df,tweets_19.df,
                   tweets_20.df,tweets_21.df,tweets_22.df,tweets_23.df,tweets_24.df,
                   tweets_25.df,tweets_26.df,tweets_27.df,tweets_28.df)
# back up: write.csv(tweets.df,"tweets.df.csv")
```

Next, remove tweets with no geo_enabled and not English, and filter tweets with extreme locations. After that, prepare the five-state sub-datasets, for further popularity analysis:

```
#Remove tweets with no geo_enabled and not English
tweets.df <- tweets.df[tweets.df$lang=="en",]
tweets.df <- tweets.df[tweets.df$geo_enabled==TRUE,]
#clean up extreme values
tweets.df <- tweets.df[tweets.df$place_lat >25 & tweets.df$place_lat <50 &
                         tweets.df$place_lon > -125 & tweets.df$place_lon< -66,]
#Filter different places
ca <- data.frame(filter(tweets.df, grepl(', CA', full_name)))
il <- data.frame(filter(tweets.df, grepl(', IL', full_name)))
ma <- data.frame(filter(tweets.df, grepl(', MA', full_name)))
ny <- data.frame(filter(tweets.df, grepl(', NY', full_name)))
dc <- data.frame(filter(tweets.df, grepl(', DC', full_name)))
#Adjust based on time zone
ma <- ma %>% filter(time_mark >= 1 & time_mark <= 24)
```

```
ny <- ny %>% filter(time_mark >= 1 & time_mark <= 24)
dc <- dc %>% filter(time_mark >= 1 & time_mark <= 24)
ca <- ca %>% filter(time_mark >= 4 & time_mark <= 27)
ca$time_mark <- ca$time_mark - 3
il <- il %>% filter(time_mark >= 2 & time_mark <= 25)
il$time_mark <- il$time_mark - 1
```

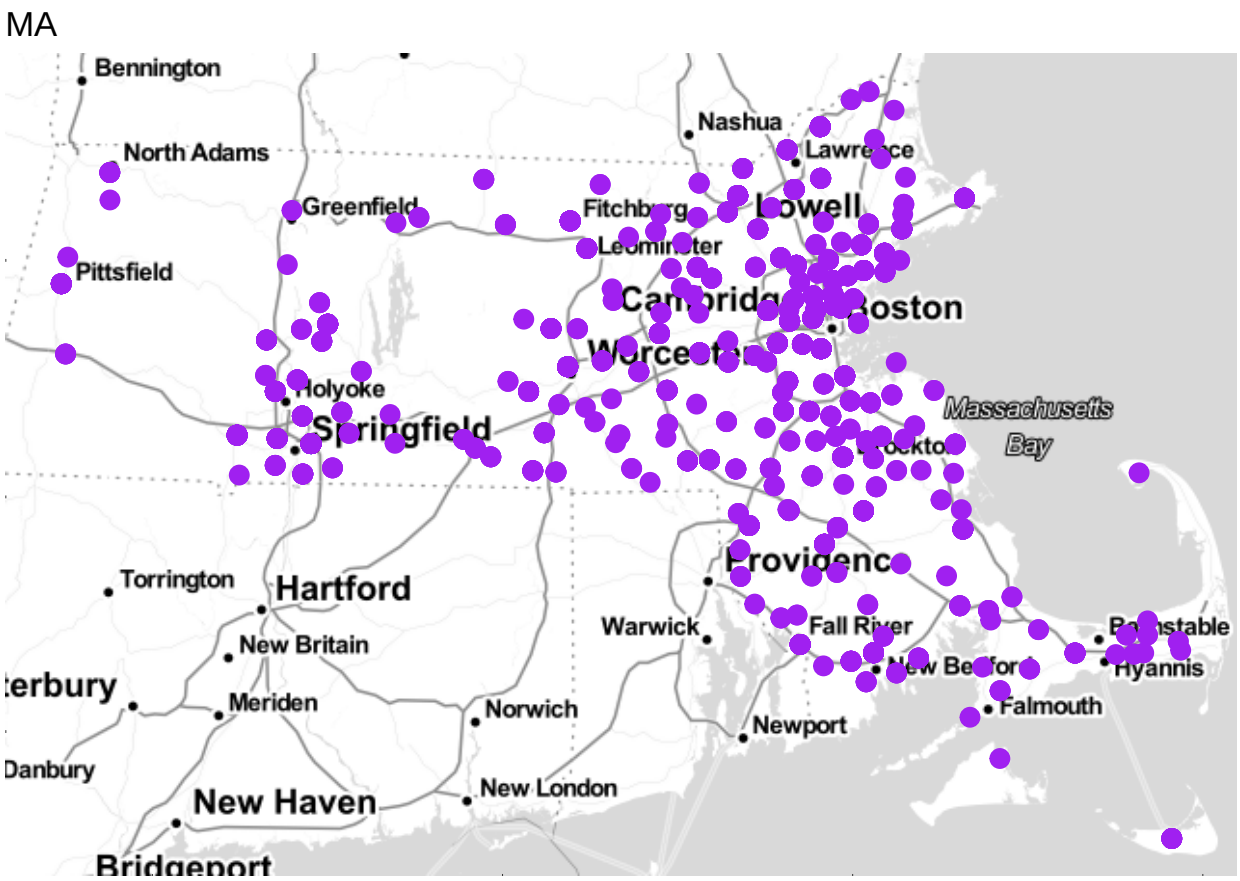
3 Geolocation Analysis

First of all, we want to generate an interactive distribution plot of all twitters using ShinyApp to show the trend of all twitter's distribution within the continent of United States through timeline in one day(and a little more at both tails).

The app is in another project's folder within this repository named "app".

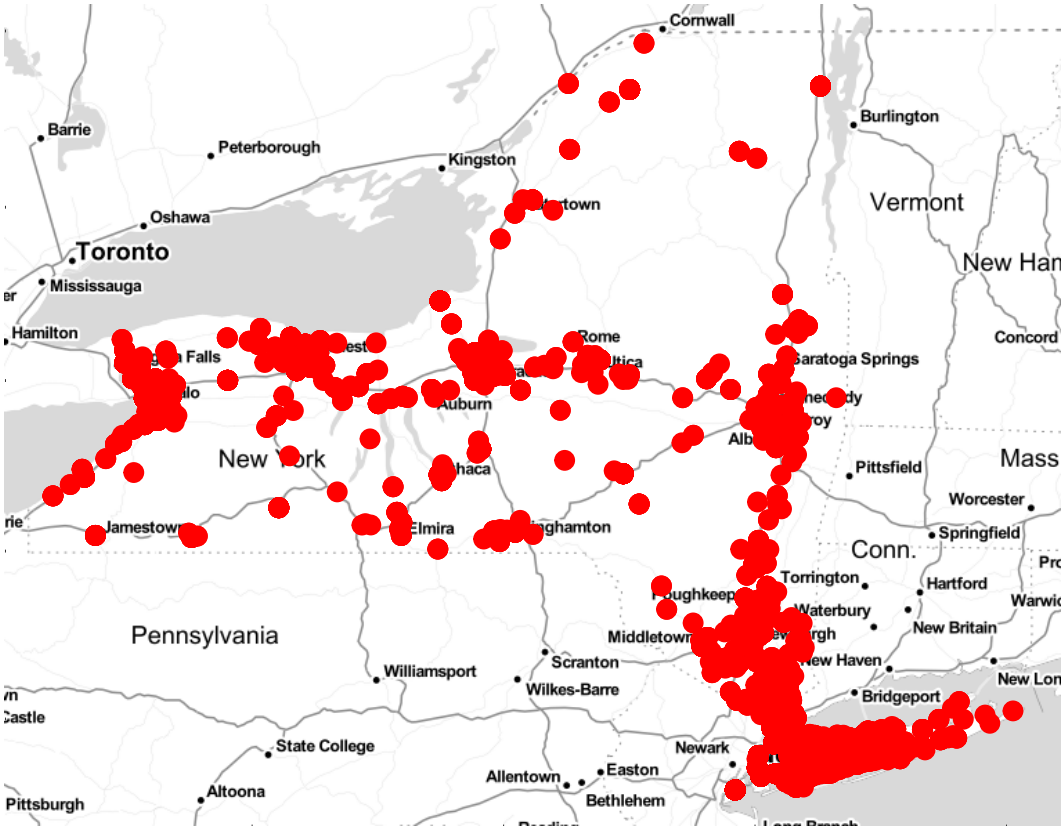
After that we also plot the Twitter activity in MA, NY, IL and CA respectively.

```
qmplot(place_lon, place_lat, data = ma, colour = I('purple'), size = I(3),
       mapcolor = "color", main="MA")
```



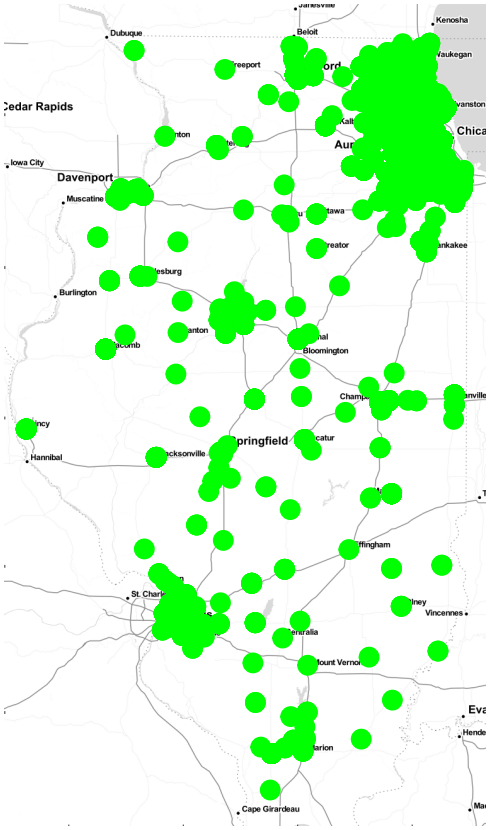

```
qmplot(place_lon, place_lat, data = ny, colour = I('red'), size = I(3),
       mapcolor = "color", main="NY")
```

NY

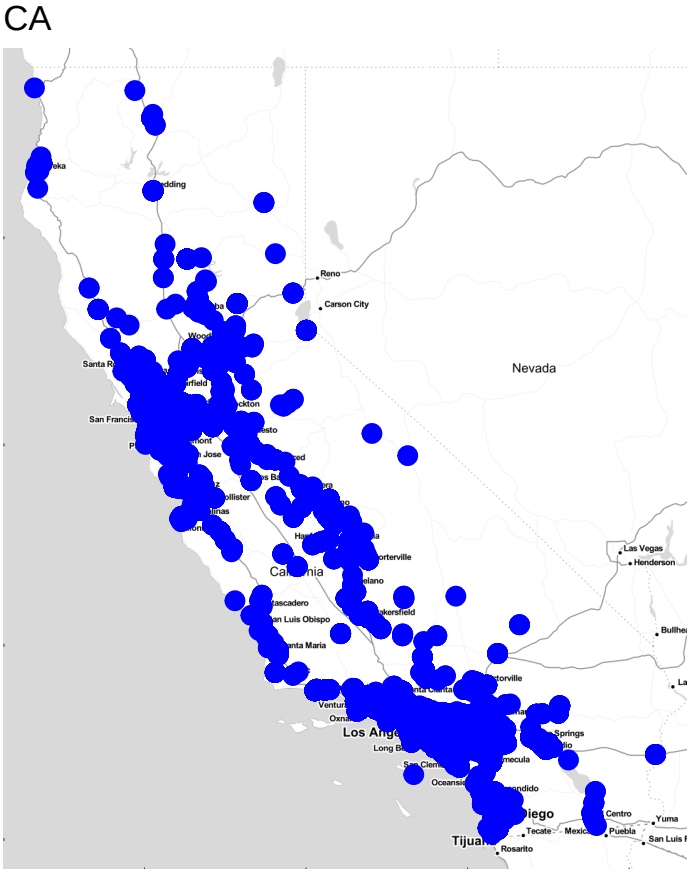



```
qmplot(place_lon, place_lat, data = il, colour = I('green'), size = I(3),
       mapcolor = "color", main="IL")
```

IL




```
qmplot(place_lon, place_lat, data = ca, colour = I('blue'), size = I(3),
       mapcolor = "color",main="CA")
```



There is no doubt that most tweets are posted in some major cities and in the following sections we will analyze this pattern in different states and cities.

```
#time_mark start from 0
ca[,"time_mark"] <- ca[,"time_mark"]-1;ny[,"time_mark"] <- ny[,"time_mark"]-1
il[,"time_mark"] <- il[,"time_mark"]-1;dc[,"time_mark"] <- dc[,"time_mark"]-1
ma[,"time_mark"] <- ma[,"time_mark"]-1

#add new variable state
ca[,"state"] <- rep("CA",nrow(ca));ny[,"state"] <- rep("NY",nrow(ny))
il[,"state"] <- rep("IL",nrow(il));dc[,"state"] <- rep("DC",nrow(dc))
ma[,"state"] <- rep("MA",nrow(ma))

#get city name
ma$full_name <- as.character(ma$full_name);ny$full_name <- as.character(ny$full_name)
il$full_name <- as.character(il$full_name);ca$full_name <- as.character(ca$full_name)
dc$full_name <- as.character(dc$full_name)

get_city <- function(x){
  city_name <- c()
  name <- strsplit(x$full_name,",")
  for(i in 1:nrow(x)){
    city_name <- c(city_name,name[[i]][1])
  }
  return(city_name)
}
```

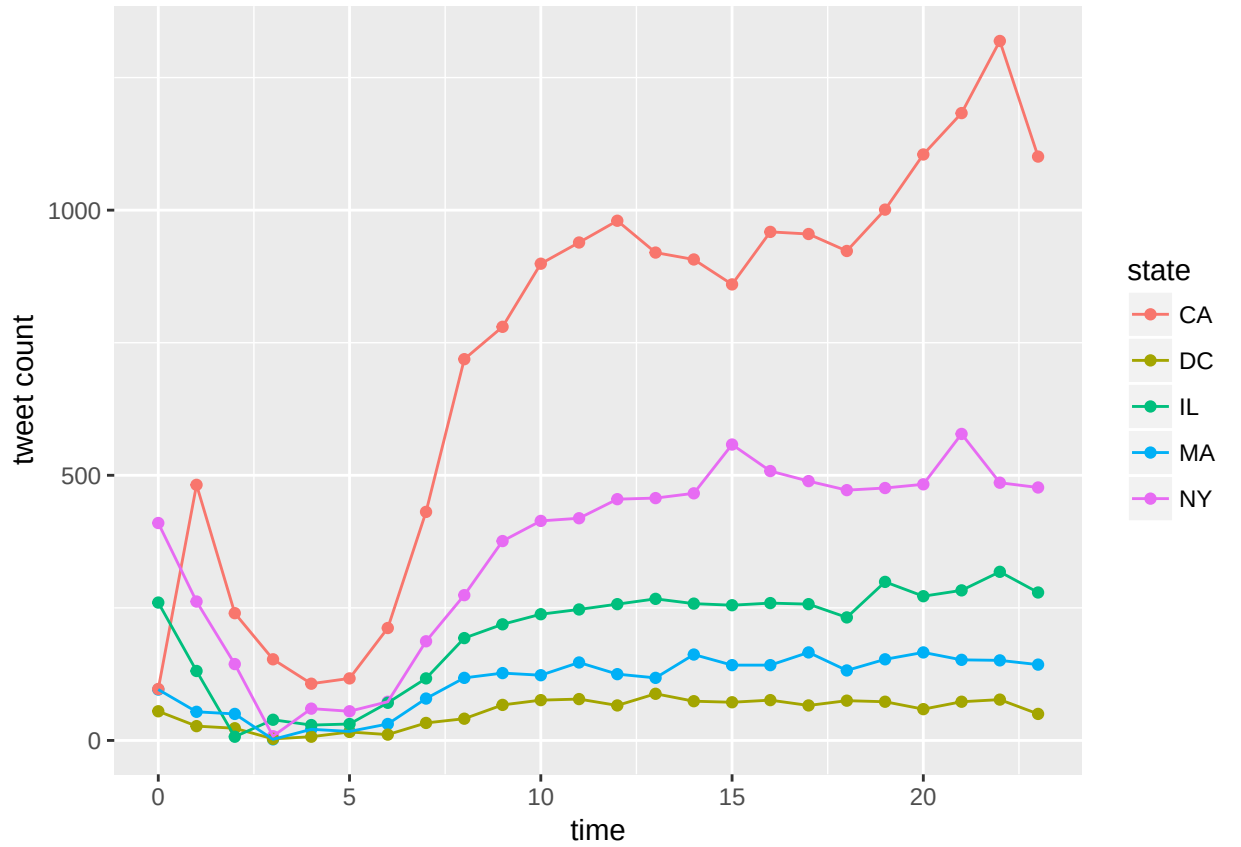
```
# transform to factor
ma[,"city"] <- factor(get_city(ma));ny[,"city"] <- factor(get_city(ny))
il[,"city"] <- factor(get_city(il));ca[,"city"] <- factor(get_city(ca))
dc[,"city"] <- factor(get_city(dc))

#combine together
total <- rbind(rbind(rbind(rbind(ca,ny),il),dc),ma)
total$lang <- as.character(total$lang)
total <- filter(total,lang=="en")

#factor time_mark
ma[,"time_mark"] <- factor(ma[,"time_mark"]);ca[,"time_mark"] <- factor(ca[,"time_mark"])
ny[,"time_mark"] <- factor(ny[,"time_mark"]);dc[,"time_mark"] <- factor(dc[,"time_mark"])
il[,"time_mark"] <- factor(il[,"time_mark"])

total_time_count <- data.frame("Num_tweet"=c(summary(ma$time_mark),summary(ca$time_mark),
                        summary(ny$time_mark),summary(dc$time_mark),summary(il$time_mark)))
total_time_count[,"time"] <- c(0:23,0:23,0:23,0:23,0:23)
total_time_count[,"state"] <- c(rep("MA",24),rep("CA",24),rep("NY",24),
                                rep("DC",24),rep("IL",24))

ggplot(total_time_count,aes(x=time,y=Num_tweet))+geom_point(aes(color=state))+
  geom_line(aes(color=state))+ylab("tweet count")
```



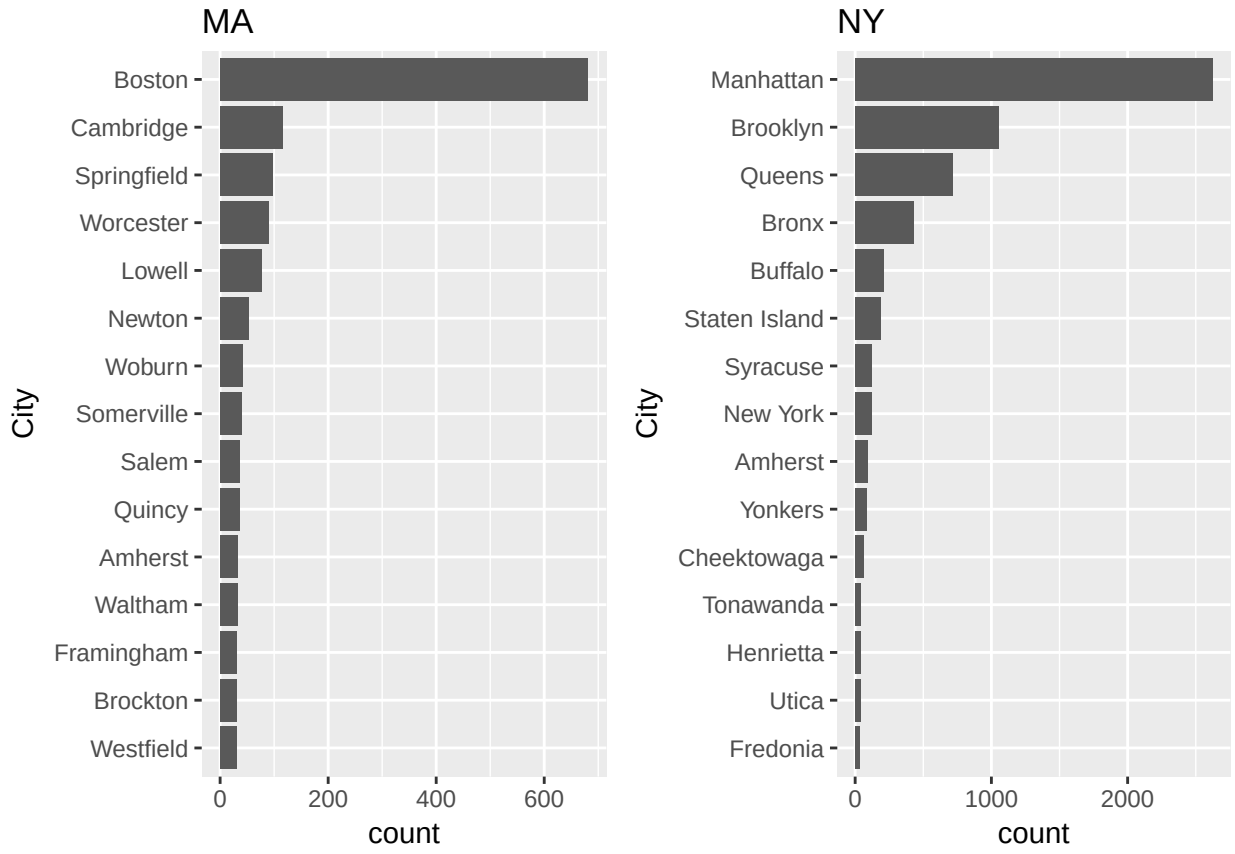
Here is the plot of the number of tweets posted in different time in one day for five states. It is obvious that from 3am to 5am the number of tweets is lowest and from 21pm to 23pm the number is highest. CA has the

highest number of tweets, since its population is large, and the number goes up sharply in the morning. Whereas the curves for IL, MA and DC are more smoothy and have the similar pattern.

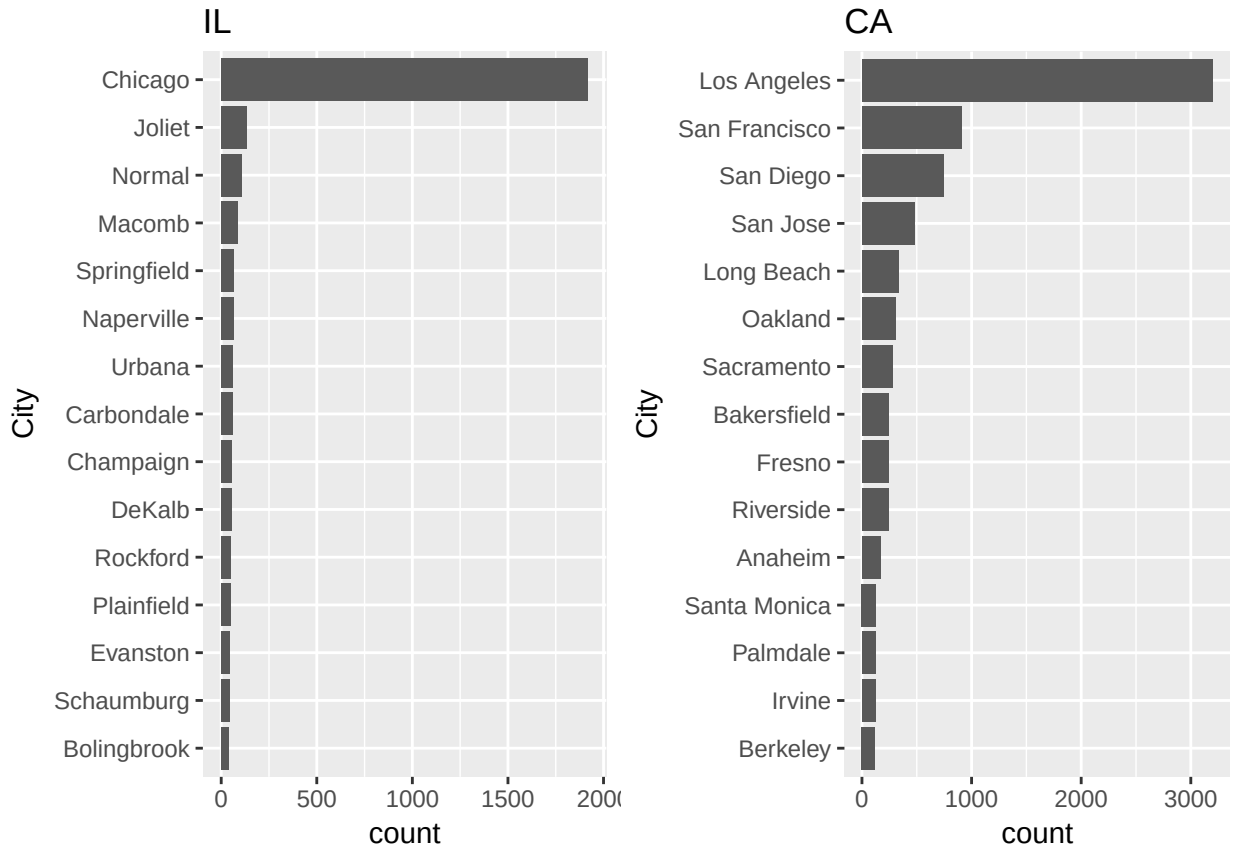
Let us take a look at the number of tweets in different city.

```
# tweet number for every city in state
count <- summary(ma$city)[1:15];ma_city_count <- as.data.frame(count)
count <- summary(ny$city)[1:15];ny_city_count <- as.data.frame(count)
count <- summary(il$city)[1:15];il_city_count <- as.data.frame(count)
count <- summary(ca$city)[1:15];ca_city_count <- as.data.frame(count)


macityplot <- ggplot(ma_city_count,aes(reorder(rownames(ma_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("MA")
nycityplot <- ggplot(ny_city_count,aes(reorder(rownames(ny_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("NY")
grid.arrange(macityplot, nycityplot, ncol=2)
```



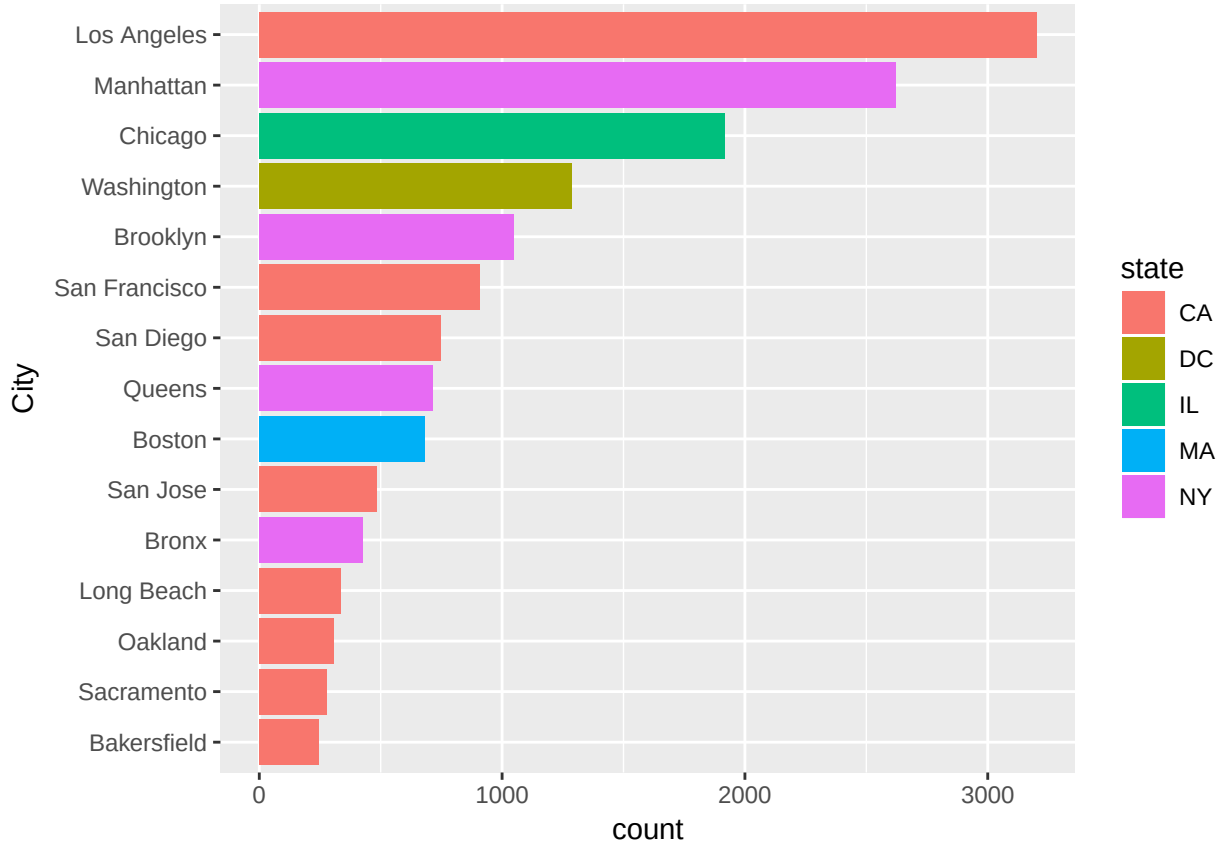

```
ilcityplot <- ggplot(il_city_count,aes(reorder(rownames(il_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("IL")
cacityplot <- ggplot(ca_city_count,aes(reorder(rownames(ca_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("CA")
grid.arrange(ilcityplot, cacityplot, ncol=2)
```



In the above I plot TOP10 cities of the number of tweets in four states. Not surprisingly, in MA and IL most tweets are posted in Boston and Chicago, and the sum of remaining nine cities is just equal to this super city. In CA and NY, more cities have large number of tweets. Apart from Los Angeles and Manhattan, San Francisco, San Diego, San Jose, Brooklyn, Queens, Bronx aslo have large number of tweets.

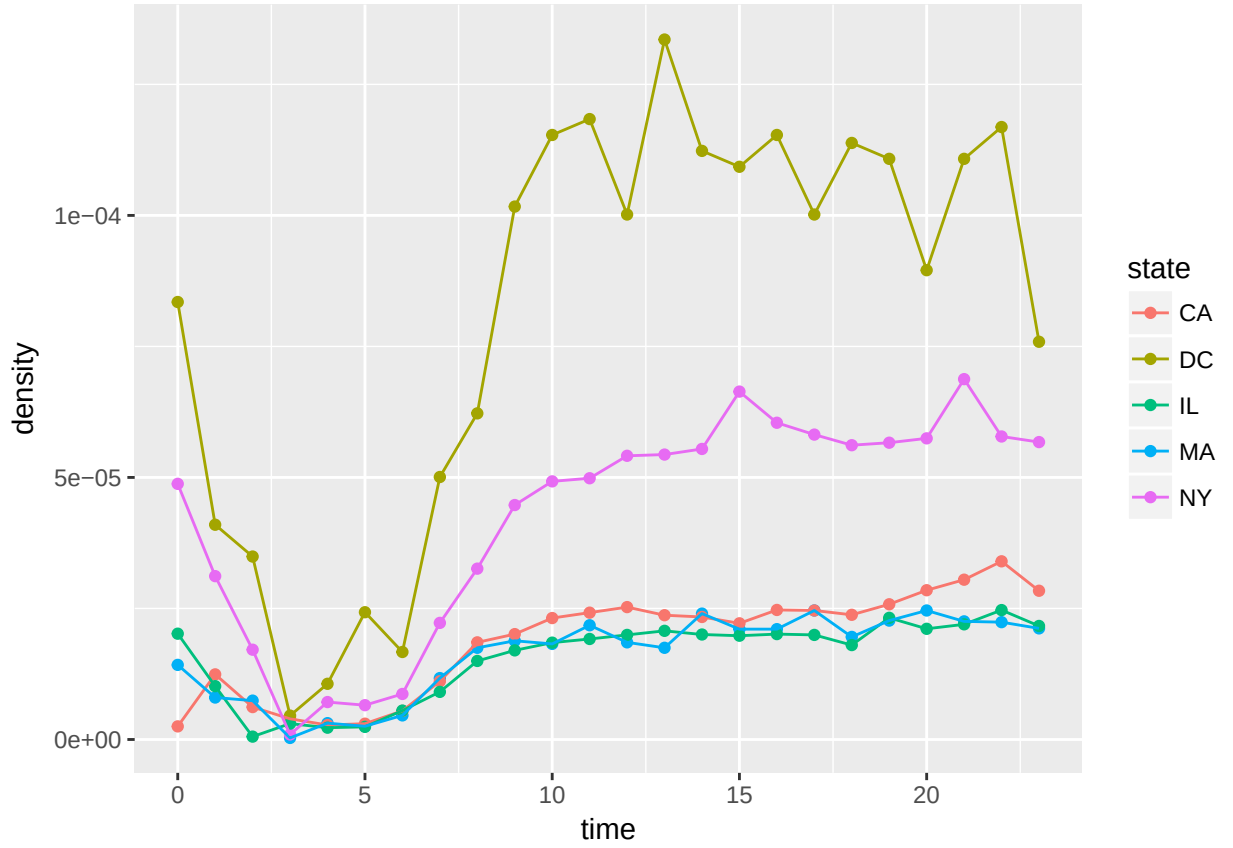
```
#the most tweets in all five states
get_city2 <- function(x){
  city_name <- c()
  name <- strsplit(x,",")
  for(i in 1:length(x)){
    city_name <- c(city_name,name[[i]][1])
  }
  return(city_name)
}
get_state2 <- function(x){
  state_name <- c()
  name <- strsplit(x,",")
  for(i in 1:length(x)){
    state_name <- c(state_name,name[[i]][2])
  }
  return(state_name)
}
total$full_name <- as.factor(total$full_name)
count <- summary(total$full_name)[1:15];total_city_count <- as.data.frame(count)
total_city_count[,"cs"] <- row.names(total_city_count)
total_city_count[,"city"] <- get_city2(as.character(total_city_count[,"cs"]))
total_city_count[,"state"] <- get_state2(as.character(total_city_count[,"cs"]))
```

```
ggplot(total_city_count,aes(reorder(city,count),count))+
  geom_bar(stat = "identity",aes(fill=state))+coord_flip()+xlab("City")
```



Now combining all cities in five states, what are TOP15 cities of the number of tweets? Here we can see Los Angeles is still TOP1 and CA has 8 cities in TOP15, NY has 3, and DC, IL, MA has 1 respectively.

```
#tweet density
total_time_count[,"population"] <- c(rep(6.745*10^6,24),rep(38.8*10^6,24),
                                     rep(8.406*10^6,24),rep(658893,24),rep(12.88*10^6,24))
total_time_count[,"density"] <- total_time_count$Num_tweet/total_time_count$population
ggplot(total_time_count,aes(x=time,y=density))+geom_point(aes(color=state))+
  geom_line(aes(color=state))
```



But it is not fair to compare two cities if their population are greatly different. Thus I define density here by the number of tweets per person in this state and make a plot of tweet density for five states. Now DC stands out, having the largest density at any time in one day, and MA, IL have the same pattern for density this time.

But do they truly have the same density pattern? First I want to test whether they have the same density variance in one day?

```
#usual test for the mean density between IL and MA, first test variance
den_IL <- filter(total_time_count,state=="IL")[,"density"]
den_MA <- filter(total_time_count,state=="MA")[,"density"]
var.test(den_IL,den_MA)
```

```
##
##  F test to compare two variances
##
## data:  den_IL and den_MA
## F = 0.97027, num df = 23, denom df = 23, p-value = 0.9429
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.4197322 2.2429153
## sample estimates:
## ratio of variances
##          0.9702699
```

The p-value is 0.9553, which indicates we could not reject the null Hypothesis. We are 95% confident that MA and IL ahve the same density variance in one day.

Next I use a t test for these two small samples to figure out whether they have the same density mean.

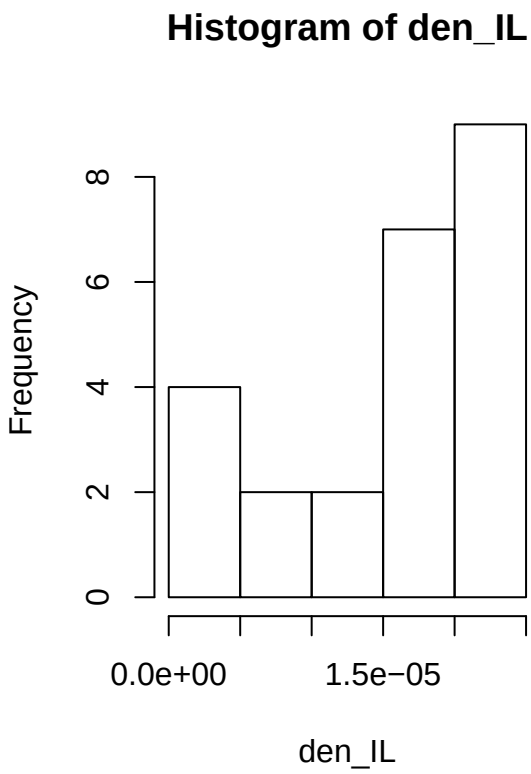
```
t.test(den_IL,den_MA)
```

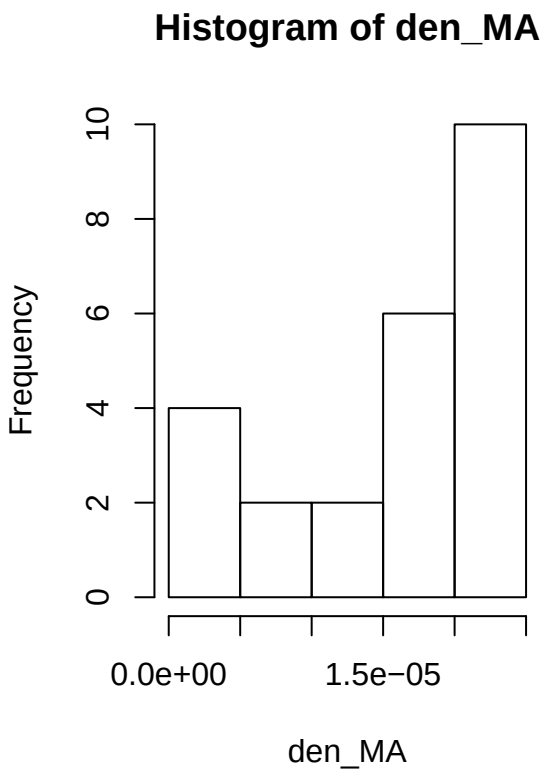
```
##
##  Welch Two Sample t-test
##
## data:  den_IL and den_MA
## t = -0.26215, df = 45.99, p-value = 0.7944
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.034489e-06  3.874257e-06
## sample estimates:
##    mean of x    mean of y
## 1.558618e-05 1.616630e-05
```

The p-value is 0.7997, which indicates we could not reject the null Hypothesis. We are 95% confident that MA and IL ahve the same density mean in one day.

But there are two assumptions for this t test: one is he two samples are randomly selected in an independent manner from the two target populations, and th other is both sampled populations have distributions that are approximately normal. We are not sure about the second assumption and the sample size is too small that it will affect the accuracy of test. In this case we could use permutation test.

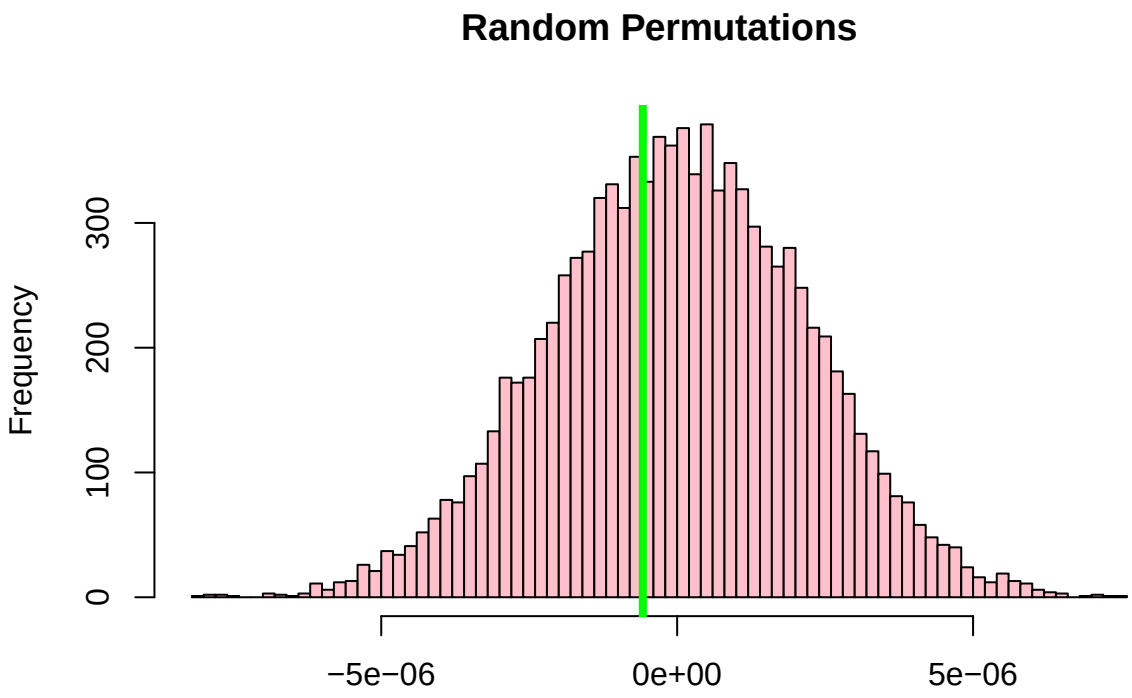
```
#permutation test for the mean density between IL and MA
par(mfrow=c(1,2))
hist(den_IL)
hist(den_MA)
```





From the histogram of the densities, they seem to have the same dristribution.

```
data <- c(den_IL, den_MA)
l1 <- length(den_IL)
l2 <- length(den_MA)
lt <- l1+l2
test.diff <- mean(den_IL) - mean(den_MA)
it <- function(n){
  M = NULL
  for(i in 1:n){
    s = sample(data, lt, FALSE)
    m1 = mean(s[1:l1]) - mean(s[(l1+1):lt])
    M = c(M,m1)
  }
  return(M)
}
examples <- it(10000)
par(mfrow=c(1,1))
hist(examples, col = "pink", breaks = 100, main="Random Permutations", xlab="")
abline(v = test.diff, col = "green", lwd = 4)
```



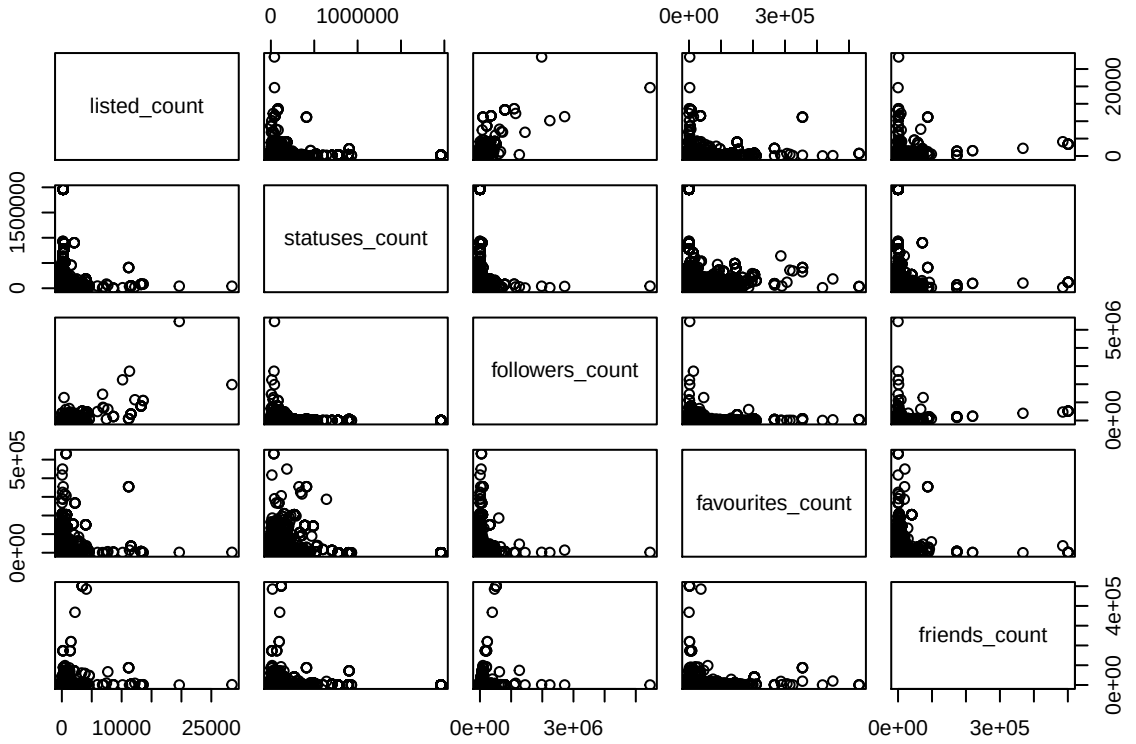

```
(sum(examples<test.diff)+sum(examples>-test.diff))/(10000) #two tail p-value
```

```
## [1] 0.7914
```

The two tail p-value of this permutation test is 0.8002, which gives us the same conclusion from the t test. So we are 95% confident that MA and IL have the same density mean in one day.

4 Popularity Analysis

```
#five count number
pairs(~listed_count+statuses_count+followers_count+favourites_count+friends_count, data=total)
```




```
cor(total[,c(3,5,6,7,8)])
```

```
##                 listed_count statuses_count followers_count
## listed_count       1.00000000     0.07120091      0.69389857
## statuses_count     0.07120091     1.00000000      0.01545478
## followers_count    0.69389857     0.01545478      1.00000000
## favourites_count   0.12208206     0.06819005      0.04835951
## friends_count      0.19402294     0.02817246      0.20494359
##                 favourites_count friends_count
## listed_count          0.12208206    0.19402294
## statuses_count        0.06819005    0.02817246
## followers_count       0.04835951    0.20494359
## favourites_count      1.00000000    0.07657018
## friends_count         0.07657018    1.00000000
```

Follower counts have long provided a decent indicator of a Twitter account's popularity - though they are relatively easily gamed if you put your mind to it. This is obviously another indicator of popularity - the more lists you are on, the more popular you probably are. In turn, one would guess that in most cases there is a high correlation between how many followers an account has, and how many lists they are on. And is there any relationship among listed count, statuses count, followers count, favourites count and friends count?

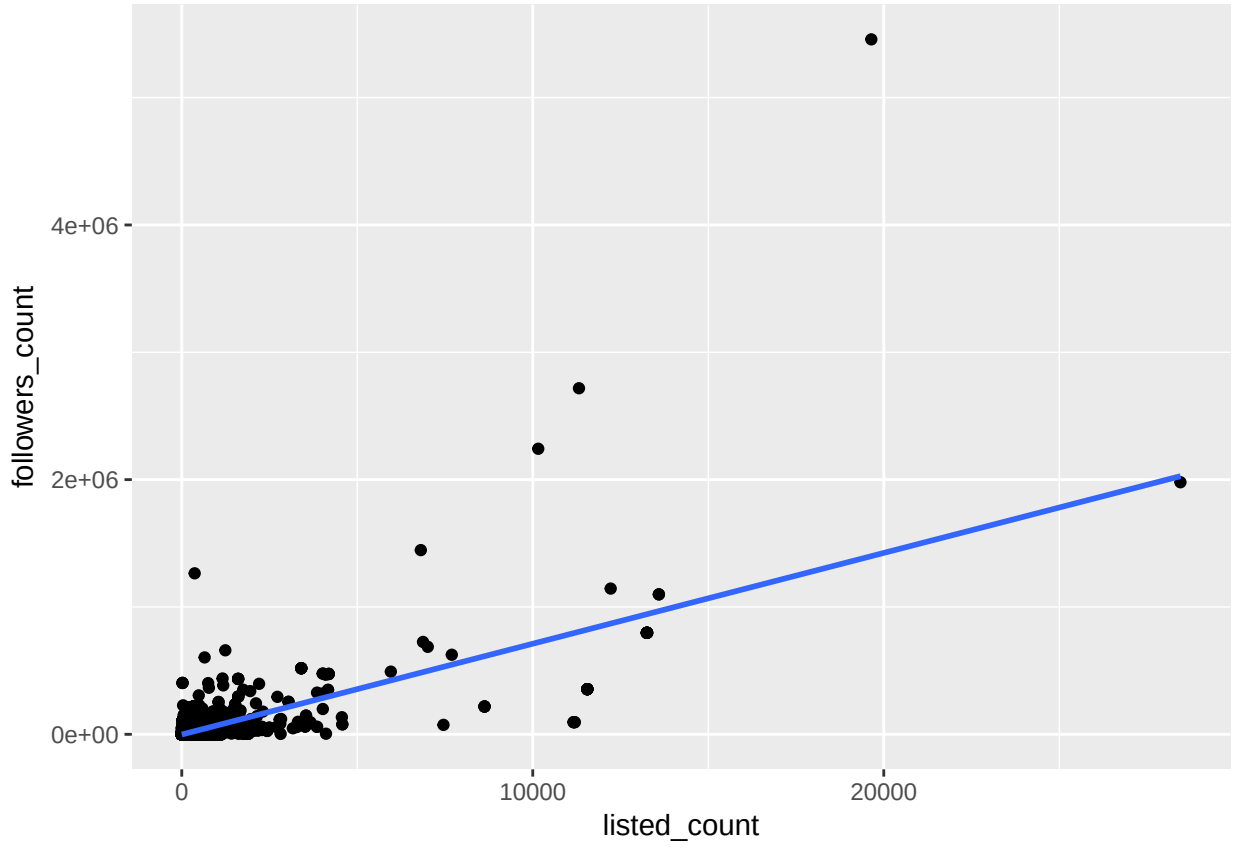
I make pair plots of these variables and calculate the correlation amony them. From the output, followers count and listed count have strong linear relation and friends count also have slight linear relation with these two variables.

So first make a simple linear regression between followers count and listed count.

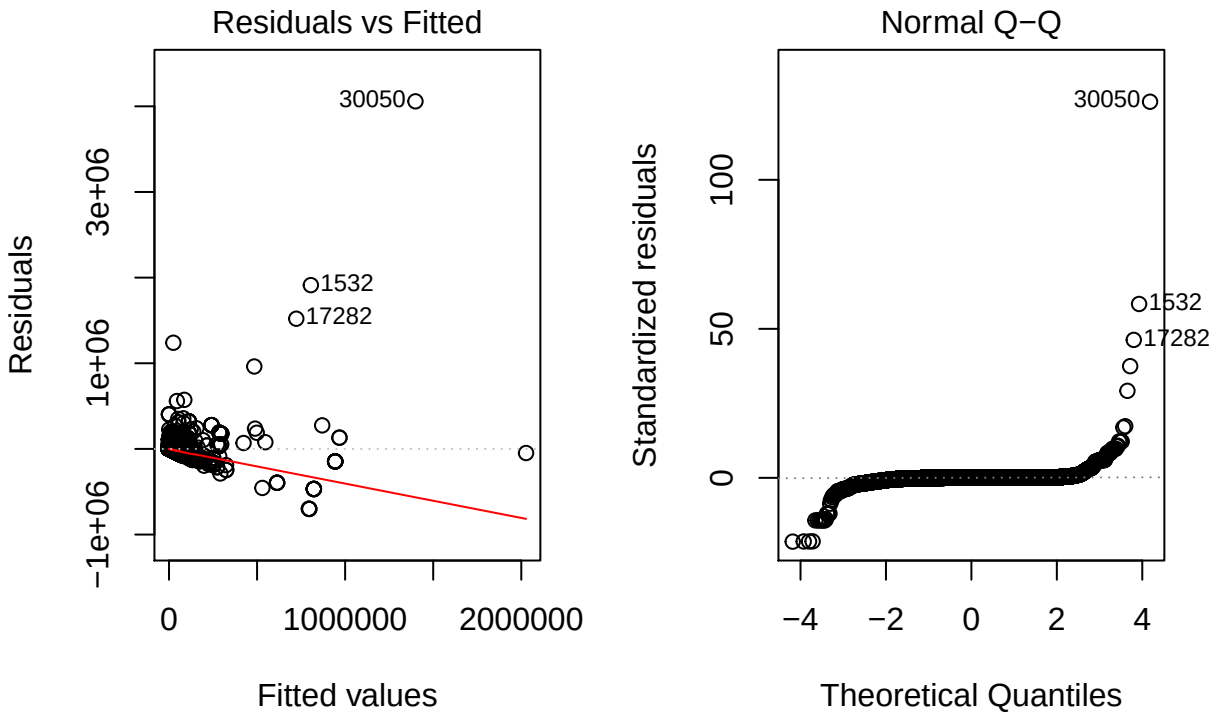
```r
#followers and listed have relation
model_fl <- lm(followers_count~listed_count,data=total)
summary(model_fl)
```

```
##
## Call:
## lm(formula = followers_count ~ listed_count, data = total)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -701632     266    1497    1833 4058074
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1550.4769   180.0613  -8.611   <2e-16 ***
## listed_count    71.3025     0.3972 179.495   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33080 on 34695 degrees of freedom
## Multiple R-squared:  0.4815, Adjusted R-squared:  0.4815
## F-statistic: 3.222e+04 on 1 and 34695 DF,  p-value: < 2.2e-16
```

```r
ggplot(total,aes(x=listed_count,y=followers_count))+geom_point()+geom_smooth(method = "lm")
```



```
par(mfrow=c(1,2))
plot(model_fl,1)
plot(model_fl,2)
```



The model is not bad with two significant variables and the R-squared is 0.4815, which means that nearly half of the variability in followers count canbe explained by listed count. But Normal Q-Q plot and Residuals vs Fitted plot are not good, indicating that there are heteroskedasticity and unlinear problems in the model.

5 Text mining

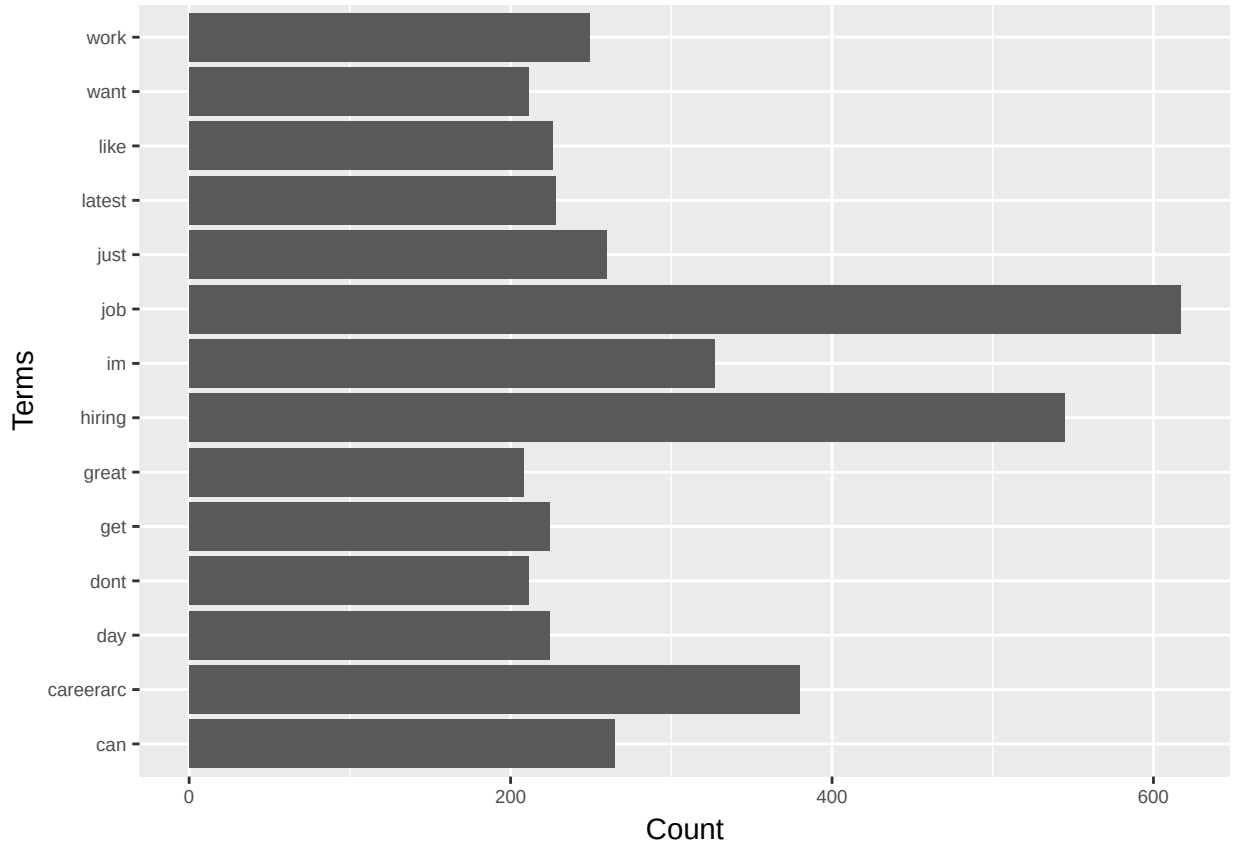
```
#Sample for wordcloud
tweets_sample.df <- tweets_10.df
tweets_sample.df <- tweets_sample.df[tweets_sample.df$lang=="en",]
tweets_sample.df <- tweets_sample.df[tweets_sample.df$geo_enabled==TRUE,]
tweets_sample.df <- tweets_sample.df[tweets_sample.df$place_lat >25 &
                    tweets_sample.df$place_lat <50 & tweets_sample.df$place_lon > -125 &
                    tweets_sample.df$place_lon< -66,]
tweets_sample.df$geo_enabled <- NULL
```

I extract a sample of tweets data and want to conduct textmining on this sample data.

```
# build a corpus, and specify the source to be character vectors
myCorpus <- Corpus(VectorSource(tweets_sample.df$text))
# remove anything other than English letters or space(!!!)
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))
# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
# remove stopwords
myStopwords <- c(stopwords('english'),"use", "see", "used", "via", "amp")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
# remove extra whitespace
myCorpus <- tm_map(myCorpus, stripWhitespace)
# remove punctuation
myCorpus <- tm_map(myCorpus, removePunctuation)
```

First, because there are so many characters do not make sense in our data, I need to clean the text data.I build a corpus, and specify the source to be character vectors, remove anything other than English letters or space, convert everything to lower case and remove URLs. And then stopwords, extra whitespace, punctuation should also be removed.

```
# Build Term Document Matrix
tdm <- TermDocumentMatrix(myCorpus, control = list(wordLengths = c(1, Inf)))
term.freq <- rowSums(as.matrix(tdm))
term.freq2 <- subset(term.freq, term.freq >= 200)
df <- data.frame(term = names(term.freq2), freq = term.freq2)
par(mfrow=c(1,1))
ggplot(df, aes(x=term, y=freq)) + geom_bar(stat="identity") + xlab("Terms") +
  ylab("Count") + coord_flip() + theme(axis.text=element_text(size=7))
```



Now we are able to see the most frequent words in the sample Twitter data.

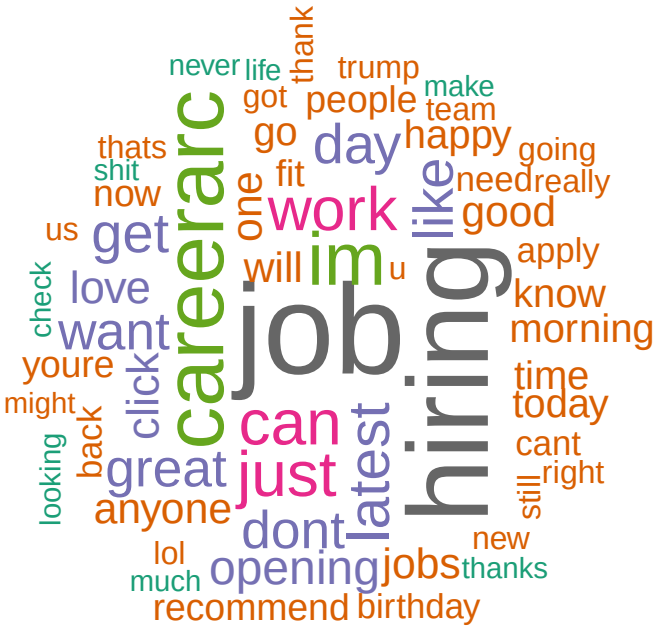
Build word cloud

```
m <- as.matrix(tdm)

# calculate the frequency of words and sort it by frequency
word.freq <- sort(rowSums(m), decreasing = T)

# colors
pal <- brewer.pal(8, "Dark2")

# plot word cloud
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 800,
          random.order = F, colors = pal, max.words = 60)
```



Furthermore, I want to do some clustering analysis on the sample text and I use K-means clustering with 6 centers first. The output gives us six clusters and all the words in the clusters are highly frequent words.

```
# remove sparse terms
tdm2 <- removeSparseTerms(tdm,sparse = 0.965)
# showing the words that are left gor the analysis
print(dimnames(tdm2)$Terms)
```

```
##  [1] "can"       "careerarc" "day"       "dont"      "get"
##  [6] "great"     "hiring"    "im"        "job"       "just"
## [11] "latest"    "like"      "want"      "work"
```

```
m2 <- as.matrix(tdm2)
m3 <- t(m2)   # transpose the matrix to cluster documents
set.seed(122)   # set a fixed random seed
k <- 6   # number of clusters
kmeansResult <- kmeans(m3,k)
round(kmeansResult$centers, digits = 3)   # cluster centers
```

```
##      can careerarc   day  dont   get great hiring    im   job  just latest
## 1 0.014     0.000 0.031 0.031 0.059 0.010  0.000 1.087 0.007 0.094  0.000
## 2 0.000     0.000 0.047 0.000 0.039 0.023  0.002 0.000 0.014 0.047  0.008
## 3 0.192     0.716 0.000 0.000 0.002 0.194  0.998 0.000 1.004 0.000  0.362
## 4 1.026     0.000 0.045 0.019 0.097 0.000  0.000 0.006 0.104 0.078  0.000
## 5 0.000     0.000 0.067 0.067 0.067 0.022  0.067 0.090 0.079 0.034  0.034
## 6 0.006     0.000 0.028 1.090 0.119 0.023  0.000 0.028 0.000 0.113  0.000
```

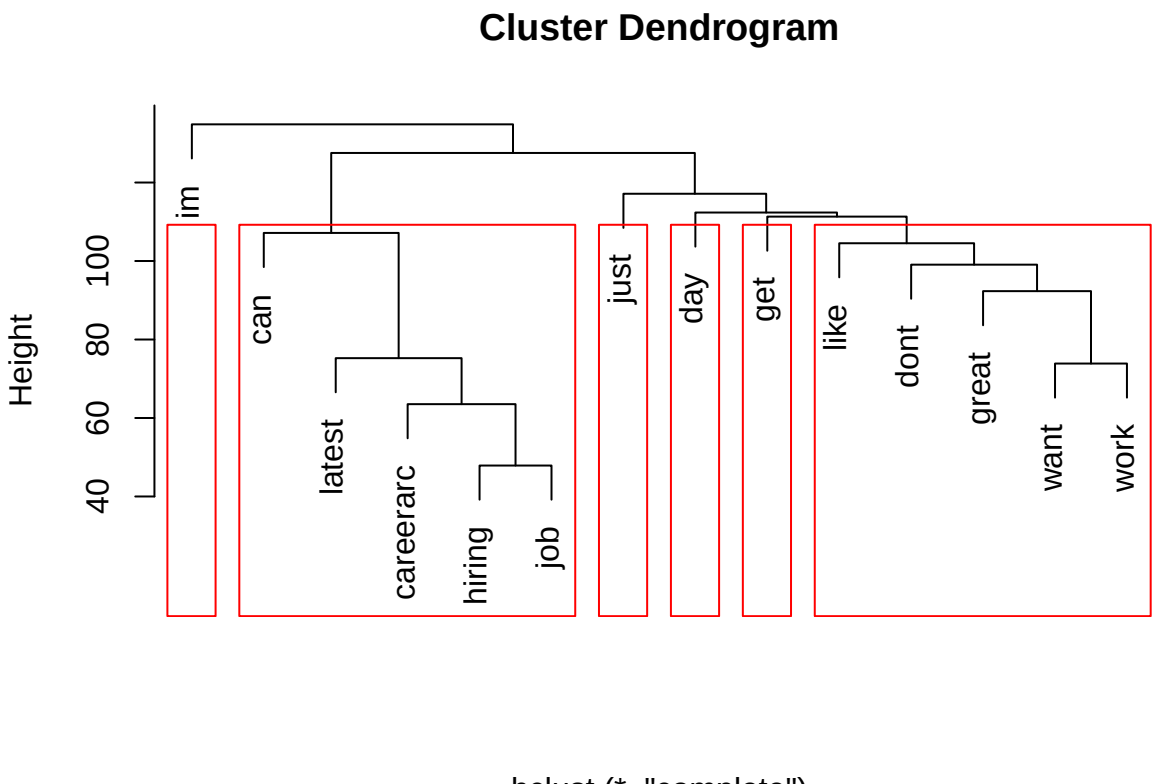
```
##    like  want  work
## 1 0.083 0.017 0.000
## 2 0.038 0.016 0.000
## 3 0.000 0.211 0.290
## 4 0.052 0.006 0.013
## 5 0.045 0.124 1.034
## 6 0.158 0.079 0.006
```

```
for(i in 1:k){
  cat(paste("cluster ", i, ": ", sep = ""))
  s <- sort(kmeansResult$centers[i,],decreasing = T)
  cat(names(s)[1:5],"\n")
  #print the tweet of every cluster
}
```

```
## cluster 1: im just like get day
## cluster 2: just day get like great
## cluster 3: job hiring careerarc latest work
## cluster 4: can job get just like
## cluster 5: work want im job day
## cluster 6: dont like get just want
```

Then I use hierarchical clustering to see if we could have the same results. Hierarchical clustering is an agglomerative algorithm that each observation starts i its own cluster and pairs of clusters are merged as one moves up the hierarchy.

```
# cluster terms
distMatrix <- dist(scale(m2))
fit <- hclust(distMatrix, method = "complete")
# show cluster dendrogram
p <- plot(fit, xlab="")
p <- rect.hclust(fit, k=6)
```



As is shown in the cluster Dendrogram, 'can', 'latest', 'careerarc', 'hiring', 'job' are in one cluster, which really makes sense that they are all related to jobs. But other clusters are dfficult to interpret and the words like 'dont', 'like' should de deleted in the early steps. So we need look back to the text data cleaning and find out how we can make sure every word in my output is meaningful.

6 Conclusion

In this Twitter data analysis project, (how to get the data and clean the data). Then I analyze the number of tweets and the density of tweets that are posted in different states and cities, making some plots to compare in different situations. I also use t test and permutation test to check the whether MA and IL have the same density mean in one day. Next, I find the relations among followers count, liseted count and friends count which could represent the popularity in Twitter. Finally, I plot the most frequent words with their wordcloud and conduct clustering analysis on the text data.

7 References

1. Text Mining in Twitter with R, https://www.youtube.com/watch?v=B0ySyj_0shc
2. Jeff Gentry, Twitter client for R, http://geoffjentry.hexdump.org/twitteR.pdf
3. David Kahle, Hadley Wickham, ggmap: Spatial Visualization with ggplot2, http://stat405.had.co.nz/ggmap.pdf