Where in the World is Taylor Swift?

Jeffrey Uslan

September 25, 2015

Synopsis

The goal of this analysis is to identify a single twitter user with a high volume of tweets which are also geotagged. The data has been acquired using the "streamR" package in the R statistical programming environment. This package allows are to interface with the twitter API. The processed tweets will be divided into 100 bins according to latitudinal and longitudinal coordinates. Probabilities will be assigned simply by percent of tweets submitted from that grid location.

Deliverables

Along with this report the following items will be delivered for review by Sotera staff.

- Analysis code
- Analysis data
- Output probabilites

Settings and Libraries

```
library(streamR)
library(Rfacebook)
library(smappR)
library(ROAuth)
library(ggplot2)
library(RCurl)
library(httr)
library(twitteR)
library(grid)
library(maps)
library(dplyr)
library(tidyr)
```

Identifying a User

The top 3 followed twitter users without political ramifications were considered (President Barack Obama is the 3 most followed user at the writing of this report). The users to be considered are Katy Perry, Justin Bieber, and Taylor Sqift.

```
potential_users=c("katyperry","justinbieber","taylorswift13")
user_ids=c("21447363","27260086","17919972")
potential_user.df=data.frame(potential_users,user_ids)
```

Before commiting computational resources a sampling of each potential subject is taken. This takes the form of minutes of access with the twitter API. For development purposes it is critically important that the owner of the application is a follower of any queried user.

```
for (i in 1:nrow(potential_user.df)){
  user=as.character(potential_user.df$potential_users[i])
  user_id=as.character(potential_user.df$user_ids[i])

  tweets2.df=data.frame()

  minutes=5
  for (i in 1:minutes){
    try(file.remove("tweets2.json"))
    filterStream("tweets2.json", timeout = 60, follow=c(user_id),
                 oauth = my_oauth)
    tmp<-try(parseTweets("tweets2.json", verbose  = FALSE))
    if (class(tmp) != "try-error") tweets2.df <- rbind(tweets2.df,tmp)
    if (sum(duplicated(tweets2.df))>0) tweets2.df=tweets2.df[!duplicated(tweets2.df),]
  }
  save(tweets2.df, file=paste0("./",user,"2.rda"))
}
```

Below we assess the validity of the potential subjects.

```
tweet_tot_n=NULL
tweet_geo_n=NULL
tweet_geo_lat_sd=NULL
tweet_geo_lon_sd=NULL
for (i in 1:nrow(potential_user.df)){
  user=as.character(potential_user.df$potential_users[i])
  user_id=as.character(potential_user.df$user_ids[i])
  load(paste0("./",user,"2.rda"))

  tweet_tot_n=c(tweet_tot_n,nrow(tweets2.df))
  tweet_geo_n=c(tweet_geo_n,sum(!is.na(tweets2.df$place_lat)))
  tweet_geo_lat_sd=c(tweet_geo_lat_sd,round(sd(tweets2.df$place_lat,na.rm=TRUE),2))
  tweet_geo_lon_sd=c(tweet_geo_lon_sd,round(sd(tweets2.df$place_lon,na.rm=TRUE),2))

}
potential_user.df=data.frame(potential_user.df,
                             Total_sample=tweet_tot_n,
                             Geo_tagged_count=tweet_geo_n,
                             Geo_lat_sd=tweet_geo_lat_sd,
                             Geo_lon_sd=tweet_geo_lon_sd)

print(potential_user.df[,c(1,3:6)])
```

```
##   potential_users Total_sample Geo_tagged_count Geo_lat_sd Geo_lon_sd
## 1       katyperry           17                2       1.28       1.90
## 2    justinbieber          289                1         NA         NA
## 3   taylorswift13           37                2      69.85      25.18
```

From this table we decide to proceed with Taylor Swift as the subject of analysis. She has the same amount of geotagged tweets as Katy Perry and the greatest geographic variability.

Gathering Data

Now that Taylor Swift has been identified as the ideal analysis subject we can dedicate more resources to gathering her data. This will take the form of three hours of wall clock time to gather data. The twitter API is highly sensitive to unstable bandwidth so it is recommonded that short queries be submitted and saved perioudically. The code below executes 180 minute queries.

```
user="taylorswift13"
user_id="17919972"

tweets3.df=data.frame()
minutes=180
for (i in 1:minutes){
  try(file.remove("tweets3.json"))
  filterStream("tweets3.json", timeout = 60, follow=c(user_id),
               oauth = my_oauth)
  tmp<-try(parseTweets("tweets3.json", verbose  = FALSE))
  if (class(tmp) != "try-error") tweets3.df <- rbind(tweets3.df,tmp)
  if (sum(duplicated(tweets3.df))>0) tweets3.df=tweets3.df[!duplicated(tweets3.df),]
}
```

Now we combine the new results with the intitial sample and remove duplicates.

```
save(tweets3.df, file=paste0("./",user,"3.rda"))

load(paste0("./",user,"2.rda"))

final_data=rbind(tweets2.df,tweets3.df)
if (sum(duplicated(final_data))>0) final_data=final_data[!duplicated(final_data),]
save(final_data, file=paste0("./final_data.rda"))
```

The Final Data

The final dataset has 1211 tweets, 45 of which are geotagged. All other tweets are removed for the analysis.

```
final_data=final_data[!is.na(final_data$place_lat),]
final_data=final_data[!is.na(final_data$place_lon),]
```

Summaries of the final data.

```
final_data=tbl_df(final_data)
final_data %>% summarise(Min_Longitude=min(place_lat),Max_Longitude=max(place_lat),
                         SD_Longitude=sd(place_lat))
```

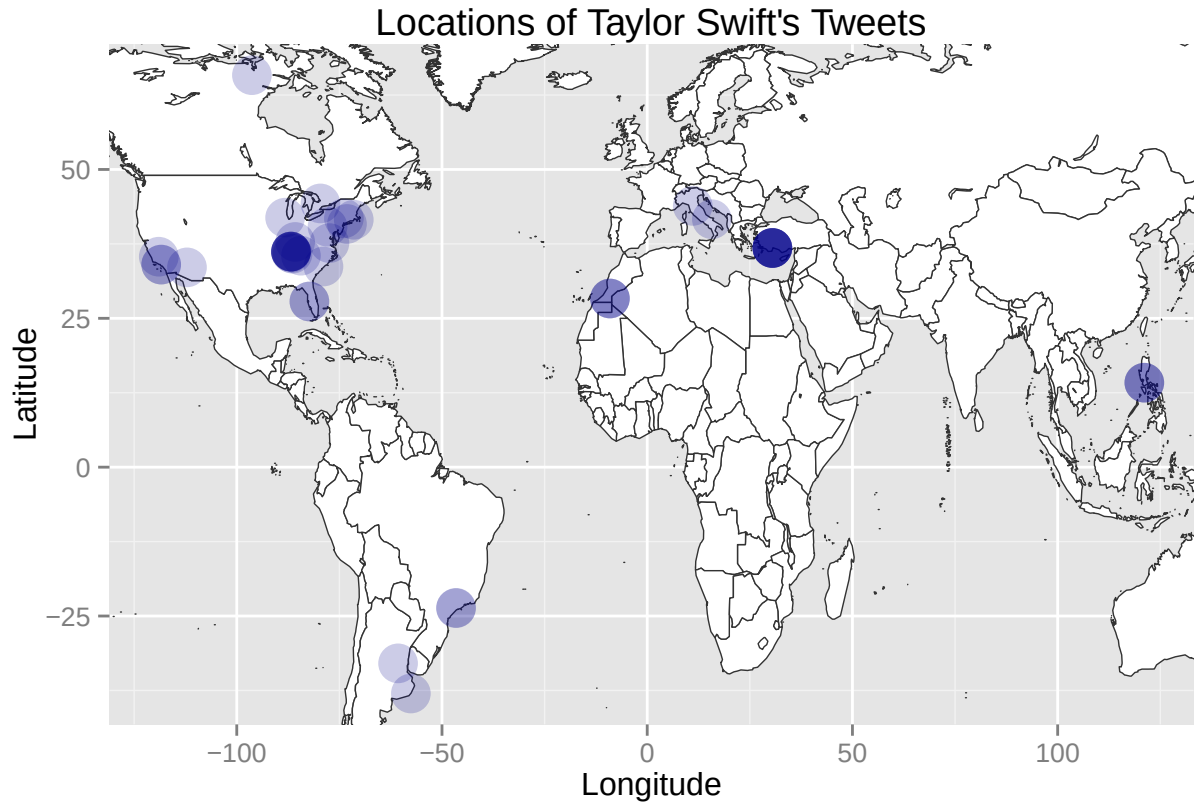
```
## Source: local data frame [1 x 3]
##
##   Min_Longitude Max_Longitude SD_Longitude
##           (dbl)         (dbl)        (dbl)
## 1     -38.01159      65.83811     20.48968
```

```
final_data %>% summarise(Min_Longitude=min(place_lon),Max_Longitude=max(place_lon),
                         SD_Longitude=sd(place_lon))
```

```
## Source: local data frame [1 x 3]
##
##   Min_Longitude Max_Longitude SD_Longitude
##           (dbl)         (dbl)        (dbl)
## 1     -118.9998      121.1089     65.80528
```

The plot below shows the origin location of the 45 tweets in this study

```
map.data <- map_data(map="world")
points <- data.frame(x = as.numeric(final_data$place_lon), y = as.numeric(final_data$place_lat))
ggplot(map.data) +
  geom_map(aes(map_id = region), map = map.data, fill = "white",color = "grey20", size = 0.25) +
  geom_point(data = points,aes(x = x, y = y), size = 7, alpha = 1/5, color = "darkblue") +
  ggtitle("Locations of Taylor Swift's Tweets")+xlab("Longitude")+ylab("Latitude")
```



Dividing the grid

Next we divide the observed geotagged data into 100 evenly sized grids.

```
grids_x=10
lon_min=floor(min(final_data$place_lon))
lon_max=ceiling(max(final_data$place_lon))
grid_lon=seq(lon_min,lon_max,((lon_max-lon_min)/grids_x))

lat_min=floor(min(final_data$place_lat))
lat_max=ceiling(max(final_data$place_lat))
grid_lat=seq(lat_min,lat_max,((lat_max-lat_min)/grids_x))

final_data$lon_grid=NA
final_data$lat_grid=NA
for (i in 2:(grids_x+1)){
  lon_ind=which(final_data$place_lon<=grid_lon[i] & final_data$place_lon>=grid_lon[i-1])
  final_data$lon_grid[lon_ind]=(i-1)
  lat_ind=which(final_data$place_lat<=grid_lat[i] & final_data$place_lat>=grid_lat[i-1])
  final_data$lat_grid[lat_ind]=(i-1)
}

final_data$grid=paste(final_data$lat_grid,final_data$lon_grid)
```

Grid Distributions

The following table shows the distibution of paired grid locations.

```
final_data %>% group_by(grid) %>% summarise(Count=n(),
                                            Probability=round(n()/nrow(final_data),4),
                                            Longitude_Minimum=min(place_lon),
                                            Longitude_Maximum=max(place_lon),
                                            Latitude_Minimum=min(place_lat),
                                            Latitude_Maximum=max(place_lat)) %>%
  arrange(desc(Count))
```

```
## Source: local data frame [11 x 7]
##
##     grid Count Probability Longitude_Minimum Longitude_Maximum
##    (chr) (int)       (dbl)             (dbl)             (dbl)
## 1    8 2    17      0.3778        -88.218062        -71.497913
## 2    8 7     8      0.1778         30.488127         30.488127
## 3   6 10     3      0.0667        121.108852        121.108852
## 4    7 1     3      0.0667       -118.411907       -112.124677
## 5    7 2     3      0.0667        -82.306722        -78.878866
## 6    7 5     3      0.0667         -9.082743         -9.082743
## 7    1 3     2      0.0444        -60.697406        -57.581430
## 8    2 4     2      0.0444        -46.595546        -46.564870
## 9    8 6     2      0.0444         11.091270         15.755051
## 10  10 1     1      0.0222        -96.307307        -96.307307
## 11   8 1     1      0.0222       -118.999827       -118.999827
## Variables not shown: Latitude_Minimum (dbl), Latitude_Maximum (dbl)
```

The following table shows the distributions longitudinal ordinates.

```
final_data %>% group_by(lon_grid) %>% summarise(Count=n(),
                                                Probability=round(n()/nrow(final_data),4),
                                                Longitude_Minimum=min(place_lon),
                                                Longitude_Maximum=max(place_lon)) %>%
  arrange(desc(Count))
```

```
## Source: local data frame [8 x 5]
##
##   lon_grid Count Probability Longitude_Minimum Longitude_Maximum
##      (dbl) (int)       (dbl)             (dbl)             (dbl)
## 1        2    20      0.4444        -88.218062        -71.497913
## 2        7     8      0.1778         30.488127         30.488127
## 3        1     5      0.1111       -118.999827        -96.307307
## 4        5     3      0.0667         -9.082743         -9.082743
## 5       10     3      0.0667        121.108852        121.108852
## 6        3     2      0.0444        -60.697406        -57.581430
## 7        4     2      0.0444        -46.595546        -46.564870
## 8        6     2      0.0444         11.091270         15.755051
```

The following table shows the distributions of latitudinal ordinates.

```
final_data %>% group_by(lat_grid) %>% summarise(Count=n(),
                                                Probability=round(n()/nrow(final_data),4),
                                                Latitude_Minimum=min(place_lat),
                                                Latitude_Maximum=max(place_lat)) %>%
  arrange(desc(Count))
```

```
## Source: local data frame [6 x 5]
##
##   lat_grid Count Probability Latitude_Minimum Latitude_Maximum
##      (dbl) (int)       (dbl)            (dbl)            (dbl)
## 1        8    28      0.6222         35.34690         44.29355
## 2        7     9      0.2000         27.82737         34.02079
## 3        6     3      0.0667         14.18862         14.18862
## 4        1     2      0.0444        -38.01159        -32.95162
## 5        2     2      0.0444        -23.68280        -23.62480
## 6       10     1      0.0222         65.83811         65.83811
```