

My First R Markdown File

Ryan Wesslen

September 28, 2017

Loading Data

Let's get started. First, we can write text to describe what we're doing next.

For example, let's load `tidyverse`.

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages -----

## filter(): dplyr, stats
## lag():    dplyr, stats
```

Next, let's use the `read_csv()` function load our dataset.

```
tweets <- read_csv("../data/CharlotteTweets20Sample.csv")
```

```
## Parsed with column specification:
## cols(
##   body = col_character(),
##   postedTime = col_datetime(format = ""),
##   actor.id = col_double(),
##   displayName = col_character(),
##   actor.postedTime = col_datetime(format = ""),
##   summary = col_character(),
##   friendsCount = col_integer(),
##   followersCount = col_integer(),
##   statusesCount = col_integer(),
##   actor.location.displayName = col_character(),
##   generator.displayName = col_character(),
##   geo.type = col_character(),
##   point_long = col_double(),
##   point_lat = col_double(),
##   urls.0.expanded_url = col_character(),
##   klout_score = col_integer(),
##   hashtags = col_character(),
##   user_mention_screen_names = col_character()
## )
```

Recall – what's going on with the path?

Why have we not needed to set our working directory?

View Data

We can use the `###` parameter to specify a new section. For example, in this section, let's consider looking at the first five tweets.

```
head(tweets$body, n = 3)
```

```
## [1] "Treon to WR is a really good move by Mac, very familiar with playbook. Very good runner in open  
## [2] "primus #vsco #vscocam #primus #primussucks #charlotte #bojanglescoliseum #livemusic... https://  
## [3] "WOAH!!!!!!"
```

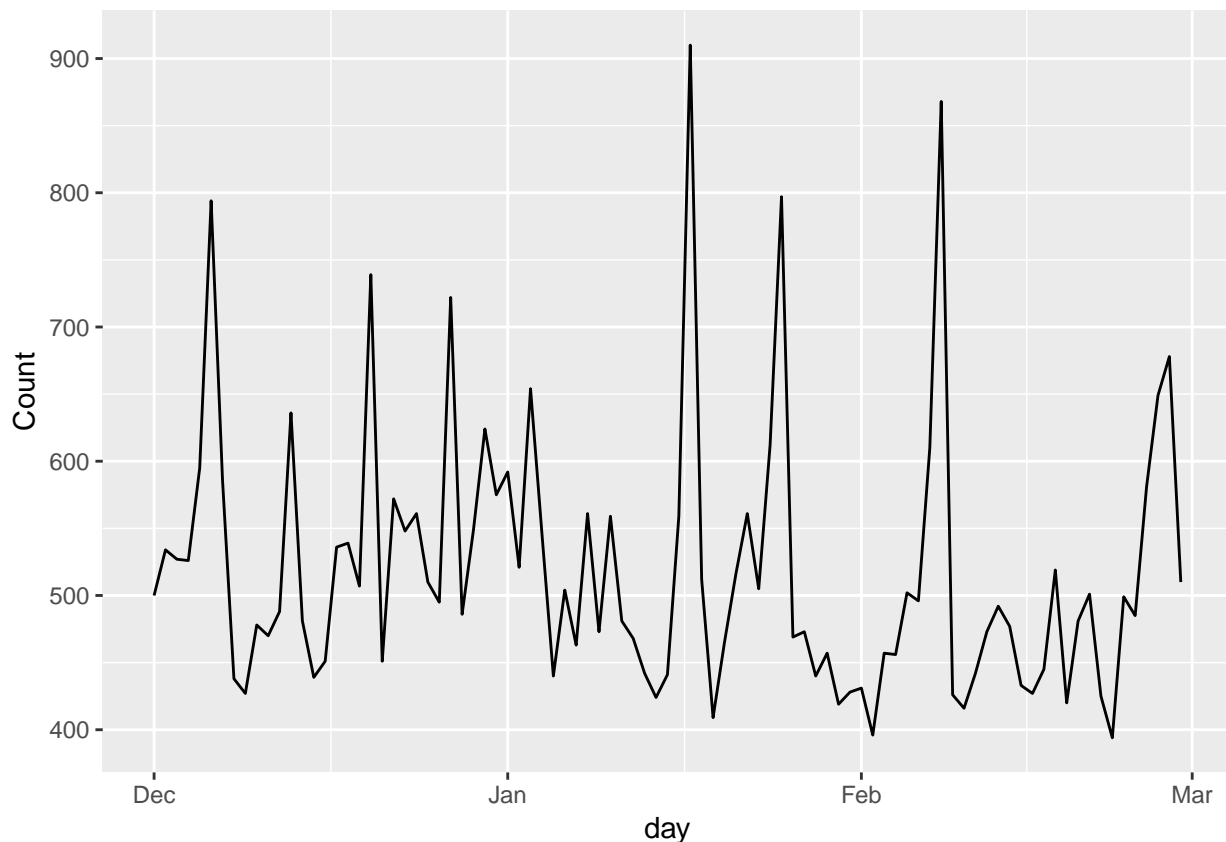
Getting a time series of the Tweets

Next, let's convert the date to allow us to plot our data.

```
# converts YYYY-MM-DD HH:MM:SS to string of YYYY-MM-DD  
tweets$day <- as.Date(tweets$postedTime)  
  
# recall from dplyr  
dayCount <- tweets %>%  
  group_by(day) %>%  
  summarise(Count=n())
```

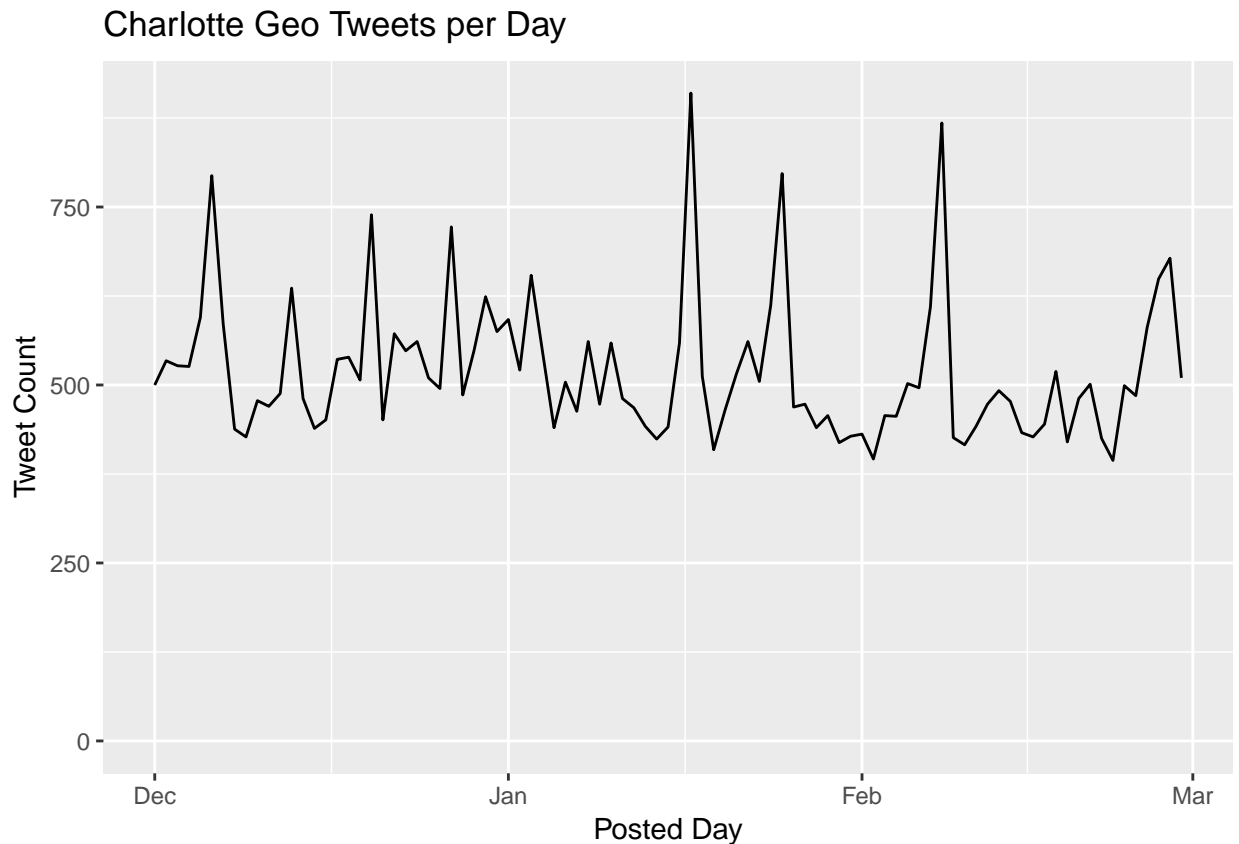
Next, we're going to use `ggplot2` to plot our time series. We'll formally introduce `ggplot` more next week.

```
ggplot(dayCount, aes(x = day, y = Count)) +  
  geom_line()
```



Not bad. We can add a few parameters to this plot to make it pretty.

```
ggplot(dayCount, aes(x = day, y = Count)) +
  geom_line() +
  labs(x = "Posted Day", y = "Tweet Count", title = "Charlotte Geo Tweets per Day") +
  expand_limits(y = 0) # set y axis to start at zero
```



What the heck are those spikes?

Hashtag & Mentions

One way we can check is to look at the most prominent hashtags and mentions.

To do this, we can run a pre-specified function (regular expression) that will keep only hashtags or handles from the body of our tweet.

```
getCommonHashtags <- function(text, n=20){
  hashtags <- regmatches(text, gregexpr("#(\\d|\\w)+",text))
  hashtags <- unlist(hashtags)
  tab <- table(hashtags)
  return(head(sort(tab, dec=TRUE), n=n))
}

getCommonHandles <- function(text, n=20){
  handles <- regmatches(text, gregexpr('@([0-9_A-Za-z]+)',text, perl=TRUE))
  handles <- unlist(handles)
  tab <- table(handles)
  return(head(sort(tab, dec=TRUE), n=n))
}
```

For example, now we can run:

```
getCommonHashtags(tweets$body)
```

```
## hashtags
## #KeepPounding #Charlotte #NC #realestate #charlotte
##          487          458          443          429          256
## #keeppounding #traffic #trndnl #photo #listing
##          234          212          205          185          159
##          #clt #Repost #realtor #CIAA #SB50
##          154          144          133          123          109
## #Panthers #CIAA2016 #Concord #CLT #panthers
##          95          89          88          86          86
```

or ...

```
getCommonHandles(tweets$body)
```

```
## handles
## @cltairport @Panthers @midnight @panthers
##          218          212          180          130
## @EthanDolan @ness @hornets @GraysonDolan
##          117          80          78          77
## @SportsCenter @F3theFort @realDonaldTrump @nodabrewing
##          69          64          64          63
## @CampersHaven @marcuslemonis @F3Isotope @marielawtf
##          58          52          48          43
## @oakley @LifeTimeFitness @ChickenNGreens @wcnc
##          39          38          36          36
```

Ah... maybe they're Carolina Panther related!

Searching for Panther Tweets

Let's use a different regular expression and save only the tweets that include three keywords...

```
panthers <- c("#keeppounding", "#panthers", "@panthers")

# find only the Tweets that contain words in the first list
hit <- grepl(paste(panthers, collapse = "|"), tolower(tweets$body))

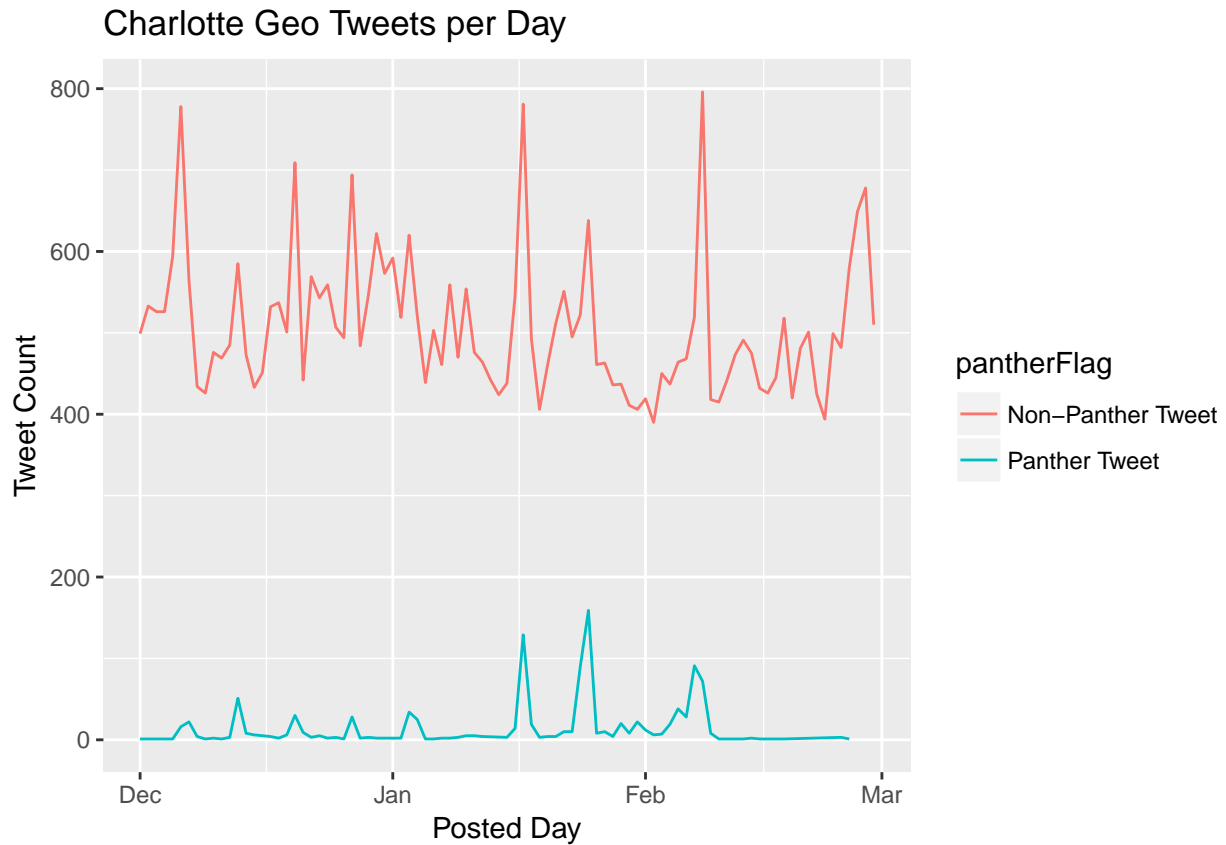
# create a column to separate panther tweets and non-Panther tweets
tweets$pantherFlag <- ifelse(hit, "Panther Tweet", "Non-Panther Tweet")
```

With our new flag, let's rerun our time series but differentiate between Panther and non-Panther tweets.

```
dayCount <- tweets %>%
  group_by(day, pantherFlag) %>%
  summarise(Count=n())
```

Next, we're going to use ggplot2 to plot our time series. We'll formally introduce ggplot more next week.

```
ggplot(dayCount, aes(x = day, y = Count, color = pantherFlag)) +
  geom_line() +
  labs(x = "Posted Day", y = "Tweet Count", title = "Charlotte Geo Tweets per Day") +
  expand_limits(y = 0) # set y axis to start at zero
```



This is a good start, but it still looks like we may have missed some Tweets.

Querying Twitter data is an incredibly hard problem (much harder than many researchers realize!).

For example, King, Lam and Roberts (2017) investigates this problem and finds that the choice of keywords can drastically affect results.

For a deeper analysis, see my Fall 2016 Twitter Workshop Day 1 Exercise.