An Introduction to the Twitter API and `rtweet`

Social Media and Web Analytics @ TiSEM

Lachlan Deer

Last updated: 11 April, 2021

Twitter: A quick intro

- A social media platform
- Users post short messages called tweets
- Users follow other users who's content they are interested in
  - User has followers - people who follow that user
  - *and* other users who follow their (i.e. the user's) account - called 'friends'
- Users can:
  - 'like' others tweets
  - reply to others tweets
  - re-broadcast an existing tweet to their followers (retweeting)
  - mention each other in a tweet `hey @lachlandeer, do you like #socialmediamarketing ?`
- There's a convention to use hashtags (`#socialmediamarketing`) to connect their tweet to a broader discussion on a topic

Advantages and Disadvantages of Twitter Data

Advantages:

- Twitter APIs are open and accessible
- Twitter is relatively research friendly compared to some other social media platforms
- Easy enough to find conversations due to hashtag norms
- Length of tweets is short, so analysis of tweet text is not too difficult

Disadvantages

- Historical search is limited to approx. last 5-7 days for a free account
- There's limits on the number of tweets we can get for free
- Free accounts only have access to a 1% sample of tweets, which may not be accurate
  - And indeed does have some biases
- The percentage of tweets with geographic tagging is small
  - Think 2-5% of all tweets.

Note that a "free" account refers to us not wanting to pay Twitter to get access to more data. It is possible to pay for historical access & for a larger sample of tweets (up to using 100%). But it does get expensive (trust me).

Twitter APIs

Twitter has two separate APIs:

- REST API
  - Allows you programmatic access to read and write Twitter data.

    - Can search the recent history of tweets and look up specific users.
- Streaming API
    - Allows you to access the public data flowing through Twitter in real-time.
    - Requires your R session to be running continuously
        * Can capture a much larger sample of tweets and avoiding rate limits in REST API.

So What is is an API Anyway?

- API: Application Programming Interface
- Uhhhhh - what does that actually mean?
    - Think of it as a piece of intermediate software that allows to applications to talk to each other
    - In our case: our R session and Twitter's database

R Packages for accessing Twitter APIs

There are several packages for accessing an searching data on Twitter. Popular packages in R include:

- `rtweet`
    - A relatively recent addition to the R package universe
    - Allows you to access both the REST and streaming APIs.
    - **This will be our package of choice**
- `twitteR`
    - Has been most popular package for R
    - Only allows you to access the REST API.
    - Not actively updated
- `streamR`
    - Older (last updated in 2018)
    - More complicated
    - Allows you to query the Streaming API from R

Setting up R to Access Tweets

To get up and running you will need:

- A Twitter account
    - For verification purposes by Twitter
    - Set one up if you don't have one … you can provide very little information and still have an account.
    - Essentially you need a username and a password
- Pop-up blockers disabled
    - Only for the first time you use the `rtweet` package
- R installed and running
    - I'll assume you are using RStudio
- `rtweet` installed
    - Via `install.packages("rtweet")` if you have not done it already
        * If you have problems verifying your account, try also installing `httpuv` (I think you will have thus installed already though)

Note: For students in the course prefer to use Server versions of RStudio. RStudio Server will not be able to follow this authentication process since you cannot run an interactive session of R that connects to your web browser.

Searching Tweets: The Basics

- First, load the package:

```r
library(rtweet)
```

- The function `search_tweets()` returns Twitter data matching a search query
  - Only using data from the last 5-7 days
  - Only using a 1% sample of all tweets posted
  - There's a max of 18,000 tweets returned per search
    - (we can get around this if needed)
- Let's try it out ... lets find tweets that use the hashtag #ajax (the football club, not the cleaning spray – we hope!)

```r
tweets_ajax <- search_tweets('#ajax')
```

- By default, we will get 100 tweets, that are a mix of original tweets and retweets.

Authentication on your first search

If you have not used `rtweet` before, here's where the authentication part becomes important. You'll have a browser window pop up, and you will need to authenticate yourself using `OAuth`. Provide your username, password and click yes or accept etc to anything you are asked. Once you've done that - the search will run and data should be returned.

Note: You won't have to authenticate yourself again - your credentials will be saved on your computer for next time.

Components of Twitter Data

- Now we have the data stored as `tweets_ajax`.

- We have 100 observations:

```r
nrow(tweets_ajax)
```

```
[1] 100
```

- Each observation has 90 columns of data:

```r
ncol(tweets_ajax)
```

```
[1] 90
```

- Can get some detail about the column names:

```r
names(tweets_ajax)
```

```
 [1] "user_id"                 "status_id"
 [3] "created_at"              "screen_name"
 [5] "text"                    "source"
 [7] "display_text_width"      "reply_to_status_id"
 [9] "reply_to_user_id"        "reply_to_screen_name"
[11] "is_quote"                "is_retweet"
[13] "favorite_count"          "retweet_count"
[15] "quote_count"             "reply_count"
[17] "hashtags"                "symbols"
[19] "urls_url"                "urls_t.co"
[21] "urls_expanded_url"       "media_url"
[23] "media_t.co"              "media_expanded_url"
[25] "media_type"              "ext_media_url"
[27] "ext_media_t.co"          "ext_media_expanded_url"
[29] "ext_media_type"          "mentions_user_id"
```

```
[31] "mentions_screen_name"    "lang"
[33] "quoted_status_id"        "quoted_text"
[35] "quoted_created_at"       "quoted_source"
[37] "quoted_favorite_count"   "quoted_retweet_count"
[39] "quoted_user_id"          "quoted_screen_name"
[41] "quoted_name"             "quoted_followers_count"
[43] "quoted_friends_count"    "quoted_statuses_count"
[45] "quoted_location"         "quoted_description"
[47] "quoted_verified"         "retweet_status_id"
[49] "retweet_text"            "retweet_created_at"
[51] "retweet_source"          "retweet_favorite_count"
[53] "retweet_retweet_count"   "retweet_user_id"
[55] "retweet_screen_name"     "retweet_name"
[57] "retweet_followers_count" "retweet_friends_count"
[59] "retweet_statuses_count"  "retweet_location"
[61] "retweet_description"     "retweet_verified"
[63] "place_url"               "place_name"
[65] "place_full_name"         "place_type"
[67] "country"                 "country_code"
[69] "geo_coords"              "coords_coords"
[71] "bbox_coords"             "status_url"
[73] "name"                    "location"
[75] "description"
 [ reached getOption("max.print") -- omitted 15 entries ]
```

I haven't printed the output here because it is quite long.

Aside: Twitter data in it's native format

- Twitter data doesn't actually come in this rows and columns format.
- The `rtweet` package has formatted it for you
- Twitter data is stored as, and thus returned by default as JSON data
  - JSON: JavaScript Object Notation
    * Think of it as each tweet is stored as a list of information. Sometimes an element of the list is itself a list
  - JSON data is a set of *attributes* and *values*.
- Tweets can have over 150 metadata components.
- Here's a stylized example:
- So the `rtweet` package has taken the attributes and converted them to column names, with the entries in each column being the values.

Columns We May be Interested In

We might be interested in the following columns:

- `screen_name`: the user name of the account who wrote the tweet
- `created_at`: date and time tweet was posted
- `text`: content of the tweet
- `retweet_count`: number of times tweet has been retweeted
- `is_quote`: is tweet a quote of another
- `is_retweet`: is tweet a retweet
- `mentions_screen_name`: names of other users mentioned in tweet

- `retweet_screen_name`: name of the user who was retweeted
- `reply_to_user_screen_name`: If tweet was a reply, who was the reply to
- `hashtags`: any hashtags used in the tweet
- `friends_count`: number of people a user follows
- `follower_count`: number of other users this user is following
- `language`: language of the tweet

You should spend some more time exploring what is in each of the 90 columns

Searching Tweets: Some Additional Parameters

We can do much more with the `search_tweets()` function:

- Get a larger volume of tweets, say 3,000:

```
tweets_ajax <- search_tweets('#ajax',
                             n = 3000)
```

- If we want a large number, we'll likely hit the 18,000 tweets per 15 min allowance, and we'd need to try again. Can do that 'automatically':

```
tweets_ajax <- search_tweets('#ajax',
                             n = 50000,
                             retryonratelimit = TRUE)
```

This might take a while because if there are 50,000 tweets using #ajax over the last 5-7 days - we have to wait for 15 mins twice during the search. (Can you explain why?)

Searching Tweets: Longer Queries

- What if we want tweets mentioning multiple terms?
  - Suppose interest in tweets about 'transfers' by '#ajax'. Write as follows:

```
tweets_ajax <- search_tweets('#ajax transfer',
                             n = 3000
                             )
```

`white space is treated as 'AND' by default`

- Can also think of `term1` or `term2`

```
tweets_af <- search_tweets('#ajax OR #feyenoord',
                            n = 3000,
                            retryonratelimit = TRUE)
```

- Want an exact phrase:

```
tweets_data <- search_tweets('"data science"',
                             n = 3000
                             )
```

You can mix and match as you need.

Searching Tweets: Adding Filters

What if we want to add restrictions our search. We can do that too. I'll present different examples. You can mix and match these together as needed.

- Don't include retweets

```
tweets_ajax <- search_tweets('#ajax',
                             n = 3000,
                             include_rts = FALSE
                             )
```

Or, equivalently:

```
tweets_ajax <- search_tweets('#ajax
                             -filter:retweets',
                             n = 3000
                             )
```

- Don't include quote tweets:

```
tweets_ajax <- search_tweets('#ajax
                                -filter:quote',
                             n = 3000
                             )
```

- Don't include replies:

```
tweets_ajax <- search_tweets('#ajax
                                -filter:retweet
                                -filter:quote
                                -filter:replies',
                             n = 3000
                             )
```

- Don't include retweets, quotes or replies:

```
tweets_ajax <- search_tweets('#ajax
                                -filter:replies',
                             n = 3000
                             )
```

- Can also filter by language.
  - Need a two letter language code to do so.
  - Examples:
    * English: en
    * Dutch: nl
    * Farsi: fa
    * French: fr
    * German: de
    * Indonesian: id
    * Italian: it
    * Japanese: ja
    * Portuguese: pt
    * Spanish: es
    * Mandarin - Simplified: zh
    * Mandarin - Traditional: zh-tw

```
tweets_ajax <- search_tweets('#ajax',
                             n = 3000,
                             lang = "nl"
                             )
```

Searching Tweets: Adding Filters by Geography

- Can you filter by geography?
  - Can - but you need geographic coordinates
  - `rtweet` provides the function `lookup_coords()`
  - But you need a Google Maps Billing Code…
    - In the demonstration video and GitHub repo I provide the geocode for the Netherlands
    - If you want another for your final project, contact me on Slack
- Suppose you want to restrict to the Netherlands:

```
nl_geo <- lookup_coords('Netherlands')
tweets_ajax <- search_tweets('#ajax',
                             n = 3000,
                             geocode = nl_geo)
```

- Suppose you wanted to tweets within a 50km radius of Rotterdam:

```
rdam_geo <- '51.9244,4.4777,50km'
tweets_ajax <- search_tweets('#ajax',
                             n = 3000,
                             geocode = rdam_geo)
```

Getting User Timelines

- Suppose you want recent tweets by a certain user.
  - Need their screen_name or their user_id

```
tweets_by_ajax <- get_timeline('afcajax')
```

equivalent to:

```
# use Ajax's user_id rather than screen name
tweets_by_ajax <- get_timeline('153000691')
```

- Can get up to 3,200 tweets per user:
  - That's the limit Twitter gives us

```
tweets_by_ajax <- get_timeline('afcajax',
                               n = 3200)
```

- Multiple users:

```
football_accounts <- c("afcajax", "feyenoord", "psv")

tweets_by_ajax <- get_timeline(football_accounts,
                               n = 3200)
```

Getting Friends and Followers:

- Get friends of a user (an account a user follows)

```
ajax_friends <- get_friends('afcajax')
```

- Do it for many users:

```
# use the list of football accounts defined previously
football_friends <- get_friends(football_accounts)
```

- Get `user_id`'s of accounts following Ajax:

```r
ajax_followers <- get_followers('afcajax')
```

Returns 5000 by default. If you want them all, this would work:

```r
ajax_followers <- get_followers('afcajax',
                                n = 1500000,
                                retryonratelimit = TRUE)
```

but you would need a stable internet connection and some quite time as the rate limits will make this slow.

Stream Tweets

- Can get a small "*random*" sample of all publicly available tweets
  - Default is to stream for 30 seconds

```r
streamed <- stream_tweets("")
```

- Can go longer, for example 3 mins:

```r
streamed <- stream_tweets("",
                          timeout = 180)
```

- Can filter what comes back
  - If I wanted tweets mentioning Ajax, feyenoord or PSV:

```r
streamed <- stream_tweets("ajax,feyenoord,psv",
                          timeout = 180)
```

- And add geo-restrictions:

```r
streamed <- stream_tweets("ajax,feyenoord,psv",
                          timeout = 180,
                          geocode = nl_geo
                          )
```

More Information?

Want to know *even* more?

- These slides are pretty good
  - They're by one of `rtweet`'s developers!
- Remark: I am neither an Ajax fan or a football fan - the example fit my purpose though...

Acknowledgements

The content in these notes is in part derived from the following sources:

- Practice Getting Data from the Twitter API by Benjamin Soltoff
- Analyzing Twitter Data by Somya Vivek
- Filtering Tweets by Vivek Vijayaraghavan