Twitter client for R

Jeff Gentry

March 18, 2014

1 Introduction

Twitter is a popular service that allows users to broadcast short messages ('*tweets*') for others to read. Over the years this has become a valuable tool not just for standard social media purposes but also for data mining experiments such as sentiment analysis. The *twitteR* package is intended to provide access to the Twitter API within R, allowing users to grab interesting subsets of Twitter data for their analyses.

This document is not intended to be exhaustive nor comprehensive but rather a brief introduction to some of the more common bits of functionality and some basic examples of how they can be used. In the last section I've included a variety of links to people using *twitteR* to solve real world problems.

2 Some Initial Notes

2.1 Support mailing list

While this package doesn't generate a huge volume of emails to me, I have found that the same questions tends to come up repeatedly (often when something has been broken!). I also field requests for advice on practical application of this package which is an area that I'm far from expert at. I've set up a mailing list to better manage emails from users as this way, with the idea being that there'll now be a searchable archive and perhaps other users might be able to chime in. The URL for this mailing list is `http://lists.hexdump.org/listinfo.cgi/twitter-users-hexdump.org`

3 Authentication with OAuth

As of March 2013 OAuth authentication is *required* for all Twitter transactions. You will need to follow these instructions to continue.

OAuth is an authentication mechanism gaining popularity which allows applications to provide client functionality to a web service without granting an end user's credentials to the client itself. This causes a few wrinkles for cases

like ours, where we're accessing Twitter programatically. *twitteR* uses the *httr* package under the hood to manage this.

The first step is to create a Twitter application for yourself. Go to `https://twitter.com/apps/new` and log in. After filling in the basic info, go to the "Settings" tab and select "Read, Write and Access direct messages". Make sure to click on the save button after doing this. In the "Details" tab, take note of your consumer key and consumer secret.

In your R session, you'll want to do the following with the appropriate values from the web page:

```
>    setup_twitter_oauth("API key", "API secret")
```

This will authenticate via *httr*, I recommend looking at that package's `Token` man page for more information regarding how to manage the authentication and caching processes.

If you are in a headless environment or otherwise don't want to deal with the browser based authentication dance, you can get your access token and secret from your apps webpage and call `setup_twitter_oauth` a bit differently:

```
>   setup_twitter_oauth("API key", "API secret", "Access token", "Access secret")
```

4 Getting Started

This document is intended to demonstrate basic techniques rather than an exhaustive tour of the functionality. For more in depth examples I recommend exploring the mailing list or StackOverflow. I've also included some links to examples of `twitteR` being used in the real word at the end.

```
> library(twitteR)

> setup_twitter_oauth("API key", "API secret")
```

5 Exploring Twitter

5.1 Searching Twitter

The `searchTwitter` function can be used to search for tweets that match a desired term. Example searches are such things as hashtags, basic boolean logic such as AND and OR. It is worth looking at `https://dev.twitter.com/docs/using-search` for an example of what can and can not be done here. The `n` argument can be used to specify the number of tweets to return, defaulting to 25. Note that while `searchTwitter` will wrap an arbitrary number of actual search calls to provide the number of tweets requested, the Twitter API has limitations on just how much it will actually return. In general you can only go back a handful of days worth of tweets.

```
>    tweets <- searchTwitter('#rstats', n=50)
>    head(tweets)

[[1]]
[1] "DaneGWendell: RT @jent103: Sometimes I make #rstats do exactly what I want, and at thos

[[2]]
[1] "diervilla: fantastic resource for #Rstats http://t.co/5oPgFNkVq0"

[[3]]
[1] "onofreliscano: RT @Rbloggers: Secrets of Teaching R: \n\n(This article was first publis

[[4]]
[1] "swcarpentry: RT @rstats_bot: RT @jamesaeddy After working in @MATLAB for years, @swcarp

[[5]]
[1] "denisohora: RT @jent103: Sometimes I make #rstats do exactly what I want, and at those

[[6]]
[1] "TraineeGeek: RT @hadleywickham: are you a web expert interested in #rstats? we (@rstudi
```

There's a handy utility method, `strip_retweets` which attempts to do exactly what it describes. It takes a list of *status* objects (e.g. from `searchTwitter`) and by default will remove official API-based retweets from Twitter, i.e. cases where the retweet button was pressed instead of the manual *RT @soandso .....* method. However there are two arguments, `strip_manual` and `strip_mt`, corresponding to the manual retweets described above and modified retweets (MT). If either of these are `TRUE`, the appropriate type of tweets will also be removed, but leaving everything to the left of the RT/MT.

Note that this example may or may not do anything depending on the data available when this vignette was compiled.

```
>   head(strip_retweets(tweets, strip_manual=TRUE, strip_mt=TRUE))

[[1]]
[1] "diervilla: fantastic resource for #Rstats http://t.co/5oPgFNkVq0"

[[2]]
[1] "jamesaeddy: After working in @MATLAB for years, @swcarpentry is starting to make me a b

[[3]]
[1] "hadleywickham: are you a web expert interested in #rstats? we (@rstudioapp) are looking

[[4]]
[1] "ucfagls: When using knitr &amp;amp; writing a journal article tracked on github, should you

[[5]]
```

```
[1] "Rbloggers: Secrets of Teaching R: \n\n(This article was first published on   Revolution

[[6]]
[1] "GGorczynski: Great list - The Stack Overflow's R Top 5 âĂŞ http://t.co/YOKmNndGDO #Rsta
```

5.2 Looking at users

To take a closer look at a Twitter user (including yourself!), run the command `getUser`. This will only work correctly with users who have their profiles public, or if you're authenticated and granted access. You can also see things such as a user's followers, who they follow, retweets, and more. The `getUser` function returns a *user* object, which can then be polled for further information.

```
>    crantastic <- getUser('crantastic')
>    crantastic$getDescription()

[1] "I like some things, and I dislike everything else."

>    crantastic$getFollowersCount()

[1] 32

>    crantastic$getFriends(n=5)

$`221454258`
[1] "rafejudkins"

$`15515656`
[1] "ChloeBennet4"

$`393987775`
[1] "Lil_Henstridge"

$`19879313`
[1] "evilhag"

$`165819788`
[1] "TorontoHydro"

>    crantastic$getFavorites(n=5)

[[1]]
[1] "AnnaKendrick47: Guys I won't be able to see the fireworks from where I am today. Can so

[[2]]
[1] "SmashKarenC: Don't get me wrong, I think it's great that Ivy sleeps with her directors

[[3]]
```

```
[1] "EmWatson: Who here actually thinks I would do 50 Shades of Grey as a movie? Like really

[[4]]
[1] "AnnaKendrick47: I can't believe they cut the scene where I walk away from that explosio

[[5]]
[1] "kevwilliamson: Who will Elena end up with?  We're so not there yet -- but if you must k
```

5.3 Conversion to data.frames

There are times when it is convenient to display the object lists as an `data.frame` structure. To do this, every class has a reference method `toDataFrame` as well as a corresponding S4 method `as.data.frame` that works in the traditional sense. Converting a single object will typically not be particularly useful by itself but there is a convenience method to convert an entire list, `twListToDF` which takes a list of objects from a single *twitteR* class:

```
>    df <- twListToDF(tweets)
>    head(df)
```

```
1                                          RT @jent103: Sometimes I make #rstats do exactl
2                                                                                        f
3 RT @Rbloggers: Secrets of Teaching R: \n\n(This article was first published on   Revolutio
4   RT @rstats_bot: RT @jamesaeddy After working in @MATLAB for years, @swcarpentry is start
5                                          RT @jent103: Sometimes I make #rstats do exactl
6            RT @hadleywickham: are you a web expert interested in #rstats? we (@rstudioapp)
  favorited favoriteCount replyToSN             created truncated replyToSID
1     FALSE             0        NA 2014-03-18 23:53:59     FALSE         NA
2     FALSE             0        NA 2014-03-18 23:22:23     FALSE         NA
3     FALSE             0        NA 2014-03-18 23:11:42     FALSE         NA
4     FALSE             0        NA 2014-03-18 23:10:50     FALSE         NA
5     FALSE             0        NA 2014-03-18 23:06:49     FALSE         NA
6     FALSE             0        NA 2014-03-18 22:54:25     FALSE         NA
                  id replyToUID
1 446072051061649408         NA
2 446064098686812161         NA
3 446061410007941121         NA
4 446061192004771840         NA
5 446060180762296321         NA
6 446057062003646464         NA
                                                                                         s
1               <a href="http://tapbots.com/software/tweetbot/mac" rel="nofollow">Tweetbot
2                                      <a href="http://www.hootsuite.com" rel="nofollow">Ho
3
4 <a href="http://itunes.apple.com/us/app/twitter/id409789998?mt=12" rel="nofollow">Twitter
```

```
5                  <a href="http://twitter.com/download/android" rel="nofollow">Twitter for
6                                          <a href="http://www.echofon.com/" rel="nofollow">
     screenName retweetCount isRetweet retweeted longitude latitude
1  DaneGWendell            7      TRUE     FALSE        NA       NA
2     diervilla            0     FALSE     FALSE        NA       NA
3 onofreliscano            1      TRUE     FALSE        NA       NA
4   swcarpentry            1      TRUE     FALSE        NA       NA
5    denisohora            7      TRUE     FALSE        NA       NA
6   TraineeGeek            7      TRUE     FALSE        NA       NA
```

5.4 Database Persistence

A question that I'm often asked is how to retrieve data from the past, generally people are doing a study on some major event that has already happened (e.g. Arab Spring, an election, etc). Using the Twitter API this is impossible as you can only go back a small amount. However, if you have the ability to look ahead, it is easy to enable a prospective study by collecting data and automatically persisting it to a database. This will then allow you to load everything into a later R session, including using tools such as *dplyr*. There's a full writeup of this functionality at `http://geoffjentry.blogspot.com/2014/02/twitter-now-supports-database.html`.

Here's a brief example:

```
>   sql_lite_file = tempfile()
>   register_sqlite_backend(sql_lite_file)
>   store_tweets_db(tweets)

[1] TRUE

>   from_db = load_tweets_db()
>   head(from_db)

[[1]]
[1] "DaneGWendell: RT @jent103: Sometimes I make #rstats do exactly what I want, and at thos

[[2]]
[1] "diervilla: fantastic resource for #Rstats http://t.co/5oPgFNkVq0"

[[3]]
[1] "onofreliscano: RT @Rbloggers: Secrets of Teaching R: \n\n(This article was first publis

[[4]]
[1] "swcarpentry: RT @rstats_bot: RT @jamesaeddy After working in @MATLAB for years, @swcarp

[[5]]
[1] "denisohora: RT @jent103: Sometimes I make #rstats do exactly what I want, and at those
```

```
[[6]]
[1] "TraineeGeek: RT @hadleywickham: are you a web expert interested in #rstats? we (@rstudi
```

5.5 Timelines

A Twitter *timeline* is simply a stream of tweets. We support two timelines, the *user timeline* and the *home timeline*. The former provides the most recent tweets of a specified user while the latter is used to display your own most recent tweets. These both return a list of *status* objects.

To look at a particular user's timeline that user must either have a public account or you must have access to their account. You can either pass in the user's name or an object of class *user* (more on this later). For this example, let's use the user *cranatic*.

```
>    cran_tweets <- userTimeline('cranatic')
>    cran_tweets[1:5]

[[1]]
[1] "cranatic: Update: Bchron, BoolNet, caribou, CePa, fmri, HTSCluster, isa2, lessR, lgcp,

[[2]]
[1] "cranatic: New: extrafont, extrafontdb, Rttf2pt1, x12GUI. http://t.co/skyrajMA #rstats"

[[3]]
[1] "cranatic: Update: drc, RcmdrPlugin.survival, rrcov, spls. http://t.co/eEoXNifB #rstats"

[[4]]
[1] "cranatic: New: hzar. http://t.co/eEoXNifB #rstats"

[[5]]
[1] "cranatic: Update: directlabels, forensim, gdata, gWidgetstcltk, gWidgetsWWW, harvestr,
```

By default this command returns the 20 most recent tweet. As with most (but not all) of the functions, it also provides a mechanism to retrieve an arbitrarily large number of tweets up to limits set by the Twitter API, which vary based on the specific type of request.

```
>    cran_tweets_large <- userTimeline('cranatic', n=100)
>    length(cran_tweets_large)

[1] 100
```

The `homeTimeline` function works nearly identically except you do not pass in a user, it uses your own timeline.

5.6 Trends

Twitter keeps track of topics that are popular at any given point of time, and allows one to extract that data. The `getTrends` function is used to pull current trend information from a given location, which is specified using a WOEID (see `http://developer.yahoo.com/geo/geoplanet/`). Luckily there are two other functions to help you identify WOEIDs that you might be interested in. The `availableTrendLocations` function will return a `data.frame` with a location in each row and the `woeid` giving that location's WOEID. Similarly the `closestTrendLocations` function is passed a latitude and longitude and will return the same style `data.frame`.

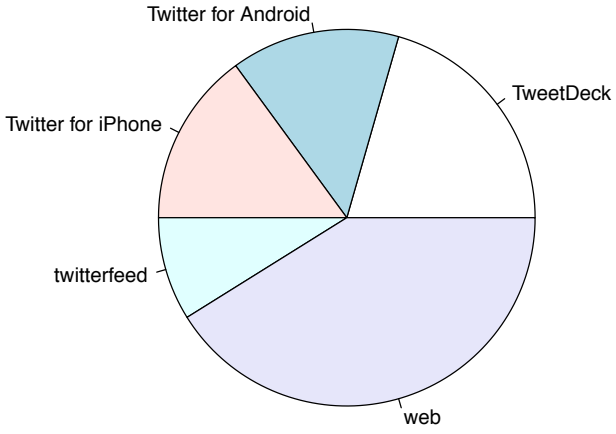
```
>   avail_trends = availableTrendLocations()
>   head(avail_trends)

       name country woeid
1 Worldwide             1
2  Winnipeg  Canada  2972
3    Ottawa  Canada  3369
4    Quebec  Canada  3444
5  Montreal  Canada  3534
6   Toronto  Canada  4118

>   close_trends = closestTrendLocations(-42.8, -71.1)
>   head(close_trends)

        name country  woeid
1 Concepcion   Chile 349860

>   trends = getTrends(2367105)
>   head(trends)

            name                                          url            query
1 #taylorday2014 http://twitter.com/search?q=%23taylorday2014 %23taylorday2014
2   #AliTellsAll   http://twitter.com/search?q=%23AliTellsAll   %23AliTellsAll
3           MCAS             http://twitter.com/search?q=MCAS             MCAS
4        Allison          http://twitter.com/search?q=Allison          Allison
5           #PLL           http://twitter.com/search?q=%23PLL           %23PLL
6        #Bruins        http://twitter.com/search?q=%23Bruins        %23Bruins
    woeid
1 2367105
2 2367105
3 2367105
4 2367105
5 2367105
6 2367105
```

5.7 A simple example

Just a quick example of how one can interact with actual data. Here we will pull the most recent results from the public timeline and see the clients that were used to post those statuses. We can look at a pie chart to get a sense for the most common clients.

Note that sources which are not the standard web interface will be presented as an anchored URL string (<A>...</A>). There are more efficient means to rip out the anchor string than how it is done below, but this is a bit more robust for the purposes of this vignette due to issues with character encoding, locales, etc.

```
>    r_tweets <- searchTwitter("#rstats", n=300)
>    sources <- sapply(r_tweets, function(x) x$getStatusSource())
>    sources <- gsub("</a>", "", sources)
>    sources <- strsplit(sources, ">")
>    sources <- sapply(sources, function(x) ifelse(length(x) > 1, x[2], x[1]))
>    source_table = table(sources)
>    pie(source_table[source_table > 10])
```



6 Examples Of twitteR In The Wild

I've found some examples around the web of people using this package for various purposes, hopefully some of these can give you good ideas on how to do things. Unfortunately I didn't give the package the most easily searched name! If you know of a good example please let me know.

NB: Many of these predate the changes to using the *httr* package, so specifics might have change, rather view these as examples of things you could do.

- Jeffrey Stanton's free book on data science discusses *twitteR*: `http://ischool.syr.edu/media/documents/2012/3/DataScienceBook1_1.pdf`
- Jeffrey Breen's sentiment analysis example: `http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment`
- Mapping your followers: `http://simplystatistics.org/2011/12/21/an-r-function-to-map-your-twitter-followers/`
- Yangchao Zhao's book on data mining w/ R `http://www.amazon.com/Data-Mining-Examples-Case-Studies/dp/0123969638`
- Gary Miner et al's book on data mining `http://www.amazon.com/Practical-Statistical-Analysis-N dp/012386979X`
- Mining Twitter with R `https://sites.google.com/site/miningtwitter/home`
- Organization or conversation in Twitter: A case study of chatterboxing `https://www.asis.org/asist2012/proceedings/Submissions/185.pdf`

7 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 3.0.2 (2013-09-25)
Platform: x86_64-apple-darwin10.8.0 (64-bit)

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] RSQLite_0.11.4 DBI_0.2-7      twitteR_1.1.7  httr_0.3

loaded via a namespace (and not attached):
[1] bit_1.1-11     bit64_0.9-3    digest_0.6.4   RCurl_1.95-4.1 rjson_0.2.13
[6] stringr_0.6.2  tools_3.0.2
```