Twitter client for R

Jeff Gentry

July 8, 2013

1 Introduction

Twitter is a popular service that allows users to broadcast short messages ('*tweets*') for others to read. These can be used to communicate wtih friends, to display headlines, for restaurants to list daily specials, and more. The *twitteR* package is intended to provide access to the Twitter API within R. Users can make access large amounts of Twitter data for data mining and other tasks.

This package is intended to be combined with the *ROAuth* package as as of March 2013 the Twitter API requires the use of OAuth authentication.

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`

2.2 Notes on API coverage

The ultimate goal is to provide full coverage of the Twitter API, although this is not currently the case. Aspects of the API will be added over time, although if there are particular places that you find missing, please contact me. Please take a look at Twitter's documentation on the 1.1 API for more information on what is possible. For most actions, `twitteR` either supports the options presented there or provides a superset of that functionality (via processing in R). The API documentation is available at `https://dev.twitter.com/docs/api/1.1`.

I've long neglected Twitter's streaming API and someone else has picked up my slack with the *streamR* package.

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. The *ROAuth* package can be used to get around this issue.

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:

```
>    cred = getTwitterOAuth(YOURKEY, YOURSECRET)
```

During this process, you'll be prompted with another URL, go to that URL with your browser and you'll be asked to approve the connection for this application. Once you do this, you'll be presented with a PIN, enter that into your R session. Your object is now verified.

To use this token in future sessions, save `cred` to a file, and from there you can use `load` in another session and call `registerTwitterOAuth`:

```
>    registerTwitterOAuth(cred)
```

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)
```

```
[1] TRUE
```

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. The `n` argument can be used to specify the number of tweets to return, defaulting to 25.

```
>    sea <- searchTwitter('#twitter', n=50)
>    sea[1:5]

[[1]]
[1] "Terianmay: @eonline yes please explain the controversy of people putting #everything on

[[2]]
[1] "dvega82: Wow. Can't believe I spelled it like that #Twitter #ifeeldumb"

[[3]]
[1] "razvanhosu7: #Twitter off"

[[4]]
[1] "eduarcortes07: Esta nueva actualizacin de #Twitter parece buena..."

[[5]]
[1] "JaniceMorse: RT @TweetSmarter: #OMG! Illiterate #Twitter Users Are Driving English Lang
```

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.

```
>    crantastic <- getUser('crantastic')
>    crantastic$getFriends(n=5)

$`703506830`
[1] "ELEMENTARYStaff"

$`1132315826`
[1] "pack"

$`143922679`
[1] "RobThomas"

$`22898904`
[1] "vanderjames"

$`538761714`
[1] "KseniaSolo"

>    crantastic$getFavorites(n=5)

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

```
[[2]]
[1] "EmWatson: Who here actually thinks I would do 50 Shades of Grey as a movie? Like really

[[3]]
[1] "AnnaKendrick47: I can't believe they cut the scene where I walk away from that explosio

[[4]]
[1] "kevwilliamson: Who will Elena end up with?  We're so not there yet -- but if you must k

[[5]]
[1] "ChristieKeith: @AliAdler Yeah, we get #invisiblescissoring to go with the #invisiblewe
```

5.3 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`.

```
>   availTrends = availableTrendLocations()
>   head(availTrends)

       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

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

        name country  woeid
1 Concepcion   Chile 349860

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

               name                                              url
1          #MazziMaz            http://twitter.com/search?q=%23MazziMaz
```

```
2   #perksofdatingme     http://twitter.com/search?q=%23perksofdatingme
3    #consofdatingme      http://twitter.com/search?q=%23consofdatingme
4 #mentionperfection   http://twitter.com/search?q=%23mentionperfection
5   #TheWorstFeeling     http://twitter.com/search?q=%23TheWorstFeeling
6     Love & Hip Hop http://twitter.com/search?q=%22Love+%26+Hip+Hop%22
                  query   woeid
1            %23MazziMaz 2367105
2     %23perksofdatingme 2367105
3      %23consofdatingme 2367105
4   %23mentionperfection 2367105
5     %23TheWorstFeeling 2367105
6 %22Love+%26+Hip+Hop%22 2367105
```

5.4 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*.

```
>    cranTweets <- userTimeline('cranatic')
>    cranTweets[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. (warning: At least as of now there is no protection from overloading the API rate limit so be reasonable with your requests).

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

```
[1] 100
```

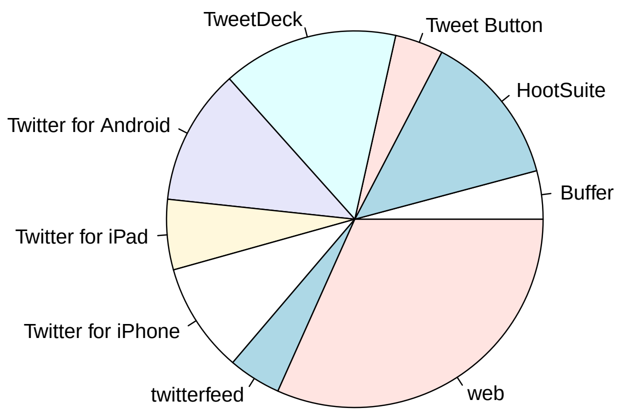
The `homeTimeline` function works nearly identically except you do not pass in a user, it uses your own timeline.

5.5 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])
```



5.6 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(r_tweets)
>    head(df)


1 RT @quantlabs: Is Microsoft #Office #Excel or #Rstats or Matlab most popular for predictiv
2            Is Microsoft #Office #Excel or #Rstats or Matlab most popular for predictive mo
3 RT @reichlab: really cool. RT @benjaminlind: R on the Cloud is now at #rstats version 3.0.
4                                                                             RT @simplystats: Us
5                                                                         NBA (Charlotte Bobcats)
6                                                                             RT @simplystats: Us
  favorited favoriteCount replyToSN             created truncated replyToSID
1     FALSE             0      <NA> 2013-07-08 23:13:33     FALSE       <NA>
2     FALSE             0      <NA> 2013-07-08 22:25:32     FALSE       <NA>
```

```
3     FALSE            0      <NA> 2013-07-08 22:23:33     FALSE       <NA>
4     FALSE            0      <NA> 2013-07-08 22:22:57     FALSE       <NA>
5     FALSE            1      <NA> 2013-07-08 22:13:03     FALSE       <NA>
6     FALSE            0      <NA> 2013-07-08 22:09:36     FALSE       <NA>
                  id replyToUID
1 354377745855291393       <NA>
2 354365664967929857       <NA>
3 354365161680797698       <NA>
4 354365014070673409       <NA>
5 354362520615665665       <NA>
6 354361654286360576       <NA>
                                                                   statusSource
1                          <a href="http://roundteam.co" rel="nofollow">RoundTeam</a>
2                     <a href="http://www.hootsuite.com" rel="nofollow">HootSuite</a>
3 <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
4 <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
5                                                                              web
6         <a href="http://tapbots.com/tweetbot" rel="nofollow">Tweetbot for iOS</a>
      screenName retweetCount isRetweet retweeted longitude latitude
1        infor3x            1      TRUE     FALSE      <NA>     <NA>
2      quantlabs            1     FALSE     FALSE      <NA>     <NA>
3    yannabraham            3      TRUE     FALSE      <NA>     <NA>
4    yannabraham            8      TRUE     FALSE      <NA>     <NA>
5  M_T_Patterson            0     FALSE     FALSE      <NA>     <NA>
6 PotardDechaine            8      TRUE     FALSE      <NA>     <NA>
```

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.

- 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 3.0.0 (2013-04-03)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
[1] C

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

other attached packages:
[1] twitteR_1.1.7  rjson_0.2.12   ROAuth_0.9.3   digest_0.6.3   RCurl_1.95-4.1
[6] bitops_1.0-5

loaded via a namespace (and not attached):
[1] tools_3.0.0
```