# Twitter client for R

Jeff Gentry

April 28, 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 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.

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

### 2.3 Notes on the classes

There are five classes in this package: `user`, `status`, `trend`, `rateLImitInfo`, and `directMessage`. As of this version they have all been implemented as

reference classes (see `setRefClass`). The first two were previously implemented as S4 classes. To help maintain backwards compatibility, the S4 methods (all accessors) have been left in for those two classes although new code should be using the new style accessors.

# 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 as well as the following:

- *requestURL:* `https://api.twitter.com/oauth/request_token`
- accessURL: `http://api.twitter.com/oauth/access_token`
- *authURL:* `http://api.twitter.com/oauth/authorize`

In your R session, you'll want to do the following:

```
>    cred <- OAuthFactory$new(consumerKey=YOURKEY,
+                             consumerSecret=YOURSECRET,
+                             requestURL=requestURL,
+                             accessURL=accessURL,
+                             authURL=authURL)
>    cred$handshake()
```

At this point, 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.

Lastly, to use that credential object within an R session, use the `registerTwitterOAuth` function. Passing your `OAuth` object to that function will cause all of the API calls to go through Twitter's OAuth mechanism instead of the standard URLs:

```
>    registerTwitterOAuth(cred)
```

The `OAuth` object, once the handshake is complete, can be saved to a file and reused. You should not ever have to redo the handshake unless you remove authorization within the Twitter website.

# 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, StackOverflow or look at the links I post at the end.

```
> library(twitteR)
```

```
[1] TRUE
```

# 5 Exploring Twitter

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 whiel 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.

## 5.1 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] "DiamondSweet16: #OMG #Gotanoldpic #webcam #sexy #cute #love #girls #instagram #twitter

[[2]]
[1] "graceellenrose: I hope my entire 5 followers since the activation of my #twitter accou

[[3]]
[1] "tyiana_x: RT @OGkushBallin: #Twitter broke up so many relationships.<ed><U+00A0><U+00BD

[[4]]
[1] "Fanbombs: hey,enhance  your clients by enhancing  your #twitter followers#fanbombs http

[[5]]
[1] "lucasamadeu_: Ol<U+00E1> pessoal, quando entro no #Twitter fico pasmo com tanta novidad
```

## 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.

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

[1] "Crantastic"
```

## 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`.

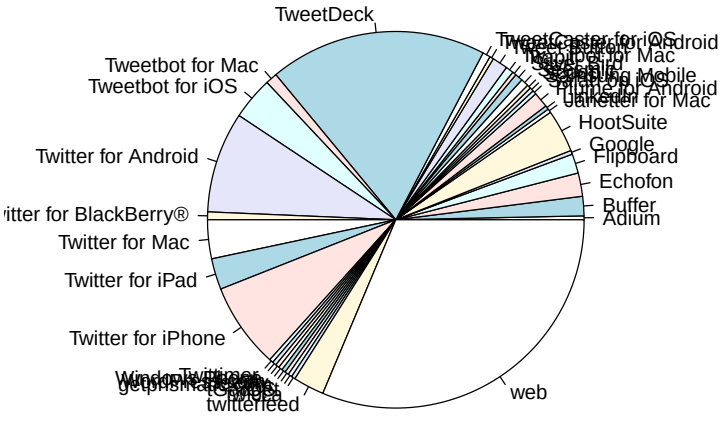
```
>   availTrends = availableTrendLocations()
>   head(availTrends)

       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
1 #PeopleIReallyWantToMeet
2         #confessionnight
3              Jason Terry
4                   #Sens
5                 #runchat
6                  #Bruins
                                                    url
1 http://twitter.com/search?q=%23PeopleIReallyWantToMeet
2         http://twitter.com/search?q=%23confessionnight
3          http://twitter.com/search?q=%22Jason+Terry%22
4                    http://twitter.com/search?q=%23Sens
5                 http://twitter.com/search?q=%23runchat
6                  http://twitter.com/search?q=%23Bruins
                       query   woeid
1 %23PeopleIReallyWantToMeet 2367105
2         %23confessionnight 2367105
3          %22Jason+Terry%22 2367105
4                   %23Sens 2367105
5                 %23runchat 2367105
6                  %23Bruins 2367105
```

## 5.4 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.

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




## 5.5 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(publicTweets)
> #   df[1:3,1:3]
```

# 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_1.1.6  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
```