Twitter client for R

Jeff Gentry

June 6, 2011

1 Disclaimer

Because vignettes are built at various points of time (often automatically), and because a lot of the examples are pulling live data from Twitter at the time of being built, it is possible that some of the content in the examples of this document will be unsavory. I've tried to use users and feeds that are unlikely to be this way, but particularly when looking at the public timeline all bets are off.

2 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.

When joined with the *ROauth* package (not currently available to Windows users), this package can be used to push the API further and directly interact by posting tweets, dealing with direct messages and enjoying enhanced API rate limitations.

3 Initial Notes

3.1 Notes for this version

Major portions of this package have been rewritten from the previous version. In the process I've tried to maintain backwards compatibility where it makes sense, although some function signatures have changed. As the dust settles and I get a better sense for how things play together, I will make another pass at moving things in the direction that I would like the to be, if this leads to functions going away there will be a deprecation cycle. Please contact me with any feedback that you might have.

3.2 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.

3.3 Notes on the classes

There are three classes in this package: `user`, `status`, 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.

4 Getting Started

We'll focus first on those sections of the package that do not require `ROAuth` authentication. The rest of this document won't be an encyclopedic report on the functionality of the package but will just show some basic techniques.

```
> library(twitteR)
```

5 Exploring Twitter

A Twitter *timeline* is simply a stream of tweets - this might be the *public timeline* which is comprised of all public tweets, it might be a user's timeline which would be all of their tweets, or it might even be a timeline to look at one's friend's tweets. Just as there are various *timelines* in Twitter, the *twitteR* package provides various interfaces to access them. The first and most obvious would be the *public timeline*, which retrieves the 20 most recent public tweets on Twitter, returned to the user as a list of *status* objects.

```
> publicTweets <- publicTimeline()
> length(publicTweets)

[1] 20

> publicTweets[1:5]

[[1]]
[1] "luis_felipe1: #queseria yo, sin ti &lt;3"

[[2]]
[1] "mediekos: Hindari stress ditempat kerja dg dengarkan musik favorit setiap pagi juga ber
```

```
[[3]]
[1] "eericsouza: voooooooooooooltei, mais computador . kkkkkkkkk :]"

[[4]]
[1] "HumorAyA: NecY kYdYrli \xe7ox etibar etmYk ... #Tenso"

[[5]]
[1] "Natural_Jovy: I ppl swear their crazy its so funny"
```

```
> publicTweets[[1]]$getScreenName()
```

```
[1] "luis_felipe1"
```

```
> publicTweets[[1]]$getCreated()
```

```
[1] "2011-06-07 00:16:53 UTC"
```

```
> publicTweets[[1]]$getText()
```

```
[1] "#queseria yo, sin ti &lt;3"
```

Similarly, we can look at a particular user's timeline. This will only work properly if that user has a public account or you are authenticated and have access to that account, and can take either a 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: CITAN, coxme, dclone, dynaTree, FAwR, mefa, mefa4, memisc, mosaic, nl

[[2]]
[1] "cranatic: New: bsml. http://bit.ly/k5mVN2 #rstats"

[[3]]
[1] "cranatic: Update: diversitree, gtcorr, raster, RgoogleMaps, tileHMM, tm.plugin.dc, WGCN

[[4]]
[1] "cranatic: Update: DoE.wrapper, Epi, GeneReg, R2Cuba, SubpathwayMiner. http://bit.ly/90f

[[5]]
[1] "cranatic: Update: GenABEL, MIfuns, MIfuns, OrdFacReg, PairViz, RExcelInstaller, SPOT. h
```

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
```

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] "Katrina_Diem: RT @ruhanirabin: 7 Sneaky Ways to Use #Twitter to Spy on Your Competitior

[[2]]
[1] "bgponzoni: My #twitter age is 1 year 293 days 0 hours 5 minutes 6 seconds. Find out you

[[3]]
[1] "rodrigodeves: My #twitter age is 1 year 145 days 6 hours 8 minutes 27 seconds. Find out

[[4]]
[1] "carol_melzinha: RT @thiagomelo3210: @carol_melzinha vicio No #twitter  HAHAHA."

[[5]]
[1] "dwoods1984: Lmmfao RT @UrFavoriteGYN: #Twitter......#1 DATING WEBSITE IN THE WORLD #Mov
```

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.

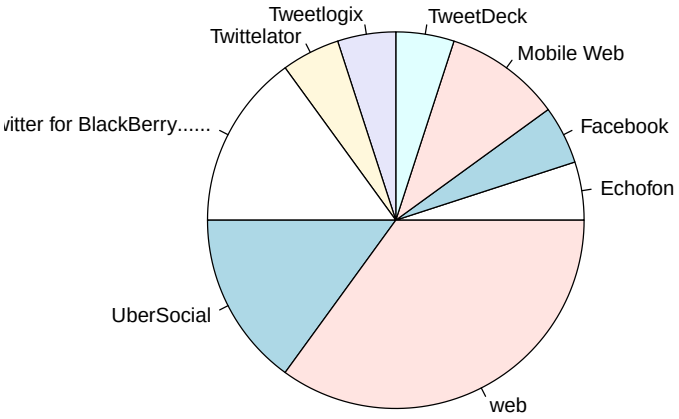
```
> crantastic <- getUser("crantastic")
> crantastic

[1] "Crantastic"
```

5.3 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.

```
> sources <- sapply(publicTweets, 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))
```



6 Authentication with OAuth

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`. Set the "Application Type" as "Client", and "Default

Access Type" as "Read & Write". This will provide you with two strings, a consumer key and a consumer secret. Record these for your future use.

Three other pieces of information you will need:

- *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.

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.

7 Session Information

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

```
R version 2.13.0 (2011-04-13)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

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

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

other attached packages:
[1] twitteR_0.99.8 RJSONIO_0.7-2  RCurl_1.6-4    bitops_1.0-4.1

loaded via a namespace (and not attached):
[1] tools_2.13.0
```