Package ‘streamR’

December 9, 2018

Title Access to Twitter Streaming API via R

Description Functions to access Twitter's filter, sample, and user streams, and to parse the output into data frames.

Version 0.4.5

Author Pablo Barbera <pablo.barbera@nyu.edu>

Maintainer Pablo Barbera <pablo.barbera@nyu.edu>

Depends R (>= 2.12.0), RCurl, rjson, ndjson

Suggests ROAuth (>= 0.9.0)

License GPL-2

Collate 'filterStream.R' 'parseTweets.R' 'sampleStream.R' 'userStream.R' 'streamR-package.R' 'readTweets.R'

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-12-09 20:00:03 UTC

R topics documented:

streamR-package 2
createOAuthToken 2
example_tweets 3
filterStream 3
parseTweets 5
readTweets 7
sampleStream 8
userStream 10

Index **13**

`streamR-package` *Access to Twitter Streaming APIs via R*

Description

This package provides a series of functions that allow R users to access Twitter's filter, sample, and user streams, and to parse the output into data frames.

Author(s)

Pablo Barbera `<pablo.barbera@nyu.edu>`

See Also

`filterStream`, `sampleStream`, `userStream`, `readTweets`, `parseTweets`

`createOAuthToken` *Create OAuth token without handshake.*

Description

This function generates a OAuth token using the consumer key, consumer secret, access token and access token secret available in the "Keys and Access Token" tab of the "Application Management" website on Twitter's developers website.

Usage

```
createOAuthToken(consumerKey, consumerSecret, accessToken, accessTokenSecret)
```

Arguments

`consumerKey`	Consumer key for OAuth token
`consumerSecret`	Consumer secret for OAuth token
`accessToken`	Access token for OAuth token
`accessTokenSecret`	Access token secret for OAuth token

example_tweets	*Ten sample tweets published by @twitterapi*

Description

A vector of string characters that contains ten sample tweets in plain text.

Usage

```
data(example_tweets)
```

Source

http://www.twitter.com/twitterapi

filterStream	*Connect to Twitter Streaming API and return public statuses that match one or more filter predicates.*

Description

`filterStream` opens a connection to Twitter's Streaming API that will return public statuses that match one or more filter predicates. Tweets can be filtered by keywords, users, language, and location. The output can be saved as an object in memory or written to a text file.

Usage

```
filterStream(file.name = NULL, track = NULL, follow = NULL,
  locations = NULL, language = NULL, timeout = 0, tweets = NULL,
  oauth = NULL, verbose = TRUE)
```

Arguments

`file.name`	string, name of the file where tweets will be written. "" indicates output to the console, which can be redirected to an R object (see examples). If the file already exists, tweets will be appended (not overwritten).
`track`	string or string vector containing keywords to track. See the `track` parameter information in the Streaming API documentation for details.
`follow`	string or numeric, vector of Twitter user IDs, indicating the users whose public statuses should be delivered on the stream. See the `follow` parameter information in the Streaming API documentation for details.
`locations`	numeric, a vector of longitude, latitude pairs (with the southwest corner coming first) specifying sets of bounding boxes to filter public statuses by. See the `locations` parameter information in the Streaming API documentation for details.

`language`	string or string vector containing a list of BCP 47 language identifiers. If not `NULL` (default), function will only return tweets that have been detected as being written in the specified languages. Note that this parameter can only be used in combination with any of the other filter parameters. See documentation for details.
`timeout`	numeric, maximum length of time (in seconds) of connection to stream. The connection will be automatically closed after this period. For example, setting `timeout` to 10800 will keep the connection open for 3 hours. The default is 0, which will keep the connection open permanently.
`tweets`	numeric, maximum number of tweets to be collected when function is called. After that number of tweets have been captured, function will stop. If set to `NULL` (default), the connection will be open for the number of seconds specified in `timeout` parameter.
`oauth`	an object of class `oauth` that contains the access token to the user's twitter session OR a list with details to create a new access token. See examples for more details.
`verbose`	logical, default is `TRUE`, which generates some output to the R console with information about the capturing process.

Details

`filterStream` provides access to the statuses/filter Twitter stream.

It will return public statuses that match the keywords given in the `track` argument, published by the users specified in the `follow` argument, written in the language specified in the `language` argument, and sent within the location bounding boxes declared in the `locations` argument.

Note that location bounding boxes do not act as filters for other filter parameters. In the fourth example below, we capture all tweets containing the term rstats (even non-geolocated tweets) OR coming from the New York City area. For more information on how the Streaming API request parameters work, check the documentation at: `https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters`.

Also note that the `language` parameter needs to be used in combination with another filter option (either keywords or location).

If any of these arguments is left empty (e.g. no user filter is specified), the function will return all public statuses that match the other filters. At least one predicate parameter must be specified.

Note that when no file name is provided, tweets are written to a temporary file, which is loaded in memory as a string vector when the connection to the stream is closed.

The total number of actual tweets that are captured might be lower than the number of tweets requested because blank lines, deletion notices, and incomplete tweets are included in the count of tweets downloaded.

Author(s)

Pablo Barbera <`pablo.barbera@nyu.edu`>

See Also

`sampleStream`, `userStream`, `parseTweets`

Examples

```
## Not run:

## An example of an authenticated request using the ROAuth package,
## where consumerkey and consumer secret are fictitious.
## You can obtain your own at dev.twitter.com
  library(ROAuth)
  requestURL <- "https://api.twitter.com/oauth/request_token"
  accessURL <- "https://api.twitter.com/oauth/access_token"
  authURL <- "https://api.twitter.com/oauth/authorize"
  consumerKey <- "xxxxxyyyyyzzzzzz"
  consumerSecret <- "xxxxxxyyyyyzzzzzzz111111222222"
  my_oauth <- OAuthFactory$new(consumerKey=consumerKey,
    consumerSecret=consumerSecret, requestURL=requestURL,
    accessURL=accessURL, authURL=authURL)
  my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))

## Alternatively, it is also possible to create a token without the handshake:
 my_oauth <- list(consumer_key = "CONSUMER_KEY",
   consumer_secret = "CONSUMER_SECRET",
   access_token="ACCESS_TOKEN",
   access_token_secret = "ACCESS_TOKEN_SECRET")

## capture 10 tweets mentioning the "Rstats" hashtag
  filterStream( file.name="tweets_rstats.json",
     track="rstats", tweets=10, oauth=my_oauth )

## capture tweets published by Twitter's official account
  filterStream( file.name="tweets_twitter.json",
     follow="783214", timeout=600, oauth=my_oauth )

## capture tweets sent from New York City in Spanish only, and saving as an object in memory
  tweets <- filterStream( file.name="", language="es",
      locations=c(-74,40,-73,41), timeout=600, oauth=my_oauth )

## capture tweets mentioning the "rstats" hashtag or sent from New York City
  filterStream( file="tweets_rstats.json", track="rstats",
      locations=c(-74,40,-73,41), timeout=600, oauth=my_oauth )


## End(Not run)
```

`parseTweets` *Converts tweets in JSON format to data frame.*

Description

This function parses tweets downloaded using `filterStream`, `sampleStream` or `userStream` and returns a data frame. If tweet contains 280-character text it will return the complete text and not only 140 characters.

Usage

```
parseTweets(tweets, simplify = FALSE, verbose = TRUE, legacy = FALSE)
```

Arguments

`tweets`	A character string naming the file where tweets are stored or the name of the object in memory where the tweets were saved as strings.
`simplify`	If `TRUE` it will return a data frame with only tweet and user fields (i.e., no geographic information or url entities).
`verbose`	logical, default is `TRUE`, which will print in the console the number of tweets that have been parsed.
`legacy`	logical, default is `FALSE`. Read tweets using old method (reading lines into memory and parsing line by line). Try using `legacy=TRUE` if getting errors with default options. Note that legacy mode will only return up to 140 characters per tweet.

Details

`parseTweets` parses tweets downloaded using the `filterStream`, `sampleStream` or `userStream` functions and returns a data frame where each row corresponds to one tweet and each column represents a different field for each tweet (id, text, created_at, etc.).

The total number of tweets that are parsed might be lower than the number of lines in the file or object that contains the tweets because blank lines, deletion notices, and incomplete tweets are ignored.

To parse json to a twitter list, see `readTweets`. That function can be significantly faster for large files, when only a few fields are required.

Note also that the `retweet_count` field contains the number of times a given tweet was retweeted at the time it was captured from the API, or for automatic retweets the number of times the original tweet was retweeted.

Author(s)

Pablo Barbera <`pablo.barbera@nyu.edu`>

See Also

`filterStream`, `sampleStream`, `userStream`

Examples

```
## The dataset example_tweets contains 10 public statuses published
## by @twitterapi in plain text format. The code below converts the object
## into a data frame that can be manipulated by other functions.

data(example_tweets)
tweets.df <- parseTweets(example_tweets, simplify=TRUE, legacy=TRUE)

## Not run:
```

```
## A more complete example, that shows how to capture a user's home timeline
## for one hour using authentication via OAuth, and then parsing the tweets
## into a data frame.

 library(ROAuth)
 reqURL <- "https://api.twitter.com/oauth/request_token"
 accessURL <- "https://api.twitter.com/oauth/access_token"
 authURL <- "https://api.twitter.com/oauth/authorize"
 consumerKey <- "xxxxxyyyyyzzzzzz"
 consumerSecret <- "xxxxxxyyyyyzzzzzzz111111222222"
 my_oauth <- OAuthFactory$new(consumerKey=consumerKey,
                              consumerSecret=consumerSecret,
                              requestURL=reqURL,
                              accessURL=accessURL,
                              authURL=authURL)
 my_oauth$handshake()
 userStream( file="my_timeline.json", with="followings",
         timeout=3600, oauth=my_oauth )
 tweets.df <- parseTweets("my_timeline.json")

## End(Not run)
```

`readTweets` *Converts tweets in JSON format to R list.*

Description

This function parses tweets downloaded using `filterStream`, `sampleStream` or `userStream` and returns a list.

Usage

```
readTweets(tweets, verbose = TRUE)
```

Arguments

`tweets`	A character string naming the file where tweets are stored or the name of the object in memory where the tweets were saved as strings.
`verbose`	logical, default is `TRUE`, which will print in the console the number of tweets that have been parsed.

Details

This function is the first step in the `parseTweets` function and is provided now as an independent function for convenience purposes. In cases where only one field is needed, it can be faster to extract it directly from the JSON data read in R as a list. It can also be useful to extract fields that are not parsed by `parseTweets`, such as hashtags or mentions.

The total number of tweets that are parsed might be lower than the number of lines in the file or object that contains the tweets because blank lines, deletion notices, and incomplete tweets are ignored.

Author(s)

Pablo Barbera <pablo.barbera@nyu.edu>

See Also

parseTweets.

Examples

```
## The dataset example_tweets contains 10 public statuses published
## by @twitterapi in plain text format. The code below converts the object
## into a list and extracts only the text.

data(example_tweets)
tweets.list <- readTweets(example_tweets)
only.text <- unlist(lapply(tweets.list, '[[', 'text'))
## it can be done with an explicit loop:
only.text <- c()
for (i in 1:length(tweets.list)){
   only.text[i] <- tweets.list[[i]]['text']
}
print(unlist(only.text))
```

---

sampleStream — *Connect to Twitter Streaming API and return a small random sample of all public statuses.*

Description

sampleStream opens a connection to Twitter's Streaming API that will return a small random sample of public statuses, around 1% at any given time.

Usage

```
sampleStream(file.name, timeout = 0, tweets = NULL, oauth = NULL,
  verbose = TRUE)
```

Arguments

`file.name`	string, name of the file where tweets will be written. "" indicates output to the console, which can be redirected to an R object. If the file already exists, tweets will be appended (not overwritten).
`timeout`	numeric, maximum length of time (in seconds) of connection to stream. The connection will be automatically closed after this period. For example, setting `timeout` to 10800 will keep the connection open for 3 hours. The default is 0, which will keep the connection open permanently.
`tweets`	numeric, maximum number of tweets to be collected when function is called. After that number of tweets have been captured, function will stop. If set to `NULL` (default), the connection will be open for the number of seconds specified in `timeout` parameter.
`oauth`	an object of class `oauth` that contains the access token to the user's twitter session OR a list with details to create a new access token. See examples for more details.
`verbose`	logical, default is `TRUE`, which generates some output to the R console with information about the capturing process.

Details

For more information, check the documentation at: `https://developer.twitter.com/en/docs/tweets/sample-realtime/overview/GET_statuse_sample`

Note that when no file name is provided, tweets are written to a temporary file, which is loaded in memory as a string vector when the connection to the stream is closed.

The total number of actual tweets that are captured might be lower than the number of tweets requested because blank lines, deletion notices, and incomplete tweets are included in the count of tweets downloaded.

Author(s)

Pablo Barbera <`pablo.barbera@nyu.edu`>

See Also

`filterStream`, `userStream`, `parseTweets`

Examples

```
## Not run:
## capture a random sample of tweets
sampleStream( file.name="tweets_sample.json", user=FOO, password=BAR )

## An example of an authenticated request using the ROAuth package,
## where consumerkey and consumer secret are fictitious.
## You can obtain your own at dev.twitter.com
 library(ROAuth)
 reqURL <- "https://api.twitter.com/oauth/request_token"
 accessURL <- "https://api.twitter.com/oauth/access_token"
 authURL <- "https://api.twitter.com/oauth/authorize"
 consumerKey <- "xxxxxyyyyyzzzzzz"
 consumerSecret <- "xxxxxxyyyyyzzzzzzz111111222222"
  my_oauth <- OAuthFactory$new(consumerKey=consumerKey,
    consumerSecret=consumerSecret, requestURL=requestURL,
    accessURL=accessURL, authURL=authURL)
 my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))

## Alternatively, it is also possible to create a token without the handshake:
 my_oauth <- list(consumer_key = "CONSUMER_KEY",
   consumer_secret = "CONSUMER_SECRET",
   access_token="ACCESS_TOKEN",
   access_token_secret = "ACCESS_TOKEN_SECRET")

 sampleStream( file.name="tweets_sample.json", oauth=my_oauth )


## End(Not run)
```

---

userStream — *Connect to Twitter Streaming API and return messages for a single user.*

Description

userStream opens a connection to Twitter's Streaming API that will return statuses specific to the authenticated user. The output can be saved as an object in memory or written to a text file.

Usage

```
userStream(file.name = NULL, with = "followings", replies = NULL,
  track = NULL, locations = NULL, timeout = 0, tweets = NULL,
  oauth = NULL, verbose = TRUE)
```

Arguments

file.name	string, name of the file where tweets will be written. "" indicates output to the console, which can be redirected to an R object. If the file already exists, tweets will be appended (not overwritten).
with	string, detault is "followings", which will stream messages from accounts the authenticated user follow. If set to "user", will only stream messages from authenticated user. See the `with` parameter information in the Streaming API documentation for details: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters

`replies`	string, default is `NULL`, which will only stream replies sent by a different user if the authenticated user follows the receiver of the reply. All replies to users that the authenticated user follows will be included if this argument is set to "all". See the `replies` parameter information in the Streaming API documentation for details: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters
`track`	string or string vector containing keywords to track. See the track parameter information in the Streaming API documentation for details: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.
`locations`	numeric, a vector of longitude, latitude pairs (with the southwest corner coming first) specifying sets of bounding boxes to filter statuses by. See the locations parameter information in the Streaming API documentation for details: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters
`timeout`	numeric, maximum length of time (in seconds) of connection to stream. The connection will be automatically closed after this period. For example, setting `timeout` to 10800 will keep the connection open for 3 hours. The default is 0, which will keep the connection open permanently.
`tweets`	numeric, maximum number of tweets to be collected when function is called. After that number of tweets have been captured, function will stop. If set to `NULL` (default), the connection will be open for the number of seconds specified in `timeout` parameter.
`oauth`	an object of class `oauth` that contains the access token to the user's twitter session OR a list with details to create a new access token. See examples for more details.
`verbose`	logical, default is `TRUE`, which generates some output to the R console with information about the capturing process.

Details

This function provides access to messages for a single user.

The set of messages to be returned can include the user's tweets and/or replies, and public statuses published by the accounts the user follows, as well to replies to those accounts.

Tweets can also be filtered by keywords and location, using the `track` and `locations` arguments.

The total number of actual tweets that are captured might be lower than the number of tweets requested because blank lines, deletion notices, and incomplete tweets are included in the count of tweets downloaded.

Note that when no file name is provided, tweets are written to a temporary file, which is loaded in memory as a string vector when the connection to the stream is closed.

Author(s)

Pablo Barbera <`pablo.barbera@nyu.edu`>

See Also

`filterStream`, `sampleStream`, `parseTweets`

Examples

```
## Not run:
## The following example shows how to capture a user's home timeline
## with the Streaming API and using authentication via the ROAuth
## package, with fictitious consumerkey and consumer secret.
## You can obtain your own at dev.twitter.com
 library(ROAuth)
 requestURL <- "https://api.twitter.com/oauth/request_token"
 accessURL <- "https://api.twitter.com/oauth/access_token"
 authURL <- "https://api.twitter.com/oauth/authorize"
 consumerKey <- "xxxxxyyyyyzzzzzz"
 consumerSecret <- "xxxxxxyyyyyzzzzzzz111111222222"
 my_oauth <- OAuthFactory$new(consumerKey=consumerKey,
    consumerSecret=consumerSecret, requestURL=requestURL,
    accessURL=accessURL, authURL=authURL)
 my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))

## Alternatively, it is also possible to create a token without the handshake:
 my_oauth <- list(consumer_key = "CONSUMER_KEY",
   consumer_secret = "CONSUMER_SECRET",
   access_token="ACCESS_TOKEN",
   access_token_secret = "ACCESS_TOKEN_SECRET")

## Capturing 10 tweets from a user's timeline
 userStream( file.name="my_timeline.json", with="followings",
     tweets=10, oauth=my_oauth )

## End(Not run)
```

Index

*Topic **datasets**
    example_tweets, 3

createOAuthToken, 2

example_tweets, 3

filterStream, *2*, 3, *6*, *9*, *11*

parseTweets, *2*, *4*, 5, *7–9*, *11*

readTweets, *2*, *6*, 7

sampleStream, *2*, *4*, *6*, 8, *11*
streamR *(*streamR-package*)*, 2
streamR-package, 2

userStream, *2*, *4*, *6*, *9*, 10