Package 'twitteR'

July 2, 2014

Title R based Twitter client

Description Provides an interface to the Twitter web API

Version 1.1.7

Author Jeff Gentry <geoffjentry@gmail.com>

Maintainer Jeff Gentry <geoffjentry@gmail.com>

Depends R (>= 2.12.0), ROAuth (>= 0.9.3), RCurl, rjson (>= 0.2.12),methods

License Artistic-2.0

LazyData yes

URL http://lists.hexdump.org/listinfo.cgi/twitter-users-hexdump.org

Collate allGenerics.R base.R account.R statuses.R users.R trends.R
s4methods.R convert.R dm.R comm.R followers.R search.R toys.R utils.R zzz.R

NeedsCompilation no

Repository CRAN

Date/Publication 2013-07-09 17:42:20

R topics documented:

`decode_short_url` *A function to decode shortened URLs*

Description

Will expand a URL that has been processed by a link shortener (e.g. bit.ly). Provided as a convenience function to users who may which to perform this operation.

Usage

```
decode_short_url(url, ...)
```

Arguments

`url`	A character string, the URL to decode
`...`	Optional arguments to pass along to RCurl

Details

Uses the `longapi.org` API

Value

A character string containing either the original URL (if not shortened) or the full URL (if shortened)

Author(s)

Neil Jang

References

`longapi.org`

Examples

```
## Not run:
  decode_short_url("http://bit.ly/23226se656")

## End(Not run)
```

`directMessage-class` *Class "directMessage": A class to represent Twitter Direct Messages*

Description

Provides a model representing direct messages (DMs) from Twitter

Details

The `directMessage` class is implemented as a reference class. As there should be no backwards compatibility issues, there are no S4 methods provided as with the `user` and `status` classes. An instance of a generator for this class is provided as a convenience to the user as it is configured to handle most standard cases. To access this generator, use the object `dmFactory`. Accessor set & get methods are provided for every field using reference class `$accessors()` methodology (see `setRefClass` for more details). As an example, the `sender` field could be accessed using `object$getSender()` and `object$setSender()`.

The constructor of this object assumes that the user is passing in a JSON encoded Twitter Direct Message. It is also possible to directly pass in the arguments.

Fields

`text`: Text of the DM

`recipient`: A `user` object representing the recipient of the message

`recipientSN`: Screen name of the recipient

`recipientID`: ID number of the recipient

`sender`: A `user` object representing the sender of the message

`senderSN`: Screen name of the sender

`senderID`: ID number of the sender

`created`: When the messages was created

Methods

`destroy`: Deletes this DM from Twitter. A wrapper around `dmDestroy`

`toDataFrame`: Converts this into a one row `data.frame`, with each field representing a column. This can also be accomplished by the S4 style `as.data.frame(objectName)`.

Author(s)

Jeff Gentry

See Also

dmGet, dmSend, dmDestroy, setRefClass

Examples

```
## Not run:
  dm <- dmFactory$new(text='foo', recipientSN='blah')
  dm$getText()

  ## assume 'json' is the return from a Twitter call
  dm <- dmFactory$new(json)
  dm$getSenderID()

## End(Not run)
```

---

dmGet *Functions to manipulate Twitter direct messages*

Description

These functions allow you to interact with, send, and delete direct messages (DMs) in Twitter.

Usage

```
dmGet(n=25, sinceID=NULL, maxID=NULL, ...)
dmSent(n=25, sinceID=NULL, maxID=NULL, ...)
dmDestroy(dm, ...)
dmSend(text, user, ...)
```

Arguments

text	The text of a message to send
user	The user to send a message to, either character or an user object.
dm	The message to delete, an object of class directMessage
n	The maximum number of direct messages to return
sinceID	If not NULL, an ID representing the earliest boundary
maxID	If not NULL, an ID representing the newest ID you wish to retrieve
...	Further arguments to pass along the communication chain

Value

These functions will not work without OAuth authentication

The dmGet and dmSent functions will return a list of directMessage objects. The former will retrieve DMs sent to the user while the latter retrieves messages sent from the user.

The dmDestroy function takes a directMessage object (perhaps from either dmGet or dmSent) and will delete it from the Twitter server.

The dmSend function will send a message to another Twitter user.

Author(s)

Jeff Gentry

See Also

directMessage, registerTwitterOAuth

Examples

```
## Not run:
         dms <- dmGet()
         dms
         ## delete the first one
         dms[[1]]$destroy()
         dmDestroy(dms[[2]])
         ## send a DM
         dmSend('Testing out twitteR!', 'twitter')

## End(Not run)
```

---

favorites *A function to get favorite tweets*

Description

Returns the n most recently favorited tweets from the specified user.

Usage

```
favorites(user, n = 20, max_id = NULL, since_id = NULL, ...)
```

Arguments

user	The Twitter user to detail, can be character or an user object.
n	Number of tweets to retrieve, up to a maximum of 200
max_id	Maximum ID to search for
since_id	Minimum ID to search for
...	Optional arguments to pass along to RCurl

Value

A list of link{status} objects corresponding to the n most recent tweets

Author(s)

Jeff Gentry

References

https://dev.twitter.com/docs/api/1.1/get/favorites/list

See Also

getUser, status

Examples

```
## Not run:
    fav = favorites("barackobama", n=100)

## End(Not run)
```

`friendships` *A function to detail relations between yourself & other users*

Description

This function will accept a list of other Twitter users and will detail if they follow you and/or you follow them.

Usage

```
friendships(screen_names = character(), user_ids = character(), ...)
```

Arguments

`screen_names`	A vector of one or more Twitter screen names
`user_ids`	A vector of one or more Twitter user id values
`...`	Any other arguments to pass to RCurl

Details

The combined number of screen names and user ids may not exceed 100. Any non-existent users will be dropped from the output

Value

A data.frame, one row for each user requested with columns `name`, `screen_name`, `id`, `following` and `followed_by`. The latter two columns will be `TRUE` or `FALSE` depending on that user's relations with your account.

Author(s)

Jeff Gentry

References

https://dev.twitter.com/docs/api/1.1/get/friendships/lookup

See Also

registerTwitterOAuth

Examples

```
  ## Not run:
    friendships()

## End(Not run)
```

`getCurRateLimitInfo` *A function to retrieve current rate limit information*

Description

Will retrieve the current rate limit information for the authenticated user, displayed as a data.frame displaying specifc information for every Twitter resource

Usage

```
getCurRateLimitInfo(resources=character(), ...)
```

Arguments

`resources`	An optional character vector of specific resources to get information for
`...`	Optional arguments to pass to cURL

Details

Using the `resources` argument will filter the returned data.frame to only list the specified resource values

The full list of allowed values in `resources` is as follows: `lists`, `application`, `friendships`, `blocks`, `geo`, `users`, `followers`, `statuses`, `help`, `friends`, `direct_messages`, `account`, `favorites`, `saved_searches`, `search`, `trends`

Value

A four column data.frame with columns `resource`, `limit`, `remaining` and `reset`. These detail the specific resource name, the rate limit for that block, the number of calls remaining and the time the rate limit will be reset in UTC time.

Author(s)

Jeff Gentry

Examples

```
## Not run:
  zz <- getCurRateLimitInfo(c("lists", "users"))

## End(Not run)
```

---

getTrends *Functions to view Twitter trends*

Description

These functions will allow you to interact with the trend portion of the Twitter API

Usage

```
availableTrendLocations(...)
closestTrendLocations(lat, long, ...)
getTrends(woeid, exclude=NULL, ...)
```

Arguments

woeid	A numerical identification code describing a location, a Yahoo! Where On Earth ID
lat	A numerical latitude value, between -180 and 180 inclusive. West is negative, East is positive
long	A numerical longitude value, between -180 and 180 inclusive. South is negative, North is positive
exclude	If set to hashtags, will exclude hashtags
...	Additional arguments to be passed to RCurl

Details

The availableTrendLocations and closestTrendLocations functions will return a data.frame with three columns - name, country and woeid. The closestTrendLocations function will return the locations closest to the specified latitude and longitude.

The getTrends function takes a specified woeid and returns the trending topics associated with that woeid. It returns a data.frame with the columns being name, url, promoted_content, query and woeid - one row per trend.

Value

A data.frame with the columns specified in Details above

Author(s)

Jeff Gentry

Examples

```
## Not run:
  woeid = availableTrendLocations[1, "woeid"]
  t1 <- getTrends(woeid)

## End(Not run)
```

---

getUser *Functions to manage Twitter users*

Description

These functions allow you interact with information about a Twitter user - retrieving their base information, list of friends, list of followers, and an up to date timeline.

Usage

```
getUser(user, ...)
lookupUsers(users, includeNA=FALSE, ...)
```

Arguments

user	The Twitter user to detail, can be character or an user object.
users	A vector of either user IDs or screen names or a mix of both
includeNA	If TRUE will leave an NA element in the return list for users that don't exist
...	Optional arguments to be passed to getURL

Details

These functions will only return fully formed objects if the authenticated user is allowed to see the requested user. If that person has a private account and has not allowed you to see them, you will not be able to extract that information.

The lookupUsers function should be used in cases where there are multiple lookups going to take place, to reduce the API call load. This function requires OAuth authentication.

Value

The getUser function returns an object of class user.

The lookupUsers function will return a list of user objects, sorted in the order of the users argument, with names being the particular element of users that it matches to. If the includeNA argument is set to FALSE (default), any non-existing users will be dropped from the list.

Author(s)

Jeff Gentry

See Also

mentions

Examples

```
## Not run:
  tuser <- getUser('geoffjentry')
  users <- lookupUsers(c('geoffjentry', 'whitehouse'))

## End(Not run)
```

`import_statuses` *Functions to import twitteR objects from various sources*

Description

Functions designed to import data into twitteR objects from a variety of data sources. Currently only JSON is supported, and this entire branch of functionality should be considered experimental & under development.

Usage

```
import_statuses(raw_data, conversion_func = json_to_statuses)
import_trends(raw_data, conversion_func = json_to_trends)
import_users(raw_data, conversion_func = json_to_users)
import_obj(raw_data, conversion_func, ...)
json_to_users(raw_data)
json_to_statuses(raw_data)
json_to_trends(raw_data)
```

Arguments

`raw_data`	Data to be be parsed via the prescribed function
`conversion_func`	The function to convert `raw_data` into the specified twitteR object
`...`	Arguments to pass along to `conversion_func`

Value

A list of twitteR objects of the appropriate type, e.g. `status`, `user`, etc

Author(s)

Jeff Gentry

See Also

`status`, `user`

Examples

```
      ## Not run:
         status_list = import_statuses(list_of_status_json)

   ## End(Not run)
```

`registerTwitterOAuth` *Register OAuth credentials to twitter R session*

Description

This function is used to provide your OAuth access tokens to your twitter session. This will enable many bits of functionality as well as allow other commands to provide more options

Usage

```
getTwitterOAuth(consumer_key, consumer_secret)
registerTwitterOAuth(oauth)
```

Arguments

`consumer_key`	The consumer key supplied by Twitter
`consumer_secret`	The consumer secret supplied by Twitter
`oauth`	An object of class `OAuth`

Details

The `getTwitterOAuth` function is a wrapper around the call to `OAuthFactory` and `registerTwitterOAuth`, which will return the registered credentials. If your workflow is such that you save the credentials and register them in later R sessions, feel free to do this using `registerTwitterOAuth`

`registerTwitterOAuth` will store the `OAuth` argument in an environment which is then accessed throughout the package. When API calls are made, instead of going through `RCurl` they will go through the `ROAuth` package.

Three URLs will need to be used for the initial `OAuth` handshake, see the examples below.

Value

`TRUE` on success, otherwise an error will be thrown

Author(s)

Jeff Gentry

See Also

`OAuth`

Examples

```
## Not run:
   ## A real example, but using a fictitious consumerkey and consumer
   ## secret - you'll need to supply your own
   reqURL <- "https://api.twitter.com/oauth/request_token"
   accessURL <- "http://api.twitter.com/oauth/access_token"
   authURL <- "http://api.twitter.com/oauth/authorize"
   consumerKey <- "12345pqrst6789ABCD"
   consumerSecret <- "abcd1234EFGH5678ijkl0987MNOP6543qrst21"
   twitCred <- OAuthFactory$new(consumerKey=consumerKey,
                                consumerSecret=consumerSecret,
                                requestURL=reqURL,
                                accessURL=accessURL,
                                authURL=authURL)
   twitCred$handshake()
   registerTwitterOAuth(twitCred)

## End(Not run)
```

`searchTwitter` *Search twitter*

Description

This function will issue a search of Twitter based on a supplied search string.

Usage

```
searchTwitter(searchString, n=25, lang=NULL, since=NULL, until=NULL,
              locale=NULL, geocode=NULL, sinceID=NULL,
            retryOnRateLimit=120, ...)
Rtweets(n=25, lang=NULL, since=NULL, ...)
```

Arguments

`searchString`	Search query to issue to twitter
`n`	The maximum number of tweets to return
`lang`	If not `NULL`, restricts tweets to the given language, given by an ISO 639-1 code
`since`	If not `NULL`, restricts tweets to those since the given date. Date is to be formatted as YYYY-MM-DD
`until`	If not `NULL`, restricts tweets to those up until the given date. Date is to be formatted as YYYY-MM-DD
`locale`	If not `NULL`, will set the locale for the search. As of 03/06/11 only `ja` is effective, as per the Twitter API
`geocode`	If not `NULL`, returns tweets by users located within a given radius of the given latitude/longitude. See `Details` below for more information

sinceID	If not NULL, returns tweets with IDs greater (ie newer) than the specified ID
retryOnRateLimit	If non-zero the search command will block retry up to X times if the rate limit is experienced. This might lead to a much longer run time but the task will eventually complete if the retry count is high enough
...	Optional arguments to be passed to getURL

Details

These commands will return any authorized tweets which match the search criteria. Note that there are pagination restrictions as well as other limits on what can be searched, so it is always possible to not retrieve as many tweets as was requested with the n argument. Authorized tweets are public tweets as well as those protected tweets that are available to the user after authenticating via registerTwitterOAuth.

For the geocode argument, the values are given in the format latitude,longitude,radius, where the radius can have either mi (miles) or km (kilometers) as a unit. For example geocode='37.781157,-122.39720,1mi'.

For the sinceID argument, if the requested ID value is older than the oldest available tweets, the API will return tweets starting from the oldest ID available.

The Rtweets function is a wrapper around searchTwitter which hardcodes in a search for #rstats.

Value

A list of status objects

Author(s)

Jeff Gentry

See Also

status, registerTwitterOAuth

Examples

```
## Not run:
 searchTwitter("#beer", n=100)
        Rtweets(n=37)

 ## Search between two dates
        searchTwitter('charlie sheen', since='2011-03-01', until='2011-03-02')

 ## geocoded results
 searchTwitter('patriots', geocode='42.375,-71.1061111,10mi')

## End(Not run)
```

showStatus *A function to return one specific tweet*

Description

This function will take a numeric ID of a tweet and return it to the user

Usage

```
showStatus(id, ...)
```

Arguments

id	Numerical ID of a specific tweet
...	Optional arguments to be passed to getURL

Value

An object of class status

Author(s)

Jeff Gentry

See Also

status

Examples

```
 ## Not run:
    showStatus('123')

## End(Not run)
```

status-class *Class to contain a Twitter status*

Description

Container for Twitter status messages, including the text as well as basic information

Details

The `status` class is implemented as a reference class. This class was previously implemented as an S4 class, and for backward compatibility purposes the old S4 accessor methods have been left in, although new code should not be written with these. An instance of a generator for this class is provided as a convenience to the user as it is configured to handle most standard cases. To access this generator, use the object `statusFactory`. Accessor set & get methods are provided for every field using reference class `$accessors()` methodology (see `setRefClass` for more details). As an example, the `screenName` field could be accessed using `object$getScreenName` and `object$setScreenName`.

The constructor of this object assumes that the user is passing in a JSON encoded Twitter status. It is also possible to directly pass in the arguments.

Fields

`text`: The text of the status

`screenName`: Screen name of the user who posted this status

`id`: ID of this status

`replyToSN`: Screen name of the user this is in reply to

`replyToUID`: ID of the user this was in reply to

`statusSource`: Source user agent for this tweet

`created`: When this status was created

`truncated`: Whether this status was truncated

`favorited`: Whether this status has been favorited

`retweeted`: TRUE if this status has been retweeted

`retweetCount`: The number of times this status has been retweeted

Methods

`toDataFrame`: Converts this into a one row `data.frame`, with each field representing a column. This can also be accomplished by the S4 style `as.data.frame(objectName)`.

Author(s)

Jeff Gentry

See Also

`userTimeline`, `setRefClass`

Examples

```
## Not run:
  st <- statusFactory$new(screenName="test", text="test message")
  st$getScreenName()
  st$getText()
```

```
## Assume 'json' is the return from a Twitter call
st <- statusFactory$new(json)
st$getScreenName()

## End(Not run)
```

---

taskStatus *A function to send a Twitter DM after completion of a task*

Description

This function will run an R expression and send a direct message to a specified user on success or failure.

Usage

```
taskStatus(expr, to, msg="")
```

Arguments

expr	An R expression that will be run
to	The user to send a message to, either character or an user object.
msg	An extra message to append to the standard DM

Details

This function will run expr, and send a Direct Message (DM) upon completion which will report the expression's success or failure.

Value

Either the value of the expression or an object of class try-error.

Author(s)

Jeff Gentry

See Also

dmSend

Examples

```
## Not run:
    taskStatus(z<-5, "username", session=sess)

## End(Not run)
```

timelines *Functions to view Twitter timelines*

Description

These functions will allow you to retrieve various timelines within the Twitter universe

Usage

```
userTimeline(user, n=20, maxID=NULL, sinceID=NULL, includeRts=FALSE, ...)
homeTimeline(n=25, maxID=NULL, sinceID=NULL, ...)
mentions(n=25, maxID=NULL, sinceID=NULL, ...)
retweetsOfMe(n=25, maxID=NULL, sinceID=NULL, ...)
```

Arguments

user	The Twitter user to detail, can be character or an user object.
n	Number of tweets to retrieve, up to a maximum of 3200
maxID	Maximum ID to search for
sinceID	Minimum (not inclusive) ID to search for
includeRts	If FALSE any native retweets (not old style RT retweets) will be stripped from the results
...	Optional arguments to be passed to getURL

Value

A list of status objects

Author(s)

Jeff Gentry

See Also

getUser, status, registerTwitterOAuth

Examples

```
## Not run:
      ut <- userTimeline('barackobama', n=100)

## End(Not run)
```

twListToDF *A function to convert twitteR lists to data.frames*

Description

This function will take a list of objects from a single twitteR class and return a data.frame version of the members

Usage

```
twListToDF(twList)
```

Arguments

twList	A list of objects of a single twitteR class, restrictions are listed in details

Details

The classes supported by this function are status, user, and directMessage.

Value

A data.frame with rows corresponding to the objects in the list and columns being the fields of the class

Author(s)

Jeff Gentry

See Also

status, user, directMessage

Examples

```
  ## Not run:
    zz <- searchTwitter("#rstats")
    twListToDF(zz)

## End(Not run)
```

updateStatus *Functions to manipulate Twitter status*

Description

These functions can be used to set or delete a user's Twitter status

Usage

```
tweet(text, ...)
updateStatus(text, lat=NULL, long=NULL, placeID=NULL,
             displayCoords=NULL, inReplyTo=NULL, ...)
deleteStatus(status, ...)
```

Arguments

text	The text to use for a new status
status	An object of class status
lat	If not NULL, the latitude the status refers to. Ignored if no long parameter is provideded
long	If not NULL, the longitude the status refers to. Ignored if no lat parameter is provideded
placeID	If not NULL, provideds a place in the world. See Twitter documentation for details
displayCoords	Whether or not to put a pin on the exact coordinates a tweet has been sent from, true or false if not NULL
inReplyTo	If not NULL, denotes the status this is in reply to. Either an object of class status or an ID value
...	Optional arguments to be passed to getURL

Details

These messages will only operate properly if the user is authenticated via OAuth

The tweet and updateStatus functions are the same.

To delete a status message, pass in an object of class status, such as from the return value of updateStatus.

Value

The updateStatus function will return an object of class status.

The deleteStatus returns TRUE on success and an error if failure occurs.

Author(s)

Jeff Gentry

See Also

registerTwitterOAuth

Examples

```
   ## Not run:
      ns <- updateStatus('this is my new status message')
      ## ooops, we want to remove it!
      deleteStatus(ns)

## End(Not run)
```

---

user-class *A container object to model Twitter users*

Description

This class is designed to represent a user on Twitter, modeling information available

Details

The user class is implemented as a reference class. This class was previously implemented as an S4 class, and for backward compatibility purposes the old S4 accessor methods have been left in, although new code should not be written with these. An instance of a generator for this class is provided as a convenience to the user as it is configured to handle most standard cases. To access this generator, user the object userFactory. Accessor set & get methods are provided for every field using reference class $accessors() methodology (see setRefClass for more details). As an example, the screenName field could be accessed using object$getScreenName and object$setScreenName.

The constructor of this object assumes that the user is passing in a JSON encoded Twitter user. It is also possible to directly pass in the arguments.

Fields

name: Name of the user

screenName: Screen name of the user

id: ID value for this user

lastStatus: Last status update for the user

description: User's description

statusesCount: Number of status updates this user has had

followersCount: Number of followers for this user

favoritesCount: Number of favorites for this user

friendsCount: Number of followees for this user

url: A URL associated with this user

`created`: When this user was created

`protected`: Whether or not this user is protected

`verified`: Whether or not this user is verified

`location`: Location of the user

`listedCount`: The number of times this user appears in public lists

`followRequestSent`: If authenticated via OAuth, will be TRUE if you've sent a friend request to this user

`profileImageUrl`: URL of the user's profile image, if one exists

Methods

`getFollowerIDs(n=NULL, ...)`: Will return a vector of twitter user IDs representing followers of this user, up to a maximum of `n` values. If `n` is NULL, all followers will be returned

`getFollowers(n=NULL, ...)`: Will return a list of `user` objects representing followers of this user, up to a maximum of `n` values. If `n` is NULL, all followers will be returned

`getFriendIDs(n=NULL, ...)`: Will return a vector of twitter user IDs representing users this user follows, up to a maximum of `n` values. If `n` is NULL, all friends will be returned

`getFriends(n=NULL, ...)`: Will return a list of `user` objects representing users this user follows, up to a maximum of `n` values. If `n` is NULL, all friendss will be returned

`toDataFrame(row.names=NULL, optional=FALSE)`: Converts this into a one row `data.frame`, with each field except for `lastStatus` representing a column. This can also be accomplished by the S4 style `as.data.frame(objectName)`.

Author(s)

Jeff Gentry

See Also

`status`, `setRefClass`

Examples

```
## This example is run, but likely not how you want to do things
us <- userFactory$new(screenName="test", name="Joe Smith")
us$getScreenName()
us$getName()

## Not run:
  ## Assume 'json' is the return from a Twitter call
  us <- userFactory$new(json)
  us$getScreenName()

## End(Not run)
```

Index