Package ‘twitteR’

February 15, 2012

Title R based Twitter client

Description Provides an interface to the Twitter web API

Version 0.99.18

Author Jeff Gentry <geoffjentry@gmail.com>

Maintainer Jeff Gentry <geoffjentry@gmail.com>

Depends R (>= 2.12.0), RCurl, RJSONIO, methods

Suggests ROAuth (>= 0.9.0)

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 dm.R comm.R followers.R search.R toys.R utils.R zzz.R

Repository CRAN

Date/Publication 2012-02-08 07:05:56

R topics documented:

`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

```
  ## This example is run, but likely not how you want to do things
  dm <- dmFactory$new(text='foo', recipientSN='blah')
  dm$getText()

  ## Not run:
    ## 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)
```

---

getCurRateLimitInfo *A function to retrieve current rate limit information*

Description

Will retrieve the current rate limit information. If the user is authenticated via OAuth, will retrieve information for the user account, otherwise it will do it based on the IP address

Usage

```
getCurRateLimitInfo(...)
```

Arguments

...	Optional arguments to pass to cURL

Value

An object of class rateLimitInfo

Author(s)

Jeff Gentry

See Also

`rateLimitInfo`

Examples

```
zz <- getCurRateLimitInfo()
zz$getResetTimeInSeconds()
```

`getTrends` *Functions to view Twitter trends*

Description

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

Usage

```
getTrends(period = c("daily", "weekly"), exclude = NULL, date = NULL)
```

Arguments

`period`	One of`daily`, or `weekly`, to describe the time period to acquire data for
`exclude`	If set to `hashtags`, will exclude hashtags
`date`	For periods `daily` and `weekly`, the date to use as a starting point, in the format `YYYY-MM-DD`. Note that going back too far (roughly 10-14 days) will result in an empty result

Details

The `daily` period will return the top 20 trending topics per hour for the given date

The `weekly` period will return the top 30 trending topics for each day of the week starting with `date`

Value

A list of `trend` objects

Author(s)

Jeff Gentry

See Also

`trend`

Examples

```
t1 <- getTrends()
t2 <- getTrends('weekly', as.character(Sys.Date()-1))
```

---

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

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

  ## End(Not run)
```

---

rateLimitInfo-class *Class* "rateLimitInfo"*: A class to represent rate limit information*

Description

This class provides a model representing rate limit information from the Twitter API

Details

The `rateLimitInfo` class is implemented as a reference class. The only S4 method provided is a `show` method. 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 `rateLimitInfoFactory`. Accessor set & get methods are provided for every field using reference class `$accessors()` methodology (see `setRefClass` for more details). As an example, the `hourlyLimit` field could be accessed using `object$getHourlyLimit()` and `object$setHourlyLImit()`.

Fields

`remainingHits`**:** Number of remaining API calls before rate limit

`resetTimeInSeconds`**:** Length of time before the rate limit counter resets

`hourlyLimt`**:** Number of API calls allowed per hour

`resetTime`**:** Date of the rate limit reset

Methods

show `signature(object = "rateLimitInfo")`: displays the remaining hits for this address

Author(s)

Jeff Gentry

See Also

`getCurRateLimitInfo`, `setRefClass`

Examples

```
   zz <- getCurRateLimitInfo()
   zz$getRemainingHits()
```

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

```
registerTwitterOAuth(oauth)
```

Arguments

oauth	An object of class OAuth

Details

This function 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
   requestURL <- "https://api.twitter.com/oauth/request_token"
   accessURL = "http://api.twitter.com/oauth/access_token"
   authURL = "http://api.twitter.com/oauth/authorize"
   consumerKey = "FAKEDATA"
   consumerSecret = "FAKEDATA"
   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, ...)
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
`...`	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

```
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')
```

---

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

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

`publicTimeline`, `userTimeline`, `setRefClass`

Examples

```
   ## This example is run, but likely not how you want to do things
   st <- statusFactory$new(screenName="test", text="test message")
   st$getScreenName()
   st$getText()

   ## Not run:
     ## 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

```
publicTimeline(...)
userTimeline(user, n=20, maxID=NULL, sinceID=NULL, ...)
homeTimeline(n=25, maxID=NULL, sinceID=NULL, ...)
mentions(n=25, maxID=NULL, sinceID=NULL, ...)
retweetedByMe(n=25, maxID=NULL, sinceID=NULL, ...)
retweetedToMe(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
`...`	Optional arguments to be passed to `getURL`

Details

The `publicTimeline` function will return a current snapshot of the public timeline.

The `userTimeline` function will only work if the user requested has a public timeline, or you have previously registered a `OAuth` object using `registerTwitterOAuth` and are authorized to view that content.

The other functions will provide various views into timelines available to the user. They all require authentication via `OAuth`.

Value

A list of `status` objects

Author(s)

Jeff Gentry

See Also

`getUser`, `status`, `registerTwitterOAuth`

Examples

```
pt <- publicTimeline()
pt
ut <- userTimeline('barackobama', n=100)
```

`trend-class` *Class "trend": A class to represent Twitter trends*

Description

Provides a model representing trends from Twitter

Details

The `trend` 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 `trendFactory`. Accessor set & get methods are provided for every field using reference class `$accessors()` methodology (see `setRefClass` for more details). As an example, the `date` field could be accessed using `object$getDate()` and `object$setDate()`.

Fields

`name`**:** Name of the trend

`date`**:** `POSIXct` representation of the date associated with this trend

`promoted_content`**:** Logical, if this was promoted by Twitter

`events`**:** Appears to be unused by Twitter

`woeid`**:** Yahoo based location code, currently unimplemented

`country`**:** Country associated with the trend, currently unimplemented

`countryCode`**:** Country code associated with the trend, currently unimplemented

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

`getTrends`, `setRefClass`

Examples

```
## It is suggested that one does not call the constructor directly,
## but if desired this is the appropriate structure
zz <- trendFactory$new(name='foo', date=Sys.Date())
zz$getName()

## Instead, use getTrends:
xx <- getTrends()
xx[[1]]$getName()
```

`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`, `directMessage`, `rateLimitInfo`, and `trend`.

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`, `rateLimitInfo`, `trend`

Examples

```
  zz <- searchTwitter("#rstats")
  twListToDF(zz)
```

`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

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