# Package 'twitteR'

October 14, 2022

**Title** R Based Twitter Client

**Description** Provides an interface to the Twitter web API.

**Version** 1.1.9

**Author** Jeff Gentry <geoffjentry@gmail.com>

**Maintainer** Jeff Gentry <geoffjentry@gmail.com>

**Depends** R (>= 2.12.0)

**Imports** methods, bit64, rjson, DBI (>= 0.3.1), httr (>= 1.0.0)

**Suggests** RSQLite, RMySQL

**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 oauth.R comm.R followers.R search.R
db.R df_columns.R db_connections.R db_utils.R db_search.R
toys.R utils.R zzz.R

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-07-29 00:27:59

## R topics documented:


decode_short_url . . . . . 2
directMessage-class . . . . . 3
dmGet . . . . . 4
favorites . . . . . 5
friendships . . . . . 6
getCurRateLimitInfo . . . . . 7
getTrends . . . . . 8
getUser . . . . . 9
get_latest_tweet_id . . . . . 10
import_statuses . . . . . 11

load_tweets_db . . . . . 12
registerTwitterOAuth . . . . . 13
register_db_backend . . . . . 14
retweets . . . . . 15
searchTwitter . . . . . 16
search_twitter_and_store . . . . . 18
setup_twitter_oauth . . . . . 19
showStatus . . . . . 20
status-class . . . . . 21
strip_retweets . . . . . 22
taskStatus . . . . . 23
timelines . . . . . 24
twListToDF . . . . . 25
updateStatus . . . . . 26
user-class . . . . . 27
use_oauth_token . . . . . 29

**Index** **30**

---

`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/rest/reference/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=resource_families, ...)
```

## Arguments

| | |
|---|---|
| `resources` | A character vector of specific resources to get information for |
| `...` | Optional arguments to pass to cURL |

## Details

By default, all known resource families will be polled. These families are contained in the object `resource_families`. If you would like to filter this down you may tweak the `resources` argument.

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 GET |

## 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)
```

---

# get_latest_tweet_id *A function to retrieve the most recent tweet ID from a database*

---

## Description

Given a registered database backend which contains a table of tweets, will return the ID of the most recent tweet stored in that table

## Usage

```
get_latest_tweet_id(table_name = "tweets")
```

## Arguments

| | |
|---|---|
| `table_name` | The name of the table in the database containing tweets |

## Details

A wrapper around a `select max(id)` on the `table_name`

## Value

The ID of the most recent tweet in the table, or a `stop` if the table is empty

## Author(s)

Jeff Gentry

## See Also

register_db_backend

## Examples

```
## Not run:
   register_sqlite_backend("sqlit_file")
   get_latest_tweet_id("rstats_tweets")

## 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)
```

---

load_tweets_db *Functions to persist/load twitteR data to a database*

---

## Description

These functions allow a user to store twitteR based data to a database backend as well as retrieving previously stored data

## Usage

```
store_tweets_db(tweets, table_name="tweets")
store_users_db(users, table_name="users")
load_users_db(as.data.frame = FALSE, table_name = "users")
load_tweets_db(as.data.frame = FALSE, table_name = "tweets")
```

## Arguments

| | |
|---|---|
| tweets | A list of status objects to persist to the database |
| users | A list of user objects to persist to the database |
| as.data.frame | if TRUE, data will be returned as a data.frame instead of twitteR objects |
| table_name | The database table to use for storing and loading |

## Value

store_tweets_db and store_users_db return TRUE of FALSE based on their success or not. The loading functions return either a data.frame of the data (representing the underlying table) or a list of the appropriate twitteR objects.

## Author(s)

Jeff Gentry

## See Also

register_db_backend, register_sqlite_backend, register_mysql_backend

## Examples

```
## Not run:
 register_sqlite_backend("/path/to/sqlite/file")
 tweets = searchTwitter("#scala")
 store_tweets_db(tweets)
 from_db = load_tweets_db()

## End(Not run)
```

---

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

---

## Description

These functions are deprecated

## 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

These functions are deprecated, see `setup_twitter_oauth`

## Value

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

## Author(s)

Jeff Gentry

## See Also

`setup_twitter_oauth`

## Examples

```
## Not run:
   fakeExample = 5

## End(Not run)
```

register_db_backend *Functions to setup a database backend for twitteR*

## Description

twitteR can have a database backend registered from which to store and load tweet and user data. These functions provide mechanisms for setting up the connection within twitteR

## Usage

```
register_db_backend(db_handle)
register_sqlite_backend(sqlite_file, ...)
register_mysql_backend(db_name, host, user, password, ...)
```

## Arguments

| | |
|---|---|
| db_handle | A DBI connection |
| sqlite_file | File path for a SQLite file |
| db_name | Name of the database to connect to |
| host | Hostname the database is on |
| user | username to connect to the database with |
| password | password to connect to the database with |
| ... | extra arguments to pass to dbConnect |

## Details

Currently only RSQLite and RMySQL are supported. To use either of these DBI implementations the appropriate packages will need to be installed.

The register_sqlite_backend and register_mysql_backend are convenience wrappers to both create the DBI connection and call register_db_backend for you.

## Value

The DBI connection, invisibly

## Author(s)

Jeff Gentry

## See Also

store_tweets_db, store_users_db, load_tweets_db, load_users_db

## Examples

```
## Not run:
 register_sqlite_backend("/path/to/sqlite/file")
 tweets = searchTwitter("#scala")
 store_tweets_db(tweets)
 from_db = load_tweets_db()

## End(Not run)
```

---

`retweets` *Functions to work with retweets*

---

## Description

These functions can be used to return retweets or users who retweeted a tweet

## Usage

```
retweets(id, n = 20, ...)
```

## Arguments

| | |
|---|---|
| `id` | The ID of the tweet to get retweet information on |
| `n` | The number of results to return, up to 100 |
| `...` | Further arguments to pass on to httr |

## Value

For `retweets` the n most recent retweets of the original tweet.

For `retweeters` the n most recent users who have retweeted this tweet.

## Author(s)

Jeff Gentry

## See Also

`showStatus`

## Examples

```
## Not run:
   retweets("21947795900469248")

   st = showStatus("21947795900469248")
   retweeters(st$getId())

## 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, maxID=NULL,
      resultType=NULL, retryOnRateLimit=120, ...)
Rtweets(n=25, lang=NULL, since=NULL, ...)
```

## Arguments

| | |
|---|---|
| searchString | Search query to issue to twitter. Use "+" to separate query terms. |
| 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 |
| maxID | If not NULL, returns tweets with IDs smaller (ie older) than the specified ID |
| resultType | If not NULL, returns filtered tweets as per value. See details for allowed values. |
| 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 GET |

## 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.

The searchString is always required. Terms can contain spaces, and multiple terms should be separated with "+".

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.

For the maxID argument, tweets upto this ID value will be returned starting from the oldest ID available. Useful for paging.

The resultType argument specifies the type of search results received in API response. Default is mixed. Allowed values are mixed (includes popular + real time results), recent (returns the most recent results) and popular (returns only the most popular results).

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

## 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')

  ## using resultType
  searchTwitter('world cup+brazil', resultType="popular", n=15)
  searchTwitter('from:hadleywickham', resultType="recent", n=10)


## End(Not run)
```

search_twitter_and_store *A function to store searched tweets to a database*

## Description

A convenience function designed to wrap the process of running a twitter search and pushing the results to a database. If this is called more than once, the search will start with the most recent tweet already stored.

## Usage

```
search_twitter_and_store(searchString, table_name = "tweets", lang = NULL,
  locale = NULL, geocode = NULL, retryOnRateLimit = 120, ...)
```

## Arguments

| | |
|---|---|
| searchString | The search string to use, e.g. as one would in searchTwitter |
| table_name | The database to store the tweets to, see register_db_backend |
| lang | If not NULL, restricts tweets to the given language, given by an ISO 639-1 code |
| 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 in link{searchTwitter} |
| 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 GET |

## Details

All arguments but table_name are being passed directly to searchTwitter.

This function will check if table_name exists, and if so will also use a sinceID of the most recent ID in the table. The search is performed, the returned tweets are stored in the database via store_tweets_db.

## Value

The number of tweets stored

## Note

Jeff Gentry

## See Also

register_db_backend, searchTwitter, store_tweets_db

## Examples

```
  ## Not run:
       register_sqlite_backend("sqlit_file")
       n = search_twitter_and_store("#rstats", "rstats_tweets")

## End(Not run)
```

---

# setup_twitter_oauth *Sets up the OAuth credentials for a twitteR session*

---

## Description

This function wraps the OAuth authentication handshake functions from the httr package for a twitteR session

## Usage

```
setup_twitter_oauth(consumer_key, consumer_secret, access_token=NULL, access_secret=NULL)
```

## Arguments

| | |
|---|---|
| consumer_key | The consumer key supplied by Twitter |
| consumer_secret | The consumer secret supplied by Twitter |
| access_token | The access token supplied by Twitter |
| access_secret | The access secret supplied by Twitter |

## Details

The httr package can cache authentication. See Token for details

If both access_token and access_secret are set (i.e. not NULL), these will be supplied directly to the OAuth authentication instead of the browser based authentication dance one would normally experience. This requires you to already know the access tokens for your Twitter app. The usefuleness of this feature is primarily in a headless environment where a web browser is not available.

## Value

This is called for its side effect

## Author(s)

Jeff Gentry

## See Also

Token, GET, POST

## Examples

```
 ## Not run:
    setup_twitter_oauth("CONSUMER_KEY", "CONSUMER_SECRET")

## End(Not run)
```

---

showStatus *Functions to return statuses*

---

## Description

These functions can be used to retrieve specific tweets from the server

## Usage

```
showStatus(id, ...)
lookup_statuses(ids, ...)
```

## Arguments

| | |
|---|---|
| id | ID of a specific tweet, should be a String, but numbers are accepted |
| ids | A vector of IDs to lookup, should be Strings but numbers are accepted |
| ... | Optional arguments to be passed to GET (or POST, see Details) |

## Details

Ideally a POST request would be used for lookup_statuses, however currently there is a problem (issue 78 on github) and GET is used.

## Value

For showStatus, an object of class status

For lookup_statuses, a list of status objects. Note that these will not be in the same order as the ids argument and that any id which could not be retrieved will not be present.

## Author(s)

Jeff Gentry

## See Also

status

## Examples

```
 ## Not run:
    showStatus('123')
    lookup_statuses(c("123", "234", "456"))

## 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)
```

---

## strip_retweets *A function to remove retweets*

---

### Description

Given a list of status objects, will remove retweets from the list to provide a "pure" set of tweets.

### Usage

```
strip_retweets(tweets, strip_manual = TRUE, strip_mt = TRUE)
```

### Arguments

| | |
|---|---|
| tweets | A list of status objects |
| strip_manual | If TRUE will remove old style manual retweets |
| strip_mt | If TRUE will remove modified tweets (MT) |

### Details

Newer style retweets are summarily removed regardless of options.

Older style retweets (aka manual retweets) are tweets of the form RT @user blah blah. If strip_manual is TRUE, tweets containing the RT string will have everything including and to the right of the RT will be removed. Everything to the left of the RT will remain, as this should be original content.

If strip_mt is TRUE, tweets will be stripped in the same manner as strip_manual but using the string MT

## Value

A list of `status` objects with retweeted content removed

## Author(s)

Jeff Gentry

## See Also

`status`

## Examples

```
## Not run:
  tweets = searchTwitter("stuff")
  no_retweets = strip_retweets(tweets)

## 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,
  excludeReplies=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 |
| excludeReplies | if TRUE any replies are stripped from the results |
| ... | Optional arguments to be passed to GET |

### Value

A list of status objects

### Author(s)

Jeff Gentry

## See Also

getUser, status

## 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, mediaPath=NULL,
             bypassCharLimit=FALSE, ...)
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 |
| mediaPath | If not NULL, file path to a supported media format (PNG, JPG and GIF) to be included in the status update |
| bypassCharLimit | If TRUE will not enforce the incoming tweet is less than 140 characters. This can be useful when dealing with autoshortened links |
| ... | Optional arguments to be passed to GET |

## 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

## 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)
```

use_oauth_token *Sets up the OAuth credentials for a twitteR session from an existing Token object*

## Description

This function uses an existing httr OAuth Token in the Twitter session

## Usage

```
use_oauth_token(twitter_token)
```

## Arguments

| | |
|---|---|
| twitter_token | An httr Token object |

## Details

This function is an escape hatch for nonstandard OAuth scenarios. Use setup_twitter_token unless it doesn't work for your use case.

## Value

This is called for its side effect

## Author(s)

Anand Patil

## See Also

Token

## Examples

```
## Not run:
   library(httr)
   library(twitteR)
   token <- Token2.0$new(
     params = list(as_header=TRUE),
     app = oauth_app("fun.with.twitter", "no.key", "no.secret"),
     endpoint = oauth_endpoints("twitter"),
     credentials = list(access_token = "AAAAAAAAAAAAAAAAAAA%3DAAAAAAAAAAAAAA"),
     cache = FALSE
   )

   use_oauth_token(token)

## End(Not run)
```

# Index


∗ **classes**
directMessage-class, 3
status-class, 21
user-class, 27
∗ **interface**
dmGet, 4
favorites, 5
friendships, 6
getCurRateLimitInfo, 7
getTrends, 8
getUser, 9
import_statuses, 11
registerTwitterOAuth, 13
searchTwitter, 16
setup_twitter_oauth, 19
showStatus, 20
taskStatus, 23
timelines, 24
twListToDF, 25
updateStatus, 26
use_oauth_token, 29
∗ **utilities**
decode_short_url, 2
get_latest_tweet_id, 10
load_tweets_db, 12
register_db_backend, 14
search_twitter_and_store, 18
strip_retweets, 22
[[,twitterObjList-method *(status-class)*, 21

as.data.frame,status-method *(status-class)*, 21
as.data.frame,twitterObj-method *(status-class)*, 21
as.data.frame,user-method *(user-class)*, 27
availableTrendLocations *(getTrends)*, 8

buildStatus *(status-class)*, 21
buildUser *(user-class)*, 27

closestTrendLocations *(getTrends)*, 8
created *(user-class)*, 27
created,status-method *(status-class)*, 21
created,user-method *(user-class)*, 27

data.frame, *3*, *21*, *25*, *28*
decode_short_url, 2
deleteStatus *(updateStatus)*, 26
description *(user-class)*, 27
description,user-method *(user-class)*, 27
directMessage, *4, 5*, *25*
directMessage *(directMessage-class)*, 3
directMessage-class, 3
dmDestroy, *3*, *4*
dmDestroy *(dmGet)*, 4
dmFactory *(directMessage-class)*, 3
dmGet, *4*, 4
dmSend, *4*, *24*
dmSend *(dmGet)*, 4
dmSent *(dmGet)*, 4

favorited *(status-class)*, 21
favorited,status-method *(status-class)*, 21
favorites, 5
favoritesCount *(user-class)*, 27
favoritesCount,user-method *(user-class)*, 27
followersCount *(user-class)*, 27
followersCount,user-method *(user-class)*, 27
followRequestSent *(user-class)*, 27
followRequestSent,user-method *(user-class)*, 27
friendsCount *(user-class)*, 27
friendsCount,user-method *(user-class)*, 27
friendships, 6

GET, *9*, *16*, *18*, *20*, *24*, *26*
get_latest_tweet_id, 10
getCurRateLimitInfo, 7
getTrends, 8
getTwitterOAuth *(*registerTwitterOAuth*)*, 13
getUser, *6*, 9, *25*

homeTimeline *(*timelines*)*, 24

id *(*status-class*)*, 21
id,status-method *(*status-class*)*, 21
id,user-method *(*user-class*)*, 27
import_obj *(*import_statuses*)*, 11
import_statuses, 11
import_trends *(*import_statuses*)*, 11
import_users *(*import_statuses*)*, 11

json_to_statuses *(*import_statuses*)*, 11
json_to_trends *(*import_statuses*)*, 11
json_to_users *(*import_statuses*)*, 11

lastStatus *(*user-class*)*, 27
lastStatus,user-method *(*user-class*)*, 27
listedCount *(*user-class*)*, 27
listedCount,user-method *(*user-class*)*, 27
load_tweets_db, 12, *14*
load_users_db, *14*
load_users_db *(*load_tweets_db*)*, 12
location *(*user-class*)*, 27
location,user-method *(*user-class*)*, 27
lookup_statuses *(*showStatus*)*, 20
lookupUsers *(*getUser*)*, 9

mentions, *10*
mentions *(*timelines*)*, 24

name *(*user-class*)*, 27
name,user-method *(*user-class*)*, 27

POST, *20*
profileImageUrl *(*user-class*)*, 27
profileImageUrl,user-method *(*user-class*)*, 27
protected *(*user-class*)*, 27
protected,user-method *(*user-class*)*, 27

register_db_backend, *11*, *12*, 14, *18*, *19*
register_mysql_backend, *12*
register_mysql_backend *(*register_db_backend*)*, 14
register_sqlite_backend, *12*
register_sqlite_backend *(*register_db_backend*)*, 14
registerTwitterOAuth, *5*, *7*, 13, *16*
replyToSID *(*status-class*)*, 21
replyToSID,status-method *(*status-class*)*, 21
replyToSN *(*status-class*)*, 21
replyToSN,status-method *(*status-class*)*, 21
replyToUID *(*status-class*)*, 21
replyToUID,status-method *(*status-class*)*, 21
resource_families *(*getCurRateLimitInfo*)*, 7
retweetCount *(*status-class*)*, 21
retweetCount,status-method *(*status-class*)*, 21
retweeted *(*status-class*)*, 21
retweeted,status-method *(*status-class*)*, 21
retweeters *(*retweets*)*, 15
retweets, 15
retweetsOfMe *(*timelines*)*, 24
Rtweets *(*searchTwitter*)*, 16

screenName *(*user-class*)*, 27
screenName,status-method *(*status-class*)*, 21
screenName,user-method *(*user-class*)*, 27
search_twitter_and_store, 18
searchTwitteR *(*searchTwitter*)*, 16
searchTwitter, 16, *18*, *19*
setRefClass, *3*, *4*, *21*, *22*, *27*, *28*
setup_twitter_oauth, *13*, 19
show,directMessage-method *(*directMessage-class*)*, 3
show,status-method *(*status-class*)*, 21
show,twitterObjList-method *(*status-class*)*, 21
show,user-method *(*user-class*)*, 27
showStatus, *15*, 20
status, *6*, *11*, *17*, *20*, *22–28*
status *(*status-class*)*, 21
status-class, 21
statusesCount *(*user-class*)*, 27

statusesCount,user-method *(user-class)*, 27
statusFactory *(status-class)*, 21
statusSource *(status-class)*, 21
statusSource,status-method *(status-class)*, 21
statusText *(status-class)*, 21
statusText,status-method *(status-class)*, 21
stop, *10*
store_tweets_db, *14*, *18, 19*
store_tweets_db *(load_tweets_db)*, 12
store_users_db, *14*
store_users_db *(load_tweets_db)*, 12
strip_retweets, 22

taskStatus, 23
text,status-method *(status-class)*, 21
timelines, 24
Token, *19, 20*, *29*
truncated *(status-class)*, 21
truncated,status-method *(status-class)*, 21
tweet *(updateStatus)*, 26
tweetCount *(user-class)*, 27
tweetCount,user-method *(user-class)*, 27
twListToDF, 25

updateStatus, 26
use_oauth_token, 29
user, *4, 5*, *9–11*, *23–25*
user *(user-class)*, 27
user-class, 27
userFactory *(user-class)*, 27
userTimeline, *22*
userTimeline *(timelines)*, 24
userURL *(user-class)*, 27
userURL,user-method *(user-class)*, 27

verified *(user-class)*, 27
verified,user-method *(user-class)*, 27