Package ‘rtweet’

July 21, 2022

Type Package

Title Collecting Twitter Data

Version 1.0.2

Description An implementation of calls designed to collect and organize Twitter data via Twitter's REST and stream Application Program Interfaces (API), which can be found at the following URL: <https://developer.twitter.com/en/docs>.

License MIT + file LICENSE

URL https://docs.ropensci.org/rtweet/,
https://github.com/ropensci/rtweet/

BugReports https://github.com/ropensci/rtweet/issues

Depends R (>= 4.0.0)

Imports bit64 (>= 4.0.5), curl (>= 4.3.2), httr (>= 1.3.0), jsonlite (>= 0.9.22), lifecycle (>= 1.0.0), methods, progress (>= 1.2.2), rlang (>= 0.4.10), tibble (>= 1.3.4), utils, withr (>= 2.5.0)

Suggests askpass (>= 1.1), covr (>= 3.5.1), dplyr (>= 1.0.9), ggplot2 (>= 3.3.5), httpuv (>= 1.6.5), igraph (>= 1.3.2), knitr (>= 1.39), magick (>= 2.7.3), maps (>= 3.4.0), openssl (>= 2.0.2), rappdirs (>= 0.3.3), rmarkdown (>= 2.14), testthat (>= 3.1.0), vcr (>= 0.6.0), webshot (>= 0.5.3)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Michael W. Kearney [aut] (<https://orcid.org/0000-0002-0730-4694>),
Lluís Revilla Sancho [aut, cre]
(<https://orcid.org/0000-0001-9747-2570>),
Hadley Wickham [aut] (<https://orcid.org/0000-0003-4757-117X>),

Andrew Heiss [rev] (<https://orcid.org/0000-0002-3948-3914>),
Francois Briatte [rev],
Jonathan Sidi [ctb] (<https://orcid.org/0000-0002-4222-1819>)

Maintainer Lluís Revilla Sancho <lluis.revilla@gmail.com>

Repository CRAN

Date/Publication 2022-07-21 09:30:02 UTC

R topics documented:

as_screenname . . . 3
auth_as . . . 4
auth_get . . . 5
auth_save . . . 6
auth_setup_default . . . 7
auth_sitrep . . . 7
direct_messages . . . 8
do_call_rbind . . . 10
emojis . . . 11
flatten . . . 11
get_favorites . . . 12
get_followers . . . 14
get_friends . . . 15
get_mentions . . . 17
get_retweets . . . 19
get_timeline . . . 20
get_token . . . 22
get_trends . . . 22
langs . . . 24
lat_lng . . . 24
lists_members . . . 26
lists_memberships . . . 27
lists_statuses . . . 29
lists_subscribers . . . 31
lists_subscriptions . . . 32
lists_users . . . 34
lookup_coords . . . 35
lookup_friendships . . . 36
lookup_tweets . . . 37
lookup_users . . . 38
my_friendships . . . 39
network_data . . . 40
parse_stream . . . 41
plain_tweets . . . 42
post_destroy . . . 42
post_favorite . . . 43
post_follow . . . 44
post_friendship . . . 45

post_list 46
post_message 48
post_tweet 49
rate_limit 51
read_twitter_csv 52
round_time 53
rtweet_user 53
search_fullarchive 55
search_tweets 58
search_users 61
stopwordslangs 63
stream_tweets 63
trends_available 65
ts_data 66
ts_plot 67
tweets_with_users 68
tweet_embed 68
tweet_shot 69
tweet_threading 70
users_data 71
user_block 72
write_as_csv 72

Index **74**

`as_screenname` *Mark a user id as a screen name*

Description

There are two ways to identify a Twitter user: a screen name (e.g. "justinbieber") or a user identifier (e.g. "27260086"). User identifiers look like regular numbers, but are actually 64-bit integers which can't be stored in R's numeric vectors. For this reason, rtweet always returns ids as strings.

Unfortunately this introduces an ambiguity, because user names can also consist solely of numbers (e.g. "123456") so it's not obvious whether a string consisting only of numbers is a screen name or a user id. By default, rtweet will assume its a user id, so if you have a screen name that consists only of numbers, you'll need to use `as_screenname()` to tell rtweet that it's actually a screen name.

Note that in general, you are best off using user ids; screen names are not static and may change over longer periods of time.

Usage

```
as_screenname(x)
```

Arguments

`x`	A character vector of Twitter screen names.

See Also

Other users: lists_subscribers(), lookup_users(), search_users()

Examples

```
if (auth_has_default()) {
# Look up user with id
lookup_users("25594077")

# Look up user with name 25594077
lookup_users(as_screenname("123456"))
}
```

---

auth_as *Set default authentication for the current session*

Description

auth_as() sets up the default authentication mechanism used by all rtweet API calls. See rtweet_user() to learn more about the three available authentication options.

Usage

```
auth_as(auth = NULL)
```

Arguments

auth	One of the following options: • NULL, the default, will look for rtweet's "default" authentication which uses your personal Twitter account. If it's not found, it will call auth_setup_default() to set it up. • A string giving the name of a saved auth file made by auth_save(). • An auth object created by rtweet_app(), rtweet_bot(), or rtweet_user().

Value

Invisibly returns the previous authentication mechanism.

See Also

auth_sitrep() to help finding and managing authentications.

Other authentication: auth_get(), auth_save(), auth_setup_default(), rtweet_user()

Examples

```
## Not run:
# Use app auth for the remainder of this session:
my_app <- rtweet_app()
auth_as(my_app)

# Switch back to the default user based auth
auth_as()

# Load auth saved by auth_save()
auth_as("my-saved-app")

## End(Not run)
```

---

auth_get — *Get the current authentication mechanism*

Description

If no authentication has been set up for this session, `auth_get()` will call `auth_as()` to set it up.

Usage

```
auth_get()
```

Value

The current token used.

See Also

Other authentication: `auth_as()`, `auth_save()`, `auth_setup_default()`, `rtweet_user()`

Examples

```
## Not run:
auth_get()

## End(Not run)
```

auth_save *Save an authentication mechanism for use in a future session*

Description

Use auth_save() with auth_as() to avoid repeatedly entering app credentials, making it easier to share auth between projects. Use auth_list() to list all saved credentials.

Usage

```
auth_save(auth, name)

auth_list()
```

Arguments

auth	One of rtweet_app(), rtweet_bot(), or rtweet_user().
name	Name of the file to use.

Details

The tokens are saved on tools::R_user_dir("rtweet", "config").

Value

Invisible the path where the authentication is saved.

See Also

auth_sitrep() to help finding and managing authentications.

Other authentication: auth_as(), auth_get(), auth_setup_default(), rtweet_user()

Examples

```
## Not run:
# save app auth for use in other sessions
auth <- rtweet_app()
auth_save(auth, "my-app")

# later, in a different session...
auth_as("my-app")
# Show all authentications stored
auth_list()

## End(Not run)
```

auth_setup_default — *Set up default authentication*

Description

You'll need to run this function once per computer so that rtweet can use your personal Twitter account. See `rtweet_app()`/rtweet_bot and `auth_save()` for other authentication options.

Usage

```
auth_setup_default()

auth_has_default()
```

Details

It will use the current logged in account on the default browser to detect the credentials needed for rtweet and save them as "default". If a default is found it will use it instead.

Value

`auth_setup_default()`: Invisibly returns the previous authentication mechanism. `auth_has_default()`: A logical value `TRUE` if there is a default authentication.

See Also

Other authentication: `auth_as()`, `auth_get()`, `auth_save()`, `rtweet_user()`

Examples

```
## Not run:
if (!auth_has_default() && interactive()) {
   auth_setup_default()
}

## End(Not run)
```

auth_sitrep — *Twitter Tokens sitrep*

Description

Get a situation report of your current tokens; useful for upgrading from rtweet 0.7.0 to 1.0.0 and diagnosing problems with tokens.

Usage

```
auth_sitrep()
```

Details

Prints rtweet tokens on the old folder (rtweet < 0.7.0) and on the new (rtweet > 1.0.0) default location. For each folder it reports apps and then users and bots authentications. For users authentications it reports the user_id, so that you can check who is that user.

Users should follow its advise, if there is no advise but there are still some problems authenticating regenerate the authentications.

Value

Invisibly, TRUE if some problems were found and FALSE otherwise

Note

It is safe to use in public, as instead of the tokens or keys it reports a letter.

See Also

`auth_as()`

Examples

```
auth_sitrep()
```

---

direct_messages *Get direct messages sent to and received by the authenticating user from the past 30 days*

Description

Returns all Direct Message events (both sent and received) within the last 30 days. Sorted in reverse-chronological order. Includes detailed information about the sender and recipient.

Usage

```
direct_messages(
  n = 50,
  cursor = NULL,
  next_cursor = NULL,
  parse = TRUE,
  token = NULL,
  retryonratelimit = NULL,
  verbose = TRUE
)
```

Arguments

`n` Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`cursor` Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

`next_cursor` **[Deprecated]** Use `cursor` instead.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

`retryonratelimit`
If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

`verbose` Show progress bars and other messages indicating current progress?

Value

A list with one element for each page of results.

References

https://developer.twitter.com/en/docs/twitter-api/v1/direct-messages/sending-and-receiving/api-reference/list-events

Examples

```
## Not run:

## get my direct messages
dms <- direct_messages()
```

```
## inspect data structure
str(dms)


## End(Not run)
```

---

do_call_rbind — *Binds list of data frames while preserving attribute (tweets or users) data.*

Description

Row bind lists of tweets/users data whilst also preserving and binding users/tweets attribute data.

Usage

```
do_call_rbind(x)
```

Arguments

x	List of parsed tweets data or users data, each of which presumably contains an attribute of the other (i.e., users data contains tweets attribute; tweets data contains users attribute).

Value

A single merged (by row) data frame (tbl) of tweets or users data that also contains as an attribute a merged (by row) data frame (tbl) of its counterpart, making it accessible via the `users_data()` or `tweets_data()` extractor functions.

Examples

```
if (auth_has_default()) {

## lapply through three different search queries
lrt <- lapply(
  c("rstats OR tidyverse", "data science", "python"),
  search_tweets,
  n = 100
)

## convert list object into single parsed data rame
rt <- do_call_rbind(lrt)

## preview tweets data
rt

## preview users data
users_data(rt)
```

```
}
```

---

emojis *Defunct: Emojis codes and descriptions data.*

Description

This data comes from "Unicode.org", http://unicode.org/emoji/charts/full-emoji-list.html. The data are codes and descriptions of Emojis.

Format

A tibble with two variables and 2,623 observations.

---

flatten *flatten/unflatten data frame*

Description

Converts list columns that containing all atomic elements into character vectors and vice versa (for appropriate named variables according to the rtweet package)

Usage

```
flatten(x)

unflatten(x)
```

Arguments

x	Data frame with list columns or converted-to-character (flattened) columns.

Details

If recursive list columns are contained within the data frame, relevant columns will still be converted to atomic types but output will also be accompanied with a warning message.

`flatten` flattens list columns by pasting them into a single string for each observations. For example, a tweet that mentions four other users, for the mentions_user_id variable, it will include the four user IDs separated by a space.

‘unflatten“ splits on spaces to convert into list columns any columns with the following names: hashtags, symbols, urls_url, urls_t.co, urls_expanded_url, media_url, media_t.co, media_expanded_url, media_type, ext_media_url, ext_media_t.co, ext_media_expanded_url, mentions_user_id, mentions_screen_name, geo_coords, coords_coords, bbox_coords, mentions_screen_name

Value

If flattened, then data frame where non-recursive list columns—that is, list columns that contain only atomic, or non-list, elements—have been converted to character vectors. If unflattened, this function splits on spaces columns originally returned as lists by functions in rtweet package. See details for more information.

See Also

Other datafiles: `read_twitter_csv()`, `write_as_csv()`

Other datafiles: `read_twitter_csv()`, `write_as_csv()`

`get_favorites` *Get tweets liked/favorited by one or more users*

Description

Returns up to 3,000 tweets liked/favorited for each `user`.

Usage

```
get_favorites(
  user,
  n = 200,
  since_id = NULL,
  max_id = NULL,
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL
)
```

Arguments

`user`	Character vector of screen names or user ids. See `as_screenname()` for more details.
`n`	Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible. The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`. You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

since_id
: Supply a vector of ids or a data frame of previous results to find tweets **newer** than since_id.

max_id
: Supply a vector of ids or a data frame of previous results to find tweets **older** than max_id.

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose
: Show progress bars and other messages indicating current progress?

token
: Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

Value

A tibble with one row for each tweet.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-favorites-list

See Also

Other tweets: get_mentions(), get_timeline(), lists_statuses(), lookup_tweets(), search_tweets()

Examples

```
if (auth_has_default()) {
# get likes for a single user
kfc <- get_favorites("KFC")
kfc
# get newer likes since last request
newer <- get_favorites("KFC", since_id = kfc)

# get likes from multiple users
favs <- get_favorites(c("Lesdoggg", "pattonoswalt", "meganamram"))
favs
}
```

get_followers *Get user IDs for accounts following target user.*

Description

Returns a list of user IDs for the accounts following specified user.

Usage

```
get_followers(
  user,
  n = 5000,
  cursor = "-1",
  retryonratelimit = NULL,
  parse = TRUE,
  verbose = TRUE,
  token = NULL,
  page = lifecycle::deprecated()
)
```

Arguments

`user` Character vector of screen names or user ids. See `as_screenname()` for more details.

`n` Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`cursor` Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

`retryonratelimit` If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

parse	If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.
verbose	Show progress bars and other messages indicating current progress?
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.
page	**[Deprecated]** Please use cursor instead.

Value

A tibble data frame with one column named "from_id" with the followers and another one "to_id" with the user used as input.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-followers-ids

Examples

```
if (auth_has_default()) {
users <- get_followers("_R_Foundation")
users

# use `cursor` to find the next "page" of results
more_users <- get_followers("_R_Foundation", cursor = users)

}
```

---

get_friends *Get user IDs of accounts followed by target user(s).*

Description

Returns a list of user IDs for the accounts following BY one or more specified users.

Usage

```
get_friends(
  users,
  n = 5000,
  retryonratelimit = NULL,
  cursor = "-1",
  parse = TRUE,
  verbose = TRUE,
  token = NULL,
  page = lifecycle::deprecated()
)
```

Arguments

users — Screen name or user ID of target user from which the user IDs of friends (accounts followed BY target user) will be retrieved.

n — Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

retryonratelimit — If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

cursor — Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

parse — If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

verbose — Show progress bars and other messages indicating current progress?

token — Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

page — **[Deprecated]** Please use `cursor` instead.

Details

Generally, you should not need to set `n` to more than 5,000 since Twitter limits the number of people that you can follow (i.e. to follow more than 5,000 people at least 5,000 people need to follow you).

Value

A tibble data frame with two columns, "from_id" for name or ID of target user and "to_id" for accounts ID they follow.

Note

If a user is protected the API will omit all requests so you'll need to find which user is protected. rtweet will warn you and the output will be `NA`.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-friends-ids

Examples

```
if (auth_has_default()) {
users <- get_friends("ropensci")
users
}
```

`get_mentions` *Get mentions for the authenticating user.*

Description

Returns data on up to 200 of the most recent mentions (Tweets containing a users' screen_name) of the authenticating user. The timeline returned is the equivalent of the one seen when you view your mentions on twitter.com.

Usage

```
get_mentions(
  n = 200,
  since_id = NULL,
  max_id = NULL,
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL,
  ...
)
```

Arguments

`n`	Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible. The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`. You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.
`since_id`	Supply a vector of ids or a data frame of previous results to find tweets **newer** than `since_id`.

max_id
: Supply a vector of ids or a data frame of previous results to find tweets **older** than max_id.

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose
: Show progress bars and other messages indicating current progress?

token
: Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

...
: Other arguments passed as parameters in composed API query.

Value

Tibble of mentions data.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/overview

See Also

Other tweets: get_favorites(), get_timeline(), lists_statuses(), lookup_tweets(), search_tweets()

Examples

```
if (auth_has_default()) {
tw <- get_mentions()
tw

# newer mentions
get_mentions(since_id = tw)
}
```

get_retweets — *Get the most recent retweets/retweeters*

Description

`get_retweets()` returns the 100 most recent retweets of a tweet; `get_retweeters()` retursn the 100 most recent users who retweeted them.

Usage

```
get_retweets(status_id, n = 100, parse = TRUE, token = NULL, ...)

get_retweeters(status_id, n = 100, parse = TRUE, token = NULL)
```

Arguments

`status_id`	Tweet/status id.
`n`	Number of results to retrieve. Must be <= 100.
`parse`	If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.
`...`	Other arguments used as parameters in the query sent to Twitter's rest API, for example, `trim_user = TRUE`.

Value

Tweets data of the most recent retweets/retweeters of a given status

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-retweets-id

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-retweeters-ids get_timeline *Get one or more user timelines*

Description

get_timeline() returns the timeline of any Twitter user (i.e. what they have tweeted). get_my_timeline() returns the home timeline for the authenticated user (i.e. the tweets you see when you log into Twitter).

Usage

```
get_timeline(
  user = NULL,
  n = 100,
  since_id = NULL,
  max_id = NULL,
  home = FALSE,
  parse = TRUE,
  check = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL,
  ...
)

get_my_timeline(
  n = 100,
  since_id = NULL,
  max_id = NULL,
  parse = TRUE,
  check = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL,
  ...
)
```

Arguments

user	Character vector of screen names or user ids. See as_screenname() for more details.
n	Desired number of results to return. Results are downloaded in pages when n is large; the default value will download a single page. Set n = Inf to download as many results as possible. The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use retryonratelimit = TRUE.

You are not guaranteed to get exactly n results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request n = 150 and the page size is 200, you'll get 200 results back.

since_id — Supply a vector of ids or a data frame of previous results to find tweets **newer** than since_id.

max_id — Supply a vector of ids or a data frame of previous results to find tweets **older** than max_id.

home — Logical, indicating whether to return a "user" timeline (the default, what a user has tweeted/retweeted) or a "home" timeline (what the user would see if they logged into twitter).

parse — If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

check — **[Deprecated]**

retryonratelimit — If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose — Show progress bars and other messages indicating current progress?

token — Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

... — Further arguments passed on as parameters in API query.

Details

At most up to 3,200 of a user's most recent Tweets can be retrieved.

Value

A tbl data frame of tweets data with users data attribute.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/overview

See Also

Other tweets: get_favorites(), get_mentions(), lists_statuses(), lookup_tweets(), search_tweets()

Examples

```
if (auth_has_default()) {
tw <- get_timeline("_R_Foundation")
tw

# get tweets that arrived since the last request
get_timeline("_R_Foundation", since_id = tw)
# get earlier tweets
get_timeline("_R_Foundation", max_id = tw)

# get timelines for multiple users
tw <- get_timeline(c("_R_Foundation", "rOpenSci", "Bioconductor"))
tw
}
```

---

get_token *Fetch Twitter OAuth token*

Description

[Deprecated] Please use `auth_get()` instead.

Usage

```
get_token()

get_tokens()
```

See Also

Other tokens: `create_token()`, `rate_limit()`

---

get_trends *Get Twitter trends data.*

Description

Get Twitter trends data.

Usage

```
get_trends(
  woeid = 1,
  lat = NULL,
  lng = NULL,
  exclude_hashtags = FALSE,
  token = NULL,
  parse = TRUE
)
```

Arguments

woeid	Numeric, WOEID (Yahoo! Where On Earth ID) or character string of desired town or country. Users may also supply latitude and longitude coordinates to fetch the closest available trends data given the provided location. Latitude/longitude coordinates should be provided as WOEID value consisting of 2 numeric values or via one latitude value and one longitude value (to the appropriately named parameters). To browse all available trend places, see `trends_available()`
lat	Optional alternative to WOEID. Numeric, latitude in degrees. If two coordinates are provided for WOEID, this function will coerce the first value to latitude.
lng	Optional alternative to WOEID. Numeric, longitude in degrees. If two coordinates are provided for WOEID, this function will coerce the second value to longitude.
exclude_hashtags	Logical, indicating whether or not to exclude hashtags. Defaults to FALSE–meaning, hashtags are included in returned trends.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.
parse	If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

Value

Tibble data frame of trends data for a given geographical area.

See Also

Other trends: `trends_available()`

Examples

```
if (auth_has_default()) {

## Retrieve available trends
trends <- trends_available()
trends
```

```
## Store WOEID for Worldwide trends
worldwide <- trends$woeid[grep("world", trends$name, ignore.case = TRUE)[1]]

## Retrieve worldwide trends datadata
ww_trends <- get_trends(worldwide)

## Preview trends data
ww_trends

## Retrieve trends data using latitude, longitude near New York City
nyc_trends <- get_trends(lat = 40.7, lng = -74.0)

## should be same result if lat/long supplied as first argument
nyc_trends <- get_trends(c(40.7, -74.0))

## Preview trends data
nyc_trends

## Provide a city or location name using a regular expression string to
## have the function internals do the WOEID lookup/matching for you
(luk <- get_trends("london"))

}
```

---

langs — *Defunct: Language codes recognized by Twitter data.*

Description

This data comes from the Library of Congress, http://www.loc.gov/standards/iso639-2/ISO-639-2_utf-8.txt. The data are descriptions and codes associated with internationally recognized languages. Variables include translations for each language represented as bibliographic, terminologic, alpha, english, and french.

Format

A tibble with five variables and 486 observations.

---

lat_lng — *Adds single-point latitude and longitude variables to tweets data.*

Description

Appends parsed Twitter data with latitude and longitude variables using all available geolocation information.

Usage

```
lat_lng(
  x,
  coords = c("coords_coords", "bbox_coords", "geo_coords"),
  prefs = "bbox_coords"
)
```

Arguments

x	Parsed Twitter data as returned by various rtweet functions. This should be a data frame with variables such as "bbox_coords", "coords_coords", and "geo_coords" (among other non-geolocation Twitter variables).
coords	Names of variables containing latitude and longitude coordinates. Priority is given to bounding box coordinates (each obs consists of eight entries) followed by the supplied order of variable names. Defaults to "bbox_coords", "coords_coords", and "geo_coords") (which are the default column names of data returned by most status-oriented rtweet functions).
prefs	Preference of coordinates to use as default, must be in `coords`.

Details

On occasion values may appear to be outliers given a previously used query filter (e.g., when searching for tweets sent from the continental US). This is typically because those tweets returned a large bounding box that overlapped with the area of interest. This function converts boxes into their geographical midpoints, which works well in the vast majority of cases, but sometimes includes an otherwise puzzling result.

Value

Returns updated data object with full information latitude and longitude vars.

See Also

Other geo: `lookup_coords()`

Examples

```
if (auth_has_default()) {

## stream tweets sent from the US
rt <- stream_tweets(lookup_coords("usa"), timeout = 10)

## use lat_lng to recover full information geolocation data
rtl_loc <- lat_lng(rt)
rtl_loc
}
```

`lists_members` *Get Twitter list members (users on a given list).*

Description

Get Twitter list members (users on a given list).

Usage

```
lists_members(
  list_id = NULL,
  slug = NULL,
  owner_user = NULL,
  n = 5000,
  cursor = "-1",
  token = NULL,
  retryonratelimit = NULL,
  verbose = TRUE,
  parse = TRUE,
  ...
)
```

Arguments

`list_id`	required The numerical id of the list.
`slug`	required You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_user parameters.
`owner_user`	optional The screen name or user ID of the user
`n`	Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible. The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`. You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.
`cursor`	Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose
: Show progress bars and other messages indicating current progress?

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

...
: Other arguments used as parameters in query composition.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/get-lists-members

See Also

Other lists: lists_statuses(), lists_subscribers(), lists_subscriptions(), lists_users()

Examples

```
if (auth_has_default()) {

## get list members for a list of rstats experts using list_id
(rstats <- lists_members("785434502382383105"))

## get list members for an rstats list using list topic slug
## list owner's screen name
rstats <- lists_members(slug = "rstats", owner_user = "scultrera")
rstats

}
```

---

lists_memberships *Get Twitter list memberships (lists containing a given user)*

Description

Due to deleted or removed lists, the returned number of memberships is often less than the provided n value. This is a reflection of the API and not a unique quirk of rtweet.

Usage

```
lists_memberships(
  user = NULL,
  n = 200,
  cursor = "-1",
  filter_to_owned_lists = FALSE,
  token = NULL,
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  previous_cursor = NULL
)
```

Arguments

`user` Character vector of screen names or user ids. See `as_screenname()` for more details.

`n` Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`cursor` Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

`filter_to_owned_lists` When `TRUE`, will return only lists that authenticating user owns.

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`retryonratelimit` If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

verbose Show progress bars and other messages indicating current progress?

previous_cursor
[Deprecated] Please use cursor instead.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/get-lists-memberships

Examples

```
if (auth_has_default()) {

## get up to 1000 Twitter lists that include Nate Silver
R_foundation <- lists_memberships("_R_Foundation", n = 1000)

## view data
R_foundation

}
```

lists_statuses *Get a timeline of tweets authored by members of a specified list.*

Description

Get a timeline of tweets authored by members of a specified list.

Usage

```
lists_statuses(
  list_id = NULL,
  slug = NULL,
  owner_user = NULL,
  since_id = NULL,
  max_id = NULL,
  n = 200,
  include_rts = TRUE,
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL
)
```

Arguments

list_id — required The numerical id of the list.

slug — required You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.

owner_user — optional The screen name or user ID of the user who owns the list being requested by a slug.

since_id — Supply a vector of ids or a data frame of previous results to find tweets **newer** than `since_id`.

max_id — Supply a vector of ids or a data frame of previous results to find tweets **older** than `max_id`.

n — Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

include_rts — optional When set to either true, t or 1, the list timeline will contain native retweets (if they exist) in addition to the standard stream of tweets. The output format of retweeted tweets is identical to the representation you see in home_timeline.

parse — If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

retryonratelimit — If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

verbose — Show progress bars and other messages indicating current progress?

token — Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

Value

data

See Also

Other lists: lists_members(), lists_subscribers(), lists_subscriptions(), lists_users()

Other tweets: get_favorites(), get_mentions(), get_timeline(), lookup_tweets(), search_tweets()

Examples

```
if (auth_has_default()) {
  (rladies <- lists_statuses(list_id = "839186302968848384"))
  (rladies <- lists_statuses(slug = "rladies1", owner_user = "RLadiesGlobal"))
}
```

---

lists_subscribers *Get subscribers of a specified list.*

Description

Get subscribers of a specified list.

Usage

```
lists_subscribers(
  list_id = NULL,
  slug = NULL,
  owner_user = NULL,
  n = 5000,
  cursor = "-1",
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL
)
```

Arguments

list_id	required The numerical id of the list.
slug, owner_user	The list name (slug) and owner.
n	Desired number of results to return. Results are downloaded in pages when n is large; the default value will download a single page. Set n = Inf to download as many results as possible. The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use retryonratelimit = TRUE. You are not guaranteed to get exactly n results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request n = 150 and the page size is 200, you'll get 200 results back.

`cursor` Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`retryonratelimit`
If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

`verbose` Show progress bars and other messages indicating current progress?

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/get-lists-subscribers

See Also

Other lists: `lists_members()`, `lists_statuses()`, `lists_subscriptions()`, `lists_users()`

Other users: `as_screenname()`, `lookup_users()`, `search_users()`

Examples

```
if (auth_has_default()) {
    ## get subscribers of rladies list
    rstats <- lists_subscribers(slug = "rladies1", owner_user = "rladiesglobal")
}
```

`lists_subscriptions` *Get list subscriptions of a given user but does not include the user's own lists.*

Description

Get list subscriptions of a given user but does not include the user's own lists.

Usage

```
lists_subscriptions(
  user,
  n = 20,
  cursor = "-1",
  parse = TRUE,
  retryonratelimit = NULL,
  verbose = TRUE,
  token = NULL
)
```

Arguments

`user` Character vector of screen names or user ids. See `as_screenname()` for more details.

`n` Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`cursor` Which page of results to return. The default will return the first page; you can supply the result from a previous call to continue pagination from where it left off.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`retryonratelimit` If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

`verbose` Show progress bars and other messages indicating current progress?

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/get-lists-subscriptions

See Also

Other lists: lists_members(), lists_statuses(), lists_subscribers(), lists_users()

Examples

```
if (auth_has_default()) {
## get ropensci subscriptions
rstats <- lists_subscriptions(user = "rladiesglobal", n = 1000)
}
```

---

lists_users *Get all lists a specified user subscribes to, including their own.*

Description

Get all lists a specified user subscribes to, including their own.

Usage

```
lists_users(user = NULL, reverse = FALSE, token = NULL, parse = TRUE)
```

Arguments

user	Character vector of screen names or user ids. See as_screenname() for more details.
reverse	optional Set this to true if you would like owned lists to be returned first. See description above for information on how this parameter works.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.
parse	If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

Value

data

See Also

Other lists: lists_members(), lists_statuses(), lists_subscribers(), lists_subscriptions()

Examples

```
if (auth_has_default()) {

## get lists subscribed to by R_Foundation
lists_users("ropensci")

}
```

`lookup_coords` *Get coordinates of specified location.*

Description

Convenience function for looking up latitude/longitude coordinate information for a given location. Returns data as a special "coords" object, which is specifically designed to interact smoothly with other relevant package functions. NOTE: USE OF THIS FUNCTION REQUIRES A VALID GOOGLE MAPS API KEY.

Usage

```
lookup_coords(address, components = NULL, apikey = NULL, ...)
```

Arguments

`address`	Desired location typically in the form of place name, subregion, e.g., address = "lawrence, KS". Also accepts the name of countries, e.g., address = "usa", address = "brazil" or states, e.g., address = "missouri" or cities, e.g., address = "chicago". In most cases using only address should be sufficient.
`components`	Unit of analysis for address e.g., components = "country:US". Potential components include postal_code, country, administrative_area, locality, route.
`apikey`	A valid Google Maps API key. If NULL, `lookup_coords()` will look for a relevant API key stored as an environment variable (e.g., `GOOGLE_MAPS_KEY`).
`...`	Additional arguments passed as parameters in the HTTP request

Details

Since Google Maps implemented stricter API requirements, sending requests to Google's API isn't very convenient. To enable basic uses without requiring a Google Maps API key, a number of the major cities throughout the word and the following two larger locations are baked into this function: 'world' and 'usa'. If 'world' is supplied then a bounding box of maximum latitutde/longitude values, i.e., `c(-180, -90, 180, 90)`, and a center point `c(0, 0)` are returned. If 'usa' is supplied then estimates of the United States' bounding box and mid-point are returned. To specify a city, provide the city name followed by a space and then the US state abbreviation or country name. To see a list of all included cities, enter `rtweet:::citycoords` in the R console to see coordinates data.

Value

Object of class coords.

See Also

Other geo: lat_lng()

Examples

```
## Not run:

## get coordinates associated with the following addresses/components
sf <- lookup_coords("san francisco, CA", "country:US")
usa <- lookup_coords("usa")
lnd <- lookup_coords("london")
bz <- lookup_coords("brazil")

## pass a returned coords object to search_tweets
bztw <- search_tweets(geocode = bz)

## or stream tweets
ustw <- stream_tweets(usa, timeout = 10)


## End(Not run)
```

---

lookup_friendships *Lookup friendship information between two specified users.*

Description

Gets information on friendship between two Twitter users.

Usage

```
lookup_friendships(source, target, parse = TRUE, token = NULL)
```

Arguments

source	Screen name or user id of source user.
target	Screen name or user id of target user.
parse	If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-friendships-show

See Also

Other friends: my_friendships()

---

lookup_tweets *Get tweets data for given statuses (status IDs).*

Description

Get tweets data for given statuses (status IDs).

Usage

```
lookup_tweets(
  statuses,
  parse = TRUE,
  token = NULL,
  retryonratelimit = NULL,
  verbose = TRUE
)
```

Arguments

statuses
: User id or screen name of target user.

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

token
: Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose
: Show progress bars and other messages indicating current progress?

Value

A tibble of tweets data.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-lookup

See Also

Other tweets: `get_favorites()`, `get_mentions()`, `get_timeline()`, `lists_statuses()`, `search_tweets()`

Examples

```
if (auth_has_default()) {
statuses <- c(
  "567053242429734913",
  "266031293945503744",
  "440322224407314432"
)

## lookup tweets data for given statuses
tw <- lookup_tweets(statuses)
tw
}
```

---

lookup_users — *Get Twitter users data for given users (user IDs or screen names).*

Description

Get Twitter users data for given users (user IDs or screen names).

Usage

```
lookup_users(
  users,
  parse = TRUE,
  token = NULL,
  retryonratelimit = NULL,
  verbose = TRUE
)
```

Arguments

users	User id or screen name of target user.
parse	If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

`retryonratelimit`
: If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

`verbose`
: Show progress bars and other messages indicating current progress?

Value

A tibble of users data.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-users-lookup

See Also

Other users: `as_screenname()`, `lists_subscribers()`, `search_users()`

Examples

```
if (auth_has_default()) {
    users <- c("twitter", "rladiesglobal", "_R_Foundation")
    users <- lookup_users(users)
    users

    # latest tweet from each user
    tweets_data(users)
}
```

`my_friendships` *Lookup friendship information between users.*

Description

Gets information on friendship between authenticated user and up to 100 other users.

Usage

```
my_friendships(user, parse = FALSE, token = NULL)
```

Arguments

`user`	Character vector of screen names or user ids. See `as_screenname()` for more details.
`parse`	If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-friendships-lookup

See Also

Other friends: `lookup_friendships()`

`network_data` *Network data*

Description

- `network_data()` returns a data frame that can easily be converted to various network classes.
- `network_graph()` returns a igraph object

Usage

```
network_data(x, e = c("mention", "retweet", "reply", "quote"))

network_graph(x, e = c("mention", "retweet", "reply", "quote"))
```

Arguments

`x`	Data frame returned by rtweet function
`e`	Type of edge/link–i.e., "mention", "retweet", "quote", "reply". This must be a character vector of length one or more. This value will be split on punctuation and space (so you can include multiple types in the same string separated by a comma or space). The values "all" and "semantic" are assumed to mean all edge types, which is equivalent to the default value of `c("mention", "retweet", "reply", "quote")`

Details

Retrieve data to know which users are connected to which users.

Value

A from/to data edge data frame

An igraph object

See Also

network_graph

Examples

```
if (auth_has_default()) {
  ## search for #rstats tweets
  rstats <- search_tweets("#rstats", n = 200)

  ## create from-to data frame representing retweet/mention/reply connections
  rstats_net <- network_data(rstats, c("retweet","mention","reply"))

  ## view edge data frame
  rstats_net

  ## view user_id->screen_name index
  attr(rstats_net, "idsn")

  ## if igraph is installed...
  if (requireNamespace("igraph", quietly = TRUE)) {

    ## (1) convert directly to graph object representing semantic network
    rstats_net <- network_graph(rstats)

    ## (2) plot graph via igraph.plotting
    plot(rstats_net)
  }
}
```

`parse_stream` *Parser of stream*

Description

Converts Twitter stream data (JSON file) into parsed data frame.

Usage

```
parse_stream(path, ...)
```

Arguments

`path`	Character, name of JSON file with data collected by `stream_tweets()`.
`...`	Unused, keeping it for back compatibility.

See Also

`stream_tweets()`

Examples

```
## Not run:
stream_tweets(timeout = 1, file_name = "stream.json", parse = FALSE)
parse_stream("stream.json")

## End(Not run)
```

`plain_tweets` *Clean up character vector (tweets) to more of a plain text.*

Description

Removes links, linebreaks, fancy spaces and apostrophes and convert everything to ASCII text. Deprecated to be defunct for next release as there are better text processing tools.

Usage

```
plain_tweets(x)
```

Arguments

x	The desired character vector or data frame/list with named column/element "text" to be cleaned and processed.

Value

Data reformatted with ascii encoding and normal ampersands and without URL links, line breaks, fancy spaces/tabs, fancy apostrophes,

---

`post_destroy` *Delete status of user's Twitter account*

Description

Deletes a status of user's profile.

Usage

```
post_destroy(destroy_id, token = NULL)
```

Arguments

`destroy_id`	To delete a status, supply the single status ID here. If a character string is supplied, overriding the default (NULL), then a destroy request is made (and the status text and media attachments) are irrelevant.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/post-statuses-destroy-id

Examples

```
if (auth_has_default()) {
pt <- post_tweet("Running #rtweet examples")
crt <- httr::content(pt)
post_destroy(crt$id_str)
}
```

`post_favorite` *Favorites target status id.*

Description

Favorites target status id.

Usage

```
post_favorite(
  status_id,
  destroy = FALSE,
  include_entities = FALSE,
  token = NULL
)
```

Arguments

`status_id`	Status id of target tweet.
`destroy`	Logical indicating whether to post (add) or remove (delete) target tweet as favorite.
`include_entities`	Logical indicating whether to include entities object in return.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

References

Create: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/post-favorites-create Destroy: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/post-favorites-destroy

See Also

Other post: `post_follow()`, `post_friendship()`, `post_tweet()`

Examples

```
if (auth_has_default()) {
   rt <- search_tweets("#rstats", n = 1)
   post_favorite(rt$id_str)
}
```

`post_follow` *Follows target Twitter user.*

Description

Follows target Twitter user.

Usage

```
post_follow(
  user,
  destroy = FALSE,
  mute = FALSE,
  notify = FALSE,
  retweets = TRUE,
  token = NULL
)

post_unfollow_user(user, token = NULL)

post_mute(user, token = NULL)
```

Arguments

`user`	Character vector of screen names or user ids. See `as_screenname()` for more details.
`destroy`	Logical indicating whether to post (add) or remove (delete) target tweet as favorite.
`mute`	Logical indicating whether to mute the intended friend (you must already be following this account prior to muting them)

notify	Logical indicating whether to enable notifications for target user. Defaults to false.
retweets	Logical indicating whether to enable retweets for target user. Defaults to true.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

References

Update: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/post-friendships-update Create: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/post-friendships-create Destroy: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/post-friendships-destroy
Mute: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/mute-block-report-use api-reference/post-mutes-users-create

See Also

Other post: post_favorite(), post_friendship(), post_tweet()

Examples

```
if (auth_has_default()) {
    post_follow("_R_Foundation")
    post_follow("rtweet_test", mute = TRUE) # Mute
}
```

---

post_friendship *Updates friendship notifications and retweet abilities.*

Description

Updates friendship notifications and retweet abilities.

Usage

```
post_friendship(user, device = FALSE, retweets = FALSE, token = NULL)
```

Arguments

user	Character vector of screen names or user ids. See as_screenname() for more details.
device	Logical indicating whether to enable or disable device notifications from target user behaviors. Defaults to false.
retweets	Logical indicating whether to enable or disable retweets from target user behaviors. Defaults to false.

token — Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/post-friendships-update

See Also

Other post: post_favorite(), post_follow(), post_tweet()

post_list — *Manage Twitter lists*

Description

Create, add users, and destroy Twitter lists

Usage

```
post_list(
  users = NULL,
  name = NULL,
  description = NULL,
  private = FALSE,
  destroy = FALSE,
  list_id = NULL,
  slug = NULL,
  token = NULL
)
```

Arguments

users	Character vectors of users to be added to list.
name	Name of new list to create.
description	Optional, description of list (single character string).
private	Logical indicating whether created list should be private. Defaults to false, meaning the list would be public. Not applicable if list already exists.
destroy	Logical indicating whether to delete a list. Either list_id or slug must be provided if destroy = TRUE.
list_id	Optional, numeric ID of list.
slug	Optional, list slug.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

Value

Response object from HTTP request.

References

Create: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-create Destroy: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-destroy Add users: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-members-create, https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-members-create_all Remove users: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-members-destroy, https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/create-manage-lists/api-reference/post-lists-members-destroy_all

Examples

```
## Not run:

## R related Twitter accounts
users <- c("_R_Foundation", "R_dev_news", "rweekly_live", "RConsortium", "rstats4ds",
  "icymi_r", "rstatstweet", "RLadiesGlobal")

## create r-accounts list with 8 total users
(r_lst <- post_list(users,
  "r-accounts", description = "R related accounts"))

## view list in browser
browseURL(sprintf("https://twitter.com/%s/lists/r-accounts",
  rtweet:::api_screen_name()))

## search for more rstats users
r_users <- search_users("rstats", n = 200)

## filter and select more users to add to list
more_users <- r_users$screen_name[r_users$verified]

## add more users to list- note: can only add up to 100 at a time
post_list(users = more_users, slug = "r-accounts")

## view updated list in browser (should be around 100 users)
browseURL(sprintf("https://twitter.com/%s/lists/r-accounts",
  rtweet:::api_screen_name()))

drop_users <- "icymi_r"

## drop these users from the R list
post_list(users = drop_users, slug = "r-accounts",
  destroy = TRUE)
```

```
## view updated list in browser (should be around 100 users)
browseURL(sprintf("https://twitter.com/%s/lists/r-accounts",
  rtweet:::api_screen_name()))

## delete list entirely
post_list(slug = "r-accounts", destroy = TRUE)


## End(Not run)
```

---

post_message *Posts direct message from user's Twitter account*

Description

Posts direct message from user's Twitter account

Usage

```
post_message(text, user, media = NULL, token = NULL)
```

Arguments

text	Character, text of message.
user	Character vector of screen names or user ids. See as_screenname() for more details.
media	File path to image or video media to be included in tweet.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

References

https://developer.twitter.com/en/docs/twitter-api/v1/direct-messages/sending-and-receiving/api-reference/new-event

post_tweet — *Posts status update to user's Twitter account*

Description

Posts status update to user's Twitter account

Usage

```
post_tweet(
  status = "my first rtweet #rstats",
  media = NULL,
  token = NULL,
  in_reply_to_status_id = NULL,
  destroy_id = NULL,
  retweet_id = NULL,
  auto_populate_reply_metadata = FALSE,
  media_alt_text = NULL,
  lat = NULL,
  long = NULL,
  display_coordinates = FALSE
)
```

Arguments

`status` Character, tweet status. Must be 280 characters or less.

`media` Length 1 character vector with a file path to video media **OR** up-to length 4 character vector with file paths to static images to be included in tweet. **The caller is responsible for managing this.**

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

`in_reply_to_status_id` Status ID of tweet to which you'd like to reply. Note: in line with the Twitter API, this parameter is ignored unless the author of the tweet this parameter references is mentioned within the status text.

`destroy_id` To delete a status, supply the single status ID here. If a character string is supplied, overriding the default (NULL), then a destroy request is made (and the status text and media attachments) are irrelevant.

`retweet_id` To retweet a status, supply the single status ID here. If a character string is supplied, overriding the default (NULL), then a retweet request is made (and the status text and media attachments) are irrelevant.

`auto_populate_reply_metadata` If set to TRUE and used with in_reply_to_status_id, leading @mentions will be looked up from the original Tweet, and added to the new Tweet from there. Defaults to FALSE.

media_alt_text
: attach additional alt text metadata to the media you are uploading. Should be same length as media (i.e. as many alt text entries as there are media entries). See the official API documentation for more information.

lat
: A numeric value representing the latitude of the location the tweet refers to. Range should be between -90 and 90 (north). Note that you should enable the "Precise location" option in your account via *Settings and privacy > Privacy and Safety > Location*. See the official Help Center section.

long
: A numeric value representing the longitude of the location the tweet refers to. Range should be between -180 and 180 (west). See lat parameter.

display_coordinates
: Put a pin on the exact coordinates a tweet has been sent from. Value should be TRUE or FALSE. This parameter would apply only if you have provided a valid lat/long pair of valid values.

References

Tweet: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/post-statuses-update Retweet: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/post-statuses-retweet-id Media: https://developer.twitter.com/en/docs/twitter-api/v1/media/upload-media/api-reference/post-media-metadata-create Alt-text: https://developer.twitter.com/en/docs/twitter-api/v1/media/upload-media/api-reference/post-media-metadata-create

See Also

Other post: post_favorite(), post_follow(), post_friendship()

Examples

```
if (auth_has_default()) {
## generate data to make/save plot (as a .png file)
x <- rnorm(300)
y <- x + rnorm(300, 0, .75)
col <- c(rep("#002244aa", 50), rep("#440000aa", 50))
bg <- c(rep("#6699ffaa", 50), rep("#dd6666aa", 50))

## create temporary file name
tmp <- tempfile(fileext = ".png")

## save as png
png(tmp, 6, 6, "in", res = 127.5)
par(tcl = -.15, family = "Inconsolata",
    font.main = 2, bty = "n", xaxt = "l", yaxt = "l",
    bg = "#f0f0f0", mar = c(3, 3, 2, 1.5))
plot(x, y, xlab = NULL, ylab = NULL, pch = 21, cex = 1,
     bg = bg, col = col,
     main = "This image was uploaded by rtweet")
grid(8, lwd = .15, lty = 2, col = "#00000088")
dev.off()
```

```
## post tweet with media attachment
post_tweet("a tweet with media attachment", media = tmp,
           media_alt_text = "Random  points example of rtweet::post_tweet.
           rtweet requires alt text with all media")

# example of replying within a thread
## first post
post_tweet(status="first in a thread")

## lookup status_id
my_timeline <- get_my_timeline()

## ID for reply
reply_id <- my_timeline$id_str[1]

## post reply
post_tweet("second in the thread",
  in_reply_to_status_id = reply_id)
}
```

`rate_limit` *Rate limit helpers*

Description

- `rate_limit()` returns a tibble of info about all rate limits
- `rate_limit_reset()` returns the next reset for a endpoint
- `rate_limit_wait()` waits for the next reset for an endpoint

You should not need to use these function in the usual operation of rtweet because all paginated functions will wait on your behalf if you set `retryonratelimit = TRUE`.

Usage

```
rate_limit(resource_match = NULL, token = NULL)

rate_limit_reset(endpoint, token = NULL)

rate_limit_wait(endpoint, token = NULL)
```

Arguments

resource_match	An optional regular expression used to filter the resources listed in returned rate limit data.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.
endpoint	Name of Twitter endpoint like `"lookup/users"`, `"/media/upload"`, or `"/feedback/show/:id"`.

References

https://developer.twitter.com/en/docs/twitter-api/v1/developer-utilities/rate-limit-status

See Also

Other tokens: create_token(), get_token()

Examples

```
if (auth_has_default()) {
rate_limit()
}
```

---

read_twitter_csv *Read comma separated value Twitter data.*

Description

Reads Twitter data that was previously saved as a CSV file.

Usage

```
read_twitter_csv(file, unflatten = FALSE)
```

Arguments

file	Name of CSV file.
unflatten	Logical indicating whether to unflatten (separate hasthags and mentions columns on space, converting characters to lists), defaults to FALSE.

Value

A tbl data frame of Twitter data

See Also

Other datafiles: flatten(), write_as_csv()

Examples

```

## Not run:

## read in data.csv
rt <- read_twitter_csv("data.csv")


## End(Not run)
```

round_time — *A generic function for rounding date and time values*

Description

A generic function for rounding date and time values

Usage

```
round_time(x, n, tz)
```

Arguments

x	A vector of class POSIX or Date.
n	Unit to round to. Defaults to mins. Numeric values treated as seconds. Otherwise this should be one of "mins", "hours", "days", "weeks", "months", "years" (plural optional).
tz	Time zone to be used, defaults to "UTC" (Twitter default)

Value

If POSIXct then POSIX. If date then Date.

Examples

```
## class posixct
round_time(Sys.time(), "12 hours")

## class date
unique(round_time(seq(Sys.Date(), Sys.Date() + 100, "1 day"), "weeks"))
```

rtweet_user — *Authentication options*

Description

There are three ways that you can authenticate with the Twitter API:

- `rtweet_user()` interactively authenticates an existing Twitter user. This form is most appropriate if you want rtweet to control your Twitter account.
- `rtweet_app()` authenticates as a Twitter application. An application can't perform actions (i.e. it can't tweet) but otherwise has generally higher rate limits (i.e. you can do more searches). See details at https://developer.twitter.com/en/docs/twitter-api/v1/rate-limits. This form is most appropriate if you are collecting data.

- `rtweet_bot()` authenticates as bot that takes actions on behalf of an app. This form is most appropriate if you want to create a Twitter account that is run by a computer, rather than a human.

To use `rtweet_app()` or `rtweet_bot()` you will need to create your own Twitter app following the instructions in `vignette("auth.Rmd")`. `rtweet_user()` *can be* used with your own app, but generally there is no need to because it uses the Twitter app provided by rtweet.

Use `auth_as()` to set the default auth mechanism for the current session, and `auth_save()` to save an auth mechanism for use in future sessions.

Usage

```
rtweet_user(api_key = NULL, api_secret = NULL)

rtweet_bot(api_key, api_secret, access_token, access_secret)

rtweet_app(bearer_token)
```

Arguments

`api_key, api_secret`	Application API key and secret. These are generally not required for `tweet_user()` since the defaults will use the built-in rtweet app.
`access_token, access_secret`	Access token and secret.
`bearer_token`	App bearer token.

Details

Authenticate methods to use the Twitter API.

Security

All of the arguments to these functions are roughly equivalent to passwords so should generally not be typed into the console (where they the will be recorded in `.Rhistory`) or recorded in a script (which is easy to accidentally share). Instead, call these functions without arguments since the default behaviour is to use ask_pass that if possible uses `askpass::askpass()` to interactively safely prompt you for the values.

See Also

Other authentication: `auth_as()`, `auth_get()`, `auth_save()`, `auth_setup_default()`

Examples

```
## Not run:
rtweet_user()
rtweet_bot()
rtweet_app()

## End(Not run)
```

search_fullarchive *Premium Twitter searches*

Description

Search 30day or fullarchive premium products. There is a limit of 5000 tweets and 25000 for the fullarchive and 30day endpoints respectively. In addition, there are some limits in the number of requests that are possible on a certain amount of time, this have already been taken into account. See the info provided by Twitter.

Usage

```
search_fullarchive(
  q,
  n = 100,
  fromDate = NULL,
  toDate = NULL,
  continue = NULL,
  env_name = NULL,
  premium = FALSE,
  safedir = NULL,
  parse = TRUE,
  token = NULL
)

search_30day(
  q,
  n = 100,
  fromDate = NULL,
  toDate = NULL,
  env_name = NULL,
  continue = NULL,
  premium = FALSE,
  safedir = NULL,
  parse = TRUE,
  token = NULL
)
```

Arguments

q	Search query on which to match/filter tweets. See details for information about available search operators.
n	Desired number of results to return. Results are downloaded in pages when n is large; the default value will download a single page. Set n = Inf to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`fromDate` Oldest date-time (YYYYMMDDHHMM) from which tweets should be searched for.

`toDate` Newest date-time (YYYYMMDDHHMM) from which tweets should be searched for.

`continue` A character string with the next results of a query. You must make the exact same query as the original, including `q`, `toDate`, and `fromDate`.

`env_name` Name/label of developer environment to use for the search.

`premium` A logical value if the environment is paid (TRUE) or sandboxed, the default (FALSE). It limits the number of results retrieved so the number of API queries needed to retrieve `n` results.

`safedir` Name of directory to which each response object should be saved. If the directory doesn't exist, it will be created. If NULL (the default) then a dir will be created in the current working directory. To override/deactivate safedir set this to FALSE.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`token` Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.

Details

Note: The `env_name` must match the ones you set up for the token you are using.

Value

A tibble data frame of Twitter data.

Developer Account

Users must have an approved developer account and an active/labeled environment to access Twitter's premium APIs. For more information, to check your current Subscriptions and Dev Environments, or to apply for a developer account visit `https://developer.twitter.com`.

Search operators

Note: Bolded operators ending with a colon should be immediately followed by a word or quoted phrase (if appropriate)–e.g., `lang:en`

Keyword

- **""** ~~ match exact phrase
- **#** ~~ hashtag
- **@** ~~ at mentions)
- **url:** ~~ found in URL
- **lang:** ~~ language of tweet

Accounts of interest

- **from:** ~~ authored by
- **to:** ~~ sent to
- **retweets_of:** ~~ retweet author

Tweet attributes

- **is:retweet** ~~ only retweets
- **has:mentions** ~~ uses mention(s)
- **has:hashtags** ~~ uses hashtags(s)
- **has:media** ~~ includes media(s)
- **has:videos** ~~ includes video(s)
- **has:images** ~~ includes image(s)
- **has:links** ~~ includes URL(s)
- **is:verified** ~~ from verified accounts

Geospatial

- **bounding_box:[west_long south_lat east_long north_lat]** ~~ lat/long coordinates box
- **point_radius:[lon lat radius]** ~~ center of search radius
- **has:geo** ~~ uses geotagging
- **place:** ~~ by place
- **place_country:** ~~ by country
- **has:profile_geo** ~~ geo associated with profile
- **profile_country:** ~~ country associated with profile
- **profile_region:** ~~ region associated with profile
- **profile_locality:** ~~ locality associated with profile

References

Endpoint: https://developer.twitter.com/en/docs/twitter-api/premium/search-api/api-reference/premium-search Full archive limits https://developer.twitter.com/en/pricing/search-fullarchive 30day limits https://developer.twitter.com/en/pricing/search-30day

Examples

```
## Not run:
## search fullarchive for up to 300 rstats tweets sent in Jan 2014
rt <- search_fullarchive("#rstats", n = 300, env_name = "SetYourLabel",
  fromDate = "201401010000", toDate = "201401312359")

toDate <- format(Sys.time() - 60 * 60 * 24 * 7, "%Y%m%d%H%M")

## search 30day for up to 300 rstats tweets sent before the last week
rt <- search_30day("#rstats", n = 300,
  env_name = "SetYourLabel", toDate = toDate)

## End(Not run)
```

`search_tweets` *Get tweets data on statuses identified via search query.*

Description

Returns Twitter statuses matching a user provided search query. ONLY RETURNS DATA FROM THE PAST 6-9 DAYS.

search_tweets2 Passes all arguments to search_tweets. Returns data from one OR MORE search queries.

Usage

```
search_tweets(
  q,
  n = 100,
  type = c("mixed", "recent", "popular"),
  include_rts = TRUE,
  geocode = NULL,
  since_id = NULL,
  max_id = NULL,
  parse = TRUE,
  token = NULL,
  retryonratelimit = NULL,
  verbose = TRUE,
  ...
)

search_tweets2(...)
```

Arguments

`q` Query to be searched, used to filter and select tweets to return from Twitter's REST API. Must be a character string not to exceed maximum of 500 characters. Spaces behave like boolean "AND" operator. To search for tweets containing at least one of multiple possible terms, separate each search term with spaces and "OR" (in caps). For example, the search `q = "data science"` looks for tweets containing both "data" and "science" located anywhere in the tweets and in any order. When "OR" is entered between search terms, `query = "data OR science"`, Twitter's REST API should return any tweet that contains either "data" or "science." It is also possible to search for exact phrases using double quotes. To do this, either wrap single quotes around a search query using double quotes, e.g., `q = '"data science"'` or escape each internal double quote with a single backslash, e.g., `q = "\"data science\""`.

Some other useful query tips:

- Exclude retweets via `"-filter:retweets"`
- Exclude quotes via `"-filter:quote"`
- Exclude replies via `"-filter:replies"`
- Filter (return only) verified via `"filter:verified"`
- Exclude verified via `"-filter:verified"`
- Get everything (firehose for free) via `"-filter:verified OR filter:verified"`
- Filter (return only) tweets with links to news articles via `"filter:news"`
- Filter (return only) tweets with media `"filter:media"`

`n` Desired number of results to return. Results are downloaded in pages when `n` is large; the default value will download a single page. Set `n = Inf` to download as many results as possible.

The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use `retryonratelimit = TRUE`.

You are not guaranteed to get exactly `n` results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request `n = 150` and the page size is 200, you'll get 200 results back.

`type` Character string specifying which type of search results to return from Twitter's REST API. The current default is `type = "recent"`, other valid types include `type = "mixed"` and `type = "popular"`.

`include_rts` Logical, indicating whether to include retweets in search results. Retweets are classified as any tweet generated by Twitter's built-in "retweet" (recycle arrows) function. These are distinct from quotes (retweets with additional text provided from sender) or manual retweets (old school method of manually entering "RT" into the text of one's tweets).

`geocode` Geographical limiter of the template "latitude,longitude,radius" e.g., `geocode = "37.78,-122.40,1mi"`.

`since_id` Supply a vector of ids or a data frame of previous results to find tweets **newer** than `since_id`.

max_id
: Supply a vector of ids or a data frame of previous results to find tweets **older** than max_id.

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

token
: Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, NULL, consults the option rtweet.retryonratelimit so that you can globally set it to TRUE, if desired.

  If you expect a query to take hours or days to perform, you should not rely soley on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.

verbose
: Show progress bars and other messages indicating current progress?

...
: Further arguments passed as query parameters in request sent to Twitter's REST API. To return only English language tweets, for example, use lang = "en". For more options see Twitter's API documentation.

Details

Twitter API documentation recommends limiting searches to 10 keywords and operators. Complex queries may also produce API errors preventing recovery of information related to the query. It should also be noted Twitter's search API does not consist of an index of all Tweets. At the time of searching, the search API index includes between only 6-9 days of Tweets.

Value

List object with tweets and users each returned as a data frame.

A tbl data frame with additional "query" column.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets

See Also

Other tweets: get_favorites(), get_mentions(), get_timeline(), lists_statuses(), lookup_tweets()

Examples

```
if (auth_has_default()) {
tweets <- search_tweets("weather")
tweets
```

```
# data about the users who made those tweets
users_data(tweets)

# Retrieve all the tweets made since the previous request
# (there might not be any if people aren't tweeting about the weather)
newer <- search_tweets("weather", since_id = tweets)
# Retrieve tweets made before the previous request
older <- search_tweets("weather", max_id = tweets)

# Restrict to English only, and ignore retweets
tweets2 <- search_tweets("weather", lang = "en", include_rts = FALSE)
}
if (auth_has_default()) {

## search using multiple queries
st2 <- search_tweets2(
  c("\"data science\"", "rstats OR python"),
  n = 500
)

## preview tweets data
st2

## preview users data
users_data(st2)

## check breakdown of results by search query
table(st2$query)

}
```

`search_users` *Search for users*

Description

Search for Twitter users. The Twitter API limits the results to at most 1,000 users.

Usage

```
search_users(q, n = 100, parse = TRUE, token = NULL, verbose = TRUE)
```

Arguments

q	As string providing the search query. Try searching by interest, full name, company name, or location. Exact match searches are not supported.

n
: Desired number of results to return. Results are downloaded in pages when n is large; the default value will download a single page. Set n = Inf to download as many results as possible.

  The Twitter API rate limits the number of requests you can perform in each 15 minute period. The easiest way to download more than that is to use retryonratelimit = TRUE.

  You are not guaranteed to get exactly n results back. You will get fewer results when tweets have been deleted or if you hit a rate limit. You will get more results if you ask for a number of tweets that's not a multiple of page size, e.g. if you request n = 150 and the page size is 200, you'll get 200 results back.

parse
: If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

token
: Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

verbose
: Show progress bars and other messages indicating current progress?

Value

Data frame with one row for each matching user.

References

https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-users-search

See Also

Other users: as_screenname(), lists_subscribers(), lookup_users()

Examples

```
if (auth_has_default()) {
users <- search_users("#rstats", n = 300)
users

# latest tweet from each user
tweets_data(users)
}
```

stopwordslangs — *Defunct: Twitter stop words in multiple languages data.*

Description

This data comes form a group of Twitter searches conducted at several times during the calendar year of 2017. The data are commonly observed words associated with 10 different languages, including `c("ar", "en", "es", "fr", "in", "ja", "pt", "ru", "tr", "und")`. Variables include `"word"` (potential stop words), `"lang"` (two or three word code), and `"p"` (probability value associated with frequency position along a normal distribution with higher values meaning the word occurs more frequently and lower values meaning the words occur less frequently).

Format

A tibble with three variables and 24,000 observations

stream_tweets — *Collect a live stream of Twitter data*

Description

Streams public statuses to a file via one of the following four methods:

1. Sampling a small random sample of all publicly available tweets
2. Filtering via a search-like query (up to 400 keywords)
3. Tracking via vector of user ids (up to 5000 user_ids)
4. Location via geo coordinates (1-360 degree location boxes)

Learn more in `vignette("stream", package = "rtweet")`

Usage

```
stream_tweets(
  q = "",
  timeout = 30,
  parse = TRUE,
  token = NULL,
  file_name = NULL,
  verbose = TRUE,
  append = TRUE,
  ...
)
```

Arguments

`q`	Query used to select and customize streaming collection method. There are four possible methods: 1. The default, `q = ""`, returns a small random sample of all publicly available Twitter statuses. 2. To filter by keyword, provide a comma separated character string with the desired phrase(s) and keyword(s). 3. Track users by providing a comma separated list of user IDs or screen names. 4. Use four latitude/longitude bounding box points to stream by geo location. This must be provided via a vector of length 4, e.g., `c(-125, 26, -65, 49)`.
`timeout`	Integer specifying number of seconds to stream tweets for. Stream indefinitely with `timeout = Inf`. The stream can be interrupted at any time, and `file_name` will still be valid file.
`parse`	Use `FALSE` to opt-out of parsing the tweets.
`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.
`file_name`	Character with name of file. If not specified, will write to a temporary file `stream_tweets*.json`.
`verbose`	If `TRUE`, display a progress bar.
`append`	If `TRUE`, will append to the end of `file_name`; if `FALSE`, will overwrite.
`...`	Other arguments passed in to query parameters.

Value

A tibble with one row per tweet

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/sample-realtime/api-reference/get-statuses-sample, https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/overview

Stream: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/sample-realtime/api-reference/get-statuses-sample Filter: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/api-reference/post-statuses-filter

See Also

`parse_stream()`.

Examples

```
## Not run:
# stream tweets mentioning "#rstats" for 10 seconds
rstats1 <- stream_tweets("#rstats", timeout = 10, file_name = "rstats.json")
rstats1

# Download another 10s worth of data to the same file
rstats2 <- stream_tweets("#rstats", timeout = 10, file_name = "rstats.json",
                         append = TRUE)

# stream tweets about continental USA for 10 seconds
usa <- stream_tweets(location = lookup_coords("usa"), file_name = "usa.json",
                     timeout = 10)


## End(Not run)
```

`trends_available` *Available Twitter trends along with associated WOEID.*

Description

Available Twitter trends along with associated WOEID.

Usage

```
trends_available(token = NULL, parse = TRUE)
```

Arguments

`token`	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See `auth_as()` for details.
`parse`	If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

Value

Data frame with WOEID column. WOEID is a Yahoo! Where On Earth ID.

References

https://developer.twitter.com/en/docs/twitter-api/v1/trends/locations-with-trending-topics/api-reference/get-trends-available

See Also

Other trends: `get_trends()`

Examples

```
if (auth_has_default()) {
## Retrieve available trends
trends <- trends_available()
trends

}
```

---

ts_data *Converts tweets data into time series-like data object.*

Description

Returns data containing the frequency of tweets over a specified interval of time.

Usage

```
ts_data(data, by = "days", trim = 0L, tz = "UTC")
```

Arguments

data	Data frame or grouped data frame.
by	Desired interval of time expressed as numeral plus one of "secs", "mins", "hours", "days", "weeks", "months", or "years". If a numeric is provided, the value is assumed to be in seconds.
trim	Number of observations to trim off the front and end of each time series
tz	Time zone to be used, defaults to "UTC" (Twitter default)

Value

Data frame with time, n, and grouping column if applicable.

Examples

```
if (auth_has_default()) {

## handles of women senators
orgs <- c("_R_Foundation", "ropensci")

## get timelines for each
orgs_tml <- get_timeline(orgs, n = 100)

## get single time series for tweets
ts_data(orgs_tml)

## using weekly intervals
ts_data(orgs_tml, "weeks")
```

```
}
```

ts_plot *Plots tweets data as a time series-like data object.*

Description

Creates a ggplot2 plot of the frequency of tweets over a specified interval of time.

Usage

```
ts_plot(data, by = "days", trim = 0L, tz = "UTC", ...)
```

Arguments

`data`	Data frame or grouped data frame.
`by`	Desired interval of time expressed as numeral plus one of "secs", "mins", "hours", "days", "weeks", "months", or "years". If a numeric is provided, the value is assumed to be in seconds.
`trim`	The number of observations to drop off the beginning and end of the time series.
`tz`	Time zone to be used, defaults to "UTC" (Twitter default)
`...`	Other arguments passed to `ggplot2::geom_line()`.

Value

If ggplot2 is installed then a `ggplot2::ggplot()` plot object.

Examples

```
if (auth_has_default()) {
## search for tweets containing "rstats"
rt <- search_tweets("rstats", n = 100)

## plot frequency in 1 min intervals
ts_plot(rt, "mins")

## examine all Twitter activity using weekly intervals
ts_plot(rt, "hours")
}
```

tweets_with_users *Parsing data into tweets/users data tibbles*

Description

For internal use only

Usage

```
tweets_with_users(x)

users_with_tweets(x)
```

Arguments

x	A list of responses, with one element for each page.

Value

A tweets/users tibble with users/tweets attribute.

tweet_embed *Create a Tweet Embed*

Description

Twitter API GET call to retieve the tweet in embedded form.

Usage

```
tweet_embed(screen_name, status_id, ...)
```

Arguments

screen_name	character, screen name of the user
status_id	character, status id
...	parameters to pass to the GET call. See https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-oembed for details.

Value

character

See Also

httr::GET(),httr::content()

Examples

```
name   <- 'kearneymw'
status <- '1087047171306856451'

tweet_embed(screen_name = name, status_id = status)

tweet_embed(
 screen_name = name,
 status_id = status,
 hide_thread = TRUE,
 hide_media = FALSE,
 align = 'center'
)
```

---

tweet_shot   *Capture an image of a tweet/thread*

Description

Provide a status id or a full Twitter link to a tweet and this function will capture an image of the tweet — or tweet + thread (if there are Twitter-linked replies) — from the mobile version of said tweet/thread.

Usage

```
tweet_shot(statusid_or_url, zoom = 3, scale = TRUE)
```

Arguments

statusid_or_url	a valid Twitter status id (e.g. "947082036019388416") or a valid Twitter status URL (e.g. "https://twitter.com/jhollist/status/947082036019388416").
zoom	a positive number >= 1. See the help for [webshot::webshot()] for more information.
scale	auto-scale the image back to 1:1? Default it TRUE, which means magick will be used to return a "normal" sized tweet. Set it to FALSE to perform your own image manipulation.

Details

For this to work, you will need to ensure the packages in Suggests: are installed as they will be loaded upon the first invocation of this function.

Use the zoom factor to get more pixels which may improve the text rendering of the tweet/thread.

Value

magick object

Examples

```
## Not run:
if (auth_has_default()) {
    shot1 <- tweet_shot("947061504892919808")
    plot(shot1)
    shot2 <- tweet_shot("https://twitter.com/ma_salmon/status/947061504892919808")
    plot(shot2)
}

## End(Not run)
```

---

tweet_threading *Collect statuses contained in a thread*

Description

Return all statuses that are part of a thread (Replies from a user to their own tweets). By default the function traverses first backwards from the origin status_id of the thread up to the root, then checks if there are any child statuses that were posted after the origin status.

Usage

```
tweet_threading(tw, traverse = c("backwards", "forwards"), verbose = FALSE)
```

Arguments

tw	lookup_tweets() output containing at least the last status in the thread or an id of a tweet.
traverse	character, direction to traverse from origin status in tw. It is not recommended to change the default if you don't know at which point of a thread you are starting.
verbose	logical, output to console status of traverse.

Details

The backwards method looks up the tweet which is replying to, so it works if starting from the last tweet of the thread.

The forwards method looks for newer replies to the tweet provided. If the tweet doesn't have a reply it won't be able to find anything. The forwards method is limited by the timeline API (See get_timeline()).

Value

Tweets in a structure like lookup_tweets().

Examples

```
if (auth_has_default()) {
tw_thread <- tweet_threading("1461776330584956929")
tw_thread
}
```

---

users_data *Get tweets from users, or users from tweets*

Description

Twitter API endpoints that return tweets also return data about the users who tweeted, and most endpoints that return users also return their last tweet. Showing these additional columns would clutter the default display, so rtweet instead stores in special attributes and allows you to retrieve them with the `user_data()` and `tweets_data()` helpers.

Usage

```
users_data(tweets)

tweets_data(users)
```

Arguments

`tweets`	A data frame of tweets.
`users`	A data frame of users.

Value

`user_data()` returns a data frame of users; `tweets_data()` returns a data frame of tweets.

Examples

```
if (auth_has_default()) {
# find users from tweets
tweets <- search_tweets("r")
users_data(tweets)

# from tweets from users
users <- search_users("r")
tweets_data(users)
}
```

user_block — *Blocking or unblocking twitter users*

Description

`user_block(...)` blocks or unblocks a target twitter user. `user_unblock(...)` is synonymous to `user_block(..., unblock=TRUE)`.

Usage

```
user_block(user, unblock = FALSE, token = NULL)

user_unblock(user, token = NULL)
```

Arguments

user	Character vector of screen names or user ids. See as_screenname() for more details.
unblock	Logical indicating whether to unblock the intended friend.
token	Expert use only. Use this to override authentication for a single API call. In most cases you are better off changing the default for all calls. See auth_as() for details.

References

Block: https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/mute-block-report-use api-reference/post-blocks-create

Examples

```
## Not run:
user_block("rtweet_test")
user_unblock("rtweet_test")
user_block("rtweet_test", unblock=TRUE) #<-same as the above

## End(Not run)
```

write_as_csv — *Save Twitter data as a comma separated value file.*

Description

Saves as flattened CSV file of Twitter data.

Usage

```
write_as_csv(x, file_name, prepend_ids = TRUE, na = "", fileEncoding = "UTF-8")

save_as_csv(x, file_name, prepend_ids = TRUE, na = "", fileEncoding = "UTF-8")
```

Arguments

x	Data frame returned by an rtweet function.
file_name	Desired name to save file as. If `file_name` does not include the extension ".csv" it will be added automatically.
prepend_ids	Logical indicating whether to prepend an "x" before all Twitter IDs (for users, statuses, lists, etc.). It's recommended when saving to CSV as these values otherwise get treated as numeric and as a result the values are often less precise due to rounding or other class-related quirks. Defaults to true.
na	Value to be used for missing (NA)s. Defaults to empty character, "".
fileEncoding	Encoding to be used when saving to CSV. defaults to "UTF-8".

Value

Saved CSV files in current working directory.

See Also

Other datafiles: `flatten()`, `read_twitter_csv()`

Other datafiles: `flatten()`, `read_twitter_csv()`

Index

∗ **authentication**
auth_as, 4
auth_get, 5
auth_save, 6
auth_setup_default, 7
rtweet_user, 53
∗ **datafiles**
flatten, 11
read_twitter_csv, 52
write_as_csv, 72
∗ **friends**
lookup_friendships, 36
my_friendships, 39
∗ **geo**
lat_lng, 24
lookup_coords, 35
∗ **lists**
lists_members, 26
lists_statuses, 29
lists_subscribers, 31
lists_subscriptions, 32
lists_users, 34
∗ **parsing**
do_call_rbind, 10
∗ **post**
post_favorite, 43
post_follow, 44
post_friendship, 45
post_tweet, 49
∗ **premium endpoints**
search_fullarchive, 55
∗ **tokens**
get_token, 22
rate_limit, 51
∗ **trends**
get_trends, 22
trends_available, 65
∗ **ts_data**
ts_plot, 67
∗ **tweets**
get_favorites, 12
get_mentions, 17
get_timeline, 20
lists_statuses, 29
lookup_tweets, 37
search_tweets, 58
∗ **users**
as_screenname, 3
lists_subscribers, 31
lookup_users, 38
search_users, 61

as_screenname, 3, *32*, *39*, *62*
as_screenname(), *12*, *14*, *20*, *28*, *33*, *34*, *40*, *44*, *45*, *48*, *72*
as_userid *(*as_screenname*)*, 3
askpass::askpass(), *54*
auth_as, 4, *5–7*, *54*
auth_as(), *5*, *6*, *8*, *9*, *13*, *15*, *16*, *18*, *19*, *21*, *23*, *26*, *28*, *30*, *32–34*, *36–38*, *40*, *43*, *45*, *46*, *48*, *49*, *51*, *54*, *56*, *60*, *62*, *64*, *65*, *72*
auth_get, *4*, 5, *6*, *7*, *54*
auth_get(), *22*
auth_has_default *(*auth_setup_default*)*, 7
auth_list *(*auth_save*)*, 6
auth_save, *4*, *5*, 6, *7*, *54*
auth_save(), *4*, *7*, *54*
auth_setup_default, *4–6*, 7, *54*
auth_setup_default(), *4*
auth_sitrep, 7
auth_sitrep(), *4*, *6*

create_token, *22*, *52*

direct_messages, 8
do_call_rbind, 10

emojis, 11

favorite_tweet *(post_favorite)*, 43
flatten, 11, *52*, *73*
follow_user *(post_follow)*, 44
friendship_update *(post_friendship)*, 45

get_favorites, 12, *18*, *21*, *31*, *38*, *60*
get_followers, 14
get_friends, 15
get_mentions, *13*, 17, *21*, *31*, *38*, *60*
get_my_timeline *(get_timeline)*, 20
get_retweeters *(get_retweets)*, 19
get_retweets, 19
get_timeline, *13*, *18*, 20, *31*, *38*, *60*
get_timeline(), *70*
get_timelines *(get_timeline)*, 20
get_token, 22, *52*
get_tokens *(get_token)*, 22
get_trends, 22, *65*
ggplot2::geom_line(), *67*
ggplot2::ggplot(), *67*

httr::content(), *69*
httr::GET(), *69*

langs, 24
lat_lng, 24, *36*
lists_members, 26, *31*, *32*, *34*
lists_memberships, 27
lists_statuses, *13*, *18*, *21*, *27*, 29, *32*, *34*, *38*, *60*
lists_subscribers, *4*, *27*, *31*, 31, *34*, *39*, *62*
lists_subscriptions, *27*, *31*, *32*, 32, *34*
lists_users, *27*, *31*, *32*, *34*, 34
lookup_coords, *25*, 35
lookup_friendships, 36, *40*
lookup_statuses *(lookup_tweets)*, 37
lookup_tweets, *13*, *18*, *21*, *31*, 37, *60*
lookup_tweets(), *70*
lookup_users, *4*, *32*, 38, *62*

mute_user *(post_follow)*, 44
my_friendships, *37*, 39

network_data, 40
network_graph *(network_data)*, 40

parse_stream, 41
plain_tweets, 42
post_destroy, 42
post_favorite, 43, *45*, *46*, *50*
post_favourite *(post_favorite)*, 43
post_follow, *44*, 44, *46*, *50*
post_friendship, *44*, *45*, 45, *50*
post_list, 46
post_message, 48
post_mute *(post_follow)*, 44
post_status *(post_tweet)*, 49
post_tweet, *44–46*, 49
post_unfollow_user *(post_follow)*, 44

rate_limit, *22*, 51
rate_limit_reset *(rate_limit)*, 51
rate_limit_wait *(rate_limit)*, 51
read_twitter_csv, *12*, 52, *73*
round_time, 53
rtweet_app *(rtweet_user)*, 53
rtweet_app(), *4*, *6*, *7*
rtweet_bot, *7*
rtweet_bot *(rtweet_user)*, 53
rtweet_bot(), *4*, *6*
rtweet_user, *4–7*, 53
rtweet_user(), *4*, *6*

save_as_csv *(write_as_csv)*, 72
search_30day *(search_fullarchive)*, 55
search_fullarchive, 55
search_tweets, *13*, *18*, *21*, *31*, *38*, 58
search_tweets2 *(search_tweets)*, 58
search_users, *4*, *32*, *39*, 61
stopwordslangs, 63
stream_tweets, 63
stream_tweets(), *41*

trends_available, *23*, 65
trends_available(), *23*
ts_data, 66
ts_plot, 67
tweet_embed, 68
tweet_shot, 69
tweet_threading, 70
tweets_data *(users_data)*, 71
tweets_data(), *10*
tweets_with_users, 68

unflatten *(flatten)*, 11
unfollow_user *(post_follow)*, 44
user_block, 72
user_unblock *(user_block)*, 72
users_data, 71

users_data(), *10*
users_with_tweets (tweets_with_users), 68

write_as_csv, *12*, *52*, 72