Package ‘rtweet’

October 17, 2023

Type Package

Title Collecting Twitter Data

Version 1.2.1

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


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), htr2 (>= 0.2.2), 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), rmarkdown (>= 2.14), spelling (>= 2.2.1), testthat (>= 3.1.0), vcr (>= 0.6.0), webside (>= 0.5.3)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

Language en-US

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 2023-10-16 22:40:02 UTC

R topics documented:

- as_screenname .............................................. 4
- auth_as .................................................. 5
- auth_get ............................................... 6
- auth_save ............................................... 6
- auth_setup_default ...................................... 7
- auth_sitrep ............................................. 8
- clean_tweets ........................................... 9
- client_as .............................................. 9
- client_get ........................................... 10
- client_has_default .................................... 11
- client_save ........................................... 12
- direct_messages ....................................... 13
- do_call_rbind ........................................ 14
- emojis .................................................. 15
- entity .................................................... 16
- Expansions ............................................. 17
- Fields .................................................... 18
- flatten ............................................... 19
- get_favorites ......................................... 20
- get_followers ......................................... 22
- get_friends .......................................... 23
- get_mentions ......................................... 25
- get_retweets ......................................... 27
- get_timeline ......................................... 28
- get_token ............................................. 30
- get_trends ........................................... 30
- ids ..................................................... 32
- langs ............................................... 33
- lat_lng ............................................... 33
- links ................................................... 34
- lists_members ......................................... 35
- lists_memberships .................................... 37
- lists_statuses ...................................... 38
- lists_subscribers ................................... 40
- lists_subscriptions ................................ 42
- lists_users .......................................... 43
- list_followers ....................................... 44
R topics documented:

- list_get .................................................. 45
- list_members ........................................... 46
- list_membership ........................................ 48
- list_tweets .............................................. 49
- lookup.coords .......................................... 50
- lookup.friendships ................................. 51
- lookup.tweets ......................................... 52
- lookup.users ........................................... 53
- my_friendships ........................................ 55
- network_data .......................................... 55
- parse_stream .......................................... 57
- plain_tweets .......................................... 57
- post_destroy .......................................... 58
- post_favorite .......................................... 59
- post_follow ............................................ 60
- post_friendship ....................................... 61
- post_list ................................................ 62
- post_message .......................................... 63
- post_tweet ............................................. 64
- rate_limit .............................................. 66
- read_twitter.csv ..................................... 67
- retrieve_errors ....................................... 68
- round_time ............................................. 69
- rtweet_client .......................................... 69
- rtweet_user ............................................ 70
- rules ..................................................... 72
- search_fullarchive .................................. 73
- search_tweets ......................................... 76
- search_users .......................................... 79
- set_fields .............................................. 80
- set_scopes ............................................. 81
- stopwordslangs ....................................... 82
- stream ................................................. 83
- stream_tweets ........................................ 85
- trends_available ..................................... 87
- ts_data .................................................. 88
- ts_plot .................................................. 89
- tweets_with_users ................................... 90
- tweet_counts_recent ................................. 90
- tweet_delete .......................................... 91
- tweet_embed .......................................... 92
- tweet_get ............................................. 93
- tweet_liking_users .................................. 94
- tweet_post ............................................. 95
- tweet_quoted ......................................... 96
- tweet_retweeted_by .................................. 97
- tweet_search_all ..................................... 98
- tweet_search_recent ................................. 99
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

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

  # Look up user with name 5594077
  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

```r
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

```r
## 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()
auth_setup_default

**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**

```r
## 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**

```r
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

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

auth_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()
**clean_tweets**

*Clean text of tweets*

**Examples**

```r
auth_sitrep()
```

---

**Description**

Removes from the text, users mentions, hashtags, urls and media. Some urls or other text might remain if it is not recognized as an entity by the API.

**Usage**

```r
clean_tweets(x, clean = c("users", "hashtags", "urls", "media"))
```

**Arguments**

- `x` Tweets
- `clean` Type of elements to be removed.

**Value**

A vector with the text without the entities selected

**Examples**

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

  # tweets
  clean_tweets(tweets)
}
```

---

**client_as**

*Set default client for the current session*

**Description**

`client_as()` sets up the default client used by rtweet API calls with PKCE. See `rtweet_user()` to learn more about the three available authentication options.

**Usage**

```r
client_as(client = NULL)
```
Arguments

client  One of the following options:

- NULL, the default, will look for rtweet's "default" authentication which uses your personal default Twitter client. If it's not found, it will call `client_setup_default()` to set it up.
- A string giving the name of a saved auth file made by `client_save()`.
- A client object created by `rtweet_client()`.

Value

Invisibly returns the previous authentication mechanism.

See Also

Other client: `client_get()`, `client_has_default()`, `client_save()`

Examples

```r
## 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
client_as()
client_list()

## End(Not run)
```

---

### client_get

*Get the current client*

**Description**

If no client has been set up for this session, `client_get()` will call `client_as()` to set it up.

**Usage**

```r
client_get()
```

**Value**

The current client used.

**See Also**

Other client: `client_as()`, `client_has_default()`, `client_save()`
client_has_default

Examples

```r
## Not run:
client_get()

## End(Not run)
```

---

client_has_default  Set up default client

Description

You'll need to run this function once per computer so that rtweet can use your client.

Usage

```r
client_has_default()
client_setup_default()
```

Details

It will use the current default account for rtweet and save them as "rtweet". If a default is found it
will use it instead.

Value

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

See Also

Other client: `client_as()`, `client_get()`, `client_save()`

Examples

```r
## Not run:
if (!client_has_default()) {
  client_setup_default()
}

## End(Not run)
```
client_save

Save an authentication mechanism for use in a future session

Description

Use `client_save()` with `client_as()` to avoid repeatedly entering app credentials, making it easier to share auth between projects. Use `client_list()` to list all saved credentials.

Usage

```r
client_save(client)

client_list()
```

Arguments

- `client` A client `rtweet_client()`.

Details

The tokens are saved on the clients folder in `tools::R_user_dir("rtweet", "config")`.

Value

Invisible the path where the client is saved.

See Also

- `auth_sitrep()` to help finding and managing authentications.

Other client: `client_as()`, `client_get()`, `client_has_default()`

Examples

```r
## Not run:
# save app client for use in other sessions
client <- rtweet_client()
client_save(client)

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

## End(Not run)
```
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 Use this to override authentication for a single API call. In many 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,
do_call_rbind

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 solely 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


Examples

```r
## 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

```r
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 frame
  rt <- do_call_rbind(lrt)
  ## preview tweets data
  rt
  ## preview users data
  users_data(rt)
}
```

Description

This data comes from "Unicode.org", [https://unicode.org/emoji/charts/full-emoji-list.html](https://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.
Extract methods

Description

Extract entities of the tweets linked to a tweet id.

Usage

entity(x, entity, ...)

Arguments

x A tweets object of the rtweet package.
entity A entity to extract data from.
... Other possible arguments currently ignored.

Details

The position of where does this occur is not provided.

Value

Some information about those entities and the tweet id it comes from. for users mentions the ids of the mentioned users are "user_id", "user_id_str" (not "id_str")

Examples

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

  ## lookup tweets data for given statuses
  tw <- lookup_tweets(statuses)
  entity(tw, "urls")
}
```
Expansions

Description

Twitter parameters to add more fields on the returned values.

Usage

```r
set_expansions(
  tweet = tweet_expansions(),
  user = user_expansions(),
  list = list_expansions()
)
```

tweet_expansions(attachments = TRUE, referenced_tweets = TRUE)

user_expansions()

list_expansions()

Arguments

tweet, user, list

tweet_expansions(), user_expansions() and tweet_expansions().

attachments Add attachments values? Default yes.

referenced_tweets Add referenced_tweets values? Default yes.

Details

The set_expansions can be used to prepare the arguments for other rtweet functions.

Value

A character with the characters of valid expansions.

References


See Also

Fields, set_fields()
### Examples

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>tweet_expansions()</td>
</tr>
<tr>
<td>user_expansions()</td>
</tr>
<tr>
<td>set_expansions()</td>
</tr>
</tbody>
</table>

### Description

Arguments of expansion that select which values are returned. Fields are possible for:

- **Tweets**
- **Users**
- **Media**
- **Polls**
- **Places**
- **Lists**

### Usage

- media_fields
- place_fields
- poll_fields
- list_fields
- tweet_fields
- user_fields
- metrics_fields

### Format

- An object of class character of length 13.
- An object of class character of length 8.
- An object of class character of length 5.
- An object of class character of length 6.
- An object of class character of length 18.
- An object of class character of length 15.
- An object of class character of length 4.
flatten

References


See Also

Expansions, set_fields()

Examples

media_fields
place_fields
poll_fields
tweet_fields
user_fields

flatten
flattens list columns by pasting them into a single string for each observation. 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
‘unflatten’ splits on spaces to convert 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

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 observation. 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
get_favorites

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()

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.
get_favorites

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 solely 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  Use this to override authentication for a single API call. In many 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


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 solely on retryonratelimit because it does not handle other common failure modes like temporarily losing your internet connection.
get_friends

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 Use this to override authentication for a single API call. In many 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


Examples

```r
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

```r
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 solely 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

Use this to override authentication for a single API call. In many 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.
get_mentions

References

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.
### `get_mentions`

- **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 solely 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**
  Use this to override authentication for a single API call. In many 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


### See Also

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

### Examples

```r
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()` returns the 100 most recent users who retweeted them.

**Usage**

```r
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`: Use this to override authentication for a single API call. In many 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**

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 solely 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**
Use this to override authentication for a single API call. In many 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**

**See Also**
Other tweets: get_favorites(), get_mentions(), lists_statuses(), lookup_tweets(), search_tweets()
Examples

```r
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**: Use this to override authentication for a single API call. In many 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

```r
if (auth_has_default()) {

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

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

## Preview trends data
```r
ww_trends
```

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

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

## Preview trends data
```r
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
```r
(luk <- get_trends("london"))
```

---

### ids

**Extract the ids**

#### Description

Extract the ids of the rtweet data if present. Depending on the object type it returns either users ids, tweet ids or rules ids.

#### Usage

```r
ids(x, ...)
```

#### Arguments

- **x**: An object of the rtweet package.
- ...: Other arguments currently unused.

#### Examples

```r
if (auth_has_default()) {
  users <- lookup_users(c("twitter", "rladiesglobal", ".R_Foundation"))
  ids(users)
  followers <- get_followers(".R_Foundation")
  head(ids(followers))
  friends <- get_friends(".R_Foundation")
}
Defunct: Language codes recognized by Twitter data.

Description


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

```r
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

```r
if (auth_has_default()) {
  ## stream tweets sent from the US
  rt <- search_tweets(geocode = lookup_coords("usa"))

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

links

Create the links

Description

create the links from the rtweet data present. Depending on the object type it returns either users links, tweet links or rules links.

Usage

`links(x, ...)`

Arguments

x  An object of the rtweet package.
...

Other arguments currently unused.
Examples

```r
if (auth_has_default()) {
  users <- lookup_users(c("twitter", "rladiesglobal", ".R_Foundation"))
  links(users)
  followers <- get_followers(".R_Foundation")
  head(links(followers))
  friends <- get_friends(".R_Foundation")
  head(links(friends))
}
```

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

**Description**

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

**Usage**

```r
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.
lists_members

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
Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See \texttt{auth\_as()} for details.

retryonratelimit
If \texttt{TRUE}, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If \texttt{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, \texttt{NULL}, consults the option \texttt{rtweet.retryonratelimit} so that you can globally set it to \texttt{TRUE}, if desired.

If you expect a query to take hours or days to perform, you should not rely solely on \texttt{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 \texttt{TRUE}, the default, returns a tidy data frame. Use \texttt{FALSE} to return the “raw” list corresponding to the JSON returned from the Twitter API.

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

References


See Also

Other lists: \texttt{lists\_statuses()}, \texttt{lists\_subscribers()}, \texttt{lists\_subscriptions()}, \texttt{lists\_users()}

Examples

```r
if (auth_has_default()) {

## get list members for a list of rstats experts using list_id
(rstats <- lists_members(list_id = "1260528710559694850"))

## get list members for an rstats list using list topic slug
## list owner’s screen name
rstats <- lists_members(slug = "r-people", owner_user = "Lluis_Revilla")

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

```r
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**: Use this to override authentication for a single API call. In many 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.
By default, if `retryonratelimit` is `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 solely on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

Show progress bars and other messages indicating current progress?

[Deprecated] Please use `cursor` instead.

References


Examples

```r
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**

```r
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 re-
quested 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 re-
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 solely
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
Use this to override authentication for a single API call. In many 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

```r
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**

```r
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 solely 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**

Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

**References**


**See Also**

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

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

**Examples**

```r
if (auth_has_default()) {
  ## get subscribers of rladies list
  rstats <- lists_subscribers(slug = "rladies", 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 solely 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**: Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See `auth_as()` for details.

**References**


**See Also**

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

**Examples**

```r
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**: Use this to override authentication for a single API call. In many 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")
}

---

list_followers  List followers of a specified list

Description

Looks up the followers of a list.

Usage

list_followers(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

ids  A list id.
n  Number of users to query.
expansions  Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields  Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
...  Other parameters passed to the body of the request.
### list_get

**Description**

Looks up information about a list.

**Usage**

```r
list_get(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

- **token**
  
  These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.

- **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**
  
  A logical value to provide more information about paginated queries.

**Value**

A data.frame with the user information of who is following the list: id, name, and username. Other information depends on the expansions and fields requested. Accepted values are:

- **Expansions:** `set_expansions(tweet = NULL, list = NULL)`.
- **Fields:** `set_fields(media = NULL, poll = NULL, place = NULL, list = NULL)`.

**References**


**Examples**

```r
if (FALSE) {
  lf <- list_followers("1150793074420998146")
}
```
list_members

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ids</td>
<td>A list id.</td>
</tr>
<tr>
<td>n</td>
<td>Number of users to query.</td>
</tr>
<tr>
<td>expansions</td>
<td>Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with <code>set_expansions()</code>).</td>
</tr>
<tr>
<td>fields</td>
<td>Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with <code>set_fields()</code>).</td>
</tr>
<tr>
<td>...</td>
<td>Other parameters passed to the body of the request.</td>
</tr>
<tr>
<td>token</td>
<td>These endpoints only accept a bearer token (can be created via <code>rtweet_app()</code>). In most cases you are better of changing the default for all calls via <code>auth_as()</code>.</td>
</tr>
<tr>
<td>parse</td>
<td>If TRUE, the default, returns a tidy data frame. Use FALSE to return the &quot;raw&quot; list corresponding to the JSON returned from the Twitter API.</td>
</tr>
<tr>
<td>verbose</td>
<td>A logical value to provide more information about paginated queries.</td>
</tr>
</tbody>
</table>

Value

A data.frame with the user information of who is included in the list: id, name, and username. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: `set_expansions(tweet = NULL, user = NULL)`
- Fields: `set_fields(place = NULL, poll = NULL, media = NULL, tweet = NULL)`.

References


Examples

```r
if (FALSE) {
    lg <- list_get("1306285118877831168")
}
```

---

list_members  

List of members from a specified List

Description

Looks up the users of a list.
Usage

list_members(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ..., 
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

ids
A list id.
n
Number of users to query.
expansions
Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields
Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
...
Other parameters passed to the body of the request.
token
These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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
A logical value to provide more information about paginated queries.

Value

A data.frame with the user information of who is included in the list: id, name, and username. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: set_expansions(tweet = NULL, list = NULL)
- Fields: set_fields(place = NULL, poll = NULL, media = NULL, list = NULL).

References


Examples

if (FALSE) {
  lm <- list_members("1306285118877831168")
}
list_membership

Lists a specified user is a member of.

Description

Lists a specified user is a member of.

Usage

```r
list_membership(
  ids,  
  n = 100,  
  expansions = NULL,  
  fields = NULL,  
  ...,  
  token = NULL,  
  parse = TRUE,  
  verbose = FALSE
)
```

Arguments

- **ids**: A list id.
- **n**: Number of users to query.
- **expansions**: Set `NULL` to not use any expansion, set `NA` to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set `NULL` to not use any field, get all allowed fields with `NA`, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other parameters passed to the body of the request.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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**: A logical value to provide more information about paginated queries.

Value

A `data.frame` with information of the list: `id, name`.

Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: `set_expansions(tweet = NULL, user = NULL)`.
- Fields: `set_fields(place = NULL, poll = NULL, media = NULL)`.
References


Examples

```r
if (FALSE) {
  lm <- list_membership("20815041")
}
```

---

**list_tweets**  
*Lists tweets of a specified list*

**Description**

Looks up the followers of a list.

**Usage**

```r
list_tweets(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

**Arguments**

- `ids`  
  A list id.
- `n`  
  Number of users to query.
- `expansions`  
  Set `NULL` to not use any expansion, set `NA` to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- `fields`  
  Set `NULL` to not use any field, get all allowed fields with `NA`, provide a list with the fields you want (create it with `set_fields()`).
- `...`  
  Other parameters passed to the body of the request.
- `token`  
  These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- `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`  
  A logical value to provide more information about paginated queries.
Value
A data.frame with the user information of who is following the list: edit_history_tweet_ids, id and
text. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: set_expansions(list = NULL).
- Fields: set_fields(list = NULL).

References
get-lists-id-tweets

Examples
if (FALSE) {
  lt <- list_tweets("1150793074420998146")
}

lookup_coords
Get coordinates of specified location.

Description
Convenience function for looking up latitude/longitude coordinate information for a given loca-
tion. 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 world and the following two larger locations are baked into this function: ‘world’ and ‘usa’. If ‘world’ is supplied then a bounding box of maximum latitude/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

```r
## 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.
lookup_tweets

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 Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References


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 Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.
lookup_users

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 re-
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 solely
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
 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",
    "44032224407314432"
  )

  ## 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).
lookup_users

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 Use this to override authentication for a single API call. In many 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 solely 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


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   Look up 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  Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References


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.
post_destroy

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  Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References


Examples

```r
if (auth_has_default()) {
  pt <- post_tweet("Running #rtweet examples")
  post_destroy(ids(pt))
}
```
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         Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References


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

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

```r
post_unfollow_user(user, token = NULL)
```

```r
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**
  - Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See `auth_as()` for details.

### References

post_friendship

See Also

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

Examples

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

Description

Updates friendship notifications and retweet abilities.

Usage

```r
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**: Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See `auth_as()` for details.

References


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 Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

Value

Response object from HTTP request.

References

Examples

```r
# Not run:

## R related Twitter accounts

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

## view list in browser at https://twitter.com/<user_name>/lists/r-accounts

## 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)
## view list in browser at https://twitter.com/<user_name>/lists/r-accounts

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)
## view list in browser at https://twitter.com/<user_name>/lists/r-accounts

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

## End(Not run)
```

---

**post_message**

*Posts direct message from user’s Twitter account*
post_tweet

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 Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References

api-reference/new-event

post_tweet

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
)

References

api-reference/new-event
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
Use this to override authentication for a single API call. In many 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


See Also

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

```r
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 = "1", yaxt = "1",
      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
pt <- post_tweet(status="first in a thread")
reply_id <- ids(pt)

# post reply
post_tweet("second in the thread",
            in_reply_to_status_id = reply_id)
}
```
### 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` Use this to override authentication for a single API call. In many 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


### See Also

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

### Examples

```r
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 hashtags 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

```r
## Not run:
## read in data.csv
rt <- read_twitter_csv("data.csv")
```

Description
Expose errors of the response

Usage
retrieve_errors(expr = NULL)

Arguments
expr An expression that might cause an error. If NULL it looks for the last error.

Examples

```r
if (FALSE){
  new_rule <- stream_add_rule(list(value = "rstats", tag = "rstats1"))
  stream_add_rule(list(value = "rstats", tag = "rstats2")) # ERROR
  # See the full information provided by the API:
  retrieve_errors(stream_add_rule(list(value = "rstats", tag = "rstats2")))
  retrieve_errors()
}
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"))

Description

Set up your client mechanism for the Twitter API.

Usage

rtweet_client(client_id, client_secret, app, scopes = NULL)
Arguments

client_id, client_secret

Application OAuth client ID and client Secret. These are generally not required for `rtweet_user()` since the defaults will use the built-in rtweet app.

app

Name of the client, it helps if you make it match with the name of your app. On the Twitter app the Callback URI must be `http://127.0.0.1:1410/` (the trailing / must be included).

scopes

Default scopes allowed for users using this client. Leave NULL to allow everything or choose yours with `set_scopes()`.

See Also

scopes

Examples

```r
if (interactive()) {
  rtweet_client()
}
```

rtweet_user | Authentication options

Description

Authenticate methods to use the Twitter API. See the instructions in `vignette("auth", package = "rtweet")`.

Usage

```r
rtweet_user(
  client_id = NULL,
  client_secret = NULL,
  api_key = client_id,
  api_secret = client_secret
)
```

```r
rtweet_bot(api_key, api_secret, access_token, access_secret, app = "rtweet")
```

```r
rtweet_app(bearer_token)
```

```r
rtweet_oauth2(client = NULL, scopes = NULL)
```
Arguments

client_id, client_secret  
Application OAuth client ID and client Secret. These are generally not required for `rtweet_user()` since the defaults will use the built-in rtweet app.

api_key, api_secret  
API key and secret. Deprecated in favor of client_* arguments.

access_token, access_secret  
Access token and secret.

app  
Name of the application you are building.

bearer_token  
App bearer token.

client  
Which client app will be used, see `rtweet_client()` for details.

scopes  
The permissions of the app, see `set_scopes()` for details. By default it uses the client’s scopes. Provided here in case you want to modify them.

Details

There are four 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.
- `rtweet_oauth2()` authenticates as a user using a client. This authentication is required in some endpoints.

To use `rtweet_app()`, `rtweet_bot()` or `rtweet_oauth2()` you will need to create your own Twitter app following the instructions in vignette("auth", package = "rtweet"). `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.

Value

If the validation is successful the OAuth token. For `rtweet_app()` a `rtweet_bearer`.

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.
## rules

### Extract the streaming rules

#### Description

Provides the information about the rules

#### Usage

```r
rules(x, 
```

#### Arguments

- `x`: An object returned by `stream_*_rule`
- `...`: Other arguments currently ignored.

#### See Also

- `stream_add_rule()` and `stream_rm_rule()`.
search_fullarchive

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 and the "Developer Account" section.

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 `retry_on_rate_limit = 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**  
Use this to override authentication for a single API call. In many 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](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
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**

```r
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).

type

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 Use this to override authentication for a single API call. In many 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 solely 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


See Also

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

Examples

```r
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**
Use this to override authentication for a single API call. In many 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**


**See Also**

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

**Examples**

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

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

---

### set_fields

**Create fields**

**Description**

Choose which fields are used, by default all are returned. Usually all the first 3 are accepted together and the last two too.
set_scopes

Usage

set_fields(
    media = media_fields,
    poll = poll_fields,
    tweet = tweet_fields,
    place = place_fields,
    user = user_fields,
    list = list_fields
)

Arguments

media        The fields you want from media_fields.
poll         The fields you want from poll_fields.
tweet        The fields you want from tweet_fields.
place        The fields you want from place_fields.
user         The fields you want from user_fields.
list         The fields you want from list_fields.

Value

A list with the fields requested ready to be used in your requests to the API.

See Also

Fields

Examples

set_fields()
set_fields(media = NULL)
set_fields(place = NULL, user = NULL)

---

set_scopes

Scopes of the OAuth2 token

Description

Permissions given to a token of a Twitter account. By default it allows everything.

Usage

set_scopes(read = TRUE, write = TRUE, tweet_moderate = TRUE, regenerate = TRUE)
stopwordslangs

Arguments

read Allow to read.
write Allow to write/manage?
tweet_moderate Allow to hide or show replies to your Tweets.
regenerate Allow to use the token for more than 2 hours.

Value

A character with all the possible scopes or those allowed.

References

https://developer.twitter.com/en/docs/authentication/oauth-2-0/authorization-code

Examples

set_scopes()

stopwordslangs Defunct: Twitter stop words in multiple languages data.

Description

This data comes from 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

Description
Open a streaming connection with Twitter and stores tweets for as long as you wish.

Usage

```r
filtered_stream(
  timeout,
  file = tempfile(),
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  append = TRUE,
  parse = TRUE
)
```

```r
stream_add_rule(query, dry = FALSE, token = NULL)
```

```r
stream_rm_rule(query, dry = FALSE, token = NULL)
```

```r
c sample_stream(
  timeout,
  file = tempfile(),
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  append = TRUE
)
```

Arguments

- **timeout**: time, in seconds, of the recording stream.
- **file**: Path to a file where the raw streaming should be stored.
- **expansions**: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other parameters passed to the body of the request.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
append  Append streaming to the file? Default does but it is recommended to have a new file for each call.

dry  Check if the addition or removal of the rule works.

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.

query  If NULL returns the current rules, else depending:

• In stream_add_rule it should be a list of value and tag.
• In stream_rm_rule it should be a vector of ids of rules to be removed

Details
The connection can be left open as long as you wish, the data is appended to the file provided. Be aware that the stream might have incomplete records (you won’t be able to read directly from the json file). One tweet might belong to multiple rules.

Value
The records in the streaming.

Functions
• filtered_stream(): Start a filtered stream according to the rules.
• stream_add_rule(): Add rules for the filtered streaming.
• stream_rm_rule(): Remove rules from the filtered streaming
• sample_stream(): Retrieve a sample of the tweets posted.

See Also
ids()

Examples
# Requires a bearer token
if (FALSE) {
  # How many rules do we have
  stream_add_rule(NULL)
  # Add new rule
  new_rule <- stream_add_rule(list(value = "#rstats", tag = "rstats"))
  new_rule
  # Open filtered streaming connection for 30s
  filtered_stream(file = tempfile(), timeout = 30, parse = FALSE)
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).
stream_tweets

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
Use this to override authentication for a single API call. In many 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 other arguments passed in to query parameters.

Value
A tibble with one row per tweet

References
They were removed from the website.
The webpages describing how it used to work were removed.

See Also
parse_stream().

Examples

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

# 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 Use this to override authentication for a single API call. In many 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

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

```r
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

```r
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**

```r
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**

```r
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_counts_recent   Count tweets

Description
Count tweets

Usage
tweet_counts_recent(query, ..., token = NULL, parse = TRUE, verbose = FALSE)
tweet_counts_all(query, ..., token = NULL, parse = TRUE, verbose = FALSE)

Arguments
query One query for matching Tweets.
... Other arguments passed to the API.
token These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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 A logical value to provide more information about the paginated queries (if any) and to store the data of each page.
Tweet Delete

Value

The number of tweets for a given granularity

References


Examples

```r
if (FALSE) {
  tcr <- tweet_counts_recent(query = "#rtweet", parse = FALSE)
  tca <- tweet_counts_all(query = "#rtweet", parse = FALSE)
}
```

---

tweet_delete Delete tweet

Description

Will delete a tweet

Usage

tweet_delete(id, verbose = FALSE, token = NULL)

Arguments

id At least a tweet id.
verbose A logical value to provide more information about the paginated queries (if any) and to store the data of each page.
token This endpoint accepts a OAuth2.0 authentication (can be created via `rtweet_oauth2()` or a bearer token (can be created via `rtweet_app()`).

References


See Also

tweet_post(), tweet_search_recent(), user_timeline()
Examples

```r
if (FALSE) {
    # It requires OAuth authentication
    tp <- tweet_post("Running examples of #rtweet")
    td <- tweet_delete(tp$id)
}
```

tweet_embed

Create a Tweet Embed

Description

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

Usage

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

Arguments

- `screen_name`: character, screen name of the user
- `status_id`: character, status id

Value

character

See Also

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

Examples

```r
name <- 'kearneywm'
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_get  

Get tweet information

Description

Look up tweets up to 100 at the same time.

Usage

tweet_get(
  id,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

id  
  At least a tweet id.

expansions  
  Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).

fields  
  Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).

...  
  Other arguments passed to the API.

token  
  This endpoint accepts a OAuth2.0 authentication (can be created via rtweet_oauth2()) or a bearer token (can be created via rtweet_app()).

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  
  A logical value to provide more information about the paginated queries (if any) and to store the data of each page.

References


See Also

lookup_tweets()
Examples

```r
if (FALSE){
  tweet_get("567053242429734913", parse = FALSE)
  tweet_get(c("567053242429734913", "567053242429734913"), parse = FALSE)
  tweet_get(c("567053242429734913", "567053242429734913"), parse = TRUE)
}
```

### Description

Looks up who have liked a given tweet.

### Usage

```r
tweet_liking_users(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

### Arguments

- **id** A tweet id string.
- **n** Number of tweets to query.
- **expansions** Set `NULL` to not use any expansion, set `NA` to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields** Set `NULL` to not use any field, get all allowed fields with `NA`, provide a list with the fields you want (create it with `set_fields()`).
- **...** Other arguments passed to the API.
- **token** These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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** A logical value to provide more information about paginated queries.

### References

Examples

if (FALSE) {
    tlu <- tweet_liking_users("567053242429734913", n = Inf, verbose = TRUE)
}

---

tweet_post Post a tweet

Description

This function uses the API v2 to post tweets.

Usage

tweet_post(text, ..., token = NULL)

Arguments

text Text of the tweet.

... Other accepted arguments.

token This endpoint accepts a OAuth2.0 authentication (can be created via `rtweet_oauth2()`) or a bearer token (can be created via `rtweet_app()`).

References


Examples

if (FALSE) {
    # It requires the OAuth2.0 Authentication
    tp_id <- tweet_post("Posting from #rtweet with the basic plan")
    tweet_post()
}
tweet_quoted | Get quoted tweet information

**Description**

Look up tweets quoting that tweet id.

**Usage**

```r
tweet_quoted(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

**Arguments**

- `id`: At least a tweet id.
- `n`: Number of tweets to query.
- `expansions`: Set `NULL` to not use any expansion, set `NA` to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- `fields`: Set `NULL` to not use any field, get all allowed fields with `NA`, provide a list with the fields you want (create it with `set_fields()`).
- `...`: Other arguments passed to the API.
- `token`: This endpoint accepts a OAuth2.0 authentication (can be created via `rtweet_oauth2()`) or a bearer token (can be created via `rtweet_app()`).
- `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`: A logical value to provide more information about the paginated queries (if any) and to store the data of each page.

**References**


**See Also**

`lookup_tweets()` `tweet_get()`
Examples

```r
if (FALSE) {
  tweet_quoted("1631945769748930561", parse = FALSE)
}
```

Description

Looks up who have retweeted a given tweet.

Usage

```r
tweet_retweeted_by(
  ids, n = 100, expansions = NULL, fields = NULL,
  ..., token = NULL, parse = TRUE,
  verbose = FALSE
)
```

Arguments

- `ids` A tweet id string.
- `n` Number of tweets to query.
- `expansions` Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- `fields` Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- `...` Other arguments passed to the API.
- `token` These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- `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` A logical value to provide more information about the paginated queries (if any) and to store the data of each page.

Value

A data frame with the user information of who retweeted it: id, name, and username. Other information depends on the expansions and fields requested.
References


Examples

if (FALSE) {
  rb <- tweet_retweeted_by("567053242429734913")
}

tweet_search_all Search in the Twitter archive

Description

Search in the Twitter archive

Usage

tweet_search_all(
  query,
  n = 500,
  expansions = NULL,
  fields = NULL,
  ..., 
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

query One query for matching Tweets.
n Number of tweets to query.
expansions Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
... Other arguments passed to the API.
token These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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 A logical value to provide more information about the paginated queries (if any) and to store the data of each page.
Note

OAuth2.0 requires tweet.read and users.read permissions.

References


Examples

```r
if (FALSE) {
  sa <- tweet_search_all("#rtweet", parse = FALSE)
}
```

tweet_search_recent  Search recent tweets

Description

Look up tweets from the last seven days that match a search query.

Usage

```r
tweet_search_recent(
  query,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

Arguments

- **query**: One query for matching Tweets.
- **n**: Number of tweets to query.
- **expansions**: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other arguments passed to the API.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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**: A logical value to provide more information about the paginated queries (if any) and to store the data of each page.
Note
OAuth2.0 requires tweet.read and users.read permissions.

References

Examples
if (FALSE) {
  sr <- tweet_search_recent("#rtweet", sort_order = "relevancy", parse = FALSE)
}

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

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

```r
## 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)
```

### 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

```r
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()`.
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 them in special attributes and allows you to show 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)
  full_search <- cbind(tweets, users_data(tweets))

  # from tweets from users
  users <- search_users("r")
  tweets_data(users)
  full_users <- cbind(users, 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 Use this to override authentication for a single API call. In many cases you are better off changing the default for all calls. See auth_as() for details.

References


Examples

if (auth_has_default()) {
  user_block("rtweet")
  user_unblock("rtweet")
  user_block("rtweet", unblock=TRUE) #<-same as the above
}

user_blocked

Find followers.

Description

List of users that follow the specified user ID.
Usage

```r
user_blocked(
  id,
  n = 1000,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

Arguments

- `id`: A user id string.
- `n`: Number of tweets to query.
- `expansions`: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- `fields`: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- `...`: Other arguments passed to the API.
- `token`: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- `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`: A logical value to provide more information about paginated queries.

References


Examples

```r
if (FALSE) {
  uf <- user_blocked("1599030512919650304", verbose = TRUE)
}
```

---

**user_bookmarks**

**Retrieve user bookmarks**

**Description**

Collects the 800 most recent bookmarked tweets of a user.
user_bookmarks

Usage

user_bookmarks(id,
               n = 100,
               ...,
               expansions = NULL,
               fields = NULL,
               parse = TRUE,
               token = NULL,
               verbose = FALSE)

Arguments

id Twitter user id: character string identifying your account.
n Number of tweets to retrieve.
... Other arguments passed down to the API.
expansions Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
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 This endpoint only accept a OAuth2.0 authentication (can be created via rtweet_oauth2()).
verbose A logical value

Value

A data.frame with the user information of who is following the list: edit_history_tweet_ids, id and text. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: set_expansions(list = NULL).
- Fields: set_fields(list = NULL).

Note

This endpoint requires a OAuth2.0 authentication, with tweet.read, users.read and bookmark.read permissions.

References


See Also

rtweet_oauth2(), user_self()
Examples

```r
if (FALSE) {
  # Requires token_oa2
  ub <- user_bookmarks(user_self()$id, parse = FALSE, n = Inf, token = token_oa2)
}
```

---

**user_by_username**

*Search users by username*

**Description**

Looks up users by their username.

**Usage**

```r
user_by_username(
  username,
  expansions = NULL,
  fields = NULL,
  ...
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

**Arguments**

- **username**: A user name string or up to 100.
- **expansions**: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other arguments passed to the API.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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**: A logical value to provide more information about paginated queries.

**References**

user_followers

See Also
user_search()

Examples
if (FALSE) {
  user_by_username("rOpenSci")
  user_by_username(c("Bioconductor", "R_Contributors"))
}

user_followers  Find followers.

Description
List of users that follow the specified user ID.

Usage
user_followers(
id,    
n = 100,   
expansions = NULL, 
fields = NULL,   
...,
token = NULL,  
parse = TRUE,  
verbose = FALSE)

Arguments
id   A user id string.
n   Number of tweets to query.
expansions Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
...   Other arguments passed to the API.
token These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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 A logical value to provide more information about paginated queries.
References


Examples

if (FALSE) {
  uf <- user_followers("1599030512919650304", verbose = TRUE)
}


user_following  Find which users are being followed.

Description

List of users the specified user ID is following.

Usage

user_following(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...
)

Arguments

id  A user id string.
n  Number of users to query.
expansions  Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields  Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
...  Other arguments passed to the API.
token  These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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  A logical value to provide more information about paginated queries.
user_liked_tweets

References


Examples

```r
if (FALSE) {
  uf <- user_following("1599030512919650304", verbose = TRUE)
}
```

---

user_liked_tweets  

**Liked tweets from a user**

Description

Looks up tweets liked by a user.

Usage

```r
user_liked_tweets(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

Arguments

- **id**: A tweet id string.
- **n**: Number of tweets to query.
- **expansions**: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other arguments passed to the API.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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**: A logical value to provide more information about paginated queries.
user_lists

References


Examples

if (FALSE) {
  ult <- user_liked_tweets("1599030512919650304", verbose = TRUE)
}

user_lists

Search users by username

Description

Looks up users by their username.

Usage

user_lists(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

ids              A user name string or up to 100.
n               Number of users to query.
expansions      Set NULL to not use any expansion, set NA to get all expansions, or provide a
                vector with the expansions you want (create it with set_expansions()).
fields          Set NULL to not use any field, get all allowed fields with NA, provide a list with
                the fields you want (create it with set_fields()).
...              Other parameters passed to the body of the request.
token           These endpoints only accept a bearer token (can be created via rtweet_app()).
                In most cases you are better of changing the default for all calls via auth_as().
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          A logical value to provide more information about paginated queries.
**user_list_follows**

**References**


**Examples**

```r
if (FALSE) {
  ul <- user_lists("1051050384")
}
```

---

**user_list_follows** *Lists a specified user follows*

**Description**

Looks up lists a user follows.

**Usage**

```r
user_list_follows(
  ids,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ..., 
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```

**Arguments**

- `ids` A list id.
- `n` Number of users to query.
- `expansions` Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- `fields` Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
- `...` Other parameters passed to the body of the request.
- `token` These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- `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` A logical value to provide more information about paginated queries.
user_mentions

References


Examples

if (FALSE) {
  ulf <- user_list_follows("1051050384")
}

user_mentions               Tweets mentioning a user

Description

Looks up to 800 tweets mentioning a user.

Usage

user_mentions(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,  
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

id       A user id string.
n       Number of tweets to query.
expansions Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
... Other arguments passed to the API.
token These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better off of changing the default for all calls via auth_as().
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 A logical value to provide more information about paginated queries.
user_muted

References


Examples

if (FALSE) {
  um <- user_mentions("1599030512919650304", verbose = TRUE)
}

user_muted List muted users

Description

Looks up the muted users.

Usage

user_muted(
  ids,
  n = 1000,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)

Arguments

ids A list id.
n Number of users to query.
expansions Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with set_expansions()).
fields Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with set_fields()).
... Other parameters passed to the body of the request.
token These endpoints only accept a bearer token (can be created via rtweet_app()). In most cases you are better of changing the default for all calls via auth_as().
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 A logical value to provide more information about paginated queries.
Value

A data.frame with the user information of who is following the list: id, name, and username. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: set_expansions(tweet = NULL, list = NULL).
- Fields: set_fields(media = NULL, poll = NULL, place = NULL, list = NULL).

References


See Also

user_self()

Examples

```r
if (FALSE) {
  um <- user_muted(user_self()$id)
}
```

---

**user_search**

Search users

Description

Looks up users.

Usage

```r
user_search(
  ids,
  expansions = NULL,
  fields = NULL,
  ...
)
```

Arguments

- **ids**: A user id string or up to 100.
- **expansions**: Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).
Other arguments passed to the API.

token

These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.

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

A logical value to provide more information about paginated queries.

Value

A data.frame with the id, name and username of the accounts. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: `set_expansions(tweet = NULL, list = NULL)`.
- Fields: `set_fields(media = NULL, poll = NULL, place = NULL)`.

References


See Also

`user_by_username()`

Examples

```r
if (FALSE) {
  us <- user_search(c("1599030512919650304", "2244994945"), verbose = TRUE)
}
```

---

**user_self**

Tweets from a user

Description

Looks up tweets posted by a user.

Usage

```r
user_self(
  expansions = NULL,
  fields = NULL,
  ..., 
  token = NULL,
  parse = TRUE,
  verbose = FALSE
)
```
Arguments

expansions  Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).

fields  Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).

...  Other arguments passed to the API.

token  These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.

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  A logical value to provide more information about paginated queries.

Value

A data.frame with the id, name and username of the authenticated user. Other information depends on the expansions and fields requested. Accepted values are:

- Expansions: `set_expansions(tweet = NULL, list = NULL)`.
- Fields: `set_fields(media = NULL, poll = NULL, place = NULL)`

References


Examples

```r
if (FALSE) {
  me <- user_self()
}
```

user_timeline  User timeline

Description

Looks up the timeline of a user with up to 800 tweets in the last 7 days.

Usage

```r
user_timeline(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...
  token = NULL,
)```
user_tweets

```r
parse = TRUE,
verbose = FALSE
)
```

Arguments

- **id**: A tweet id string.
- **n**: Number of tweets to query.
- **expansions**: Set `NULL` to not use any expansion, set `NA` to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).
- **fields**: Set `NULL` to not use any field, get all allowed fields with `NA`, provide a list with the fields you want (create it with `set_fields()`).
- **...**: Other arguments passed to the API.
- **token**: These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.
- **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**: A logical value to provide more information about paginated queries.

References


Examples

```r
if (FALSE) {
  ut <- user_timeline("1599030512919650304", verbose = TRUE)
}
```

---

**user_tweets**

*Tweets from a user*

**Description**

Looks up tweets posted by a user.

**Usage**

```r
user_tweets(
  id,
  n = 100,
  expansions = NULL,
  fields = NULL,
  ...,
  token = NULL,
```
Arguments

id
  A user id string.

n
  Number of tweets to query.

expansions
  Set NULL to not use any expansion, set NA to get all expansions, or provide a vector with the expansions you want (create it with `set_expansions()`).

fields
  Set NULL to not use any field, get all allowed fields with NA, provide a list with the fields you want (create it with `set_fields()`).

... Other arguments passed to the API.

token
  These endpoints only accept a bearer token (can be created via `rtweet_app()`). In most cases you are better of changing the default for all calls via `auth_as()`.

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
  A logical value to provide more information about paginated queries.

References


Examples

```r
if (FALSE) {
  ut <- user_tweets("1599030512919650304", verbose = TRUE)
}
```

---

write_as_csv

Save Twitter data as a comma separated value file.

Description

Saves as flattened CSV file of Twitter data.

Usage

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

```r
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()`
Index

* authentication
  auth_as, 5
  auth_get, 6
  auth_save, 6
  auth_setup_default, 7
  rtweet_user, 70
* client
  client_as, 9
  client_get, 10
  client_has_default, 11
  client_save, 12
* datafiles
  flatten, 19
  read_twitter_csv, 67
  write_as_csv, 118
* datasets
  Fields, 18
* friends
  lookup_friendships, 51
  my_friendships, 55
* geo
  lat_lng, 33
  lookup_coords, 50
* lists
  lists_members, 35
  lists_statuses, 38
  lists_subscribers, 40
  lists_subscriptions, 42
  lists_users, 43
* parsing
  do_call_rbind, 14
* post
  post_favorite, 59
  post_follow, 60
  post_friendship, 61
  post_tweet, 64
* premium endpoints
  search_fullarchive, 73
* tokens
  get_token, 30
  rate_limit, 66
* trends
  get_trends, 30
  trends_available, 87
* ts_data
  ts_plot, 89
* tweets
  get_favorites, 20
  get_mentions, 25
  get_timeline, 28
  lists_statuses, 38
  lookup_tweets, 52
  search_tweets, 76
* users
  as_screenname, 4
  lists_subscribers, 40
  lookup_users, 53
  search_users, 79
  as_screenname, 4, 41, 54, 80
  as_screenname(), 20, 22, 28, 37, 42, 43, 55, 60, 61, 64, 103
  as_userid(as_screenname), 4
  askpass::askpass(), 71
  auth_as, 5, 6–8, 72
  auth_get, 5, 6, 7, 8, 72
  auth_get(), 30
  auth_has_default(auth_setup_default), 7
  auth_list(auth_save), 6
  auth_save, 5, 6, 6, 8, 72
  auth_save(), 5, 7, 71
  auth_setup_default, 5–7, 7, 72
  auth_setup_default(), 5
  auth_sitrep, 8

120
auth_sitrep(), 5, 7, 12

clean_tweets, 9
client_as, 9, 10–12
client_as(), 10, 12
client_get, 10, 11, 12
client_has_default, 10, 11, 12
client_list (client_save), 12
client_save, 10, 11, 12
client_save(), 10
client_setup_default
  (client_has_default), 11
client_setup_default(), 10
create_token, 30, 67
direct_messages, 13
do_call_rbind, 14

emojis, 15
entity, 16
Expansions, 17, 19
expansions (Expansions), 17

favorite_tweet (post_favorite), 59
Fields, 17, 18
fields (Fields), 18
filtered_stream (stream), 83
flatten, 19, 68, 119
follow_user (post_follow), 60
friendship_update (post_friendship), 61

get_favorites, 20, 26, 29, 40, 53, 78
get_followers, 22
get_friends, 23
get_mentions, 21, 25, 29, 40, 53, 78
get_my_timeline (get_timeline), 28
get_retweeters (get_retweets), 27
get_retweets, 27
get_timeline, 21, 26, 28, 40, 53, 78
get_timeline(), 101
get_timelines (get_timeline), 28
get_token, 30, 67
get_tokens (get_token), 30
get_trends, 30, 87
ggplot2::geom_line(), 89
ggplot2::ggplot(), 89
httr::content(), 92
httr::GET(), 92

ids, 32
ids(), 84
langs, 33
lat_lng, 33, 51
links, 34
list_expansions (Expansions), 17
list_fields (Fields), 18
list_followers, 44
list_get, 45
list_members, 46
list_membership, 48
list_tweets, 49
lists_members, 35, 40, 41, 43, 44
lists_memberships, 37
lists_statuses, 21, 26, 29, 36, 38, 41, 43, 44, 53, 78
lists_subscribers, 4, 36, 40, 43, 44, 54, 80
lists_subscriptions, 36, 40, 41, 42, 44
lists_users, 36, 40, 41, 43, 43
lookup_coords, 34, 50
lookup_friendships, 51, 55
lookup_statuses (lookup_tweets), 52
lookup_tweets, 21, 26, 29, 40, 52, 78
lookup_tweets(), 93, 96, 101
lookup_users, 4, 41, 53, 80

media_fields (Fields), 18
metrics_fields (Fields), 18
mute_user (post_follow), 60
my_friendships, 52, 55

network_data, 55
network_graph (network_data), 55

parse_stream, 57
parse_stream(), 86
place_fields (Fields), 18
plain_tweets, 57
poll_fields (Fields), 18
post_destroy, 58
post_favorite, 59, 61, 65
post_favourite (post_favorite), 59
post_follow, 59, 60, 61, 65
post_friendship, 59, 61, 61, 65
post_list, 62
post_message, 63
post_mute (post_follow), 60
INDEX

post_status (post_tweet), 64
post_tweet, 59, 61, 64
post_unfollow_user (post_follow), 60
rate_limit, 30, 66
rate_limit_reset (rate_limit), 66
rate_limit_wait (rate_limit), 66
read_twitter_csv, 20, 67, 119
retrieve_errors, 68
round_time, 69
rtweet_app (rtweet_user), 70
rtweet_bot, 7
rtweet_bot (rtweet_user), 70
rtweet_bot(), 5, 7
rtweet_client, 69
rtweet_client(), 10, 12, 71, 72
rtweet_oauth2 (rtweet_user), 70
rtweet_oauth2(), 91, 93, 95, 96, 105
rtweet_user, 5–8, 70
rtweet_user(), 5, 7, 9
rules, 72
sample_stream (stream), 83
save_as_csv (write_as_csv), 118
search_30day (search_fullarchive), 73
search_fullarchive, 73
search_tweets, 21, 26, 29, 40, 53, 76
search_tweets2 (search_tweets), 76
search_users, 4, 41, 54, 79
set_expansions (Expansions), 17
set_fields, 80
set_fields(), 17, 19, 44, 46–49, 83, 93, 94, 96–99, 104–114, 116–118
set_scopes, 81
set_scopes(), 71
stopwordslangs, 82
stream, 83
stream_add_rule (stream), 83
stream_rm_rule (stream), 83
stream_tweets, 85
stream_tweets(), 57
trends_available, 31, 87
trends_available(), 31
ts_data, 88
ts_plot, 89
tweet_counts_all (tweet_counts_recent), 90
tweet_counts_recent, 90
tweet_delete, 91
tweet_embed, 92
tweet_expansions (Expansions), 17
tweet_expansions(), 17
tweet_fields (Fields), 18
tweet_get, 93
tweet_get(), 96
tweet_liking_users, 94
tweet_post, 95
tweet_post(), 91
tweet_quoted, 96
tweet_retweeted_by, 97
tweet_search_all, 98
tweet_search_recent, 99
tweet_search_recent(), 91
tweet_shot, 100
tweet_threading, 101
tweets_data (users_data), 102
tweets_data(), 15
tweets_with_users, 90
unflatten (flatten), 19
unfollow_user (post_follow), 60
user_block, 103
user_blocked, 103
user_bookmarks, 104
user_by_username, 106
user_by_username(), 115
user_expansions (Expansions), 17
user_expansions(), 17
user_fields (Fields), 18
user_followers, 107
user_following, 108
user_liked_tweets, 109
user_list_follows, 111
user_lists, 110
user_mentions, 112
user_muted, 113
user_search, 114
user_search(), 107
user_self, 115
user_self(), 105, 114
user_timeline, 116
user_timeline(), 91
user_tweets, 117
user_unblock (user_block), 103
users_data, 102
users_data(), 15
users_with_tweets (tweets_with_users), 90

write_as_csv, 20, 68, 118