Package 'rtweet'

October 26, 2017

Type Package

Title Collecting Twitter Data

Version 0.5.22

Date 2017-10-25

Description An implementation of calls designed to extract and organize Twitter data via Twitter's REST and stream APIs. Functions formulate and send API requests, convert response objects to more user friendly data structures---e.g., data frames---and provide some aesthetically pleasing visualizations for exploring the data.

Depends R (>= 3.1.0)

Imports bit64,
httr (>= 1.0.0),
jsonlite,
magrittr,
methods,
openssl,
readr,
tibble

License MIT + file LICENSE

LazyData TRUE

URL https://CRAN.R-project.org/package=rtweet

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

RoxygenNote 6.0.1

Suggests ggplot2,
knitr,
parallel,
rmarkdown,
testthat

VignetteBuilder knitr

R topics documented:

`as_screenname` *Coerces user identifier(s) to be evaluated as a screen name(s).*

Description

Coerces user identifier(s) to be evaluated as a screen name(s).

Usage

```
as_screenname(x)

as_userid(x)
```

Arguments

x	A vector consisting of one or more Twitter user identifiers (i.e., screen names or user IDs).

Details

Default rtweet function behaviors will treat "1234" as a user ID, but the inverse (i.e., treating "2973406683" as a screen name) should rarely be an issue. However, in those cases, users may need to mix both screen names and user IDs. To do so, make sure to combine them as a list (and not a character vector, which will override conflicting user identifier classes). See examples code for example of mixing user IDs with screen names. Note: this only works with certain functions, e.g., get_friends, get_followers.

Value

A vector of class screen_name or class user_id

See Also

Other users: lists_subscribers, lookup_users, search_users, tweets_with_users, users_data

Examples

```
## Not run:
## get friends list for user with the handle "1234"
get_friends(as_screenname("1234"))

## as_screenname coerces all elements to class "screen_name"
sns <- as_screenname(c("kearneymw", "1234", "jack"))
class(sns)

## print will display user class type
sns

## BAD: combine user id and screen name using c()
users <- c(as_userid("2973406683"), as_screenname("1234"))
class(users)

## GOOD: combine user id and screen name using list()
users <- list(as_userid("2973406683"), as_screenname("1234"))
users

## get friend networks for each user
get_friends(users)


## End(Not run)
```

create_token *Creating Twitter authorization token(s).*

Description

Sends request to generate oauth 1.0 tokens. Twitter also allows users to create user-only (oauth 2.0) access token. Unlike the 1.0 tokens, oauth 2.0 tokens are not at all centered on a host user. Which means these tokens cannot be used to send information (follow requests, Twitter statuses, etc.). If you have no interest in those capabilities, then 2.0 oauth tokens do offer some higher rate limits. At the current time, the difference given the functions in this package is trivial, so I have yet to verified oauth 2.0 token method. Consequently, I encourage you to use 1.0 tokens.

Usage

```
create_token(app = "mytwitterapp", consumer_key, consumer_secret,
  cache = TRUE)
```

Arguments

app	Name of user created Twitter application
consumer_key	Application API key
consumer_secret	Application API secret User-owned app must have Read and write access level and Callback URL of http://127.0.0.1:1410.
cache	Logical indicating whether to cache the token as a .httr-oauth file. The default is TRUE, which means the cached token file will be added to the user's working directory. Ideally, users will store their token as an environment variable (see the tokens vignette for instructions), but the cache file works as long as always return to the same working directory.

Value

Twitter oauth token(s) (Token1.0).

See Also

https://developer.twitter.com/en/docs/basics/authentication/overview/oauth

Other tokens: get_tokens, rate_limit

emojis *emojis data*

Description

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

Usage

```
emojis
```

Format

A tibble with two variables and 2,623 observations.

Examples

```
emojis
```

`get_favorites` *Get tweets data for statuses favorited by one or more target users.*

Description

Get tweets data for statuses favorited by one or more target users.

Usage

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

Arguments

`user`	Vector of user names, user IDs, or a mixture of both.
`n`	Specifies the number of records to retrieve. Defaults to 200. 3000 is the max number of favorites returned per token. Due to suspended or deleted content, this function may return fewer tweets than the desired (n) number. Must be of length 1 or of length equal to the provided number of users.
`since_id`	Returns results with an status_id greater than (that is, more recent than) the specified status_id. There are limits to the number of tweets returned by the REST API. If the limit is hit, since_id is adjusted (by Twitter) to the oldest ID available.
`max_id`	Returns results with status_id less (older) than or equal to (if hit limit) the specified status_id.
`parse`	Logical, indicating whether to return parsed vector or nested list (fromJSON) object. By default, `parse = TRUE` saves you the time [and frustrations] associated with disentangling the Twitter API return objects.
`token`	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

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

See Also

Other tweets: `get_timeline`, `lists_statuses`, `lookup_statuses`, `search_tweets`, `tweets_data`, `tweets_with_users`

Examples

```
## Not run:
## get max number of statuses favorited by KFC
kfc <- get_favorites("KFC", n = 3000)
kfc

## get 400 statuses favorited by each of three users
favs <- get_favorites(c("Lesdoggg", "pattonoswalt", "meganamram"))
favs

## End(Not run)
```

`get_followers` *Get user IDs for accounts following target user(s).*

Description

Get user IDs for accounts following target user(s).

Usage

```
get_followers(user, n = 5000, page = "-1", retryonratelimit = FALSE,
  parse = TRUE, verbose = TRUE, token = NULL)
```

Arguments

`user`	Screen name or user ID of target user from which the user IDs of followers will be retrieved.
`n`	Number of followers to return. Defaults to 5000, which is the max number of followers returned by a single API request. Twitter allows up to 15 of these requests every 15 minutes, which means 75,000 is the max number of followers to return without waiting for the rate limit to reset. If this number exceeds either 75,000 or the remaining number of possible requests for a given token, then the returned object will only return what it can (less than n) unless retryonratelimit is set to true.
`page`	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object. If `parse = TRUE` then the cursor value can be extracted from the return object by using the `next_cursor` function.
`retryonratelimit`	If you'd like to retrieve more than 75,000 followers in a single call, then set `retryonratelimit = TRUE` and this function will use base `Sys.sleep` until rate limits reset and the desired n is achieved or the number of total followers is exhausted. This defaults to FALSE. See details for more info regarding possible issues with timing misfires.
`parse`	Logical, indicating whether to return parsed vector or nested list (fromJSON) object. By default, `parse = TRUE` saves you the time [and frustrations] associated with disentangling the Twitter API return objects.
`verbose`	Logical indicating whether or not to print messages. Only relevant if retryonratelimit = TRUE. Defaults to TRUE, prints sleep times and followers gathered counts.

token — OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Details

When `retryonratelimit = TRUE` this function internally makes a rate limit API call to get information on (a) the number of requests remaining and (b) the amount of time until the rate limit resets. So, in theory, the sleep call should only be called once between waves of data collection. However, as a fail safe, if a system's time is calibrated such that it expires before the rate limit reset, or if, in another session, the user dips into the rate limit, then this function will wait (use Sys.sleep for a second time) until the next rate limit reset. Users should monitor and test this before making especially large calls as any systematic issues could create sizable inefficiencies.

Value

A tibble data frame of follower IDs (one column named "user_id").

See Also

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

Other ids: get_friends, max_id

Examples

```
## Not run:
## get 5000 ids of users following the KFC account
(kfc <- get_followers("KFC"))

## get max number [per fresh token] of POTUS follower IDs
(pres <- get_followers("potus", n = 75000))

## resume data collection (warning: rate limits reset every 15 minutes)
pres2 <- get_followers("potus", n = 75000, page = next_cursor(pres))

## store next cursor in object before merging data
nextpage <- next_cursor(pres2)

## merge data frames
pres <- rbind(pres, pres2)

## store next cursor as an attribute in the merged data frame
attr(pres, "next_cursor") <- next_page

## view merged ddata
pres


## End(Not run)
```

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

Description

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

Usage

```
get_friends(users, n = 5000, retryonratelimit = FALSE, page = "-1",
  parse = TRUE, verbose = TRUE, token = NULL)
```

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	Number of friends (user IDs) to return. Defaults to 5,000, which is the maximum returned by a single API call. Users are limited to 15 of these requests per 15 minutes. Twitter limits the number of friends a user can have to 5,000. To follow more than 5,000 accounts (to have more than 5 thousand "friends") accounts must meet certain requirements (e.g., a certain ratio of followers to friends). Consequently, the vast majority of users follow fewer than five thousand accounts. This function has been oriented accordingly (i.e., it assumes the maximum value of n is 5000). To return more than 5,000 friends for a single user, call this function multiple times with requests after the first using the page parameter.
retryonratelimit	If you'd like to retrieve 5,000 or fewer friends for more than 15 target users, then set `retryonratelimit = TRUE` and this function will use base `Sys.sleep` until rate limits reset and the desired number of friend networks is retrieved. This defaults to FALSE. See details for more info regarding possible issues with timing misfires.
page	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object. This is only relevant if a user has over 5000 friends (follows more than 5000 accounts).
parse	Logical, indicating whether to return parsed vector or nested list (fromJSON) object. By default, `parse = TRUE` saves you the time [and frustrations] associated with disentangling the Twitter API return objects.
verbose	Logical indicating whether or not to include output messages. Defaults to TRUE, which includes printing a success message for each inputted user.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Details

When `retryonratelimit = TRUE` this function internally makes a rate limit API call to get information on (a) the number of requests remaining and (b) the amount of time until the rate limit resets. So, in theory, the sleep call should only be called once between waves of data collection.

However, as a fail safe, if a system's time is calibrated such that it expires before the rate limit reset, or if, in another session, the user dips into the rate limit, then this function will wait (use Sys.sleep for a second time) until the next rate limit reset. Users should monitor and test this before making especially large calls as any systematic issues could create sizable inefficiencies.

Value

A tibble data frame with two columns, "user" for name or ID of target user and "user_id" for follower IDs.

See Also

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

Other ids: get_followers, max_id

Examples

```
## Not run:
## get user ids of accounts followed by Donald Trump
(djt <- get_friends("realDonaldTrump"))

## get user ids of accounts followed by (friends) KFC, Trump, and Nate Silver.
(fds <- get_friends(c("kfc", "jack", "NateSilver538")))


## End(Not run)
```

---

get_mentions *GET statuses/mentions_timeline*

Description

Returns the 20 most recent mentions (Tweets containing a users's screen_name) for the authenticating user. The timeline returned is the equivalent of the one seen when you view your mentions on twitter.com. This method can only return up to 800 tweets.

Returns the 20 most recent mentions (Tweets containing a users's screen_name) for the authenticating user. The timeline returned is the equivalent of the one seen when you view your mentions on twitter.com. This method can only return up to 800 tweets.

Usage

```
get_mentions(n = 100, since_id = NULL, max_id = NULL, ..., parse = TRUE,
  token = NULL)

get_mentions_call(n = 100, since_id = NULL, max_id = NULL, ...,
  parse = TRUE, token = NULL)
```

Arguments

n	Specifies the number of Tweets to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. We include retweets in the count, even if include_rts is not supplied. It is recommended you always send include_rts=1 when using this API method.
since_id	Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.
max_id	Returns results with an ID less than (that is, older than) or equal to the specified ID.
...	Other args passed as parameters in composed API query.
parse	Logical indicating whether to convert the response object into an R list. Defaults to TRUE.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

Tibble of mentions data.

A tibble of mentions data.

---

get_retweeters *GET statuses/retweeters/ids*

Description

Returns a collection of up to 100 user IDs belonging to users who have retweeted the Tweet specified by the id parameter.

Usage

```
get_retweeters(status_id, n = 100, cursor = "-1", parse = TRUE,
  token = NULL)
```

Arguments

status_id	required The status ID of the desired status.
n	Specifies the number of records to retrieve. Best if intervals of 100.
cursor	semi-optional Causes the list of IDs to be broken into pages of no more than 100 IDs at a time. The number of IDs returned is not guaranteed to be 100 as suspended users are filtered out after connections are queried. If no cursor is provided, a value of -1 will be assumed, which is the first "page." The response from the API will include a previous_cursor and next_cursor to allow paging back and forth. See our cursor docs for more information. While this method supports the cursor parameter, the entire result set can be returned in a single cursored collection. Using the count parameter with this method will not provide segmented cursors for use with this parameter.

parse — Logical indicating whether to convert the response object into an R list. Defaults to TRUE.

token — OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

data

`get_retweets` *GET statuses/retweets/:id*

Description

Returns a collection of the 100 most recent retweets of the Tweet specified by the id parameter. Twitter's API is currently limited to 100 or fewer retweeters.

Usage

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

Arguments

status_id — required The numerical ID of the desired status.

n — optional Specifies the number of records to retrieve. Must be less than or equal to 100.

parse — Logical indicating whether to convert the response object into an R list. Defaults to TRUE.

... — Other arguments used as parameters in the query sent to Twitter's rest API, for example, `trim_user = TRUE`.

token — OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

data

`get_timeline` *Get one or more user timelines (tweets posted by target user(s)).*

Description

Get one or more user timelines (tweets posted by target user(s)).

Usage

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

Arguments

`user`	Vector of user names, user IDs, or a mixture of both.
`n`	Number of tweets to return per timeline. Defaults to 100. Must be of length 1 or equal to length of user.
`max_id`	Character, status_id from which returned tweets should be older than.
`home`	Logical, indicating whether to return a user-timeline or home-timeline. By default, home is set to FALSe, which means `get_timeline` returns tweets posted by the given user. To return a user's home timeline feed, that is, the tweets posted by accounts followed by a user, set the home to false.
`parse`	Logical, indicating whether to return parsed (data.frames) or nested list (fromJSON) object. By default, `parse = TRUE` saves users from the time [and frustrations] associated with disentangling the Twitter API return objects.
`check`	Logical indicating whether to remove check available rate limit. Ensures the request does not exceed the maximum remaining number of calls. Defaults to TRUE.
`token`	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
`...`	Futher arguments passed on as parameters in API query.

Value

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

See Also

https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline

Other tweets: `get_favorites`, `lists_statuses`, `lookup_statuses`, `search_tweets`, `tweets_data`, `tweets_with_users`

Examples

```
## Not run:
## get most recent 3200 tweets posted by Donald Trump's account
djt <- get_timeline("realDonaldTrump", n = 3200)

## data frame where each observation (row) is a different tweet
djt

## users data for realDonaldTrump is also retrieved
users_data(djt)

## retrieve timelines of mulitple users
tmls <- get_timeline(c("KFC", "ConanOBrien", "NateSilver538"), n = 1000)

## it's returned as one data frame
tmls

## count observations for each timeline
table(tmls$screen_name)


## End(Not run)
```

`get_tokens` *Fetching Twitter authorization token(s).*

Description

Call function used to fetch and load Twitter oauth tokens. Since Twitter app key should be stored privately, users should save the path to token(s) as an environment variable. This allows Tokens to be instantly [re]loaded in future sessions. See the "tokens" vignette for instructions on obtaining and using access tokens.

Usage

```
get_tokens()

get_token()
```

Details

This function will search for tokens using R, internal, and global environment variables (in that order).

Value

Twitter oauth token(s) (Token1.0).

See Also

Other tokens: `create_token`, `rate_limit`

Examples

```
## Not run:
## fetch default token(s)
token <- get_tokens()

## print token
token


## End(Not run)
```

`get_trends` *Get Twitter trends data.*

Description

Get Twitter trends data.

Usage

```
get_trends(woeid = 1, lat = NULL, lng = NULL, exclude = 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. Latitutde/longitude coordinates should be provided as woeid value consisting of 2 numeric values or via one lat value and one lng 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 providded for woeid, this function will coerce the first value to lat.
`lng`	Optional alternative to woeid. Numeric, longitude in degrees. If two coordinates are providded for woeid, this function will coerce the second value to lng.
`exclude`	Logical, indicating whether or not to exclude hashtags
`token`	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
`parse`	Logical, indicating whether or not to parse return trends data. Defaults to true.

Value

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

See Also

Other trends: `trends_available`

Examples

```
## Not run:
## Retrieve available trends
trends <- trends_available()
trends

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

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

## Preview trends data
ww_trends

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

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

## Preview trends data
nyc_trends

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

## End(Not run)
```

`langs` *langs data*

Description

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

Usage

```
langs
```

Format

A tibble with five variables and 486 observations.

Examples

```
langs
```

lat_lng *lat_lng*

Description

Computes lat and lng variables using all available geolocation information and adds those variables to the provided data frame.

Usage

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

Arguments

x	Parsed tweets 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 c("bbox_coords", "coords_coords", "geo_coords") (which are the default column names of data returned by most status-oriented rtweet functions).

Details

On occassion 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 lng and lat vars.

See Also

Other geo: lookup_coords

Examples

```
## Not run:
## stream tweets sent from the US
rt <- stream_tweets(lookup_coords("usa"), timeout = 10)

## use lat_lng to recover full information geolocation data
rtll <- lat_lng(rt)

## plot points
with(rtll, plot(lng, lat))


## End(Not run)
```

lists_members *Get the members of a specified Twitter list.*

Description

Get the members of a specified Twitter list.

Get the lists a specified user has been added to.

Usage

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

lists_memberships(user, n = 20, cursor = "-1",
  filter_to_owned_lists = FALSE, token = NULL, 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 who owns the list being requested by a slug.
n	Specifies the number of results to return per page (see cursor below). The default is 20, with a maximum of 5,000.
cursor	optional Breaks the results into pages. Provide a value of -1 to begin paging. Provide values as returned in the response body's next_cursor and previous_cursor attributes to page back and forth in the list.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
parse	Logical indicating whether to convert the response object into an R list. Defaults to TRUE.
...	Other args used as parameters in query composition.
user	The user id or screen_name of the user for whom to return results for.
filter_to_owned_lists	When set to true . t or 1 , will return just lists the authenticating user owns, and the user represented by user_id or screen_name is a member of.

Details

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.

Value

Either a nested list (if parsed) or an http response object.

See Also

Other lists: lists_statuses, lists_subscribers, lists_users

Examples

```
## Not run:
## get list memebers for a list of polling experts using list_id
(pollsters <- lists_members("105140588"))

## get list members for a rstats list using list slug and the screen name
## of the list owner
rstats <- lists_members(slug = "rstats", owner_user = "scultrera")
rstats


## End(Not run)
## Not run:
## get up to 200 list memberships of Nate Silver
ns538 <- lists_memberships("NateSilver538", n = 200)
ns538


## End(Not run)
```

---

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

Description

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

Usage

```
lists_statuses(list_id = NULL, slug = NULL, owner_user = NULL,
  since_id = NULL, max_id = NULL, n = 5000, include_rts = TRUE,
  parse = TRUE, token = NULL)
```

Arguments

list_id	required The numerical id of the list.
slug	required You can identify a list by its slug instead of its numerical id. If you decide to do so, note that you'll also have to specify the list owner using the owner_id or owner_screen_name parameters.
owner_user	optional The screen name or user ID of the user who owns the list being requested by a slug.
since_id	optional Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.
max_id	optional Returns results with an ID less than (that is, older than) or equal to the specified ID.

n	optional Specifies the number of results to retrieve per "page."
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	Logical indicating whether to convert the response object into an R list. Defaults to TRUE.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

data

See Also

Other lists: `lists_members`, `lists_subscribers`, `lists_users`

Other tweets: `get_favorites`, `get_timeline`, `lookup_statuses`, `search_tweets`, `tweets_data`, `tweets_with_users`

---

lists_subscribers *GET lists/subscribers*

Description

Returns the subscribers of the specified list. Private list subscribers will only be shown if the authenticated user owns the specified list.

Usage

```
lists_subscribers(list_id = NULL, slug = NULL, owner_user = NULL,
  n = 20, cursor = "-1", parse = 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_user parameters.
owner_user	optional The screen name or user ID of the user who owns the list being requested by a slug.
n	optional Specifies the number of results to return per page (see cursor below). The default is 20, with a maximum of 5,000.
cursor	semi-optional Causes the collection of list members to be broken into "pages" of consistent sizes (specified by the count parameter). If no cursor is provided, a value of -1 will be assumed, which is the first "page." The response from the API will include a previous_cursor and next_cursor to allow paging back and forth. See Using cursors to navigate collections for more information.

parse	Logical indicating whether to convert the response object into an R list. Defaults to TRUE.
token	OAuth token. By default token = NULL fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send ?tokens to console).

See Also

Other lists: lists_members, lists_statuses, lists_users

Other users: as_screenname, lookup_users, search_users, tweets_with_users, users_data

Examples

```
## Not run:
## get subscribers of new york times politics list
rstats <- lists_subscribers(slug = "new-york-times-politics", owner_user = "nytpolitics", n = 1000)


## End(Not run)
```

---

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, reverse = FALSE, token = NULL, parse = TRUE)
```

Arguments

user	The ID of the user or screen name for whom to return results. Helpful for disambiguating when a valid user ID is also a valid screen name.
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	OAuth token. By default token = NULL fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send ?tokens to console).
parse	Logical indicating whether to convert the response object into an R list. Defaults to TRUE.

Value

data

See Also

Other lists: lists_members, lists_statuses, lists_subscribers

Examples

```
## Not run:
## get lists subsribed to by Nate Silver
lists_users("NateSilver538")

## End(Not run)
```

---

lookup_coords *lookup_coords*

Description

Returns coordinates using google geocode api

Usage

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

Arguments

address	Desired location typically in the form of placename, 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.
...	Additional args passed as parameters in the http reqeust

Value

Object of class coords.

See Also

Other geo: lat_lng

Examples

```
## Not run:
## get coordinates associated with the following addresses/components
sf <- lookup_coords("san francisco, CA", "country:US")
usa <- lookup_coords("usa")
lnd <- lookup_coords("london")
bz <- lookup_coords("brazil")

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

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


## End(Not run)
```

lookup_friendships *lookup_friendships*

Description

Look up informaiton on friendship between authenticated user and up to 100 users.

Usage

```
lookup_friendships(user, parse = TRUE, token = NULL)
```

Arguments

user	Screen name or user id of target user.
parse	Logical indicating whether to return parsed data frame. Defaults to true.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

Data frame converted form returned json object. If parse is not true, the httr response object is returned instead.

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

Description

Get tweets data for given statuses (status IDs).

Usage

```
lookup_statuses(statuses, token = NULL, parse = TRUE)
```

Arguments

statuses	User id or screen name of target user.
token	OAuth token (1.0 or 2.0). By default `token = NULL` fetches a non-exhausted token from an environment variable @describeIn tokens.
parse	Logical, indicating whether or not to parse return object into data frame(s).

Value

json response object (max is 18000 per token)

See Also

https://dev.twitter.com/overview/documentation

Other tweets: get_favorites, get_timeline, lists_statuses, search_tweets, tweets_data, tweets_with_users

Examples

```
## Not run:
# lookup tweets data via status_id vector
statuses <- c("567053242429734913", "266031293945503744",
  "440322224407314432")
statuses <- lookup_statuses(statuses)
statuses

# view users data for these statuses via tweets_data()
users_data(statuses)

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

users	User id or screen name of target user.
token	OAuth token (1.0 or 2.0). By default token = NULL fetches a non-exhausted token from an environment variable @describeIn tokens.
parse	Logical, indicating whether or not to parse return object into data frame(s).

Value

json response object

See Also

https://dev.twitter.com/overview/documentation

Other users: as_screenname, lists_subscribers, search_users, tweets_with_users, users_data

Examples

```
## Not run:
## lookup vector of 1 or more user_id or screen_name
users <- c("potus", "hillaryclinton", "realdonaldtrump",
  "fivethirtyeight", "cnn", "espn", "twitter")

## get users data
usr_df <- lookup_users(users)

## view users data
usr_df

## view tweet data for these users via tweets_data()
tweets_data(usr_df)

## End(Not run)
```

---

max_id *next_cursor/max_id*

Description

Method for returning next value (used to request next page or results) object returned from Twitter APIs.

Usage

```
max_id(x)

next_cursor(x)
```

Arguments

x	Data object returned by Twitter API.

Value

Character string of next cursor value used to retrieved the next page of results. This should be used to resume data collection efforts that were interrupted by API rate limits. Modify previous data request function by entering the returned value from `next_cursor` for the `page` argument.

See Also

Other ids: `get_followers`, `get_friends`

Other extractors: `tweets_data`, `users_data`

Examples

```
## Not run:
## Retrieve user ids of accounts following POTUS
f1 <- get_followers("potus", n = 75000)

## store next_cursor in page
page <- next_cursor(f1)

## max. number of ids returned by one token is 75,000 every 15
##   minutes, so you'll need to wait a bit before collecting the
##   next batch of ids
sys.Sleep(15 * 60) ## Suspend execution of R expressions for 15 mins

## Use the page value returned from \code{next_cursor} to continue
##   where you left off.
f2 <- get_followers("potus", n = 75000, page = page)

## combine
f <- do.call("rbind", list(f1, f2))

## count rows
nrow(f)

## End(Not run)
```

---

my_direct_messages *GET direct_messages*

Description

Returns the 20 most recent direct messages sent to the authenticating user. Includes detailed information about the sender and recipient user. You can request up to 200 direct messages per call, and only the most recent 200 DMs will be available using this endpoint. Important: This method requires an access token with RWD (read, write & direct message) permissions.

Usage

```
my_direct_messages(since_id = NULL, max_id = NULL, count = 200,
  parse = TRUE, token = NULL)
```

Arguments

since_id	optional Returns results with an ID greater than (that is, more recent than) the specified ID. There are limits to the number of Tweets which can be accessed through the API. If the limit of Tweets has occured since the since_id, the since_id will be forced to the oldest ID available.
max_id	optional Returns results with an ID less than (that is, older than) or equal to the specified ID.
count	optional Specifies the number of direct messages to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of Tweets to return because suspended or deleted content is removed after the count has been applied.

parse	Logical indicating whether to convert response object into nested list. Defaults to true.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

Value

Return object converted to nested list. If status code of response object is not 200, the response object is returned directly.

`parse_stream` *Converts Twitter stream data (json file) into parsed data frame.*

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`.
...	Other arguments passed on to interal data_from_stream function.

Value

A tbl of tweets data with attribute of users data

See Also

Other stream tweets: `stream_tweets`

Examples

```
## Not run:
## run and save stream to json file
stream_tweets(
  "the,a,an,and", timeout = 60,
  file_name = "theaanand.json",
  parse = FALSE
)

## parse stream file into tibble data frame
rt <- parse_stream("theaanand.json")

## End(Not run)
```

plain_tweets *plain_tweets*

Description

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

Usage

```
plain_tweets(x)
```

Arguments

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

Value

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

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	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable tokens.

See Also

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

Examples

```
## Not run:
rt <- search_tweets("rstats")
r <- lapply(rt$user_id, post_favorite)

## End(Not run)
```

---

post_follow *Follows target twitter user.*

Description

Follows target twitter user.

Usage

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

post_unfollow_user(user, token = NULL)

post_mute(user, token = NULL)
```

Arguments

user	Screen name or user id of target user.
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	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable tokens.

See Also

Other post: post_favorite, post_friendship, post_tweet

Examples

```
## Not run:
post_follow("BarackObama")

## End(Not run)
```

post_friendship *Updates friendship notifications and retweet abilities.*

Description

Updates friendship notifications and retweet abilities.

Usage

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

Arguments

user	Screen name or user id of target user.
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	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable tokens.

See Also

Other post: post_favorite, post_follow, post_tweet

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

Description

Posts direct message from user's Twitter account

Usage

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

Arguments

text	Character, text of message.
user	Screen name or user ID of message target.
media	File path to image or video media to be included in tweet.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable tokens.

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

Description

Posts status update to user's Twitter account

Usage

```
post_tweet(status = "my first rtweet #rstats", media = NULL, token = NULL,
  in_reply_to_status_id = NULL)
```

Arguments

status	Character, tweet status. Must be 140 characters or less.
media	File path to image or video media to be included in tweet.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable tokens.
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.

See Also

Other post: post_favorite, post_follow, post_friendship

Examples

```
## Not run:
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))
tmp <- tempfile(fileext = "png")
png(tmp, 6, 6, "in", res = 127.5)
par(tcl = -.15, family = "Inconsolata",
    font.main = 2, bty = "n", xaxt = "l", yaxt = "l",
    bg = "#f0f0f0", mar = c(3, 3, 2, 1.5))
plot(x, y, xlab = NULL, ylab = NULL, pch = 21, cex = 1,
     bg = bg, col = col,
     main = "This image was uploaded by rtweet")
grid(8, lwd = .15, lty = 2, col = "#00000088")
dev.off()
browseURL(tmp)
post_tweet(".Call(\"oops\", ...)",
           media = tmp)

# example of replying within a thread
post_tweet(status="first in a thread")
my_timeline <- get_timeline(self_user_name, n=1, token=twitter_token)
reply_id <- my_timeline[1,]$status_id
post_tweet(status="second in the thread", in_reply_to_status_id=reply_id)
```

```
## End(Not run)
```

`rate_limit` *Get rate limit data for given Twitter access tokens.*

Description

Get rate limit data for given Twitter access tokens.

Usage

```
rate_limit(token = NULL, query = NULL, parse = TRUE)

rate_limits(token = NULL, query = NULL, parse = TRUE)
```

Arguments

`token`	One or more OAuth tokens. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
`query`	Specific API (path) or a character function name, e.g., `query = "get_timelines"`, used to subset the returned data. If null, this function returns entire rate limit request object as a tibble data frame. Otherwise, query returns specific values matching the query of interest; e.g., `query = "lookup/users"` returns remaining limit for user lookup requests; `type = "followers/ids"` returns remaining limit for follower id requests; `type = "friends/ids"` returns remaining limit for friend id requests.
`parse`	Logical indicating whether to parse response object into a data frame.

Details

If multiple tokens are provided, this function will return the names of the associated [token] apps as new variable (column) or as a named element (if parse = FALSE).

Value

Tibble data frame with rate limit information pertaining to the limit (max allowed), remaining (specific to token), reset (mins until reset), and reset_at (time of rate limit reset). If query is specified, only relevant rows are returned.

See Also

https://developer.twitter.com/en/docs/developer-utilities/rate-limit-status/api-reference/get-application-rate_limit_status

Other tokens: `create_token`, `get_tokens`

save_as_csv *save_as_csv*

Description

Converts and saves data table generated from rtweet package as csv file(s).

Usage

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

Arguments

x	Data table to be saved (tweets or user object) generated via rtweet function like `search_tweets`. If x is a list object containing both tweets and users data (which is currently the output for many of the rtweet functions), then a CSV file is created and saved for each object using the file_name provided as a base–e.g, if x is a list object from search_tweets with `file_name = "election"`, this function will save both the tweets data ("election.tweets.csv") and the user data ("election.users.csv"). If not included in file_name, the ".csv" extension will be added when writing file to disk.
file_name	Path/file name where object(s) is to be saved. If object includes both tweets and users data then provided file_name will be used as base for the two saved files. For example, `file_name = "election"` would save files as "election.tweets.csv" and "election.users.csv".
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.

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

Description

Get tweets data on statuses identified via search query.

Usage

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

Arguments

q	Query tobe 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" anywhere 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\""`.
n	Integer, specifying the total number of desired tweets to return. Defaults to 100. Maximum number of tweets returned from a single token is 18,000. To return more than 18,000 tweets, users are encouraged to set `retryonratelimit` to TRUE. See details for more information.
type	Character string specifying which type of search results to return from Twitter's REST API. The current default is `type = "recent"`, other valid types include `type = "mixed"` and `type = "popular"`.
include_rts	Logical, indicating whether to include retweets in search results. Retweets are classified as any tweet generated by Twitter's built-in "retweet" (recycle arrows) function. These are distinct from quotes (retweets with additional text provided from sender) or manual retweets (old school method of manually entering "RT" into the text of one's tweets).
geocode	Geographical limiter of the template "latitude,longitude,radius" e.g., `geocode = "37.78,-122.40,1mi"`.
max_id	Character string specifying the [oldest] status id beyond which search results should resume returning. Especially useful large data returns that require multiple iterations interrupted by user time constraints. For searches exceeding 18,000 tweets, users are encouraged to take advantage of rtweet's internal automation procedures for waiting on rate limits by setting `retryonratelimit` argument to TRUE. It some cases, it is possible that due to processing time and rate limits, retreiving several million tweets can take several hours or even multiple days. In these cases, it would likely be useful to leverage `retryonratelimit` for sets of tweets and `max_id` to allow results to continue where previous efforts left off.
parse	Logical, indicating whether to return parsed data.frame, if true, or nested list (fromJSON), if false. By default, `parse = TRUE` saves users from the wreck of time and frustration associated with disentangling the nasty nested list returned from Twitter's API (for proof, check rtweet's Github commit history). As Twitter's APIs are subject to change, this argument would be especially useful when changes to Twitter's APIs affect performance of internal parsers. Setting `parse = FALSE` also ensures the maximum amount of possible information is returned. By default, the rtweet parse process returns nearly all bits of information returned from Twitter. However, users may occassionally encounter new or omitted variables. In these rare cases, the nested list object will be the only way to access these variables.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).

`retryonratelimit`
: Logical indicating whether to wait and retry when rate limited. This argument is only relevant if the desired return (n) exceeds the remaining limit of available requests (assuming no other searches have been conducted in the past 15 minutes, this limit is 18,000 tweets). Defaults to false. Set to TRUE to automate process of conducting big searches (i.e., n > 18000). For many search queries, esp. specific or specialized searches, there won't be more than 18,000 tweets to return. But for broad, generic, or popular topics, the total number of tweets within the REST window of time (7-10 days) can easily reach the millions.

`verbose`
: Logical, indicating whether or not to include output processing/retrieval messages. Defaults to TRUE. For larger searches, messages include rough estimates for time remaining between searches. It should be noted, however, that these time estimates only describe the amount of time between searches and not the total time remaining. For large searches conducted with `retryonratelimit` set to TRUE, the estimated retreival time can be estimated by dividing the number of requested tweets by 18,000 and then multiplying the quotient by 15 (token cooldown time, in minutes).

`...`
: Futher 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.

Number of tweets returned will often be less than what was specified by the user. This can happen because (a) the search query did not return many results (the search pool is already thinned out from the population of tweets to begin with), (b) because user hitting rate limit for a given token, or (c) of recent activity (either more tweets, which affect pagination in returned results or deletion of tweets). To return more than 18,000 tweets in a single call, users must set `retryonratelimit` argument to true. This method relies on updating the `max_id` parameter and waiting for token rate limits to refresh between searches. As a result, it is possible to search for 50,000, 100,000, or even 10,000,000 tweets, but these searches can take hours or even days. At these durations, it would not be uncommon for connections to timeout. Users are instead encouraged to breakup data retrieval into smaller chunks by leveraging `retryonratelimit` and then using the status_id of the oldest tweet as the `max_id` to resume searching where the previous efforts left off.

Value

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

See Also

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

Other tweets: `get_favorites`, `get_timeline`, `lists_statuses`, `lookup_statuses`, `tweets_data`, `tweets_with_users`

Examples

```
## Not run:
```

```
## search for 1000 tweets mentioning Hillary Clinton
hrc <- search_tweets(q = "hillaryclinton", n = 1000)

## data frame where each observation (row) is a different tweet
hrc

## users data also retrieved. can access it via users_data()
users_data(hrc)

## search for 1000 tweets in English
djt <- search_tweets(q = "realdonaldtrump", n = 1000, lang = "en")
djt
users_data(djt)

## exclude retweets
rt <- search_tweets("rstats", n = 500, include_rts = FALSE)

## perform search for lots of tweets
rt <- search_tweets("trump OR president OR potus", n = 100000,
                    retryonratelimit = TRUE)

## plot time series of tweets frequency
ts_plot(rt, by = "mins", theme = "spacegray",
        main = "Tweets about Trump")


## End(Not run)
```

`search_users` *Get users data on accounts identified via search query.*

Description

Get users data on accounts identified via search query.

Usage

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

Arguments

`q`	Query to be searched, used in filtering relevant tweets to return from Twitter's REST API. Should be a character string not to exceed 500 characters maximum. Spaces are assumed to function like boolean "AND" operators. To search for tweets including one of multiple possible terms, separate search terms with spaces and the word "OR". For example, the search `query = "data science"` searches for tweets using both "data" and "science" though the words can appear anywhere and in any order in the tweet. However, when OR is added between search terms, `query = "data OR science"`, Twitter's REST API should return any tweet that includes either "data" or "science" appearing in the tweets. At this time, Twitter's users/search API does not allow complex searches or queries targetting exact phrases as is allowed by `search_tweets`.

n	Numeric, specifying the total number of desired users to return. Defaults to 100. Maximum number of users returned from a single search is 1,000.
parse	Logical, indicating whether to return parsed (data.frames) or nested list (fromJSON) object. By default, `parse = TRUE` saves users from the time [and frustrations] associated with disentangling the Twitter API return objects.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
verbose	Logical, indicating whether or not to output processing/retrieval messages.

Value

Data frame of users returned by query.

See Also

https://dev.twitter.com/overview/documentation

Other users: as_screenname, lists_subscribers, lookup_users, tweets_with_users, users_data

Examples

```
## Not run:
# search for 1000 tweets mentioning Hillary Clinton
pc <- search_users(q = "political communication", n = 1000)

# data frame where each observation (row) is a different user
pc

# tweets data also retrieved. can access it via tweets_data()
users_data(hrc)

## End(Not run)
```

---

stopwordslangs *stopwordslangs data*

Description

This data comes form a group of Twitter searches conducted on 2017-09-27. The data are commonly observed words associated with 10 different languages, including "ar", "en", "es", "fr", "in", "ja", "pt", ru", "tr", and "und". Variables include word (potential stop words), lang (two or three word code), and p (p 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).

Usage

```
stopwordslangs
```

Format

A tibble with three variables and 24,000 observations

Examples

```
stopwordslangs
```

`stream_tweets` *Collect a live stream of Twitter data.*

Description

Returns public statuses 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)

Usage

```
stream_tweets(q = "", timeout = 30, parse = TRUE, token = NULL,
  file_name = NULL, gzip = FALSE, verbose = TRUE, fix.encoding = TRUE,
  ...)
```

Arguments

`q`	Query used to select and customize streaming collection method. There are four possible methods. (1) The default, `q = ""`, returns a small random sample of all publicly available Twitter statuses. (2) To filter by keyword, provide a comma separated character string with the desired phrase(s) and keyword(s). (3) Track users by providing a comma separated list of user IDs or screen names. (4) Use four latitude/longitude bounding box points to stream by geo location. This must be provided via a vector of length 4, e.g., c(-125, 26, -65, 49).
`timeout`	Numeric scalar specifying amount of time, in seconds, to leave connection open while streaming/capturing tweets. By default, this is set to 30 seconds. To stream indefinitely, use `timeout = FALSE` to ensure json file is not deleted upon completion or `timeout = Inf`.
`parse`	Logical, indicating whether to return parsed data. By default, `parse = TRUE`, this function does the parsing for you. However, for larger streams, or for automated scripts designed to continuously collect data, this should be set to false as the parsing process can eat up processing resources and time. For other uses, setting parse to TRUE saves you from having to sort and parse the messy list structure returned by Twitter. (Note: if you set parse to false, you can use the `parse_stream` function to parse the json file at a later point in time.)
`token`	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
`file_name`	Character with name of file. By default, a temporary file is created, tweets are parsed and returned to parent environment, and the temporary file is deleted.

gzip	Logical indicating whether to request gzip compressed stream data. By default this is set to FALSE. After performing some tests, it appears gzip requires less bandwidth, but also returns slightly fewer tweets. Use of gzip option should, in theory, make connection more reliable (by hogging less bandwidth, there's less of a chance Twitter cuts you off for getting behind).
verbose	Logical, indicating whether or not to include output processing/retrieval messages.
fix.encoding	Logical indicating whether to internally specify encoding to prevent possible errors caused by things such as non-ascii characters.
...	Insert magical paramaters, spell, or potion here. Or filter for tweets by language, e.g., `language = "en"`.

Value

Tweets data returned as data frame with users data as attribute.

See Also

https://stream.twitter.com/1.1/statuses/filter.json

Other stream tweets: parse_stream

Examples

```
## Not run:
## stream tweets mentioning "election" for 90 seconds
e <- stream_tweets("election", timeout = 90)

## data frame where each observation (row) is a different tweet
e

## users data also retrieved, access it via users_data()
users_data(e)

## plot tweet frequency
ts_plot(e, "secs")

## stream tweets mentioning Obama for 30 seconds
djt <- stream_tweets("realdonaldtrump", timeout = 30)

## preview tweets data
djt

## get user IDs of people who mentioned trump
usrs <- users_data(djt)

## lookup users data
usrdat <- lookup_users(unique(usrs$user_id))

## preview users data
usrdat

## store large amount of tweets in files using continuous streams
## by default, stream_tweets() returns a random sample of all tweets
## leave the query field blank for the random sample of all tweets.
stream_tweets(
```

```
  timeout = (60 * 10),
  parse = FALSE,
  file_name = "tweets1"
)
stream_tweets(
  timeout = (60 * 10),
  parse = FALSE,
  file_name = "tweets2"
)

## parse tweets at a later time using parse_stream function
tw1 <- parse_stream("tweets1.json")
tw1

tw2 <- parse_stream("tweets2.json")
tw2

## streaming tweets by specifying lat/long coordinates

## stream continental US tweets for 5 minutes
usa <- stream_tweets(
  c(-125, 26, -65, 49),
  timeout = 300
)

## use lookup_coords() for a shortcut verson of the above code
usa <- stream_tweets(
  lookup_coords("usa"),
  timeout = 300
)

## stream world tweets for 5 mins, save to json file
## shortcut coords note: lookup_coords("world")
world.old <- stream_tweets(
  c(-180, -90, 180, 90),
  timeout = (60 * 5),
  parse = FALSE,
  file_name = "world-tweets.json"
)

## read in json file
rtworld <- parse_stream("word-tweets.json")

## world data set with with lat lng coords variables
x <- lat_lng(rtworld)


## End(Not run)
```

---

suggested_slugs *Returns the list of suggested user categories.*

Description

Returns the list of suggested user categories.

Returns users in a given category of the Twitter suggested user list

Usage

```
suggested_slugs(lang = NULL, token = NULL)

suggested_users(slug, lang = NULL, token = NULL)
```

Arguments

lang	optional Restricts the suggested categories to the requested language. The language must be specified by the appropriate two letter ISO 639-1 representation.
token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
slug	required The short name of list or a category

Value

List of recommended categories which can be passed along as the "slug" parameter in `suggested_users`

Recommended users

Examples

```
## Not run:
## get slugs
slugs <- suggested_slugs()

## use slugs to get suggested users
suggested_users(slugs$slug[1])


## End(Not run)
```

`trends_available` *trends_available*

Description

Returns Twitter trends based on requested WOEID.

Usage

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

Arguments

token	OAuth token. By default `token = NULL` fetches a non-exhausted token from an environment variable. Find instructions on how to create tokens and setup an environment variable in the tokens vignette (in r, send `?tokens` to console).
parse	Logical, indicating whether to return parsed (data.frames) or nested list (fromJSON) object. By default, `parse = TRUE` saves users from the time [and frustrations] associated with disentangling the Twitter API return objects.

Value

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

See Also

Other trends: `get_trends`

Examples

```
## Not run:
## Retrieve available trends
trends <- trends_available()
trends


## End(Not run)
```

---

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

Description

Converts tweets data into time series-like data object.

Usage

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

Arguments

data	Data frame or grouped data frame.
by	Desired interval of time expressed as numeral plus secs, mins, hours, days, weeks, months, 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

Value

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

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

Description

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

Usage

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

Arguments

data	Data frame or grouped data frame.
by	Desired interval of time expressed as numeral plus secs, mins, hours, days, weeks, months, years. If a numeric is provided, the value is assumed to be in seconds.
trim	The number of observatons to drop off the beginning and end of the time series.
...	Other args passed to ggplot geom_line or, if ggplot2 is not installed, then to lines plotting function.

Value

If ggplot2 is installed then a ggplot2 plot

Examples

```
## Not run:
rt <- search_tweets("rstats", n = 10000)
ts_plot(rt, "10 mins")

rt %>%
  dplyr::group_by(is_retweet) %>%
  ts_plot("hours")

## compare account activity for some important national political figures
tmls <- get_timeline(c("SenSchumer", "SenGillibrand", "realDonaldTrump"), n = 3000)
## examine all twitter activity
ts_plot(tmls, "months")
## group by screen name
ts_plot(dplyr::group_by(tmls, screen_name), "months") + theme_mwk()
## group by screen name and is_retweet
ts_plot(dplyr::group_by(tmls, screen_name, is_retweet), "months") + theme_minimal()

## End(Not run)
```

tweets_data *Extracts tweets data from users data object.*

Description

Extracts tweets data from users data object.

Usage

```
tweets_data(users)
```

Arguments

users	Parsed data object of users data as returned via search_users, lookup_users, etc.

Value

Tweets data frame.

See Also

Other tweets: get_favorites, get_timeline, lists_statuses, lookup_statuses, search_tweets, tweets_with_users

Other extractors: max_id, users_data

Examples

```
## Not run:
## get twitter user data
jack <- lookup_users("jack")

## get data on most recent tweet from user(s)
tweets_data(jack)

## search for 100 tweets containing the letter r
r <- search_tweets("r")

## print tweets data (only first 10 rows are shown)
head(r, 10)

## preview users data
head(users_data(r))

## End(Not run)
```

tweets_with_users *parsing*

Description

Parsing data into tweets/users data tibbles

Usage

```
tweets_with_users(x)

users_with_tweets(x)
```

Arguments

x	Unparsed data returned by rtweet API request.

Value

A tweets/users tibble (data frame) with users/tweets tibble attribute.

See Also

Other tweets: get_favorites, get_timeline, lists_statuses, lookup_statuses, search_tweets, tweets_data

Other users: as_screenname, lists_subscribers, lookup_users, search_users, users_data

Examples

```
## Not run:
## search with parse = FALSE
rt <- search_tweets("rstats", n = 500, parse = FALSE)

## parse to tweets data tibble with users data attribute object
tweets_with_users(rt)

## search with parse = FALSE
usr <- search_users("rstats", n = 300, parse = FALSE)

## parse to users data tibble with users data attribute object
users_with_tweets(usr)


## End(Not run)
```

users_data *Extracts users data from tweets data object.*

Description

Extracts users data from tweets data object.

Usage

```
users_data(tweets)
```

Arguments

tweets	Parsed data object of tweets data as returned via get_timeline, search_tweets, stream_tweets, etc..

Value

Users data frame from tweets returned in a tweets data object.

See Also

Other users: as_screenname, lists_subscribers, lookup_users, search_users, tweets_with_users

Other extractors: max_id, tweets_data

Examples

```
## Not run:
## search for 100 tweets containing the letter r
r <- search_tweets("r")

## print tweets data (only first 10 rows are shown)
head(r, 10)

## extract users data
head(users_data(r))

## End(Not run)
```

write_as_csv *write_as_csv*

Description

Saves as flattened CSV file Twitter data.

Usage

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

Arguments

x	Data frame with tweets and users data.
file_name	Desired name(stem) to save files as (one save for tweets, one save for users).
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.

Index