Package 'rtweet'

January 18, 2017

Type Package

Title Collecting Twitter Data

Version 0.4.0

Date 2017-01-17

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 httr (>= 1.0.0), jsonlite, magrittr, openssl

License MIT + file LICENSE

LazyData TRUE

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

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

RoxygenNote 5.0.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Michael W. Kearney [aut, cre]

Maintainer Michael W. Kearney <mkearney@ku.edu>

Repository CRAN

Date/Publication 2017-01-18 23:54:58

R topics documented:

`create_token` *create_token*

Description

Sends request to generate oauth 1.0 tokens. Twitter also allows uers to create user-only (oauth 2.0) access token. Unlike the 1.0 tokens, oath 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 personal access token object

See Also

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

Other tokens: get_tokens

---

get_favorites *get_favorites*

Description

Returns the 20 most recent Tweets liked by the authenticating or specified user.

Usage

```
get_favorites(user, n = 3000, since_id = NULL, max_id = NULL,
  parse = TRUE, clean_tweets = FALSE, as_double = FALSE, usr = TRUE,
  token = NULL)
```

Arguments

user	Screen name or user id of target user.
n	Specifies the number of records to retrieve. Must be less than or equal to 200; defaults to 3000, which 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.
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.
clean_tweets	logical indicating whether to remove non-ASCII characters in text of tweets. defaults to FALSE.
as_double	logical indicating whether to handle ID variables as double (numeric) class. By default, this is set to FALSE, meaning ID variables are treated as character vectors. Setting this to TRUE can provide performance (speed and memory) boost but can also lead to issues when printing and saving, depending on the format.
usr	Logical indicating whether to return users 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

Tweets data frame.

See Also

Other tweets: get_timeline, lookup_statuses, search_tweets, stream_tweets, tweets_data

Examples

```
## Not run:
# get ids of users following the president of the US
pres <- get_followers(user = "potus")
pres

# get ids of users following the Environmental Protection Agency
epa <- get_followers(user = "epa")
epa

## End(Not run)
```

---

get_followers *get_followers*

Description

Returns max followers per token

Usage

```
get_followers(user, n = 75000, page = "-1", parse = TRUE,
  as_double = FALSE, token = NULL)
```

Arguments

user	Screen name or user id of target user.
n	Number of followers to return. For max return, enter `n = "all"` or `n = 75000` (max per token).
page	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.
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.
as_double	logical indicating whether to handle ID variables as double (numeric) class. By default, this is set to FALSE, meaning ID variables are treated as character vectors. Setting this to TRUE can provide performance (speed and memory) boost but can also lead to issues when printing and saving, depending on the format.
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

list of follower ids and next page value (presumably this would be used in loops extracting more than 75,000 followers using either multiple tokens or by waiting out rate limits)

See Also

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

Other ids: get_friends, next_cursor

Examples

```
## Not run:
# get ids of users following the president of the US
pres <- get_followers(user = "potus")
pres

# get ids of users following the Environmental Protection Agency
epa <- get_followers(user = "epa")
epa

## End(Not run)
```

get_friends *get_friends*

Description

Requests information from Twitter's REST API regarding a user's friend network (i.e., accounts followed by a user). To request information on followers of accounts

Usage

```
get_friends(user, page = "-1", parse = TRUE, as_double = FALSE,
  token = NULL)
```

Arguments

user	Screen name or user id of target user.
page	Default page = -1 specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.
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.
as_double	logical indicating whether to handle ID variables as double (numeric) class. By default, this is set to FALSE, meaning ID variables are treated as character vectors. Setting this to TRUE can provide performance (speed and memory) boost but can also lead to issues when printing and saving, depending on the format.
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

friends User ids for everyone a user follows.

See Also

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

Other ids: get_followers, next_cursor

Examples

```
## Not run:
# get ids of users followed by the president of the US
pres <- get_friends(user = "potus")
pres

# get ids of users followed by the Environmental Protection Agency
epa <- get_friends(user = "epa")
```

```
epa

## End(Not run)
```

`get_timeline` *get_timeline*

Description

Returns timeline of tweets from a specified Twitter user. By default, get_timeline returns tweets posted by a given user. To return a user's timeline feed, that is, tweets posted by accounts you follow, set the home argument to true.

Usage

```
get_timeline(user, n = 200, max_id = NULL, home = FALSE, parse = TRUE,
  check = TRUE, usr = TRUE, token = NULL, ...)
```

Arguments

`user`	Screen name or user id of target user.
`n`	Numeric, number of tweets to return.
`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.
`usr`	Logical indicating whether to return users 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).
`...`	Futher arguments passed on to `make_url`. All named arguments that do not match the above arguments (i.e., count, type, etc.) will be built into the request. To return only English language tweets, for example, use `lang = "en"`. Or, to exclude retweets, use `include_rts = FALSE`. For more options see Twitter's API documentation.

Value

List consisting of two data frames. One with the tweets data for a specified user and the second is a single row for the user provided.

See Also

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

Other tweets: get_favorites, lookup_statuses, search_tweets, stream_tweets, tweets_data

Examples

```
## Not run:
# get 2000 from Donald Trump's account
djt <- get_timeline("realDonaldTrump", n = 2000)

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

# users data for realDonaldTrump is also retrieved.
# access it via users_data() users_data(hrc)
users_data(djt)

## End(Not run)
```

`get_tokens` *get_tokens*

Description

Call function used to load Twitter oauth tokens. Since Twitter app key should be stored private, you are encouraged to create and save an R user profile declaring the path to your Twitter tokens. This allows Tokens to be instantly [re]loaded for future sessions. It also makes it easier to write teh card - allowing internals of the functions t call your tokens for you.

Usage

```
get_tokens()
```

Value

path

See Also

Other tokens: create_token

get_trends *get_trends*

Description

Returns Twitter trends

Usage

```
get_trends(woeid = 1, exclude = FALSE, token = NULL, parse = TRUE)
```

Arguments

woeid	Numeric, WOEID (Yahoo! Where On Earth ID) or character string of desired town or country. To browse all available trend places, see trends_available
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.

Value

Trend data for a given location.

See Also

Other trends: find_woeid, trends_available

Examples

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

# Store WOEID for Worldwide trends
worldwide <- subset(trends, name == "Worldwide")[["woeid"]]

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

# Preview trends data
ww_trends

## End(Not run)
```

lookup_coords *lookup_coords*

Description

Returns lat long coordinates using google geocode api

Usage

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

Arguments

address	Desired location, e.g., "lawrence, KS"
components	Unit of analysis for address e.g., "country:US". Potential components include postal_code, country, administrative_area, locality, route.
...	Additional args passed along to params portion of http request

Value

Numeric vector with lat and long coordinates

Examples

```
## Not run:
lookupcoords("san francisco, CA", "country:US")

## 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).

lookup_statuses *lookup_tweets*

Description

Returns Twitter user data_frame object for specified user_ids or screen_names.

Usage

```
lookup_statuses(statuses, token = NULL, parse = TRUE, usr = TRUE,
  clean_tweets = FALSE, as_double = FALSE)
```

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).
usr	Logical indicating whether to return users data frame. Defaults to true.
clean_tweets	logical indicating whether to remove non-ASCII characters in text of tweets. defaults to FALSE.
as_double	logical indicating whether to handle ID variables as double (numeric) class. By default, this is set to FALSE, meaning ID variables are treated as character vectors. Setting this to TRUE can provide performance (speed and memory) boost but can also lead to issues when printing and saving, depending on the format.

Value

json response object (max is 18000 per token)

See Also

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

Other tweets: get_favorites, get_timeline, search_tweets, stream_tweets, tweets_data

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 *lookup_users*

Description

Returns Twitter user data_frame object for specified user_ids or screen_names.

Usage

```
lookup_users(users, token = NULL, parse = TRUE, tw = TRUE,
  clean_tweets = TRUE, as_double = FALSE)
```

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).
tw	Logical indicating whether to return tweets data frame. Defaults to true.
clean_tweets	logical indicating whether to remove non-ASCII characters in text of tweets. defaults to TRUE.
as_double	logical indicating whether to handle ID variables as double (numeric) class. By default, this is set to FALSE, meaning ID variables are treated as character vectors. Setting this to TRUE can provide performance (speed and memory) boost but can also lead to issues when printing and saving, depending on the format.

Value

json response object (max is 18000 per token)

See Also

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

Other users: search_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")

usr_df <- lookup_users(users)
usr_df

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

## End(Not run)
```

`mutate_coords` *mutate_coords*

Description

Initializes rt plotting sequence

Usage

```
mutate_coords(data, ...)
```

Arguments

`data`	Data frame generated via rtweet function.
`...`	Args passed to points.

`next_cursor` *next_cursor*

Description

Returns next cursor value from ids object. Return object used to retrieve next page of results from API request.

Usage

```
next_cursor(ids)
```

Arguments

`ids`	Data frame of Twitter IDs generated via `get_followers` or `get_friends`.

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`

Examples

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

## End(Not run)
```

`parser` *parser*

Description

Returns Parses tweets and users data

Usage

```
parser(rt, att = TRUE)
```

Arguments

`rt`	Nested list converted from json structure
`att`	Logical indicating whether to include user obj (users data) as attribute if tweets data provided to rt argument or tweets obj (tweets data) as attribute if users data provided to rt. Defaults to true.

parse_data *parse_data*

Description

Returns Parses tweets and users data

Usage

```
parse_data(rt, tw = TRUE)
```

Arguments

rt	Nested list converted from json structure
tw	Logical indicating whether to include user obj (users data) as attribute. Defaults to true. If was generated from users-oriented function, e.g., lookup_users(), set to false.

parse_stream *parse_stream*

Description

parse_stream

Usage

```
parse_stream(file_name, ...)
```

Arguments

file_name	name of file to be parsed. NOTE: if file was created via stream_tweets, then it will end in ".json" (see example below)
...	For developmental purposes.

Details

Reading and simplifying json files can be very slow. To make things more managable, parse_stream_xl does one chunk of Tweets at a time and then compiles the data into a data frame.

Value

Data frame of tweets data with attributes users data

Examples

```
## Not run:
## file extension automatically converted to .json whether or
## not file_name already includes .json
stream_tweets(q = "", timeout = 30,
              file_name = "rtweet-stream", parse = FALSE)
rt <- parse_stream("rtweet-stream.json")
## preview tweets data
head(rt)
## preview users data
head(users_data(rt))
## plot time series
ts_plot(rt, "secs")

## End(Not run)
```

`post_favorite` *post_favorite*

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_mute`, `post_tweet`, `post_unfollow_user`

Examples

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

## End(Not run)
```

post_follow *post_follow*

Description

Follows target twitter user.

Usage

```
post_follow(user, destroy = FALSE, mute = FALSE, notify = FALSE,
  retweets = TRUE, 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_mute, post_tweet, post_unfollow_user

Examples

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

## End(Not run)
```

post_friendship *post_friendship*

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_mute, post_tweet, post_unfollow_user

post_mute *post_mute*

Description

Mute, or hide all content coming from, current twitter friend. Wrapper function for mute version of follow_user.

Usage

```
post_mute(user, token = NULL)
```

Arguments

user	Screen name or user id of target user.
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_friendship, post_tweet, post_unfollow_user

post_tweet *post_tweet*

Description

Posts status update to user's Twitter account

Usage

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

Arguments

status	Character, tweet status. Must be 140 characters or less.
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_friendship, post_mute, post_unfollow_user

Examples

```
## Not run:
post_tweet("my first rtweet #rstats")

## End(Not run)
```

post_unfollow_user *post_unfollow*

Description

Remove, or unfollow, current twitter friend. Wrapper function for destroy version of follow_user.

Usage

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

Arguments

user	Screen name or user id of target user.
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_friendship, post_mute, post_tweet

rate_limit *rate_limit*

Description

Returns rate limit information for Twitter access tokens.

Usage

```
rate_limit(token, query = NULL, rest = TRUE, 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).
query	If null, returns entire rate limit request object as 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.
rest	Logical indicating whether to send request to REST API. At this time, this should always be TRUE.
parse	Logical indicating whether to parse response object into tidy data frame.

Value

Data frame with rate limit respones details. If query is specified, only relevant rows are returned.

See Also

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

rtweet *rtweet*

Description

rtweet provides users a range of functions designed to extract data from Twitter's REST and streaming APIs.

It has three main goals:

- Formulate and send requests to Twitter's REST and stream APIs.
- Retrieve and iterate over returned data.
- Wrangling data into tidy structures.

Examples

```
## Not run:
## for instructions on access tokens, see the tokens vignette
vignette("tokens")

## for a quick demo check the rtweet vignette
vignette("rtweet")

## End(Not run)
```

`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)
```

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".

`search_tweets` *search_tweets*

Description

Returns two data frames (tweets data and users data) using a provided search query.

Usage

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

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" 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"`.
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.
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).
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.

usr
: Logical indicating whether to return a data frame of users data. Users data is stored as an attribute. To access this data, see users_data. Useful for marginal returns in memory demand. However, any gains are likely to be negligible as Twitter's API invariably returns this data anyway. As such, this defaults to true, see users_data.

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 on to make_url. All named arguments that do not match the above arguments (i.e., count, type, etc.) will be built into the request. 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://dev.twitter.com/overview/documentation

Other tweets: `get_favorites`, `get_timeline`, `lookup_statuses`, `stream_tweets`, `tweets_data`

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 *search_users*

Description

Returns data frame of users data using a provided search query.

Usage

```
search_users(q, n = 20, parse = TRUE, tw = 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.
tw	Logical indicating whether to return tweets 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).
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: lookup_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)
```

`stream_tweets` *stream_tweets*

Description

Returns public statuses via one of three methods described below. By design, this function deciphers which method to use when processing the `stream` argument.

- 1. Filtering via a search-like query (up to 400 keywords)
- 2. Tracking via vector of user ids (up to 5000 user_ids)
- 3. 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`	Character vector with desired phrases and keywords used to filter tweets, a comma separated list of desired user IDs to track, or a set of bounding boxes to track. If left empty, the default `q = ""`, stream function will return sample of all tweets.
`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 tweets: get_favorites, get_timeline, lookup_statuses, search_tweets, tweets_data

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. can access it via users_data()
users_data(e)

# stream tweets mentioning Obama for 30 seconds
djt <- stream_tweets("realdonaldtrump", timeout = 30)
djt # prints tweets data preview
users_data(djt) # prints users data preview

# 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

## 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: `find_woeid`, `get_trends`

Examples

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

# Store WOEID for Worldwide trends
worldwide <- subset(trends, name == "Worldwide")[["woeid"]]

# Retrieve worldwide trends datadata
```

```
ww_trends <- get_trends(woeid = Worldwide)

# Preview Worldwide trends data
ww_trends

## End(Not run)
```

---

ts_filter *ts_filter*

Description

Converts text-level observations to time aggregated frequency data frame with [optional] filtered dummy variable(s).

Usage

```
ts_filter(rt, by = "days", dtname = "created_at", txt = "text",
  filter = NULL, key = NULL, na.omit = TRUE, trim = FALSE)
```

Arguments

rt	Tweets or users data frame. Technically, this argument will accept any recursive object (i.e., list or data frame) containing a named date-time (POSIXt) element or column. By default, ts_plot assumes the date-time variable is labeled "created_at", which is the default date-time label used in tweets data. However, this function should work with any data source, assuming it meets the (a) POSIXt class requirement and (b) the date-time variable is given the appropriate name (if not "created_at" then a label specified with the dtname argument).
by	Unit of time, e.g., secs, days, weeks,months, years by which to aggregate observations. By default, ts_plot tries to aggregate time by "days", but for some high-frequency data sets that only span a matter of minutes or hours, this is likely to either produce an error or a truly disappointing plot. In these cases, users are encouraged to explore smaller units of time. Conversely, high-frequency and long [in duration] data sets may be difficult to read given the default unit of time. In these cases, users should try larger units of time, e.g., "weeks" or "months". This parameter will also accept numeric quantifiers in addition to units of time. By default, for example, the provided unit of time is assumed to specify whole (1) units of time. It is posible to tweak this unit by specifying the number (or fraction) of time units, e.g., by = "2 weeks", by = "30 secs", by = ".333 days".
dtname	Name of date-time (POSIXt) column (if data frame) or element (if list). Defaults to "created_at", the default label supplied as a timestamp variable for tweets data. This function is exportable to non-Twitter data, assuming the intended data object includes a date-time variable with the same label that's supplied to the dtname parameter.

txt	Name of distinguishing variable in data frame or list to which filter is applied. Defaults to text.
filter	Vector of regular expressions with which to filter data (creating multiple time series).
key	Optional provide pretty labels for filters. Defaults to actual filters.
na.omit	Logical indicating whether to omit rows with missing (NA) values for the dtname variable. Defaults to TRUE. If FALSE and data contains missing values for the date-time variable, an error will be returned to the user.
trim	Logical indicating whether to trim extreme intervals, which often capture artificially lower frequencies. Defaults to FALSE.

---

ts_plot *ts_plot*

Description

Plots frequency of tweets as time series or, if multiple filters (text-based criteria used to subset data) are specified, multiple time series.

Usage

```
ts_plot(rt, by = "days", dtname = "created_at", txt = "text",
  na.omit = TRUE, filter = NULL, key = NULL, trim = FALSE, lwd = 1.5,
  linetype = FALSE, cols = NULL, theme = "light", main = NULL,
  subtitle = NULL, adj = TRUE, xlab = "Time", ylab = "Freq",
  box = FALSE, axes = TRUE, legend.title = NULL, ticks = 0, cex = 1,
  cex.main, cex.sub, cex.lab, cex.axis, cex.legend, mar, font.main = 1,
  xtime = NULL, plot = TRUE, ...)
```

Arguments

rt	Tweets or users data frame. Technically, this argument will accept any recursive object (i.e., list or data frame) containing a named date-time (POSIXt) element or column. By default, ts_plot assumes the date-time variable is labeled "created_at", which is the default date-time label used in tweets data. However, this function should work with any data source, assuming it meets the (a) POSIXt class requirement and (b) the date-time variable is given the appropriate name (if not "created_at" then a label specified with the dtname argument).
by	Unit of time, e.g., secs, days, weeks,months, years by which to aggregate observations. By default, ts_plot tries to aggregate time by "days", but for some high-frequency data sets that only span a matter of minutes or hours, this is likely to either produce an error or a truly disappointing plot. In these cases, users are encouraged to explore smaller units of time. Conversely, high-frequency and long [in duration] data sets may be difficult to read given the default unit of time. In these cases, users should try larger units of time, e.g., "weeks" or "months". This parameter will also accept numeric quantifiers in

	addition to units of time. By default, for example, the provided unit of time is assumed to specify whole (1) units of time. It is posible to tweak this unit by specifying the number (or fraction) of time units, e.g., `by = "2 weeks"`, `by = "30 secs"`, `by = ".333 days"`.
`dtname`	Name of date-time (POSIXt) column (if data frame) or element (if list). Defaults to "created_at", the default label supplied as a timestamp variable for tweets data. This function is exportable to non-Twitter data, assuming the intended data object includes a date-time variable with the same label that's supplied to the `dtname` parameter.
`txt`	Name of distinguishing variable in data frame or list to which filter is applied. Defaults to text.
`na.omit`	Logical indicating whether to omit rows with missing (NA) values for the dtname variable. Defaults to TRUE. If FALSE and data contains missing values for the date-time variable, an error will be returned to the user.
`filter`	Vector of regular expressions with which to filter data (creating multiple time series).
`key`	Optional provide pretty labels for filters. Defaults to actual filters.
`trim`	Logical indicating whether to trim extreme intervals, which often capture artificially lower frequencies. Defaults to FALSE.
`lwd`	Width of time series line(s). Defaults to 1.5
`linetype`	Logical indicating whether lines should be distinguished by line type.
`cols`	Colors for filters. Leave NULl for default color scheme.
`theme`	Either integer (0-8) or character string specifyng the plot theme. Options include "light", "inverse", "dark", "nerd", "gray", "spacegray", "minimal", and "apa" (my attempt at making an APA-consistent graphic).
`main`	Optional, title of the plot. By default, the title is printed on top of the plot and it is left-justified (ggplot2 style). To alter justification, see `adj`.
`subtitle`	Optional, text for plot subtitle. Inherits justification method from `main`.
`adj`	Logical indicating whether to left justify main plot title. Defaults to TRUE. To more exactly specify hornizontal location of the title, provide a numeric value between 0 (left) and 1 (right).
`xlab`	Optional, text for x-axis title, defaults to "Time".
`ylab`	Optional, text for y-axis title, defaults to "Freq"
`box`	Logical indicating whether to draw box around plot area. Defaults to false.
`axes`	Logical indicating whether to draw axes. Defaults to true. Users may set this to FALSE and supply their own axes using the base graphics axis function.
`legend.title`	Provide title for legend or ignore to leave blank (default).
`ticks`	Numeric specifying width of tick marks. Defaults to zero. If you'd like tick marks, try setting this value to 1.25.
`cex`	Global cex setting defaults to 1.0.
`cex.main`	Size of plot title (if plot title provided via `main = "title"` argument).
`cex.sub`	Size of subtitles

cex.lab	Size of axis labels
cex.axis	Size of axis text
cex.legend	Size of legend text
mar	Margins in number of lines.
font.main	Font style of main title if provided. Default is to 1, which means (non-bold) normal font, overriding R's bold default, which I think is a little to aggressive. If you disagree with me, you can make the title bold by setting this value to 2.
xtime	Format date-time labels in x-axis. Accepts any format string via strptime, e.g., xtime = "%F %H:%S".
plot	Deprecated. Use ts_filter to create time series-like data frame.
...	Arguments passed to base graphics plot function.

Examples

```
## Not run:
## stream tweets mentioning trump for 30 mins
rt <- stream_tweets(
    q = "realdonaldtrump",
    timeout = (60 * 60 * 30))

## plot tweet data aggregated by minute (default)
ts_plot(rt, by = "mins")

## use a different time increment, line width, and theme
ts_plot(rt, by = "30 secs", lwd = .75, theme = "inverse")

## filter data using regular expressions and
## plot each corresponding time series
ts_plot(rt, by = "mins",
        theme = "gray",
        main = "Partisanship in tweets about Trump",
        filter = c("democrat|liberal|libs",
                   "republican|conservativ|gop"),
        key = c("Democrats", "Republicans"))

## ts_plot also accepts data frames created via ts_filter
rt.ts <- ts_filter(
    rt, "mins",
    filter = c("democrat|liberal|libs",
               "republican|conservativ|gop"),
    key = c("Democrats", "Republicans"))
## printing should yield around 30 rows (give or take)
## since stream was 30 mins and aggregated by minute
rt.ts

## Pass data frame created by ts_filter to ts_plot
ts_plot(rt.ts, theme = "spacegray")

## the returned data frame from ts_filter also fits the
## tidyverse and includes three columns
```

```
## Column 1 - time Date-time obj of [median] time intervals
## Column 2 - freq Integer (class double) frequency counts
## Column 3 - filter Keys of different time series filters

## This makes it easy to pass the data along to ggplot
## but my themes are cooler anyway so why bother?
## library(ggplot2)
## rt.ts %>%
##     ggplot(aes(x = time, y = freq, color = filter)) +
##     geom_line()

## End(Not run)
```

`tweets_data` *tweets_data*

Description

Tweets data frame from users returned in a users data object. Typically, this involves the most recent tweet of each user, though in some cases the most recent tweet may not be available.

Usage

```
tweets_data(users)
```

Arguments

`users`	Data frame of Twitter users generated via `lookup_users` or `search_users`.

Value

Tweets data frame.

See Also

Other tweets: `get_favorites`, `get_timeline`, `lookup_statuses`, `search_tweets`, `stream_tweets`

Examples

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

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

# extract users data
users_data(r)

## End(Not run)
```

users_data *users_data*

Description

Returns users data frame from returned tweets data object.

Usage

```
users_data(tweets)
```

Arguments

tweets	Data frame of Twitter statuses (tweets) generated via get_timeline, search_tweets, or stream_tweets.

Value

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

See Also

Other users: lookup_users, search_users

Examples

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

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

# extract users data
users_data(r)

## End(Not run)
```

Index