Package ‘tfse’

July 4, 2016

Version 1.1.0

Title Twitter Follows and Selective Exposure

Description An implementation of http calls and data wrangling functions designed to interact with Twitter API's. GET and POST requests are designed to return data from Twitter's search and streaming API's. Returned [json] objects are automatically parsed into data_frames or lists. Specific consideration is given within the functions to use of more than one OAuth personal access token.

Depends R (>= 3.3.0)

Imports dplyr (>= 0.5.0), httr (>= 1.2.1), jsonlite (>= 1.0)

Suggests rvest (>= 3.2), knitr, rmarkdown

VignetteBuilder knitr

License MIT + file LICENSE

LazyData TRUE

URL https://github.com/mkearney/tfse/,
https://dev.twitter.com/overview/documentation

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

RoxygenNote 5.0.1

Author Mike Kearney [aut, cre]

Maintainer Mike Kearney <mwk@ku.edu>

R topics documented:

`check_rate_limit` *check_rate_limit*

Description

check_rate_limit

Usage

```
check_rate_limit(type, token, seconds = FALSE)
```

Arguments

`type`	If not missing, which returns entire rate limit request object, type returns specific values; e.g., `type = "lookup"` returns remaining limit for user lookup requests; `type = "followers"` returns remaining limit for follower id requests; `type = "friends"` returns remaining limit for friend id requests.
`token`	An OAuth token (1.0 or 2.0)
`seconds`	logical, indicating whether to return unix seconds

Value

response Rate limit response object or specific value of remaining requests

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

`enc_track_query` *enc_track_query*

Description

enc_track_query

Usage

```
enc_track_query(.track)
```

Arguments

`.track`	stream search string

`extend_label_df` *extend_label_df*

Description

extend_label_df

Usage

```
extend_label_df(dff, label = "other")
```

Arguments

`dff`	other, extended data.frame within larger json response object
`new`	label to represent the other data.frame

Value

data_frame

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

`filter_stream`	*filter_stream Returns public statuses that match one or more filter predicates. Multiple parameters may be specified which allows most clients to use a single connection to the Streaming API. Both GET and POST requests are supported, but GET requests with too many parameters may cause the request to be rejected for excessive URL length. Use a POST request to avoid long URLs.*

Description

The track, follow, and locations fields should be considered to be combined with an OR operator. track=foo&follow=1234 returns Tweets matching “foo” OR created by user 1234.

Usage

```
filter_stream(stream, delimited = FALSE, stall_warnings = FALSE, token,
  timeout = 120, file_name = NULL)
```

Arguments

`stream`	either follower A comma separated list of user IDs, indicating the users to return statuses for in the stream. See follow for more information. track Keywords to track. Phrases of keywords are specified by a comma-separated list. See track for more information. Or locations Specifies a set of bounding boxes to track. See locations for more information.
`delimited`	optional Specifies whether messages should be length-delimited. See delimited for more information.
`stall_warnings`	optional Specifies whether stall warnings should be delivered. See stall_warnings for more information.
`token`	OAuth token (1.0 or 2.0)

Details

The default access level allows up to 400 track keywords, 5,000 follow userids and 25 0.1-360 degree location boxes. If you need elevated access to the Streaming API, you can contact Gnip.

Value

json object

See Also

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

`from_js` *from_js*

Description

from_js

Usage

```
from_js(x)
```

Arguments

`x`	json object

`get_api` *get_api*

Description

get_api

Usage

```
get_api(url, token = NULL)
```

Arguments

`url`	API url address.

Value

Response formatted as nested list. Assumes response object is json object.

`get_followers` *get_followers*

Description

get_followers

Usage

```
get_followers(user, token, page = "-1", stringify = TRUE)
```

Arguments

`user`	Screen name or user id of target user.
`token`	OAuth token (1.0 or 2.0)
`page`	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.
`stringify`	logical, indicating whether to return user ids as strings (some ids are too long to be read as numeric). Defaults to `TRUE`

Value

user ids

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

`get_followerslist` *get_followerslist*

Description

Returns a cursored collection of user objects for users following the specified user.

Usage

```
get_followerslist(user, token, page = "-1")
```

Arguments

`user`	Screen name or user id of target user.
`token`	OAuth token (1.0 or 2.0)
`page`	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.

Value

json user object (nested list)

See Also

See `https://api.twitter.com/1.1/followers/list.json`.

`get_followers_max` *get_followers_max*

Description

get_followers_max

Usage

```
get_followers_max(user, tokens)
```

Arguments

`user`	Screen name or user id of target user
`tokens`	OAuth tokens (1.0 or 2.0)

Value

user ids

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

`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 @seealso `get_followers`.

Usage

```
get_friends(user, token, page = "-1", stringify = TRUE)
```

Arguments

`user`	Screen name or user id of target user.
`token`	OAuth token (1.0 or 2.0)
`page`	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.
`stringify`	logical, indicating whether to return user ids as strings (some ids are too long to be read as numeric). Defaults to `TRUE`

Value

friends User ids for everyone a user follows.

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

`get_friendships` *get_friendships*

Description

Returns detailed information about the relationship between two arbitrary users.

Usage

```
get_friendships(source, target, token)
```

Arguments

`source`	Screen name or user id of source user.
`target`	Screen name or user id of target user.
`token`	OAuth token (1.0 or 2.0)

Value

json object (nested list)

See Also

See https://api.twitter.com/1.1/friendships/show.json.

`get_friendslist` *get_friendslist*

Description

Returns a cursored collection of user objects for every user the specified user is following (otherwise known as their "friends").

Usage

```
get_friendslist(user, token, page = "-1")
```

Arguments

`user`	Screen name or user id of target user.
`token`	OAuth token (1.0 or 2.0)
`page`	Default `page = -1` specifies first page of json results. Other pages specified via cursor values supplied by Twitter API response object.

Value

json user object (nested list)

See Also

See https://api.twitter.com/1.1/friends/list.json.

get_friends_max *get_friends_max*

Description

get_friends_max

Usage

```
get_friends_max(user_ids, tokens, start = 1, stringify = TRUE,
  verbose = TRUE)
```

Arguments

user_ids	Data frame with column name "screen_name"
tokens	OAuth tokens (1.0 or 2.0)
start	Starting value (nth user id)
stringify	logical, indicating whether to return user ids as strings
verbose	default behavior verbose = TRUE prints asterisk for every 15 (or max of one token) user networks collected. Set verbose = FALSE to run function silently. (some ids are too long to be read as numeric). Defaults to TRUE.

Value

friends List of user ids each user follows.

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

get_friends_ply *get_friends_ply*

Description

lapply() version of get_friends_max()

Usage

```
get_friends_ply(user_ids, tokens, start = 1)
```

Arguments

`user_ids`	Data frame with column name "screen_name"
`tokens`	OAuth tokens (1.0 or 2.0)
`start`	Starting value (nth user id)

Value

friends List of user ids each user follows.

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

get_list *get_list*

Description

get_list

Usage

```
get_list(slug, owner, token, count = 5000, cursor = "-1")
```

Arguments

`slug`	Name of Twitter list
`owner`	Screen name of list owner
`token`	An OAuth token (1.0 or 2.0)
`count`	Maximum number of users to return (cannot be higher than 5000).
`cursor`	Select next or previous page or results using cursor value from returned json object

Value

json response object as nested list

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

get_lookup *get_lookup*

Description

get_lookup

Usage

```
get_lookup(users, token, df = TRUE, skip = TRUE, entities = FALSE)
```

Arguments

users	Screen name or user id of target users.
token	OAuth tokens (1.0 or 2.0)
df	logical, indicating whether to format response as data frame

Value

response object

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

get_lookup_max *get_lookup_max*

Description

get_lookup_max

Usage

```
get_lookup_max(ids, tokens, start = 1)
```

Arguments

ids	User id or screen name of target user.
tokens	OAuth tokens (1.0 or 2.0)
start	First (nth) id

Value

response object

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

---

get_package_pdf *get_package_pdf*

Description

get_package_pdf

Usage

```
get_package_pdf(package, update = FALSE, open = TRUE)
```

Arguments

`package`	Name of target package.
`update`	logical, indicating whether to replace old manual
`open`	logical, indicating whether to automatically view preview of pdf

---

get_retweets *get_retweets*

Description

Returns a collection of the 100 most recent retweets of the tweet specified by the id parameter.

Usage

```
get_retweets(tweet_id, count = 100, trim_user = TRUE, token)
```

Arguments

`tweet_id`	required, The numerical ID of the desired status
`count`	optional, Specifies the number of records to retrieve. Must be less than or equal to 100.
`trim_user`	optional, When set to TRUE each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object.
`token`	OAuth token (1.0 or 2.0)

Value

json object

See Also

https://api.twitter.com/1.1/statuses/retweets/:id.json

`get_timeline` *get_timeline*

Description

Returns a collection of the most recent Tweets posted by the user indicated by the screen_name or user_id parameters. User timelines belonging to protected users may only be requested when the authenticated user either "owns" the timeline or is an approved follower of the owner. The timeline returned is the equivalent of the one seen when you view a user's profile on twitter.com. This method can only return up to 3,200 of a user's most recent Tweets. Native retweets of other statuses by the user is included in this total, regardless of whether include_rts is set to false when requesting this resource.

Usage

```
get_timeline(user, since_id = NULL, count = NULL, max_id = NULL,
  trim_user = NULL, exclude_replies = FALSE, contributer_details = TRUE,
  include_rts = TRUE, token)
```

Arguments

`user` The screen_name or ID of the user for whom to return results for.

`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.

`count` Specifies the number of tweets to try and retrieve, up to a maximum of 200 per distinct request. 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.

`max_id` Returns results with an ID less than (that is, older than) or equal to the specified ID.

`trim_user` When set to either true, t or 1, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object.

`exclude_replies`
This parameter will prevent replies from appearing in the returned timeline. Using exclude_replies with the count parameter will mean you will receive up-to count tweets — this is because the count parameter retrieves that many tweets before filtering out retweets and replies. This parameter is only supported for JSON and XML responses.

`include_rts` When set to false, the timeline will strip any native retweets (though they will still count toward both the maximal length of the timeline and the slice selected by the count parameter). Note: If you're using the trim_user parameter in conjunction with include_rts, the retweets will still contain a full user object.

`token`	OAuth token (1.0 or 2.0)
`contributor_details`	This parameter enhances the contributors element of the status response to include the screen_name of the contributor. By default only the user_id of the contributor is included.

Value

json object (nested list)

See Also

See https://api.twitter.com/1.1/statuses/user_timeline.json.

`get_token` *get_token*

Description

get_token

Usage

```
get_token(app, consumer_key, consumer_secret)
```

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`.

Value

twitter oauth 1.0 token

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

get_wave *get_wave*

Description

get_wave

Usage

```
get_wave(ids, tokens)
```

Arguments

ids	Vector of user ids

Value

friends List of user ids each user follows.

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

get_wave_data *get_wave_data*

Description

get_wave_data

Usage

```
get_wave_data(wave, ids, tokens)
```

Arguments

wave	Wave number

Value

data Saved wave data and updated running data set

`is_screen_name` *is_screen_name*

Description

is_screen_name

Usage

```
is_screen_name(x)
```

Arguments

`x`	Twitter user id or screen name

Value

logical value indicating whether object is screen name [or user ID]

`parse_all_tweets` *parse_all_tweets*

Description

parse_all_tweets

Usage

```
parse_all_tweets(x)
```

Arguments

`x`	json response object from tweet/status Twitter API request.

Value

data_frame

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

`parse_entities` *parse_entities*

Description

parse_entities

Usage

```
parse_entities(x)
```

Arguments

`x`	statuses entities object

`parse_place` *parse_place*

Description

parse_place

Usage

```
parse_place(x)
```

Arguments

`x`	json resposne object from user lookup Twitter API call.

Value

data frame

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

parse_retweet *parse_retweet*

Description

parse_retweet

Usage

```
parse_retweet(x)
```

Arguments

x	json response object as in json_object$retweet_status

Value

data_frame

See Also

See https://dev.twitter.com/overview/documentation for more information on using Twitter's API.

parse_status *parse_status*

Description

parse_status

Usage

```
parse_status(x)
```

Arguments

x	json object from search tweets

`parse_user` *parse_user*

Description

parse_user

Usage

```
parse_user(x)
```

Arguments

`x`	json resposne object from user lookup Twitter API call.

Value

data frame

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

`prep_list` *prep_list*

Description

prep_list

Usage

```
prep_list(x, colname)
```

Arguments

`x`	data to be vectorized

prep_vector *prep_vector*

Description

prep_vector

Usage

```
prep_vector(x)
```

Arguments

x	data to be vectorized

search_tweets *search_tweets*

Description

Returns a collection of relevant Tweets matching a specified query. Please note that Twitter's search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets. Not all Tweets will be indexed or made available via the search interface. In API v1.1, the response format of the Search API has been improved to return Tweet objects more similar to the objects you'll find across the REST API and platform. However, perspectival attributes (fields that pertain to the perspective of the authenticating user) are not currently supported on this endpoint. To learn how to use Twitter Search effectively, consult our guide to Using the Twitter Search API. See Working with Timelines to learn best practices for navigating results by since_id and max_id.

Usage

```
search_tweets(q, token, geocode = NULL, lang = NULL, locale = NULL,
  result_type = "recent", count = 100, until = NULL, since_id = NULL,
  max_id = NULL, include_entities = TRUE)
```

Arguments

q	required, A UTF-8, URL-encoded search query of 500 characters maximum, including operators. Queries may additionally be limited by complexity.
token	OAuth token (1.0 or 2.0)
geocode	optional, Returns tweets by users located within a given radius of the given latitude/longitude. The location is preferentially taking from the Geotagging API, but will fall back to their Twitter profile. The parameter value is specified by "latitude,longitude,radius", where radius units must be specified as either "mi" (miles) or "km" (kilometers). Note that you cannot use the near operator via the API to geocode arbitrary locations; however you can use this geocode parameter to search near geocodes directly. A maximum of 1,000 distinct "subregions" will be considered when using the radius modifier.

`lang` optional, Restricts tweets to the given language, given by an ISO 639-1 code. Language detection is best-effort.

`locale` optional, Specify the language of the query you are sending (only ja is currently effective). This is intended for language-specific consumers and the default should work in the majority of cases.

`result_type` optional, Specifies what type of search results you would prefer to receive. The current default is “mixed.” Valid values include `'mixed'` to include both popular and real time results in the response, `'recent'` to return only the most recent results in the response, and`'popular'` to return only the most popular results in the response.

`count` optional, The number of tweets to return per page, up to a maximum of 100. Defaults to 15. This was formerly the “rpp” parameter in the old Search API.

`until` optional, Returns tweets created before the given date. Date should be formatted as YYYY-MM-DD. Keep in mind that the search index has a 7-day limit. In other words, no tweets will be found for a date older than one week. Example Values: `'2015-07-19'`.

`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.

`include_entities` optional, The entities node will be disincluded when set to false.

`callback` optional, If supplied, the response will use the JSONP format with a callback of the given name. The usefulness of this parameter is somewhat diminished by the requirement of authentication for requests to this endpoint.

Details

The GET search/tweets is part of the Twitter REST API 1.1 and is rate limited similarly to other v1.1 methods. See REST API Rate Limiting in v1.1 for information on that model. At this time, users represented by access tokens can make 180 requests/queries per 15 minutes. Using application-only auth, an application can make 450 queries/requests per 15 minutes on its own behalf without a user context.

Value

json object

See Also

https://api.twitter.com/1.1/search/tweets.json

`should_be_post` *should_be_post*

Description

should_be_post

Usage

```
should_be_post(.query, .nchar = 20)
```

Arguments

`x`	api params

Value

logical indicating whether the query exceeds the specified cutoff point

`sn2id` *sn2id*

Description

sn2id

Usage

```
sn2id(screen_name)
```

Arguments

`screen_name`	Twitter handle

Value

response Twitter account user id

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

`tfse` *tfse: Twitter Follows and Selective Exposure*

Description

An implementation of http calls and data wrangling functions designed to interact with Twitter API's. GET and POST requests are designed to return data from Twitter's search and streaming API's. Returned [json] objects are automatically parsed into data_frames or lists. Specific consideration is given within the functions to use of more than one OAuth personal access token.

`top_tweet_words` *top_tweet_words*

Description

top_tweet_words

Usage

```
top_tweet_words(tweets_text, min = 3, exclude_words = NULL)
```

Arguments

`tweets_text`	character vector of tweets text
`min`	minimum number of ocurrences to include in returned object
`exclude_words`	other words to exclude

Value

list object with top mentions and top words

`try_catch` *try_catch*

Description

try_catch

Usage

```
try_catch(x)
```

Arguments

`x`	function call

TWIT *TWIT*

Description

The Twitter Search API is part of Twitter's REST API. It allows queries against the indices of recent or popular Tweets and behaves similarly to, but not exactly like the Search feature available in Twitter mobile or web clients, such as Twitter.com search. The Twitter Search API searches against a sampling of recent Tweets published in the past 7 days. The Search API is focused on relevance and not completeness. This means that some Tweets and users may be missing from search results. If you want to match for completeness you should consider using a Streaming API instead.

Usage

```
TWIT(query, parameters = NULL, token, parse = TRUE, version = "1.1",
  timeout = 120, file_name)
```

Arguments

`query`	Twitter API request type (e.g., `"friends/ids"`, `"users/lookup"`, `"statuses/filter"`).
`parameters`	Additional parameters passed along to API call
`token`	An OAuth token (1.0 or 2.0)
`parse`	logical indicating whether to parse json response object

Value

json response object as nested list

See Also

See `https://dev.twitter.com/overview/documentation` for more information on using Twitter's API.

which_ids *which_ids*

Description

Returns integer values. Used for get_friends function.

Usage

```
which_ids(n, max_users = 3000, token = NULL)
```

Arguments

`n` starting number for users

`max_users` max number of user ids (if rate limit exceeds remaining number of users, this sets upper ceiling and reduces likelihood of API request errors)

`token` Specify token if there is reason to believe current remaning friend list request is below the rate limit max of 15. This rate limit resets every 15 minutes, so this is usually not necessary. Checking rate limits does not reduce the number of available requests, but it does slow things down.

Value

integers used to identify 15 (or token max given rate limits) users from provided list of user ids

Index