Package ‘academictwitteR’

October 12, 2022

Title  Access the Twitter Academic Research Product Track V2 API
Endpoint

Version  0.3.1

Description  Package to query the Twitter Academic Research Product Track, providing access to full-archive search and other v2 API endpoints. Functions are written with academic research in mind. They provide flexibility in how the user wishes to store collected data, and encourage regular storage of data to mitigate loss when collecting large volumes of tweets. They also provide workarounds to manage and reshape the format in which data is provided on the client side.

License  MIT + file LICENSE

URL  https://github.com/cjbarrie/academictwitteR

BugReports  https://github.com/cjbarrie/academictwitteR/issues

Encoding  UTF-8

RoxygenNote  7.1.2

Depends  R (>= 3.4)

Imports  dplyr (>= 1.0.0), httr, jsonlite, magrittr, lubridate, usethis, tibble, tidyr, tidyselect, purrr, rlang, utils

Suggests  knitr, rmarkdown, devtools, testthat (>= 3.0.0), httptest, lifecycle, covr

VignetteBuilder  knitr

Config/testthat/edition 3

NeedsCompilation  no

Author  Christopher Barrie [aut, cre] (<https://orcid.org/0000-0002-9156-990X>), Justin Chun-ting Ho [aut] (<https://orcid.org/0000-0002-7884-1059>), Chung-hong Chan [ctb] (<https://orcid.org/0000-0002-6232-7530>), Noelia Rico [ctb] (<https://orcid.org/0000-0002-6169-4523>), Tim König [ctb] (<https://orcid.org/0000-0002-2852-2690>), Thomas Davidson [ctb] (<https://orcid.org/0000-0002-5947-7490>)

Maintainer  Christopher Barrie <christopher.barrie@ed.ac.uk>

Repository  CRAN

Date/Publication  2022-02-16 15:20:09 UTC
bind_tweets

Bind information stored as JSON files

Description

This function binds information stored as JSON files. The experimental function convert_json converts individual JSON files into either "raw" or "tidy" format.

Usage

bind_tweets(data_path, user = FALSE, verbose = TRUE, output_format = NA)

convert_json(data_file, output_format = "tidy")

Arguments

data_path: string, file path to directory of stored tweets data saved as data_id.json and users_id.json
user: If FALSE, this function binds JSON files into a data frame containing tweets; data frame containing user information otherwise. Ignore if output_format is not NA
verbose: If FALSE, messages are suppressed
**build_query**

**Description**

Build tweet query according to targeted parameters.

**Usage**

```r
build_query(
    query = NULL,
    exact_phrase = NULL,
    users = NULL,
    reply_to = NULL,
    retweets_of = NULL,
```

**Details**

By default, `bind_tweets` binds into a data frame containing tweets (from `data_id.json` files). If `users` is `TRUE`, it binds into a data frame containing user information (from `users_id.json`).

**Value**

A data frame containing either tweets or user information.

**Examples**

```r
# Not run:
# bind json files in the directory "data" into a data frame containing tweets
bind_tweets(data_path = "data/")

# bind json files in the directory "data" into a data frame containing user information
bind_tweets(data_path = "data/", user = TRUE)

# bind json files in the directory "data" into a "tidy" data frame / tibble
bind_tweets(data_path = "data/", user = TRUE, output_format = "tidy")
```

---

**output_format**

*Experimental* string, if it is not NA, this function return an unprocessed data frame containing either tweets or user information. Currently, this function supports the following format(s):

- "raw": List of data frames; Note: not all data frames are in Boyce-Codd 3rd Normal Form
- "tidy": Tidy format; all essential columns are available

**data_file**

String, a single file path to a JSON file; or a vector of file paths to JSON files of stored tweets data saved as `data_id.json`.
exclude = NULL,
is_retweet = NULL,
is_reply = NULL,
is_quote = NULL,
is_verified = NULL,
remove_promoted = FALSE,
has_hashtags = NULL,
has_cashtags = NULL,
has_links = NULL,
has_mentions = NULL,
has_media = NULL,
has_images = NULL,
has_videos = NULL,
has_geo = NULL,
place = NULL,
country = NULL,
point_radius = NULL,
bbox = NULL,
lang = NULL,
conversation_id = NULL,
url = NULL
)

Arguments

query string or character vector, search query or queries

exact_phrase If TRUE, only tweets will be returned matching the exact phrase

users string or character vector, user handles to collect tweets from the specified users

reply_to string or character vector, user handles to collect replies to the specified users

retweets_of string or character vector, user handles to collect retweets of tweets by the specified users

exclude string or character vector, tweets containing the keyword(s) will be excluded

is_retweet If TRUE, only retweets will be returned; if FALSE, retweets will be excluded; if NULL, both retweets and other tweet types will be returned.

is_reply If TRUE, only replies will be returned; if FALSE, replies will be excluded; if NULL, both replies and other tweet types will be returned.

is_quote If TRUE, only quote tweets will be returned; if FALSE, quote tweets will be excluded; if NULL, both quote tweets and other tweet types will be returned.

is_verified If TRUE, only tweets from verified accounts will be returned; if FALSE, tweets from verified accounts will be excluded; if NULL, both verified account tweets and tweets from non-verified accounts will be returned.

remove_promoted If TRUE, tweets created for promotion only on ads.twitter.com are removed

has_hashtags If TRUE, only tweets containing hashtags will be returned; if FALSE, tweets containing hashtags will be excluded; if NULL, both tweets containing hashtags and tweets without hashtags will be returned.
build_query

<table>
<thead>
<tr>
<th>has_cashtags</th>
<th>If TRUE, only tweets containing cashtags will be returned; if FALSE, tweets containing cashtags will be excluded; if NULL, both tweets containing cashtags and tweets without cashtags will be returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>has_links</td>
<td>If TRUE, only tweets containing links (and media) will be returned; if FALSE, tweets containing links (and media) will be excluded; if NULL, both tweets containing links (and media) and tweets without links (and media) will be returned.</td>
</tr>
<tr>
<td>has_mentions</td>
<td>If TRUE, only tweets containing mentions will be returned; if FALSE, tweets containing mentions will be excluded; if NULL, both tweets containing mentions and tweets without mentions will be returned.</td>
</tr>
<tr>
<td>has_media</td>
<td>If TRUE, only tweets containing media such as a photo, GIF, or video (as determined by Twitter) will be returned will be returned; if FALSE, tweets containing media will be excluded; if NULL, both tweets containing media and tweets without media will be returned.</td>
</tr>
<tr>
<td>has_images</td>
<td>If TRUE, only tweets containing (recognized URLs to) images will be returned will be returned will be returned; if FALSE, tweets containing images will be excluded; if NULL, both tweets containing images and tweets without images will be returned.</td>
</tr>
<tr>
<td>has_videos</td>
<td>If TRUE, only tweets containing contain videos (recognized as native videos uploaded directly to Twitter) will be returned will be returned; if FALSE, tweets containing videos will be excluded; if NULL, both tweets containing videos and tweets without videos will be returned.</td>
</tr>
<tr>
<td>has_geo</td>
<td>If TRUE, only tweets containing geo information (Tweet-specific geolocation data provided by the Twitter user) will be returned; if FALSE, tweets containing geo information will be excluded; if NULL, both tweets containing geo information and tweets without geo information will be returned.</td>
</tr>
<tr>
<td>place</td>
<td>string, name of place e.g. &quot;London&quot;</td>
</tr>
<tr>
<td>country</td>
<td>string, name of country as ISO alpha-2 code e.g. &quot;GB&quot;</td>
</tr>
<tr>
<td>point_radius</td>
<td>numeric, a vector of two point coordinates latitude, longitude, and point radius distance (in miles)</td>
</tr>
<tr>
<td>bbox</td>
<td>numeric, a vector of four bounding box coordinates from west longitude to north latitude</td>
</tr>
<tr>
<td>lang</td>
<td>string, a single BCP 47 language identifier e.g. &quot;fr&quot;</td>
</tr>
<tr>
<td>conversation_id</td>
<td>string, return tweets that share the specified conversation ID</td>
</tr>
<tr>
<td>url</td>
<td>string, url</td>
</tr>
</tbody>
</table>

Details

This function is already called within the main get_all_tweets function.

It may also be called separately and the output saved as a character object query string to be input as query parameter to get_all_tweets.

Value

a query string
count_all_tweets

Count tweets from full archive search

Description

This function returns aggregate counts of tweets by query string or strings between specified date ranges.

Usage

count_all_tweets(
    query = NULL,
    start_tweets, 
    end_tweets, 
    bearer_token = get_bearer(), 
    n = 100, 
    file = NULL, 
    data_path = NULL, 
    export_query = TRUE, 
    bind_tweets = TRUE, 
    granularity = "day", 
    verbose = TRUE, 
    ...
)

Arguments

query string or character vector, search query or queries
start_tweets string, starting date
end_tweets string, ending date
bearer_token string, bearer token
create_compliance_job

Description

This function creates a new compliance job and upload the Tweet IDs or user IDs. By default, the parameter x with the length of one is assumed to be a text file containing either Tweet IDs or user IDs. This default behavior can be bypassed using force_ids For example, if you want to check for just a single Tweet ID.
Usage

create_compliance_job(
  x,
  type = "tweets",
  bearer_token = get_bearer(),
  force_ids = FALSE,
  verbose = TRUE
)

Arguments

x either a character vector of Tweet IDs or user IDs; or a plain text file that each line contains a Tweet ID or user ID.

type the type of the job, whether "tweets" or "users".

bearer_token string, bearer token

force_ids logical, make sure x is treated as a character vector of Tweet IDs or user IDs.

verbose If FALSE, query progress messages are suppressed

Value

the job ID (invisibly)

Examples

## Not run:
create_compliance_job(x = "tweetids.txt", type = "tweets")

## End(Not run)

get_all_tweets

Get tweets from full archive search

Description

This function collects tweets by query string or strings between specified date ranges.

Usage

get_all_tweets(
  query = NULL,
  start_tweets, end_tweets, 
  bearer_token = get_bearer(),
  n = 100,
  file = NULL,
  data_path = NULL,
get_all_tweets

export_query = TRUE,
bind_tweets = TRUE,
page_n = 500,
context_annotations = FALSE,
verbose = TRUE,

Arguments

query string or character vector, search query or queries
start_tweets string, starting date
end_tweets string, ending date
bearer_token string, bearer token
n integer, upper limit of tweets to be fetched
file string, name of the resulting RDS file
data_path string, if supplied, fetched data can be saved to the designated path as jsons
export_query If TRUE, queries are exported to data_path
bind_tweets If TRUE, tweets captured are bound into a data.frame for assignment
page_n integer, amount of tweets to be returned by per page
context_annotations If TRUE, context_annotations will be fetched. Note it will limit the page_n to 100 due restrictions of Twitter API.
verbose If FALSE, query progress messages are suppressed

Details

The function can also collect tweets by users. These may be specified alongside a query string or without. When no query string is supplied, the function collects all tweets by that user.

If a filename is supplied, the function will save the result as a RDS file.

If a data path is supplied, the function will also return tweet-level data in a data/ path as a series of JSONs beginning "data_"; while user-level data will be returned as a series of JSONs beginning "users_".

Value

When bind_tweets is TRUE (default), the function returns a data frame. Nothing otherwise.
### Examples

```r
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

get_all_tweets(query = "BLM",
              start_tweets = "2020-01-01T00:00:00Z",
              end_tweets = "2020-01-05T00:00:00Z",
              bearer_token = get_bearer(),
              data_path = "data",
              n = 500)

get_all_tweets(users = c("cbarrie", "jack"),
              start_tweets = "2021-01-01T00:00:00Z",
              end_tweets = "2021-06-01T00:00:00Z",
              bearer_token = get_bearer(),
              n = 1000)

get_all_tweets(start_tweets = "2021-01-01T00:00:00Z",
              end_tweets = "2021-06-01T00:00:00Z",
              bearer_token = get_bearer(),
              n = 1500,
              conversation_id = "1392887366507970561")

## End(Not run)
```

---

**get_bearer**

### Manage bearer token

#### Description

This function attempts to retrieve your bearer token from the environmental variable "TWITTER_BEARER". The easiest way to set up this environmental variable is to use `set_bearer()` and insert your bearer token to `.Renviron` file following the format: `TWITTER_BEARER=YOURTOKENHERE`. Replace `YOURTOKENHERE` with your own token.

#### Usage

`get_bearer()`

#### Details

Note: for `get_bearer()` to retrieve your bearer token you will need to restart the R session after storing in `.Renviron`.

#### Value

string represents your bearer token, if it the environmental variable "TWITTER_BEARER" has been preset.
get_compliance_result

Get Compliance Result

Description

This function retrieves the information for a single compliance job.

Usage

get_compliance_result(id, bearer_token = get_bearer(), verbose = TRUE)

Arguments

id string, the job id
bearer_token string, bearer token
verbose If FALSE, query progress messages are suppressed

Value

a data frame

Examples

## Not run:
get_compliance_result("1460077048991555585")
## End(Not run)

get_liked_tweets

Get liked tweets

Description

This function fetches returns tweets liked by a user or users.

Usage

get_liked_tweets(x, bearer_token = get_bearer(), ...)

Arguments

x string containing one user id or a vector of user ids
bearer_token string, bearer token
... arguments passed to other backend functions
get_liking_users

Value

a data frame

Examples

## Not run:
users <- c("2244994945", "95226101")
get_liked_tweets(users, bearer_token = get_bearer())
## End(Not run)

Description

This function fetches users who liked a tweet or tweets.

Usage

get_liking_users(x, bearer_token = get_bearer(), verbose = TRUE)

Arguments

x string containing one tweet id or a vector of tweet ids
bearer_token string, bearer token
verbose If FALSE, query progress messages are suppressed

Value

a data frame

Examples

## Not run:
tweet <- "1387744422729748486"
get_liking_users(tweet, bearer_token = get_bearer())
## End(Not run)
**get_retweeted_by**  
*Get users who has retweeted a tweet*

**Description**
This function fetches users who retweeted a tweet

**Usage**
```r
get_retweeted_by(
  x,
  bearer_token = get_bearer(),
  data_path = NULL,
  verbose = TRUE
)
```

**Arguments**
- `x`  
  string containing one tweet id or a vector of tweet ids
- `bearer_token`  
  string, bearer token
- `data_path`  
  string, if supplied, fetched data can be saved to the designated path as jsons
- `verbose`  
  If FALSE, query progress messages are suppressed

**Value**

a data frame

**Examples**

```r
## Not run:
tweets <- c("1392887366507970561","1409931481552543749")
get_retweeted_by(tweets, bearer_token = get_bearer())
## End(Not run)
```

---

**get_user_followers**  
*Get user followers*

**Description**
This function fetches users who are followers of the specified user ID.

**Usage**
```r
get_user_followers(x, bearer_token = get_bearer(), ...)
```
get_user_following

Arguments

x string containing one user id or a vector of user ids
bearer_token string, bearer token
... arguments passed to other backend functions

Value

a data frame

Examples

## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- "224494945"
get_user_following(users, bearer_token = get_bearer())

## End(Not run)

get_user_following Get user following

Description

This function fetches a list of users the specified user ID is following.

Usage

get_user_following(x, bearer_token = get_bearer(), ...)

Arguments

x string containing one user id or a vector of user ids
bearer_token string, bearer token
... arguments passed to other backend functions

Value

a data frame

Examples

## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- "224494945"
get_user_following(users, bearer_token)

## End(Not run)
get_user_id

Description

This function get the user IDs (e.g. 1349149096909668363) of given usernames, e.g. "potus".

Usage

get_user_id(
  usernames,
  bearer_token = get_bearer(),
  all = FALSE,
  keep_na = TRUE
)

Arguments

usernames character vector containing screen names to be queried
bearer_token string, bearer token
all logical, default FALSE to get a character vector of user IDs. Set it to TRUE to get a data frame, see below
keep_na logical, default TRUE to keep usernames that cannot be queried. Set it to TRUE to exclude those usernames. Only useful when all is FALSE

Value

a string vector with the id of each of the users unless all = TRUE. If all = TRUE, a data.frame with ids, names (showed on the screen) and usernames is returned.

Examples

## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- c("Twitter", "TwitterDev")
get_user_id(users, bearer_token)

## End(Not run)
## get_user_profile

### Description

This function fetches user-level information for a vector of user IDs.

### Usage

```r
get_user_profile(x, bearer_token = get_bearer())
```

### Arguments

- **x**: string containing one user id or a vector of user ids
- **bearer_token**: string, bearer token

### Value

a data frame

### Examples

```r
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- c("2244994945", "6253282")
get_user_profile(users, bearer_token)
## End(Not run)
```

## get_user_timeline

### Description

This function collects tweets by an user ID from the users endpoint.

### Usage

```r
generate_user_timeline(
  x, 
  start_tweets, 
  end_tweets, 
  bearer_token = get_bearer(), 
  n = 100, 
  file = NULL, 
  data_path = NULL,
)```
get_user_timeline

export_query = TRUE,
bind_tweets = TRUE,
page_n = 100,
verbose = TRUE,
...  
)

Arguments

x string containing one user id or a vector of user ids
start_tweets string, starting date
end_tweets string, ending date
bearer_token string, bearer token
n integer, upper limit of tweets to be fetched
file string, name of the resulting RDS file
data_path string, if supplied, fetched data can be saved to the designated path as jsons
export_query If TRUE, queries are exported to data_path
bind_tweets If TRUE, tweets captured are bound into a data.frame for assignment
page_n integer, amount of tweets to be returned by per page
verbose If FALSE, query progress messages are suppressed
... arguments will be passed to build_query() function. See ?build_query() for further information.

Details

Only the most recent 3,200 Tweets can be retrieved.
If a filename is supplied, the function will save the result as a RDS file.
If a data path is supplied, the function will also return tweet-level data in a data/ path as a series of JSONs beginning "data_"; while user-level data will be returned as a series of JSONs beginning "users_".
When bind_tweets is TRUE, the function returns a data frame.

Value

a data.frame

Examples

## Not run:

get_user_timeline("2244994945",
  start_tweets = "2020-01-01T00:00:00Z",
  end_tweets = "2021-05-14T00:00:00Z",
  bearer_token = get_bearer(),
  n = 200)

## End(Not run)
hydrate_tweets  

*Hydrate Tweets Based On Tweet IDs*

**Description**

This function is helpful for hydrating Tweet IDs (i.e. getting the full content of tweets from a list of Tweet IDs).

**Usage**

```r
hydrate_tweets(
  ids,
  bearer_token = get_bearer(),
  data_path = NULL,
  context_annotations = FALSE,
  bind_tweets = TRUE,
  verbose = TRUE,
  errors = FALSE
)
```

**Arguments**

- `ids` a character vector of Tweet IDs
- `bearer_token` string, bearer token
- `data_path` string, if supplied, fetched data can be saved to the designated path as jsons
- `context_annotations` If TRUE, context_annotations will be fetched.
- `bind_tweets` If TRUE, tweets captured are bound into a data.frame for assignment
- `verbose` If FALSE, query progress messages are suppressed
- `errors` logical, if TRUE, the error capturing mechanism is enabled. See details below.

**Details**

When the error capturing mechanism is enabled, Tweets IDs that cannot be queried (e.g. with error) are stored as `errors_*.json` files. If `bind_tweets` is TRUE, those error Tweets IDs are retained in the returned data.frame with the column `error` indicating the error.

**Value**

When `bind_tweets` is TRUE, the function returns a data frame. The `data_path` (invisibly) if `bind_tweets` is FALSE.
list_compliance_jobs

Description

This function lists all compliance jobs.

Usage

list_compliance_jobs(type = "tweets", bearer_token = get_bearer())

Arguments

type the type of the job, whether "tweets" or "users".
bearer_token string, bearer token

Value

a data frame

Examples

## Not run:
list_compliance_jobs()

## End(Not run)
resume_collection  Resume previous collection

Description
This function resumes a previous interrupted collection session.

Usage
resume_collection(data_path, bearer_token = get_bearer(), verbose = TRUE, ...)

Arguments
- data_path: string, name of an existing data_path
- bearer_token: string, bearer token
- verbose: If FALSE, query progress messages are suppressed
- ...: arguments will be passed to get_all_tweets() function. See ?get_all_tweets() for further information.

Details
For this function to work, export_query must be set to "TRUE" during the original collection.

Value
a data.frame

Examples
## Not run:
resume_collection(data_path = "data", bearer_token = get_bearer())
## End(Not run)

set_bearer  Set bearer token

Description
This function lets the user add their bearer token to the .Renviron file.

Usage
set_bearer()
Details

It is in general not safe to 1) hard code your bearer token in your R script or 2) have your bearer token in your command history.

set_bearer opens the .Renviron file for the user and provides instructions on how to add the bearer token, which requires the addition of just one line in the .Renviron file, following the format TWITTER_BEARER=YOURTOKENHERE. Replace YOURTOKENHERE with your own token.

update_collection  Update previous collection session

Description

This function continues a previous collection session with a new end date. For this function to work, export_query must be set to "TRUE" during the original collection.

Usage

update_collection(
data_path,
end_tweets,
bearer_token = get_bearer(),
verbose = TRUE,
...
)

Arguments

data_path  string, name of an existing data_path
date_tweets  string, ending date
bearer_token  string, bearer token
verbose  If FALSE, query progress messages are suppressed
...

Value

a data.frame

Examples

## Not run:
update_collection(data_path = "data", "2020-01-03T00:00:00Z", bearer_token = get_bearer())

## End(Not run)
Index

bind_tweets, 2
build_query, 3
build_query(), 9

convert_json(bind_tweets), 2
count_all_tweets, 6
create_compliance_job, 7

get_all_tweets, 5, 8
get_bearer, 10
get_compliance_result, 11
getLiked_tweets, 11
get_liking_users, 12
get_retweeted_by, 13
get_user_followers, 13
get_user_following, 14
get_user_id, 15
get_user_profile, 16
get_user_timeline, 16

hydrate_tweets, 18

list_compliance_jobs, 19

resume_collection, 20

set_bearer, 20

update_collection, 21