Package 'tidytags'

January 10, 2023

Title Importing and Analyzing 'Twitter' Data Collected with 'Twitter Archiving Google Sheets'

Version 1.1.1

License MIT + file LICENSE

Description The 'tidytags' package coordinates the simplicity of collecting tweets over time with a 'Twitter Archiving Google Sheet' (TAGS; <https://tags.hawksey.info/>) and the utility of the 'rtweet' package (<https://docs.ropensci.org/rtweet/>) for processing and preparing additional 'Twitter' metadata. 'tidytags' also introduces functions developed to facilitate systematic yet flexible analyses of data from 'Twitter'.

Language en-US

URL https://docs.ropensci.org/tidytags/ (website)
https://github.com/ropensci/tidytags

Depends R (>= 4.2)

Imports dplyr (>= 1.0), googlesheets4 (>= 1.0), rlang (>= 1.0), rtweet (>= 1.1), stringr (>= 1.4)

Suggests beepr, covr, ggplot2, ggraph, knitr, longurl, readr, rmarkdown, testthat, tibble, tidygraph, urltools, vcr (>= 1.2)

Encoding UTF-8

BugReports https://github.com/ropensci/tidytags/issues

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author K. Bret Staudt Willet [aut, cre] (<https://orcid.org/0000-0002-6984-416X>), Joshua M. Rosenberg [aut] (<https://orcid.org/0000-0003-2170-0447>), Lluís Revilla Sancho [rev] (<https://orcid.org/0000-0001-9747-2570>), Marion Louveaux [rev] (<https://orcid.org/0000-0002-1794-3748>)

Maintainer K. Bret Staudt Willet <bret.staudtwillet@fsu.edu>

Repository CRAN

Date/Publication 2023-01-10 08:10:05 UTC

R topics documented:

`add_users_data` *Retrieve user information for everyone in an edgelist*

Description

Updates an edgelist created with `create_edgelist()` by appending user data retrieved with `rtweet::lookup_users()`. The resulting dataframe adds many additional columns and appends "_sender" or "_receiver" to the column names.

Usage

```
add_users_data(edgelist)
```

Arguments

`edgelist`	An edgelist of senders and receivers, such as that returned by the function `create_edgelist()`.

Details

This function requires authentication; please see `vignette("setup", package = "tidytags")`

Value

A dataframe in the form of an edgelist (i.e., with senders and receivers) as well as numerous, appropriately named columns of details about the senders and receivers.

See Also

Read more about rtweet authentication setup at `vignette("auth", package = "rtweet")`

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)

if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content)
  add_users_data(create_edgelist(tweets_data))
}
```

`create_edgelist` *Create an edgelist where senders and receivers are defined by different types of Twitter interactions*

Description

Starting with a dataframe of Twitter data imported to R with `read_tags()` and additional metadata retrieved by `pull_tweet_data()`, `create_edgelist()` removes any statuses that are not of the requested type (e.g., replies, retweets, and quote tweets) by calling `filter_by_tweet_type()`. Finally, `create_edgelist()` pulls out senders and receivers of the specified type of statuses, and then adds a new column called `edge_type`.

Usage

```
create_edgelist(df, type = "all")
```

Arguments

`df`	A dataframe returned by `pull_tweet_data()`
`type`	The specific kind of statuses used to define the interactions around which the edgelist will be built. Choices include "reply", "retweet", or "quote". Defaults to "all".

Value

A dataframe edgelist defined by interactions through the type of statuses specified. The dataframe has three columns: `sender`, `receiver`, and `edge_type`.

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)
```

```
if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content)
  full_edgelist <- create_edgelist(tweets_data)
  full_edgelist

  reply_edgelist <- create_edgelist(tweets_data, type = "reply")
  retweet_edgelist <- create_edgelist(tweets_data, type = "retweet")
  quote_edgelist <- create_edgelist(tweets_data, type = "quote")
}
```

`filter_by_tweet_type` *Filter a Twitter dataset to only include statuses of a particular type*

Description

Starting with a dataframe of Twitter data imported to R with `read_tags()` and additional metadata retrieved by `pull_tweet_data()`, `filter_by_tweet_type()` processes the statuses by calling `process_tweets()` and then removes any statuses that are not of the requested type (e.g., replies, retweets, and quote tweets). `filter_by_tweet_type()` is a useful function in itself, but it is also used in `create_edgelist()`.

Usage

```
filter_by_tweet_type(df, type)
```

Arguments

`df`	A dataframe returned by `pull_tweet_data()`
`type`	The specific kind of statuses that will be kept in the dataset after filtering the rest. Choices for `type`include "reply", "retweet", "quote", and "original".

Value

A dataframe of processed statuses and fewer rows that the input dataframe. Only the statuses of the specified type will remain.

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)

if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content)
  only_replies <- filter_by_tweet_type(tweets_data, "reply")
  only_retweets <- filter_by_tweet_type(tweets_data, "retweet")
```

```
  only_quote_tweets <- filter_by_tweet_type(tweets_data, "quote")
  only_originals <- filter_by_tweet_type(tweets_data, "original")
}
```

`get_char_tweet_ids` *Get Twitter status ID numbers as character strings*

Description

This function is useful because Google Sheets (and hence TAGS) typically round very large numbers into an exponential form. Thus, because status ID numbers are very large, they often get corrupted in this rounding process. The most reliable way to get full status ID numbers is by using this function, `get_char_tweet_ids()`, to pull the ID numbers from the URL linking to specific statuses.

Usage

```
get_char_tweet_ids(x)
```

Arguments

`x`	A dataframe containing the column name 'status_url' (i.e., the hyperlink to specific statuses), such as that returned by `read_tags()`, or a vector of status URLs, such as as those contained in the 'status_url' column of a dataframe returned by `tidytags::read_tags()`

Value

A vector of Twitter status IDs as character strings

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)
get_char_tweet_ids(tags_content[1:10, ])
get_char_tweet_ids(tags_content$status_url[1:10])
get_char_tweet_ids(
  "https://twitter.com/tweet__example/status/1176592704647716864")
```

`get_upstream_tweets` *Collect upstream statuses and add to dataset*

Description

Because the Twitter API offers a `in_reply_to_status_id_str` column, it is possible to iteratively reconstruct reply threads in an *upstream* direction, that is, retrieving statuses composed earlier than replies in the dataset. The `get_upstream_tweets()` function collects upstream replies not previously found in the dataset. Keep in mind that there is no way to predict how far upstream you can trace back a reply thread, so running `get_upstream_tweets()` could take a while and potentially hit the Twitter API rate limit of 90,000 statuses in a 15-minute period.

Usage

```
get_upstream_tweets(df)
```

Arguments

`df`	A dataframe of statuses and full metadata from the Twitter API as returned by `pull_tweet_data()`

Details

This function requires authentication; please see `vignette("setup", package = "tidytags")`

Value

A new, expanded dataframe which includes any retrievable upstream replies

See Also

Read more about rtweet authentication setup at `vignette("auth", package = "rtweet")`

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)

if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content)
  more_replies_df <- get_upstream_tweets(tweets_data)
  more_replies_df
}
```

get_url_domain *Find the domain name of URLs, even shortened URLs*

Description

`get_url_domain()` retrieves the Web domain name from a URL, including URLs shortened with services such as bit.ly and t.co

Usage

```
get_url_domain(x, wait = 10)
```

Arguments

`x`	A list or vector of hyperlinks, whether shortened or expanded
`wait`	How long (in seconds) to wait on the `longurl::expand_urls()` function to retrieve the full, expanded URL from a shortened URL (e.g., a bit.ly). The `longurl` default is 2 seconds, but we have found that this misses a number of valid URLs. Here, we have made the default `wait = 10` seconds, but the user can adjust this as they like.

Value

A list or vector of Web domain names

See Also

Read the documentation for `longurl::expand_urls()` and `urltools::domain()`.

Examples

```
get_url_domain("https://www.tidyverse.org/packages/")
get_url_domain("https://dplyr.tidyverse.org/")
get_url_domain("http://bit.ly/2SfWO3K")
```

lookup_many_tweets *Retrieve the fullest extent of metadata for more than 90,000 statuses*

Description

This function calls `pull_tweet_data()`, but has a built-in delay of 15 minutes to allow the Twitter API to reset after looking up 90,000 statuses

Usage

```
lookup_many_tweets(x, alarm = FALSE)
```

Arguments

x	A list or vector of status ID numbers
alarm	An audible notification that a batch of 90,000 statuses has been completed

Details

This function requires authentication; please see `vignette("setup", package = "tidytags")`

Value

A dataframe of statuses and full metadata from the Twitter API

See Also

Read more about rtweet authentication setup at `vignette("auth", package = "rtweet")`

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)

if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content$id_str)
  one_tweet_data <- lookup_many_tweets("1176592704647716864")
  one_tweet_data <- lookup_many_tweets("1176592704647716864", alarm = TRUE)
  one_tweet_data
}
```

---

process_tweets *Calculate additional information using status metadata*

Description

Calculate additional information using status metadata

Usage

```
process_tweets(df)
```

Arguments

df	A dataframe of statuses and full metadata from the Twitter API as returned by pull_tweet_data()

Value

A dataframe with several additional columns: mentions_count, hashtags_count, urls_count, tweet_type, is_self_reply

Examples

```
example_url <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_url)

if (rtweet::auth_has_default()) {
  tweets_data <- lookup_many_tweets(tags_content)
  tweets_processed <- process_tweets(tweets_data)
  tweets_processed
}
```

---

pull_tweet_data *Retrieve the fullest extent of status metadata available from the Twitter API*

Description

With a TAGS archive imported into R, pull_tweet_data() uses the **rtweet** package to query the Twitter API. Using rtweet requires Twitter API keys associated with an approved developer account. Fortunately, the rtweet vignette, Authentication, provides a thorough guide to obtaining Twitter API keys and authenticating access to the Twitter API. Following the directions for "Apps," you will run the rtweet::rtweet_app() function.

Usage

```
pull_tweet_data(df = NULL, url_vector = NULL, id_vector = NULL, n = NULL)
```

Arguments

df	A dataframe of containing the column name 'status_url' (i.e., the hyperlink to specific statuses), such as that returned by read_tags()
url_vector	A vector of status URLs, such as as those contained in the 'status_url' column of a dataframe returned by tidytags::read_tags()
id_vector	A vector of statuses (i.e., ID numbers, such as those contained in the 'id_str' column of a dataframe returned by tidytags::read_tags()

n — The number of statuses to look up, by default the total number of tweet ID numbers available, but capped at 90,000 due to Twitter API limitations.

Details

This function requires authentication; please see `vignette("setup", package = "tidytags")`

Value

A dataframe of statuses and full metadata from the Twitter API

See Also

Read more about rtweet authentication setup at `vignette("auth", package = "rtweet")`

Examples

```
## Import data from a TAGS tracker:
example_tags_tracker <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
tags_content <- read_tags(example_tags_tracker)

if (rtweet::auth_has_default()) {
  ## Use any of three input parameters (TAGS dataframe, `status_url`
  ##   column, or `id_str` column)
  tweets_data_from_df <- pull_tweet_data(tags_content)
  tweets_data_from_url <-
    pull_tweet_data(url_vector = tags_content$status_url)
  tweets_data_from_ids <- pull_tweet_data(id_vector = tags_content$id_str)

  ## Specifying the parameter `n` clarifies how many statuses to look up,
  ##   but the returned values may be less than `n` because some statuses
  ##   may have been deleted or made protected since the TAGS tracker
  ##   originally recorded them.
  tweets_data_10 <- pull_tweet_data(tags_content, n = 10)

  ## Note that the following two examples will return the same thing:
  one_tweet_data <-
    pull_tweet_data(url_vector =
      "https://twitter.com/tweet__example/status/1176592704647716864")
  one_tweet_data <- pull_tweet_data(id_vector = "1176592704647716864")
  one_tweet_data
}
```

read_tags *Retrieve a TAGS archive of Twitter statuses and bring into R*

Description

Keep in mind that `read_tags()` uses the **googlesheets4** package, and one requirement is that your TAGS tracker has been "published to the web." To do this, with the TAGS page open in a web browser, navigate to `File >> Share >> Publish to the web`. The `Link` field should be 'Entire document' and the `Embed` field should be 'Web page.' If everything looks right, then click the `Publish` button. Next, click the `Share` button in the top right corner of the Google Sheets browser window, select `Get shareable link`, and set the permissions to 'Anyone with the link can view.'

Usage

```
read_tags(tags_id)
```

Arguments

`tags_id`	A Google Sheet identifier (i.e., the alphanumeric string following "https://docs.google.com/spreadsheets/d in the TAGS tracker's URL.)

Value

A tibble of the TAGS archive of Twitter statuses

See Also

Read more about `library(googlesheets4)` here.

Examples

```
example_tags <- "18clYlQeJOc6W5QRuSlJ6_v3snqKJImFhU42bRkM_OX8"
read_tags(example_tags)
```

Index