Package 'rtoot'

January 9, 2023

Title Collecting and Analyzing Mastodon Data

Version 0.3.0

Description An implementation of calls designed to collect and organize Mastodon data via its Application Program Interfaces (API), which can be found at the following URL: <https://docs.joinmastodon.org/>.

License MIT + file LICENSE

URL https://schochastics.github.io/rtoot/, https://github.com/schochastics/rtoot/

BugReports https://github.com/schochastics/rtoot/issues

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.6)

Imports clipr, curl, dplyr, httr, jsonlite, tibble

Suggests knitr, rmarkdown, testthat (>= 3.0.0), rstudioapi, vcr (>= 0.6.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author David Schoch [aut, cre] (<https://orcid.org/0000-0003-2952-4812>), Chung-hong Chan [aut] (<https://orcid.org/0000-0002-6232-7530>), Johannes Gruber [ctb] (<https://orcid.org/0000-0001-9177-1772>)

Maintainer David Schoch <david@schochastics.net>

Repository CRAN

Date/Publication 2023-01-09 08:40:02 UTC

R topics documented:

`auth_setup` *Authenticate with a Mastodon instance*

Description

Authenticate with a Mastodon instance

Usage

```
auth_setup(
  instance = NULL,
  type = NULL,
  name = NULL,
  path = NULL,
  clipboard = FALSE,
  verbose = TRUE,
```

```
  browser = TRUE
)
```

Arguments

instance	a public instance of Mastodon (e.g., mastodon.social).
type	Either "public" to create a public authentication or "user" to create authentication for your user (e.g., if you want to post from R or query your followers).
name	give the token a name, in case you want to store more than one.
path	path to store the token in. The default is to store tokens in the path returned by `tools::R_user_dir("rtoot", "config")`.
clipboard	logical, whether to export the token to the clipboard
verbose	logical whether to display messages
browser	if TRUE (default) a browser window will be opened to authenticate, else the URL will be provided so you can copy/paste this into the browser yourself

Details

If either name or path are set to FALSE, the token is only returned and not saved. If you would like to save your token as an environment variable, please set clipboard to TRUE. Your token will be copied to clipboard in the environment variable format. Please paste it into your environment file, e.g. ".Renviron", and restart your R session.

Value

A bearer token

See Also

verify_credentials(), convert_token_to_envvar()

Examples

```
## Not run:
auth_setup("mastodon.social", "public")

## End(Not run)
```

---

convert_token_to_envvar

Convert token to environment variable

Description

Convert token to environment variable

Usage

```
convert_token_to_envvar(token, clipboard = TRUE, verbose = TRUE)
```

Arguments

token	bearer token, either public or user level
clipboard	logical, whether to export the token to the clipboard
verbose	logical whether to display messages

Value

Token (in environment variable format), invisibily

Examples

```
## Not run:
x <- auth_setup("mastodon.social", "public")
envvar <- convert_token_to_envvar(x)
envvar

## End(Not run)
```

---

create_token *get a bearer token for the mastodon api*

Description

get a bearer token for the mastodon api

Usage

```
create_token(client, type = "public", browser = TRUE)
```

Arguments

client	rtoot client object created with get_client
type	one of "public" or "user". See details
browser	if TRUE (default) a browser window will be opened to authenticate, else the URL will be provided so you can copy/paste this into the browser yourself

Details

TBA

Value

a mastodon bearer token

References

https://docs.joinmastodon.org/client/authorized/

`get_account` *Query the instance for a specific user*

Description

Query the instance for a specific user

Usage

```
get_account(id, instance = NULL, token = NULL, anonymous = FALSE, parse = TRUE)
```

Arguments

`id`	character, Local ID of a user (this is not the username)
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`token`	user bearer token (read from file by default)
`anonymous`	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Value

an account

Examples

```
## Not run:
get_account("109302436954721982")

## End(Not run)
```

`get_account_blocks` *Get blocks of user*

Description

Get blocks of user

Usage

```
get_account_blocks(
  max_id,
  since_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`limit`	integer, Maximum number of results to return
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.
`retryonratelimit`	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
`verbose`	logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

tibble or list of blocked users

Examples

```
## Not run:
# needs user level token
get_account_blocks()

## End(Not run)
```

`get_account_bookmarks` *Get bookmarks of user*

Description

Get bookmarks of user

Usage

```
get_account_bookmarks(
  max_id,
  since_id,
  min_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`min_id`	character, Return results younger than this id
`limit`	integer, Maximum number of results to return
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.
`retryonratelimit`	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
`verbose`	logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

bookmarked statuses

Examples

```
## Not run:
get_account_followers("109302436954721982")

## End(Not run)
```

`get_account_favourites` *Get favourites of user*

Description

Get favourites of user

Usage

```
get_account_favourites(
  max_id,
  min_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`max_id`	character, Return results older than this id
`min_id`	character, Return results younger than this id
`limit`	integer, Maximum number of results to return
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.

verbose
: logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

tibble or list of favourited statuses

Examples

```
## Not run:
# needs user level token
get_account_favourites()

## End(Not run)
```

---

get_account_featured_tags

Get featured tags of a user

Description

Get featured tags of a user

Usage

```
get_account_featured_tags(id, token = NULL, parse = TRUE)
```

Arguments

id
: character, local ID of a user (this is not the username)

token
: user bearer token (read from file by default)

parse
: logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Details

this functions needs a user level auth token

Value

tibble or list of featured_tags

Examples

```
## Not run:
get_account_featured_tags("109302436954721982")

## End(Not run)
```

`get_account_followers` *Get followers of a user*

Description

Get followers of a user

Usage

```
get_account_followers(
  id,
  max_id,
  since_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`id`	character, local ID of a user (this is not the username)
`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`limit`	integer, maximum number of results to return. Defaults to 40.
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.
`retryonratelimit`	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
`verbose`	logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

tibble or list of followers

Examples

```
## Not run:
get_account_followers("109302436954721982")

## End(Not run)
```

---

get_account_following *Get accounts a user follows*

Description

Get accounts a user follows

Usage

```
get_account_following(
  id,
  max_id,
  since_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

id	character, local ID of a user (this is not the username)
max_id	character, Return results older than this id
since_id	character, Return results newer than this id
limit	integer, Maximum number of results to return
token	user bearer token (read from file by default)
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.

verbose
: logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

tibble or list of accounts a user follows

Examples

```
## Not run:
get_account_following("109302436954721982")

## End(Not run)
```

---

get_account_lists *Get lists containing the user*

Description

Get lists containing the user

Usage

```
get_account_lists(id, token = NULL, parse = TRUE)
```

Arguments

id
: character, local ID of a user (this is not the username)

token
: user bearer token (read from file by default)

parse
: logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Details

this functions needs a user level auth token

Value

tibble or list of lists

Examples

```
## Not run:
get_account_lists("109302436954721982")

## End(Not run)
```

`get_account_mutes` *Get mutes of user*

Description

Get mutes of user

Usage

```
get_account_mutes(
  max_id,
  since_id,
  limit = 40L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`limit`	integer, Maximum number of results to return
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.
`retryonratelimit`	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
`verbose`	logical whether to display messages

Details

this functions needs a user level auth token. If limit>40, automatic pagination is used. You may get more results than requested.

Value

tibble or list of muted users

Examples

```
## Not run:
# needs user level token
get_account_mutes()

## End(Not run)
```

`get_account_relationships` *Find out whether a given account is followed, blocked, muted, etc.*

Description

Find out whether a given account is followed, blocked, muted, etc.

Usage

```
get_account_relationships(ids, token = NULL, parse = TRUE)
```

Arguments

`ids`	vector of account ids
`token`	user bearer token (read from file by default)
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Details

this functions needs a user level auth token

Value

tibble or list of relationships

Examples

```
## Not run:
fol <- get_account_followers("109302436954721982")
get_account_relationships(fol$id)

## End(Not run)
```

`get_account_statuses` *Get statuses from a user*

Description

Get statuses from a user

Usage

```
get_account_statuses(
  id,
  max_id,
  since_id,
  min_id,
  limit = 20L,
  exclude_reblogs = FALSE,
  hashtag,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`id`	character, local ID of a user (this is not the username)
`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`min_id`	character, Return results immediately newer than this id
`limit`	integer, Maximum number of results to return
`exclude_reblogs`	logical, Whether to filter out boosts from the response.
`hashtag`	character, filter for statuses using a specific hashtag.
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`token`	user bearer token (read from file by default)
`anonymous`	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.

retryonratelimit
: If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.

verbose
: logical whether to display messages

Details

For anonymous calls only public statuses are returned. If a user token is supplied also private statuses the user is authorized to see are returned

Value

tibble or list of statuses

Examples

```
## Not run:
get_account_statuses("109302436954721982")

## End(Not run)
```

---

get_client *register a mastodon client*

Description

register a mastodon client

Usage

```
get_client(instance = "mastodon.social")
```

Arguments

instance
: server name

Value

an rtoot client object

References

https://docs.joinmastodon.org/client/token/#creating-our-application

get_context *View statuses above and below this status in the thread*

Description

Query the instance for information about the context of a specific status. A context contains statuses above and below a status in a thread.

Usage

```
get_context(id, instance = NULL, token = NULL, anonymous = FALSE, parse = TRUE)
```

Arguments

id	character, local ID of a status in the database
instance	character, the server name of the instance where the status is located. If NULL, the same instance used to obtain the token is used.
token	user bearer token (read from file by default)
anonymous	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
parse	logical, if TRUE, the default, returns a named list of two tibbles, representing the ancestors (statuses above the status) and descendants (statuses below the status). Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Value

context of a toot as tibble or list

Examples

```
## Not run:
get_context(id = "109294719267373593", instance = "mastodon.social")

## End(Not run)
```

get_fedi_instances *Get a list of fediverse servers*

Description

Get a list of fediverse servers

Usage

```
get_fedi_instances(token = NA, n = 20, ...)
```

Arguments

token	token from instances.social (this is different from your Mastodon token!)
n	number of servers to show
...	additional parameters for the instances.social API. See https://instances.social/api/doc/

Details

This function uses the API at instances.social and needs a separate token. Results are sorted by number of users

Value

tibble of fediverse instances

Examples

```
## Not run:
get_fedi_instances(n = 5)

## End(Not run)
```

---

get_instance *Get various information about a specific instance*

Description

Get various information about a specific instance

Usage

```
get_instance_general(instance = NULL, token = NULL, anonymous = TRUE)

get_instance_peers(instance = NULL, token = NULL, anonymous = TRUE)

get_instance_activity(instance = NULL, token = NULL, anonymous = TRUE)

get_instance_emoji(instance = NULL, token = NULL, anonymous = TRUE)

get_instance_directory(
  instance = NULL,
  token = NULL,
  offset = 0,
  limit = 40,
  order = "active",
  local = FALSE,
  anonymous = TRUE,
```

```
  parse = TRUE
)

get_instance_trends(
  instance = NULL,
  token = NULL,
  limit = 10,
  anonymous = TRUE
)

get_instance_rules(instance = NULL, token = NULL, anonymous = TRUE)

get_instance_blocks(instance = NULL, token = NULL, anonymous = TRUE)
```

Arguments

instance	character, the server name of the instance where the status is located. If NULL, the same instance used to obtain the token is used.
token	user bearer token (read from file by default)
anonymous	logical, should the API call be made anonymously? Defaults to TRUE but some instances might need authentication here
offset	How many accounts to skip before returning results. Default 0.
limit	integer, Maximum number of results to return
order	'active' to sort by most recently posted statuses (default) or 'new' to sort by most recently created profiles.
local	logical, show only local accounts?
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Details

get_instance_general Returns general information about the instance

get_instance_peers Returns the peers of an instance

get_instance_activity Shows the weekly activity of the instance (3 months)

get_instance_emoji Lists custom emojis available on the instance

get_instance_directory A directory of profiles that the instance is aware of

get_instance_trends Tags that are being used more frequently within the past week

get_instance_rules Prints the rules of an instance

get_instance_blocks List of domains that are blocked by an instance.

Value

instance details as list or tibble depending on call function

Examples

```
## Not run:
 get_instance_general("mastodon.social")
 get_instance_activity("mastodon.social")
 get_instance_emoji("mastodon.social")
 get_instance_peers("mastodon.social")
 get_instance_directory("mastodon.social",limit = 2)

## End(Not run)
```

`get_poll` *View a poll*

Description

View a polls attached to statuses. To discover poll ID, you will need to use `get_status()` and look into the `poll`.

Usage

```
get_poll(id, instance = NULL, token = NULL, anonymous = FALSE, parse = TRUE)
```

Arguments

`id`	character, ID of the poll in the database
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`token`	user bearer token (read from file by default)
`anonymous`	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
`parse`	logical, if `TRUE`, the default, returns a tibble. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Value

a poll

Examples

```
## Not run:
get_poll(id = "105976")

## End(Not run)
```

get_status *View information about a specific status*

Description

Query the instance for information about a specific status. get_status returns complete information of a status. get_reblogged_by returns who boosted a given status. get_favourited_by returns who favourited a given status.

Usage

```
get_status(id, instance = NULL, token = NULL, anonymous = FALSE, parse = TRUE)

get_reblogged_by(
  id,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE
)

get_favourited_by(
  id,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE
)
```

Arguments

id	character, local ID of a status in the database
instance	character, the server name of the instance where the status is located. If NULL, the same instance used to obtain the token is used.
token	user bearer token (read from file by default)
anonymous	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Value

a status or a list of users

Examples

```
## Not run:
get_status(id = "109298295023649405")
get_reblogged_by(id = "109294719267373593")
get_favourited_by(id = "109294719267373593")

## End(Not run)
```

`get_timeline_hashtag` *Get hashtag timeline*

Description

Query the instance for the timeline of a specific hashtag

Usage

```
get_timeline_hashtag(
  hashtag = "rstats",
  local = FALSE,
  only_media = FALSE,
  max_id,
  since_id,
  min_id,
  limit = 20L,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`hashtag`	character, Content of a #hashtag. The hash is optional
`local`	logical, Show only local statuses?
`only_media`	logical, Show only statuses with media attached?
`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`min_id`	character, Return results immediately newer than this id
`limit`	integer, Maximum number of results to return
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`token`	user bearer token (read from file by default)

anonymous	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.
retryonratelimit	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
verbose	logical whether to display messages

Value

statuses

References

https://docs.joinmastodon.org/methods/timelines/

Examples

```
## Not run:
get_timeline_hashtag(hashtag = "#ichbinhanna")
## anonymously
get_timeline_hashtag(hashtag = "ichbinhanna", instance = "mastodon.social", anonymous = TRUE)

## End(Not run)
```

---

get_timeline_home *Get home and list timelines*

Description

Query the instance for the timeline from either followed users or a specific list. These functions can only be called with a user token from `create_token()`.

Usage

```
get_timeline_home(
  local = FALSE,
  max_id,
  since_id,
  min_id,
  limit = 20L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
```

```
  verbose = TRUE
)

get_timeline_list(
  list_id,
  max_id,
  since_id,
  min_id,
  limit = 20L,
  token = NULL,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

local	logical, Show only local statuses?
max_id	character, Return results older than this id
since_id	character, Return results newer than this id
min_id	character, Return results immediately newer than this id
limit	integer, Maximum number of results to return
token	user bearer token (read from file by default)
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.
retryonratelimit	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
verbose	logical whether to display messages
list_id	character, Local ID of the list in the database.

Value

statuses

References

https://docs.joinmastodon.org/methods/timelines/

Examples

```
## Not run:
get_timeline_home()

## End(Not run)
```

```
## Not run:
get_timeline_list("<listid>")

## End(Not run)
```

`get_timeline_public` *Get the public timeline*

Description

Query the instance for the public timeline

Usage

```
get_timeline_public(
  local = FALSE,
  remote = FALSE,
  only_media = FALSE,
  max_id,
  since_id,
  min_id,
  limit = 20L,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE,
  retryonratelimit = TRUE,
  verbose = TRUE
)
```

Arguments

`local`	logical, Show only local statuses?
`remote`	logical, Show only remote statuses?
`only_media`	logical, Show only statuses with media attached?
`max_id`	character, Return results older than this id
`since_id`	character, Return results newer than this id
`min_id`	character, Return results immediately newer than this id
`limit`	integer, Maximum number of results to return
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`token`	user bearer token (read from file by default)
`anonymous`	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.

parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.
retryonratelimit	If TRUE, and a rate limit is exhausted, will wait until it refreshes. Most Mastodon rate limits refresh every 5 minutes. If FALSE, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point.
verbose	logical whether to display messages

Value

statuses

References

https://docs.joinmastodon.org/methods/timelines/

Examples

```
## Not run:
## as tibble
get_timeline_public()
## as list
get_timeline_public(parse = FALSE)

## End(Not run)
```

---

parse_stream *Parser of Mastodon stream*

Description

Converts Mastodon stream data (JSON file) into a parsed tibble.

Usage

```
parse_stream(path)
```

Arguments

path	Character, name of JSON file with data collected by any stream_timeline function.

Details

The stream sometimes returns invalid lines of json. These are automatically skipped. Parsing can be slow if your json contains a large amount of statuses

Value

a tibble of statuses

See Also

`stream_timeline_public()`, `stream_timeline_hashtag()`,`stream_timeline_list()`

Examples

```
## Not run:
stream_timeline_public(1,file_name = "stream.json")
parse_stream("stream.json")

## End(Not run)
```

---

post_toot *Post status update to user's Mastodon account*

Description

Be aware that excessive automated posting is frowned upon (or even against the ToS) in many instances. Make sure to check the ToS of your instance and be mindful when using this function.

Usage

```
post_toot(
  status = "my first rtoot #rstats",
  media = NULL,
  alt_text = NULL,
  token = NULL,
  in_reply_to_id = NULL,
  sensitive = FALSE,
  spoiler_text = NULL,
  visibility = "public",
  scheduled_at = NULL,
  language = NULL,
  verbose = TRUE
)
```

Arguments

`status`	character, toot status. Must be 500 characters or less.
`media`	character, path to media to add to post
`alt_text`	character, a plain-text description of the media, for accessibility purposes.
`token`	user bearer token (read from file by default)
`in_reply_to_id`	character, ID of the status being replied to, if status is a reply

sensitive	logical, mark status and attached media as sensitive?
spoiler_text	character, text to be shown as a warning or subject before the actual content. Statuses are generally collapsed behind this field.
visibility	character, Visibility of the posted status. One of public (default), unlisted, private, direct.
scheduled_at	ISO 8601 Datetime at which to schedule a status. Must be at least 5 minutes in the future.
language	ISO 639 language code for this status.
verbose	logical whether to display messages

Value

no return value, called for site effects

Examples

```
## Not run:
# post a simple status
post_toot("my first rtoot #rstats")
# post a media file with alt text
post_toot("look at this pic",media = "path/to/image",alt_text = "describe image")

## End(Not run)
```

---

post_user *Perform actions on an account*

Description

Perform actions on an account

Usage

```
post_user(id, action = "follow", comment = "", token = NULL, verbose = TRUE)
```

Arguments

id	character, user id to perform the action on
action	character, one of "(un)follow","(un)block", "(un)mute", "(un)pin","note"
comment	character (if action="note"), The comment to be set on that user. Provide an empty string or leave out this parameter to clear the currently set note.
token	user bearer token (read from file by default)
verbose	logical whether to display messages

Value

no return value, called for site effects

Examples

```
## Not run:
#follow a user
post_user("xxxxxx",action = "follow")
#unfollow a user
post_user("xxxxxx",action = "unfollow")

## End(Not run)
```

`rtoot` *Query Mastodon API*

Description

This is a minimalistic interface for querying the Mastodon API. This function is for advanced users who want to query the Mastodon API for endpoints that the R functions are not yet implemented. Please also note that the API responses will not be parsed as tibble. Refer to the official API documentation for endpoints and parameters.

Usage

```
rtoot(
  endpoint,
  ...,
  params = list(),
  token = NULL,
  instance = NULL,
  anonymous = FALSE
)
```

Arguments

`endpoint`	character, a Mastodon API endpoint. Currently, only endpoints using GET are supported
`...`	Name-value pairs giving API parameters.
`params`	list, API parameters to be submitted
`token`	user bearer token (read from file by default)
`instance`	character, the server name of the instance where the status is located. If `NULL`, the same instance used to obtain the token is used.
`anonymous`	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.

Value

a list

References

https://docs.joinmastodon.org/methods/

Examples

```
## Not run:
rtoot(endpoint = "api/v1/notifications")
rtoot(endpoint = "api/v1/notifications", limit = 8)
## same
rtoot(endpoint = "api/v1/notifications", params = list(limit = 8))
rtoot(endpoint = "api/v1/followed_tags")
## reimplement `get_timeline_public`
rtoot(endpoint = "api/v1/timelines/public", instance = "emacs.ch", local = TRUE, anonymous = TRUE)

## End(Not run)
```

---

save_auth_rtoot *save a bearer token to file*

Description

save a bearer token to file

Usage

```
save_auth_rtoot(token, name = NULL, path = NULL)
```

Arguments

token	bearer token created with create_token
name	A file name (if you want to store more than one token).
path	A path where the token is stored.

search_accounts *Search the instance for a specific user*

Description

Search the instance for a specific user

Usage

```
search_accounts(
  query,
  limit = 40,
  token = NULL,
  anonymous = FALSE,
  parse = TRUE
)
```

Arguments

query	character, search string
limit	number of search results to return. Defaults to 40
token	user bearer token (read from file by default)
anonymous	some API calls do not need a token. Setting anonymous to TRUE allows to make an anonymous call if possible.
parse	logical, if TRUE, the default, returns a tibble. Use FALSE to return the "raw" list corresponding to the JSON returned from the Mastodon API.

Value

a tibble ir list of accounts

Examples

```
## Not run:
search_accounts("schochastics")

## End(Not run)
```

`stream_timeline` *Collect live streams of Mastodon data*

Description

Collect live streams of Mastodon data

Usage

```
stream_timeline_public(
  timeout = 30,
  local = FALSE,
  file_name = NULL,
  append = TRUE,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  verbose = TRUE
)

stream_timeline_hashtag(
  hashtag = "rstats",
  timeout = 30,
  local = FALSE,
  file_name = NULL,
  append = TRUE,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  verbose = TRUE
)

stream_timeline_list(
  list_id,
  timeout = 30,
  file_name = NULL,
  append = TRUE,
  instance = NULL,
  token = NULL,
  anonymous = FALSE,
  verbose = TRUE
)
```

Arguments

`timeout`	Integer, Number of seconds to stream toots for. Stream indefinitely with timeout = Inf. The stream can be interrupted at any time, and file_name will still be a valid file.

local	logical, Show only local statuses (either statuses from your instance or the one provided in instance)?
file_name	character, name of file. If not specified, will write to a temporary file stream_toots*.json.
append	logical, if TRUE will append to the end of file_name; if FALSE, will overwrite.
instance	character, the server name of the instance where the status is located. If NULL, the same instance used to obtain the token is used.
token	user bearer token (read from file by default)
anonymous	logical, should the API call be made anonymously? Defaults to TRUE but some instances might need authentication here
verbose	logical whether to display messages
hashtag	character, hashtag to stream
list_id	character, id of list to stream

Details

stream_timeline_public stream all public statuses on any instance

stream_timeline_hashtag stream all statuses containing a specific hashtag

stream_timeline_list stream the statuses of a list

Value

does not return anything. Statuses are written to file

Examples

```
## Not run:
# stream public timeline for 30 seconds
stream_timeline_public(timeout = 30,file_name = "public.json")
# stream timeline of mastodon.social  for 30 seconds
stream_timeline_public(timeout = 30, local = TRUE,
   instance = "mastodon.social", file_name = "social.json")

# stream hashtag timeline for 30 seconds
stream_timeline_hashtag("rstats", timeout = 30, file_name = "rstats_public.json")
# stream hashtag timeline of mastodon.social  for 30 seconds
stream_timeline_hashtag("rstats", timeout = 30, local = TRUE,
   instance = "fosstodon.org", file_name = "rstats_foss.json")
# json files can be parsed with parse_stream()
parse_stream("rstats_foss.json")

## End(Not run)
```

`verify_credentials` *Verify mastodon credentials*

Description

Verify mastodon credentials

Usage

```
verify_credentials(token, verbose = TRUE)

verify_envvar(verbose = TRUE)
```

Arguments

`token`	bearer token, either public or user level
`verbose`	logical whether to display messages

Details

If you have created your token as an environment variable, use `verify_envvar` to verify your token.

Value

Raise an error if the token is not valid. Return the response from the verification API invisibly otherwise.

Examples

```
## Not run:
#read a token from a file
verify_credentials(token)

## End(Not run)
```

Index