Package 'Rinstapkg'

June 8, 2019

Title An Implementation of the 'Instagram' API Using Tidy Principles

Version 0.1.0

Date 2019-06-07

Description Provides functions to use the 'Instagram' API to get feed and user information, but also performs basic in-app functionality such as liking, commenting, following, and blocking. Use of this package means that you will not use it to spam, harass, or perform other nefarious acts. For more details on how to use the API please see this package's website <https://eric88tchong.github.io/Rinstapkg/> for more information, documentation, and examples.

URL https://github.com/eric88tchong/Rinstapkg

BugReports https://github.com/eric88tchong/Rinstapkg/issues

Encoding UTF-8

Depends R (>= 3.1.0)

License MIT + file LICENSE

LazyData true

Imports methods, httr, dplyr, jsonlite, purrr, readr, lubridate, digest, uuid, rlang

Suggests knitr, testthat, rmarkdown, here

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Eric Tchong [aut, cre], Steven M. Mortimer [ctb], Jennifer Bryan [ctb, cph], Joanna Zhao [ctb, cph]

Maintainer Eric Tchong <est2fr@virginia.edu>

Repository CRAN

Date/Publication 2019-06-08 08:20:04 UTC

R topics documented:

`as_epoch` *Convert a value to Epoch Time*

Description

This function takes an input and converts it to the Unix epoch which is the number of seconds that have elapsed since January 1, 1970 at Midnight UTC.

Usage

```
as_epoch(x)
```

Arguments

x	object to be coerced

---

check_user_id *Validate a User Id*

Description

This function checks whether a supplied user_id fits the pattern of one, mostly to alert users when supplying a username instead of an id.

Usage

```
check_user_id(user_id)
```

Arguments

user_id	numeric; the unique id to identify an Instagram user which can be found in the "pk" field on a user object

Examples

```
# check with a real user id
bieber_user_id <- 6860189 # returned using ig_get_user_id("justinbieber")
check_user_id(bieber_user_id)

# this will return a warning that we need a User Id, not a Username
#check_user_id("justinbieber")
```

---

feed *Instagram Feeds*

Description

The arguments available to all functions that return feed data from Instagram

Arguments

max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?

max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

---

ig_auth *Log in to Instagram*

Description

Log in using Basic (Username-Password) or OAuth 2.0 authenticaion. OAuth does not require sharing passwords, but will require authorizing Rinstapkg as a connected app to view and manage your account. You will be directed to a web browser, asked to sign in to your Instagram account, and to grant Rinstapkg permission to operate on your behalf. By default, these user credentials are cached in a file named .httr-oauth-Rinstapkg in the current working directory.

Usage

```
ig_auth(username = NULL, password = NULL, token = NULL,
  client_id = getOption("Rinstapkg.client_id"),
  client_secret = getOption("Rinstapkg.client_secret"),
  callback_url = getOption("Rinstapkg.callback_url"),
  cache = getOption("Rinstapkg.httr_oauth_cache"), verbose = FALSE)
```

Arguments

username	Instagram username, use the handle without the "@" sign or an email
password	Instagram password
token	optional; an actual token object or the path to a valid token stored as an .rds file
client_id, client_secret, callback_url	the "Client Id","Client Secret", and "Callback URL" when using a connected app. The Rinstapkg does not have a default App so you must specify these if using the OAuth2.0 protocol
cache	logical or character; TRUE means to cache using the default cache file .httr-oauth-Rinstapkg, FALSE means don't cache. A string means use the specified path as the cache file.
verbose	logical; do you want informative messages?

Examples

```
# log in using basic authentication (username-password)
ig_auth(username = "test@gmail.com",
        password = "test_password")
```

```
# log in using OAuth 2.0
# Via brower or refresh of .httr-oauth-Rinstapkg
options(Rinstapkg.client_id = "012345678901-99thisisatest99")
options(Rinstapkg.client_secret = "Th1s1sMyC1ientS3cr3t")
ig_auth()

# Save token and log in using it
saveRDS(.state$token, "token.rds")
ig_auth(token = "token.rds")
```

`ig_comment` *Comment or Delete a Comment on a Post*

Description

This function takes the `media_id` of a post and text for a comment or the id of a comment to manipulate the comment on the post.

Usage

```
ig_comment(media_id, comment_text, verbose = FALSE)

ig_comment_delete(media_id, comment_id, verbose = FALSE)
```

Arguments

`media_id`	numeric; the unique id to identify a post which can be found in the `id`, not the `pk` field, of posts returned via many of the functions retrieving feeds.
`comment_text`	character; the text that would be posted as a comment underneath the post
`verbose`	logical; do you want informative messages?
`comment_id`	numeric; the unique id to identify a post which can be found in the `"id"` on other comments returned via the Instagram API

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
comment_result <- ig_comment(last_post_media_id,
                             comment_text = "New Comment!")
comment_media_id <- comment_result$comment$media_id
deletion_result <- ig_comment_delete(last_post_media_id,
                                     comment_media_id)
```

ig_comment_delete_bulk *Delete Comments on a Post in Bulk*

Description

This function takes the media_id of a post as well as one or more comment_ids and then deletes all of those comments

Usage

```
ig_comment_delete_bulk(media_id, comment_id, verbose = FALSE)
```

Arguments

media_id	numeric; the unique id to identify a post which can be found in the id, not the pk field, of posts returned via many of the functions retrieving feeds.
comment_id	numeric; the unique id to identify a post which can be found in the "id" on other comments returned via the Instagram API
verbose	logical; do you want informative messages?

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
comment_result1 <- ig_comment(last_post_media_id,
                              comment_text = "New Comment #1")
comment1_media_id <- comment_result1$comment$media_id
comment_result2 <- ig_comment(last_post_media_id,
                              comment_text = "New Comment #2")
comment2_media_id <- comment_result2$comment$media_id
deletion_result <- ig_comment_delete(last_post_media_id,
                                     c(comment1_media_id,
                                       comment2_media_id))
```

ig_delete_media *Delete a Post*

Description

This function can be used to delete a post.

Usage

```
ig_delete_media(media_id, verbose = FALSE)
```

Arguments

media_id	numeric; the unique id to identify a post which can be found in the id, not the pk field, of posts returned via many of the functions retrieving feeds.
verbose	logical; do you want informative messages?

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
deletion_result <- ig_delete_media(last_post_media_id)
```

---

ig_edit_media_caption *Edit a Post's Caption*

Description

This function can be used to edit the caption of a post that has already been made.

Usage

```
ig_edit_media_caption(media_id, caption_text, verbose = FALSE)
```

Arguments

media_id	numeric; the unique id to identify a post which can be found in the id, not the pk field, of posts returned via many of the functions retrieving feeds.
caption_text	character; the text below a post's image or video.
verbose	logical; do you want informative messages?

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
edit_result <- ig_edit_media_caption(last_post_media_id,
                                     caption_text = "New, Edited Caption")
```

ig_following_recent_activity

Get news feed of recent activities for accounts you follow

Description

This function returns notifications regarding the people you follow, such as what posts they've liked or that they've started following other people.

Usage

```
ig_following_recent_activity(return_df = TRUE, verbose = FALSE)
```

Arguments

return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
verbose	logical; do you want informative messages?

Examples

```
my_following_recent_activity <- ig_following_recent_activity()
```

ig_get_followers

Get Follower Info

Description

These function returns all of the followers or users that a specific user is following

Usage

```
ig_get_followers(user_id, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)

ig_get_following(user_id, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)
```

Arguments

user_id	numeric; the unique id to identify an Instagram user which can be found in the "pk" field on a user object
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
bieber_user_id <- ig_get_user_id("justinbieber")

# By default, ig_get_followers will retrieve the top 10 pages of follower data.
# This is because IG users like Justin Bieber have 100M+ followers, so it could
# take a long time to pull. If you would really like to get all users, then set
# the max_pages argument to Inf.
bieber_followers <- ig_get_followers(bieber_user_id)
bieber_following <- ig_get_following(bieber_user_id)
```

---

ig_get_hashtag_feed *Get Feed of a Hashtag*

Description

This function filters by hashtags and returns all posts that have the same hashtag string

Usage

```
ig_get_hashtag_feed(hashtag, max_id = NULL, ranked_content = TRUE,
  return_df = TRUE, paginate = TRUE, max_pages = 10,
  verbose = FALSE)
```

Arguments

hashtag	character; do not include the hashtag (pound sign) at the beginning of the string
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call

ranked_content	logical; do you want the feed content to be sorted by rank?
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
beliebers_tagged_posts <- ig_get_hashtag_feed("beliebers")
```

---

ig_get_liked_feed *Get Feed of Liked or Saved Posts*

Description

These functions return all of the posts that you have liked or saved

Usage

```
ig_get_liked_feed(max_id = NULL, return_df = TRUE, paginate = TRUE,
  max_pages = 10, verbose = FALSE)

ig_get_saved_feed(max_id = NULL, return_df = TRUE, paginate = TRUE,
  max_pages = 10, verbose = FALSE)
```

Arguments

max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
my_liked_posts <- ig_get_liked_feed()
my_saved_posts <- ig_get_saved_feed()
```

`ig_get_location_feed` *Get Feed of a Location*

Description

This function filters by location and returns all posts that have the same location

Usage

```
ig_get_location_feed(location_id, max_id = NULL, ranked_content = TRUE,
  return_df = TRUE, paginate = TRUE, max_pages = 10,
  verbose = FALSE)
```

Arguments

`location_id`	numeric; the unique id to identify a place which can be found in the `"pk"`, `"external_id"`, or `"facebook_places_id"` fields on other objects returned via the Instagram API
`max_id`	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
`ranked_content`	logical; do you want the feed content to be sorted by rank?
`return_df`	logical; do you want to return the results as a `tbl_df` with one row per entity or as a list with one element per entity?
`paginate`	logical; do you want to paginate through results or just return the first page?
`max_pages`	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
`verbose`	logical; do you want informative messages?

Details

Note that if your location is a "group" (such as a city), the feed will include media from multiple locations within that area. But if your location is a very specific place such as a specific night club, it will usually only include media from that exact location.

See Also

https://docs.social-streams.com/article/118-find-instagram-location-id

Examples

```
# location feed for Paris, France
paris_location_feed <- ig_get_location_feed(6889842)
```

`ig_get_media_comments` *Get Media Comments and Likers*

Description

These functions return the comments and user like data from a single post.

Usage

```
ig_get_media_comments(media_id, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)

ig_get_media_likers(media_id, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)
```

Arguments

`media_id`	numeric; the unique id to identify a post which can be found in the `id`, not the `pk` field, of posts returned via many of the functions retrieving feeds.
`max_id`	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
`return_df`	logical; do you want to return the results as a `tbl_df` with one row per entity or as a list with one element per entity?
`paginate`	logical; do you want to paginate through results or just return the first page?
`max_pages`	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
`verbose`	logical; do you want informative messages?

Examples

```
bieber_user_id <- ig_get_user_id("justinbieber")
bieber_feed <- ig_get_user_feed(bieber_user_id, paginate = FALSE)
most_recent_post_comments <- ig_get_media_comments(media_id = bieber_feed$id[1])
most_recent_post_likers <- ig_get_media_likers(media_id = bieber_feed$id[1])
```

ig_get_media_info *Get Media Info*

Description

This function returns the details of a single post. It contains the same information about a post that is retrieved by many of the feed functions.

Usage

```
ig_get_media_info(media_id, verbose = FALSE)
```

Arguments

media_id	numeric; the unique id to identify a post which can be found in the id, not the pk field, of posts returned via many of the functions retrieving feeds.
verbose	logical; do you want informative messages?

Examples

```
bieber_user_id <- ig_get_user_id("justinbieber")
bieber_feed <- ig_get_user_feed(bieber_user_id, paginate = FALSE)
most_recent_post_info <- ig_get_media_info(media_id = bieber_feed$id[1])
```

ig_get_popular_feed *Get Feed of Popular Posts*

Description

This function returns current most popular posts on Instagram

Usage

```
ig_get_popular_feed(max_id = NULL, ranked_content = TRUE,
  return_df = TRUE, paginate = TRUE, max_pages = 10,
  verbose = FALSE)
```

Arguments

max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
ranked_content	logical; do you want the feed content to be sorted by rank?
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
most_popular_posts <- ig_get_popular_feed()
```

---

ig_get_user_feed *Get Feed of a User's Posts*

Description

This function uses the user_id to return all posts made by that user.

Usage

```
ig_get_user_feed(user_id, max_id = NULL, min_timestamp = NULL,
  ranked_content = TRUE, return_df = TRUE, paginate = TRUE,
  max_pages = 10, verbose = FALSE)
```

Arguments

user_id	numeric; the unique id to identify an Instagram user which can be found in the "pk" field on a user object
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
min_timestamp	integer, date, or datetime; a value identifying the oldest post by date that you would want to retrieve in this function call. Dates and various datetime objects will be converted into an integer representing time in epoch (seconds since January 1st, 1970) since that is what the API requires
ranked_content	logical; do you want the feed content to be sorted by rank?
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?

paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
bieber_user_id <- ig_get_user_id("justinbieber")
bieber_feed <- ig_get_user_feed(bieber_user_id)
```

---

ig_get_user_id *Get a User Id*

Description

This function accepts a Instagram username and returns their `user_id`, which is needed for other functions

Usage

```
ig_get_user_id(username, verbose = FALSE)
```

Arguments

username	character; the username of the Instagram user. Omit the "@" symbol that is typically used when referencing another user
verbose	logical; do you want informative messages?

Examples

```
ig_get_user_id("justinbieber")
```

ig_get_user_profile *Get a User's Profile*

Description

This function returns the details of a user's profile based on the supplied username (omitting the @ symbol)

Usage

```
ig_get_user_profile(username, verbose = FALSE)
```

Arguments

username	character; the username of the Instagram user. Omit the "@" symbol that is typically used when referencing another user
verbose	logical; do you want informative messages?

Examples

```
# get Jusin Bieber's profile and to see how many followers he has
bieber_follower_cnt <- ig_get_user_profile("justinbieber")$follower_count
```

ig_get_user_tags *Get Feed of Posts for a Tagged User*

Description

This fuction returns all of the posts that the specified user was tagged in.

Usage

```
ig_get_user_tags(user_id, max_id = NULL, ranked_content = TRUE,
  return_df = TRUE, paginate = TRUE, max_pages = 10,
  verbose = FALSE)
```

Arguments

user_id	numeric; the unique id to identify an Instagram user which can be found in the "pk" field on a user object
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
ranked_content	logical; do you want the feed content to be sorted by rank?

return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
bieber_user_id <- ig_get_user_id("justinbieber")
tagged_bieber_posts <- ig_get_user_tags(bieber_user_id)
```

---

ig_like *Like or Unlike a Post*

Description

This function takes the media_id of a post and likes/unlikes that post for you.

Usage

```
ig_like(media_id, verbose = FALSE)

ig_unlike(media_id, verbose = FALSE)
```

Arguments

media_id	numeric; the unique id to identify a post which can be found in the id, not the pk field, of posts returned via many of the functions retrieving feeds.
verbose	logical; do you want informative messages?

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
liked_result <- ig_like(last_post_media_id)
unliked_result <- ig_unlike(last_post_media_id)
```

ig_my_inbox *Get direct inbox messages for your account*

Description

This function returns direct messages for your account.

Usage

```
ig_my_inbox(return_df = TRUE, verbose = FALSE)
```

Arguments

return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
verbose	logical; do you want informative messages?

Examples

```
my_inbox <- ig_my_inbox()
```

ig_my_recent_activity *Get news feed of recent activities by you*

Description

This function returns notifications regarding the actions you have recently took, such as what posts you've liked or when you've started following other people.

Usage

```
ig_my_recent_activity(return_df = TRUE, verbose = FALSE)
```

Arguments

return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
verbose	logical; do you want informative messages?

Examples

```
my_recent_activity <- ig_my_recent_activity()
```

`ig_my_timeline` *Get the authenticated user's timeline feed*

Description

This function returns data that would appear in the authenticated user's timeline feed.

Usage

```
ig_my_timeline(max_id = NULL, min_timestamp = NULL,
  ranked_content = TRUE, return_df = TRUE, paginate = TRUE,
  max_pages = 10, verbose = FALSE)
```

Arguments

`max_id`	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
`min_timestamp`	integer, date, or datetime; a value identifying the oldest post by date that you would want to retrieve in this function call. Dates and various datetime objects will be converted into an integer representing time in epoch (seconds since January 1st, 1970) since that is what the API requires
`ranked_content`	logical; do you want the feed content to be sorted by rank?
`return_df`	logical; do you want to return the results as a `tbl_df` with one row per entity or as a list with one element per entity?
`paginate`	logical; do you want to paginate through results or just return the first page?
`max_pages`	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
`verbose`	logical; do you want informative messages?

Value

`tbl_df` or `list`

Examples

```
my_timeline <- ig_my_timeline()
```

ig_save *Save or Unsave a Post*

Description

This function takes the `media_id` of a post and adds/removes that post from the Saved folder within Instagram.

Usage

```
ig_save(media_id, verbose = FALSE)

ig_unsave(media_id, verbose = FALSE)
```

Arguments

`media_id`	numeric; the unique id to identify a post which can be found in the `id`, not the `pk` field, of posts returned via many of the functions retrieving feeds.
`verbose`	logical; do you want informative messages?

Examples

```
last_post_media_id <- ig_my_timeline(paginate = FALSE)$id[1]
save_result <- ig_save(last_post_media_id)
unsave_result <- ig_unsave(last_post_media_id)
```

ig_search_tags *Search by Hashtag*

Description

This function accepts a hashtag, without the "#" symbol and returns the similar hashtags. You can then use ig_get_hashtag_feed to return all posts associated with that hashtag.

Usage

```
ig_search_tags(hashtag, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)
```

Arguments

hashtag	character; do not include the hashtag (pound sign) at the beginning of the string
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?
max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
# search for posts using the #belieber hashtag
belieber_posts <- ig_search_tags("belieber")
```

---

ig_search_users *Search by Username*

Description

This function accepts a username, without the "@" symbol and finds matching user profiles. This is helpful when you do not know a user's name exactly and need to get their user_id. If you know the username exactly then you can use ig_get_user_profile to pull the profile information without searching.

Usage

```
ig_search_users(username, max_id = NULL, return_df = TRUE,
  paginate = TRUE, max_pages = 10, verbose = FALSE)
```

Arguments

username	character; the username of the Instagram user. Omit the "@" symbol that is typically used when referencing another user
max_id	integer; the unique id identifying the oldest post that you would want to retrieve in this function call
return_df	logical; do you want to return the results as a tbl_df with one row per entity or as a list with one element per entity?
paginate	logical; do you want to paginate through results or just return the first page?

max_pages	integer; a limit to the number of pages to retrieve from paginated endpoints. Instagram feeds have the potential to paginate forever, so by default we stop after pulling 10 pages. If you would like more or less pages returned, then modify this argument.
verbose	logical; do you want informative messages?

Examples

```
# search for usernames like Justin Bieber
bieber_users <- ig_search_users("justinbieb")
```

---

ig_sync_features *Synchronise experiments*

Description

This function performs a sync of the conditions of the app you are working with. The experiment conditions must need to be set prior to logging in using ig_auth by first using `options(Rinstapkg.experiments="...")`.

Usage

```
ig_sync_features(features = getOption("Rinstapkg.experiments"),
  verbose = FALSE)
```

Arguments

features	character; a long string with commas separating each of the experimental values to be set
verbose	logical; do you want informative messages?

Note

This is mainly for backend functionality during login.

media_type_enum *Lookup Media Type Enum*

Description

This function returns the integer value of a specified type of media

Usage

```
media_type_enum(media_type = c("PHOTO", "VIDEO", "ALBUM"))
```

Arguments

media_type	character; one of three types of media PHOTO, VIDEO, or ALBUM.

Examples

```
# photo media are mapped to 1
media_type_enum("PHOTO")

# an unknown media type returns NULL
media_type_enum("FAKE_MEDIA_TYPE")
```

Rinstapkg Rinstapkg *package*

Description

An R package connecting to the Instagram API using tidy principles

Details

An implementation of the Instagram API using tidy principles. This includes functions to get feeds and users, but also perform basic in-app functionality such as liking, commenting, following, and blocking. Use of this package means that you will not use it to spam, harass, or perform other nefarious acts.

Additional material can be found in the README on GitHub

Author(s)

Maintainer: Eric Tchong <est2fr@virginia.edu>

Other contributors:

- Steven M. Mortimer <reportmort@gmail.com> [contributor]
- Jennifer Bryan <jenny@rstudio.com> [contributor, copyright holder]
- Joanna Zhao <joanna.zhao@alumni.ubc.ca> [contributor, copyright holder]

See Also

Useful links:

- https://github.com/eric88tchong/Rinstapkg
- Report bugs at https://github.com/eric88tchong/Rinstapkg/issues

Index