Package 'tuber'

May 8, 2026

Title Client for the YouTube API

Version 1.4.0

Date 2026-03-23

Language en-US

Description Get comments posted on YouTube videos, information on how many times a video has been liked, search for videos with particular content, and much more. You can also scrape captions from a few videos. To learn more about the YouTube API, see <https://developers.google.com/youtube/v3/>.

License MIT + file LICENSE

URL https://gojiplus.github.io/tuber/, https://github.com/gojiplus/tuber

BugReports https://github.com/gojiplus/tuber/issues

Depends R (>= 4.2.0)

Imports askpass, checkmate, digest, dplyr, hms, httr, httr2, jsonlite, magrittr, mime, purrr, rlang (>= 1.1.0), tibble, tidyr, tidyselect, utils

Suggests config, future, ggplot2, knitr (>= 1.11), lintr, memoise, promises, progress, rmarkdown, testthat (>= 3.0.0), xml2

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Gaurav Sood [aut, cre], Kate Lyons [ctb], John Muschelli [ctb]

Maintainer Gaurav Sood <gsood07@gmail.com>

Repository CRAN

Date/Publication 2026-03-25 04:00:03 UTC

Contents

`add_video_to_playlist` *Add Video to Playlist*

Description

Add Video to Playlist

Usage

```
add_video_to_playlist(playlist_id, video_id, position = NULL, ...)
```

Arguments

`playlist_id`	string; Required. The ID of the playlist.
`video_id`	string; Required. The ID of the video to add.
`position`	numeric; Optional. The position of the video in the playlist. If not provided, the video will be added to the end of the playlist.
`...`	Additional arguments passed to `tuber_POST_json`.

Value

Details of the added video in the playlist.

References

https://developers.google.com/youtube/v3/docs/playlistItems/insert

Examples

```
## Not run:

# Set API token via yt_oauth() first

add_video_to_playlist(playlist_id = "YourPlaylistID", video_id = "2_gLD1jarfU")

## End(Not run)
```

`analyze_channel` *Comprehensive channel analysis*

Description

Performs a complete analysis of a YouTube channel including basic info, statistics, recent videos, and performance metrics.

Usage

```
analyze_channel(
  channel_id,
  max_videos = 50,
  auth = "key",
  include_comments = FALSE,
  ...
)
```

Arguments

`channel_id`	Channel ID to analyze
`max_videos`	Maximum number of recent videos to analyze (default: 50)
`auth`	Authentication method: "token" (OAuth2) or "key" (API key)
`include_comments`	Whether to fetch comment statistics (requires more quota)
`...`	Additional arguments passed to API functions

Value

List containing comprehensive channel analysis

Examples

```
## Not run:
# Basic channel analysis
analysis <- analyze_channel("UCuAXFkgsw1L7xaCfnd5JJOw")

# Detailed analysis with comments
```

```
detailed <- analyze_channel("UCuAXFkgsw1L7xaCfnd5JJOw",
                            max_videos = 100,
                            include_comments = TRUE)

## End(Not run)
```

`analyze_trends` *Trending analysis for search terms*

Description

Analyzes trending videos and content for specific search terms or topics.

Usage

```
analyze_trends(
  search_terms,
  max_results = 50,
  time_period = "month",
  order = "viewCount",
  region_code = NULL,
  auth = "key",
  ...
)
```

Arguments

`search_terms`	Vector of search terms to analyze
`max_results`	Maximum results per search term (default: 50)
`time_period`	Time period for analysis: "week", "month", "year", "all"
`order`	Sort order: "relevance", "date", "rating", "viewCount"
`region_code`	Region code for localized trends
`auth`	Authentication method: "token" (OAuth2) or "key" (API key)
`...`	Additional arguments passed to search functions

Value

List containing trending analysis results

Examples

```
## Not run:
# Analyze trending topics
trends <- analyze_trends(c("machine learning", "AI", "data science"))

# Regional trending analysis
trends_us <- analyze_trends("music", region_code = "US", time_period = "week")

## End(Not run)
```

`bulk_video_analysis` *Bulk video performance analysis*

Description

Analyzes performance metrics for multiple videos in bulk.

Usage

```
bulk_video_analysis(
  video_ids,
  include_comments = FALSE,
  benchmark_percentiles = c(0.25, 0.5, 0.75, 0.9),
  auth = "key",
  ...
)
```

Arguments

`video_ids`	Vector of video IDs to analyze
`include_comments`	Whether to include comment analysis
`benchmark_percentiles`	Percentiles to use for performance benchmarking
`auth`	Authentication method: "token" (OAuth2) or "key" (API key)
`...`	Additional arguments passed to API functions

Value

List containing bulk video analysis

Examples

```
## Not run:
# Analyze multiple videos
video_ids <- c("dQw4w9WgXcQ", "M7FIvfx5J10", "kJQP7kiw5Fk")
analysis <- bulk_video_analysis(video_ids)

# Include comment analysis
detailed <- bulk_video_analysis(video_ids, include_comments = TRUE)

## End(Not run)
```

change_playlist_title *Change the title of a YouTube playlist.*

Description

This function updates the title of an existing YouTube playlist using the YouTube Data API.

Usage

```
change_playlist_title(playlist_id, new_title, auth = "token")
```

Arguments

playlist_id	A character string specifying the ID of the playlist you want to update.
new_title	A character string specifying the new title for the playlist.
auth	Authentication method: "token" (OAuth2) or "key" (API key)

Value

A list containing the server response after the update attempt.

Examples

```
## Not run:
change_playlist_title(playlist_id = "YourPlaylistID", new_title = "New Playlist Title")

## End(Not run)
```

compare_channels *Compare multiple channels*

Description

Compares statistics and performance metrics across multiple YouTube channels.

Usage

```
compare_channels(
  channel_ids,
  metrics = c("subscriber_count", "video_count", "view_count"),
  auth = "key",
  simplify = TRUE,
  ...
)
```

Arguments

channel_ids	Vector of channel IDs to compare
metrics	Metrics to include in comparison
auth	Authentication method: "token" (OAuth2) or "key" (API key)
simplify	Whether to return a simplified comparison table
...	Additional arguments passed to API functions

Value

List or data frame with channel comparison

Examples

```
## Not run:
# Compare two channels
channels <- c("UCuAXFkgsw1L7xaCfnd5JJOw", "UCsXVk37bltHxD1rDPwtNM8Q")
comparison <- compare_channels(channels)

# Custom metrics comparison
comparison <- compare_channels(channels,
                               metrics = c("subscriber_count", "video_count", "view_count"))

## End(Not run)
```

---

count_emojis *Count emojis in text*

Description

Counts the number of emoji characters in text.

Usage

```
count_emojis(text)
```

Arguments

text	Character vector to count emojis in

Value

Integer vector with emoji counts for each element

Examples

```
count_emojis("Hello world")
count_emojis("Hello \U0001F44B World \U0001F30D!")
count_emojis(c("No emoji", "\U0001F600\U0001F601\U0001F602"))
```

`create_playlist` *Create New Playlist*

Description

Create New Playlist

Usage

```
create_playlist(title, description = "", status = "public", ...)
```

Arguments

`title`	string; Required. The title of the playlist.
`description`	string; Optional. The description of the playlist.
`status`	string; Optional. Default: 'public'. Can be one of: 'private', 'public', or 'unlisted'.
`...`	Additional arguments passed to `tuber_POST`.

Value

The created playlist's details.

References

https://developers.google.com/youtube/v3/docs/playlists/insert

Examples

```
## Not run:

# Set API token via yt_oauth() first

create_playlist(title = "My New Playlist", description = "This is a test playlist.")

## End(Not run)
```

delete_captions *Delete a Particular Caption Track*

Description

Delete a Particular Caption Track

Usage

```
delete_captions(id = NULL, ...)
```

Arguments

id	String. Required. id of the caption track that is being retrieved
...	Additional arguments passed to tuber_DELETE.

References

https://developers.google.com/youtube/v3/docs/captions/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_captions(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

delete_channel_sections *Delete Channel Sections*

Description

Delete a Channel Section

Usage

```
delete_channel_sections(id = NULL, ...)
```

Arguments

id	Required. ID of the channel section.
...	Additional arguments passed to tuber_DELETE.

References

https://developers.google.com/youtube/v3/docs/channelSections/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_channel_sections(c(channel_id = "UCRw8bIz2wMLmfgAgWm903cA"))

## End(Not run)
```

`delete_comments` *Delete a Particular Comment*

Description

Delete a Particular Comment

Usage

```
delete_comments(id = NULL, ...)
```

Arguments

`id`	String. Required. id of the comment being retrieved
`...`	Additional arguments passed to `tuber_DELETE`.

References

https://developers.google.com/youtube/v3/docs/comments/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_comments(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

delete_playlists *Delete a Playlist*

Description

Delete a Playlist

Usage

```
delete_playlists(id = NULL, ...)
```

Arguments

id	String. Required. id of the playlist that is to be deleted
...	Additional arguments passed to tuber_DELETE.

References

https://developers.google.com/youtube/v3/docs/playlists/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_playlists(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

delete_playlist_items *Delete a Playlist Item*

Description

Delete a Playlist Item

Usage

```
delete_playlist_items(id = NULL, ...)
```

Arguments

id	String. Required. id of the playlist item that is to be deleted
...	Additional arguments passed to tuber_DELETE.

References

https://developers.google.com/youtube/v3/docs/playlistItems/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_playlist_items(id = "YourPlaylistItemID")

## End(Not run)
```

`delete_videos` *Delete a Video*

Description

Delete a Video

Usage

```
delete_videos(id = NULL, ...)
```

Arguments

`id`	String. Required. id of the video that is to be deleted
`...`	Additional arguments passed to `tuber_DELETE`.

References

https://developers.google.com/youtube/v3/docs/playlistItems/delete

Examples

```
## Not run:

# Set API token via yt_oauth() first

delete_videos(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

extended-endpoints *Extended YouTube API Endpoints*

Description

Functions for YouTube API endpoints that were not previously covered in tuber, including live streaming, thumbnails, channel sections, and modern video features.

extract_emojis *Extract emojis from text*

Description

Extracts all emoji characters from text.

Usage

```
extract_emojis(text)
```

Arguments

text	Character vector to extract emojis from

Value

List of character vectors, one per input element, containing extracted emojis. Returns empty character vector for elements without emojis.

Examples

```
extract_emojis("Hello \U0001F44B World \U0001F30D!")
extract_emojis(c("No emoji", "\U0001F600 \U0001F601 \U0001F602"))
```

`get_all_channel_video_stats` *Get statistics on all the videos in a Channel*

Description

Efficiently collects all video IDs from a channel's uploads playlist, then fetches statistics and details using batch processing for optimal API quota usage.

Usage

```
get_all_channel_video_stats(channel_id = NULL, mine = FALSE, ...)
```

Arguments

`channel_id`	Character. Id of the channel
`mine`	Boolean. TRUE if you want to fetch stats of your own channel. Default is FALSE.
`...`	Additional arguments passed to `tuber_GET`.

Value

A `data.frame` containing video metadata along with view, like, dislike and comment counts.

If the `channel_id` is mistyped or there is no information, an empty list is returned

References

https://developers.google.com/youtube/v3/docs/channels/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_all_channel_video_stats(channel_id="UCxOhDvtaoXDAB336AolWs3A")
get_all_channel_video_stats(channel_id="UCMtFAi84ehTSYSE9Xo") # Incorrect channel ID

## End(Not run)
```

get_all_comments *Get all the comments for a video including replies*

Description

Get all the comments for a video including replies

Usage

```
get_all_comments(video_id = NULL, max_results = NULL, ...)
```

Arguments

video_id	string; Required. video_id: video ID.
max_results	Integer. Maximum number of comments to return. Default is NULL which returns all comments. Set this to avoid long-running requests on popular videos.
...	Additional arguments passed to tuber_GET.

Value

a data.frame with the following columns: authorDisplayName, authorProfileImageUrl, authorChannelUrl, authorChannelId.value, videoId, textDisplay,canRate, viewerRating, likeCount, publishedAt, updatedAt,id, moderationStatus, parentId

References

https://developers.google.com/youtube/v3/docs/commentThreads/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_all_comments(video_id = "a-UQz7fqR3w")
get_all_comments(video_id = "a-UQz7fqR3w", max_results = 100)

## End(Not run)
```

get_cached_response *Get cached response if available and valid*

Description

Get cached response if available and valid

Usage

```
get_cached_response(cache_key)
```

Arguments

cache_key	Cache key

Value

Cached response or NULL if not available/expired

get_captions *Get Particular Caption Track*

Description

For getting captions from the v3 API, you must specify the id resource. Check list_caption_tracks for more information. IMPORTANT: This function requires OAuth authentication and you must own the video or have appropriate permissions to access its captions.

Usage

```
get_captions(id = NULL, lang = "en", format = "sbv", as_raw = TRUE, ...)
```

Arguments

id	String. Required. id of the caption track that is being retrieved
lang	Optional. Default is en.
format	Optional. Default is sbv.
as_raw	If FALSE the captions be converted to a character string versus a raw vector
...	Additional arguments passed to tuber_GET.

Value

String.

References

https://developers.google.com/youtube/v3/docs/captions/download

Examples

```
## Not run:

# Set API token via yt_oauth() first
# You must own the video to download captions

get_captions(id = "y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

`get_channel_info_cached` *Get channel information with caching (for static parts)*

Description

Get channel information with caching (for static parts)

Usage

```
get_channel_info_cached(
  channel_id,
  part = "snippet,brandingSettings",
  auth = "key",
  cache_ttl = 3600,
  ...
)
```

Arguments

`channel_id`	Channel ID
`part`	Parts to retrieve (only static parts will be cached)
`auth`	Authentication method
`cache_ttl`	Cache time-to-live (default: 1 hour for channel info)
`...`	Additional arguments

Value

Channel information

`get_channel_sections` *Get channel sections*

Description

Retrieves channel sections (featured channels, playlists, etc.).

Usage

```
get_channel_sections(
  channel_id = NULL,
  section_id = NULL,
  part = "snippet,contentDetails",
  simplify = TRUE,
  auth = "key",
  ...
)
```

Arguments

`channel_id`	Channel ID
`section_id`	Specific section ID (optional)
`part`	Parts to retrieve
`simplify`	Whether to return a simplified data frame
`auth`	Authentication method: "token" (OAuth2) or "key" (API key)
`...`	Additional arguments passed to tuber_GET

Value

List or data frame with channel section information

Examples

```
## Not run:
# Get all sections for a channel
sections <- get_channel_sections(channel_id = "UCuAXFkgsw1L7xaCfnd5JJOw")

# Get specific section
section <- get_channel_sections(section_id = "UC_x5XG1OV2P6uZZ5FSM9Ttw.e-Fk7vMeOn4")

## End(Not run)
```

`get_channel_stats` *Get Channel Statistics*

Description

Get statistics and details for one or more YouTube channels efficiently using batch processing.

Usage

```
get_channel_stats(
  channel_ids = NULL,
  mine = NULL,
  part = c("statistics", "snippet"),
  simplify = TRUE,
  batch_size = 50,
  show_progress = NULL,
  auth = "token",
  ...
)

list_my_channel(...)
```

Arguments

`channel_ids`	Character vector of channel IDs to retrieve. Use `list_my_channel()` to get your own channel ID.
`mine`	Logical. Set to TRUE to get authenticated user's channel. Overrides `channel_ids`. Default: NULL.
`part`	Character vector of parts to retrieve. Default: c("statistics", "snippet").
`simplify`	Logical. If TRUE (default), returns a data frame. If FALSE, returns raw list.
`batch_size`	Number of channels per API call (max 50). Default: 50.
`show_progress`	Whether to show progress for large batches. Default: TRUE for >10 channels.
`auth`	Authentication method: "token" (OAuth2) or "key" (API key). Default: "token".
`...`	Additional arguments passed to `tuber_GET`.

Details

Valid parts include: `auditDetails`, `brandingSettings`, `contentDetails`,`contentOwnerDetails`, `id`, `localizations`, `snippet`, `statistics`, `status`,`topicDetails`.

The function automatically batches requests to minimize API quota usage: - 1 channel = 1 API call - 100 channels = 2 API calls (batched in groups of 50)

Value

When `simplify = TRUE` (default): Data frame with channel details. When `simplify = FALSE`: List with channel details.

For single channels, returns the channel object directly (not in a list). For multiple channels, returns a list with items array.

References

https://developers.google.com/youtube/v3/docs/channels/list

Examples

```
## Not run:
# Get stats for a single channel - displays console output
stats <- get_channel_stats("UCT5Cx1l4IS3wHkJXNyuj4TA")

# Get stats for multiple channels - automatically batched
channel_ids <- c("UCT5Cx1l4IS3wHkJXNyuj4TA", "UCfK2eFCRQ64Gqfrpbrcj31w")
stats <- get_channel_stats(channel_ids)

# Get as data frame
df <- get_channel_stats(channel_ids, simplify = TRUE)

# Get your own channel stats
my_stats <- get_channel_stats(mine = TRUE)

# Get additional parts
detailed <- get_channel_stats(channel_ids,
                              part = c("statistics", "snippet", "brandingSettings"))

## End(Not run)
```

`get_comments` *Get Comments*

Description

Get Comments

Usage

```
get_comments(
  filter = NULL,
  part = "snippet",
  max_results = 100,
  text_format = "html",
  page_token = NULL,
```

```
  simplify = TRUE,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: comment_id: comment ID. parent_id: parent ID.
part	Comment resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., "id, snippet", "id", etc. Default: snippet.
max_results	Maximum number of items that should be returned. Integer. Optional. Can be between 20 and 100. Default is 100.
text_format	Data Type: Character. Default is "html". Only takes "html" or "plainText". Optional.
page_token	Specific page in the result set that should be returned. Optional.
simplify	Data Type: Boolean. Default is TRUE. If TRUE, the function returns a data frame. Else a list with all the information returned.
...	Additional arguments passed to tuber_GET.

Value

Nested named list. The entry items is a list of comments along with meta information. Within each of the items is an item snippet which has an item topLevelComment$snippet$textDisplay that contains the actual comment.

When filter is comment_id, and simplify is TRUE, and there is a correct comment id, it returns a data.frame with the following cols: id, authorDisplayName, authorProfileImageUrl, authorChannelUrl,value, textDisplay, canRate, viewerRating, likeCount publishedAt, updatedAt

References

https://developers.google.com/youtube/v3/docs/comments/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_comments(filter = c(comment_id = "z13dh13j5rr0wbmzq04cifrhtuypwl4hsdk"))
get_comments(filter = c(parent_id = "z13ds5yxjq3zzptyx04chlkbhx2yh3ezxtc0k"))
get_comments(filter =
c(comment_id = "z13dh13j5rr0wbmzq04cifrhtuypwl4hsdk,
            z13dh13j5rr0wbmzq04cifrhtuypwl4hsdk"))

## End(Not run)
```

get_comment_threads *Get Comments Threads*

Description

Get Comments Threads

Usage

```
get_comment_threads(
  filter = NULL,
  part = "snippet",
  text_format = "html",
  simplify = TRUE,
  max_results = 100,
  page_token = NULL,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: video_id: video ID. channel_id: channel ID. thread_id: comma-separated list of comment thread IDs threads_related_to_channel: channel ID.
part	Comment resource requested. Required. Comma separated list of one or more of the following: id, replies, snippet. e.g., "id,snippet", "replies", etc. Default: snippet.
text_format	Data Type: Character. Default is "html". Only takes "html" or "plainText". Optional.
simplify	Data Type: Boolean. Default is TRUE. If TRUE, the function returns a data frame. Else a list with all the information returned.
max_results	Maximum number of items that should be returned. Integer. Optional. Can be 1-2000. Default is 100. If the value is greater than 100, multiple API calls are made to fetch all results. Each API call is limited to 100 items per the YouTube API.
page_token	Specific page in the result set that should be returned. Optional.
...	Additional arguments passed to tuber_GET.

Value

Nested named list. The entry items is a list of comments along with meta information. Within each of the items is an item snippet which has an item topLevelComment$snippet$textDisplay that contains the actual comment.

If simplify is TRUE, a data.frame with the following columns: authorDisplayName, authorProfileImageUrl, authorChannelUrl,authorChannelId.value, videoId, textDisplay,canRate, viewerRating, likeCount, publishedAt, updatedAt

References

https://developers.google.com/youtube/v3/docs/commentThreads/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_comment_threads(filter = c(video_id = "N708P-A45D0"))
get_comment_threads(filter = c(video_id = "N708P-A45D0"), max_results = 101)

## End(Not run)
```

---

get_live_chat_messages

Get Live Chat Messages

Description

Retrieves live chat messages for a specific live chat. Note that live chat messages can only be retrieved for active live broadcasts.

Usage

```
get_live_chat_messages(
  live_chat_id,
  part = "snippet,authorDetails",
  hl = NULL,
  max_results = 500,
  page_token = NULL,
  profile_image_size = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

live_chat_id	Character. The id of the live chat.
part	Character. Parts to retrieve. Valid values are "snippet", "authorDetails". Default is "snippet,authorDetails".
hl	Character. Language used for text values. Optional.
max_results	Integer. Maximum number of items to return. Default is 500. Max is 2000.
page_token	Character. Specific page token to retrieve. Optional.
profile_image_size	Integer. Size of the profile image to return. Optional.
simplify	Logical. Whether to return a simplified data.frame. Default is TRUE.
...	Additional arguments passed to tuber_GET.

Value

A data.frame or list of live chat messages.

References

https://developers.google.com/youtube/v3/live/docs/liveChatMessages/list

Examples

```
## Not run:
# Set API token via yt_oauth() first

messages <- get_live_chat_messages(live_chat_id = "Cg0KC...")

## End(Not run)
```

---

get_live_streams *Get live stream information*

Description

Retrieves information about live streams and premieres.

Usage

```
get_live_streams(
  stream_id = NULL,
  channel_id = NULL,
  part = "snippet,status",
  status = NULL,
  simplify = TRUE,
  auth = "token",
  ...
)
```

Arguments

stream_id	Live stream ID (optional if using other filters)
channel_id	Channel ID to get live streams for
part	Parts to retrieve
status	Filter by status: "active", "upcoming", "completed"
simplify	Whether to return a simplified data frame
auth	Authentication method: "token" (OAuth2) or "key" (API key)
...	Additional arguments passed to tuber_GET

Value

List or data frame with live stream information

Examples

```
## Not run:
# Get live streams for a channel
streams <- get_live_streams(channel_id = "UCuAXFkgsw1L7xaCfnd5JJOw")

# Get specific live stream details
stream <- get_live_streams(stream_id = "abc123", part = c("snippet", "status"))

## End(Not run)
```

---

get_playlists *Get Playlists*

Description

Get Playlists

Usage

```
get_playlists(
  filter = NULL,
  part = "snippet",
  max_results = 50,
  hl = NULL,
  page_token = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: ID of the channel playlist_id: YouTube playlist ID.
part	Required. One of the following: contentDetails, id,localizations, player, snippet, status. Default: contentDetails.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 trigger additional requests and may increase API quota usage.
hl	Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
page_token	specific page in the result set that should be returned, optional
simplify	Data Type: Boolean. Default is TRUE. If TRUE and if part requested is contentDetails, the function returns a data.frame. Else a list with all the information returned.
...	Additional arguments passed to tuber_GET.

Value

playlists When `simplify` is `TRUE`, a `data.frame` with 4 columns is returned: `kind`, `etag`, `id`, `contentDetails.itemCount`

References

https://developers.google.com/youtube/v3/docs/playlists/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_playlists(filter=c(channel_id="UCMtFAi84ehTSYSE9XoHefig"))
get_playlists(filter=c(channel_id="UCMtFAi84ehTSYSE9X")) # incorrect Channel ID

# For searching playlists by keyword, use yt_search() instead:
# yt_search(term="tutorial", channel_id="UCMtFAi84ehTSYSE9XoHefig", type="playlist")

## End(Not run)
```

---

get_playlist_items *Get Playlist Items*

Description

Get Playlist Items

Usage

```
get_playlist_items(
  filter = NULL,
  part = "contentDetails",
  max_results = 50,
  video_id = NULL,
  page_token = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`filter`	string; Required. named vector of length 1 potential names of the entry in the vector: `item_id`: comma-separated list of one or more unique playlist item IDs. `playlist_id`: YouTube playlist ID.
`part`	Required. Comma separated string including one or more of the following: `contentDetails`, `id`, `snippet`, `status`. Default: `contentDetails`.

max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. If over 50, additional requests are made until the requested amount is retrieved. Larger values may increase API quota usage.
video_id	Optional. request should return only the playlist items that contain the specified video.
page_token	specific page in the result set that should be returned, optional
simplify	returns a data.frame rather than a list.
...	Additional arguments passed to tuber_GET.

Value

playlist items

References

https://developers.google.com/youtube/v3/docs/playlistItems/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_playlist_items(filter =
                       c(playlist_id = "PLrEnWoR732-CN09YykVof2lxdI3MLOZda"))
get_playlist_items(filter =
                       c(playlist_id = "PL0fOlXVeVW9QMO3GoESky4yDgQfK2SsXN"),
                       max_results = 51)

## End(Not run)
```

---

get_playlist_item_ids *Get Playlist Item IDs*

Description

Get Playlist Item IDs

Usage

```
get_playlist_item_ids(
  filter = NULL,
  part = "contentDetails",
  max_results = 50,
  video_id = NULL,
  page_token = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: item_id: comma-separated list of one or more unique playlist item IDs. playlist_id: YouTube playlist ID.
part	Required. Comma separated string including one or more of the following: contentDetails, id, snippet, status. Default: contentDetails.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 will trigger multiple requests and may increase API quota usage.
video_id	Optional. request should return only the playlist items that contain the specified video.
page_token	specific page in the result set that should be returned, optional
simplify	returns a data.frame rather than a list.
...	Additional arguments passed to tuber_GET.

Value

playlist items

References

https://developers.google.com/youtube/v3/docs/playlists/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_playlist_items(filter =
                       c(playlist_id = "PLrEnWoR732-CN09YykVof2lxdI3MLOZda"))
get_playlist_items(filter =
                       c(playlist_id = "PL0fOlXVeVW9QMO3GoESky4yDgQfK2SsXN"),
                       max_results = 51)

## End(Not run)
```

---

get_playlist_item_videoids

Get Playlist Item Video IDs

Description

Get Playlist Item Video IDs

Usage

```
get_playlist_item_videoids(
  filter = NULL,
  part = "contentDetails",
  max_results = 50,
  video_id = NULL,
  page_token = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: item_id: comma-separated list of one or more unique playlist item IDs. playlist_id: YouTube playlist ID.
part	Required. Comma separated string including one or more of the following: contentDetails, id, snippet, status. Default: contentDetails.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 trigger multiple requests and may increase API quota usage.
video_id	Optional. request should return only the playlist items that contain the specified video.
page_token	specific page in the result set that should be returned, optional
simplify	returns a data.frame rather than a list.
...	Additional arguments passed to tuber_GET.

Value

playlist items

References

https://developers.google.com/youtube/v3/docs/playlists/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_playlist_items(filter =
                     c(playlist_id = "YourPlaylistID"))
get_playlist_items(filter =
                     c(playlist_id = "YourPlaylistID"),
                     max_results = 51)

## End(Not run)
```

get_premiere_info *Get video premiere information*

Description

Checks if videos are premieres and gets premiere scheduling information.

Usage

```
get_premiere_info(video_id, simplify = TRUE, auth = "key", ...)
```

Arguments

video_id	Video ID or vector of video IDs
simplify	Whether to return simplified data frame
auth	Authentication method: "token" (OAuth2) or "key" (API key)
...	Additional arguments passed to tuber_GET

Value

List or data frame with premiere information

Examples

```
## Not run:
# Check if video is a premiere
premiere_info <- get_premiere_info("dQw4w9WgXcQ")

# Check multiple videos for premiere status
premieres <- get_premiere_info(c("video1", "video2", "video3"))

## End(Not run)
```

get_related_videos *Get Related Videos*

Description

‘r lifecycle::badge("deprecated")‘

Takes a video id and returns related videos.

Note: YouTube deprecated the ‘relatedToVideoId‘ parameter in August 2023. This function will return an error as the API endpoint no longer works. Consider using yt_search with relevant keywords instead.

Usage

```
get_related_videos(
  video_id = NULL,
  max_results = 50,
  safe_search = "none",
  ...
)
```

Arguments

video_id	Character. Required. No default.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 trigger multiple requests and may increase API quota usage.
safe_search	Character. Optional. Takes one of three values: 'moderate', 'none' (default) or 'strict'
...	Additional arguments passed to tuber_GET.

Value

data.frame with 16 columns: video_id, rel_video_id,publishedAt, channelId, title, description,thumbnails.
thumbnails.default.width, thumbnails.default.height,thumbnails.medium.url,thumbnails.medium.width,
thumbnails.medium.height, thumbnails.high.url,thumbnails.high.width,thumbnails.high.height,
channelTitle, liveBroadcastContent

References

https://developers.google.com/youtube/v3/docs/search/list

Examples

```
## Not run:

# Set API token via yt_oauth() first
# NOTE: This function no longer works due to YouTube API deprecation

get_related_videos(video_id = "yJXTXN4xrI8")

## End(Not run)
```

---

get_stats *Get statistics of a Video or Videos*

Description

Gets view count, like count, comment count and other statistics for YouTube video(s). For unlisted videos, you must use OAuth authentication with the channel owner's credentials. Automatically uses batch processing when multiple video IDs are provided for efficiency.

Usage

```
get_stats(
  video_ids = NULL,
  include_content_details = FALSE,
  batch_size = 50,
  simplify = TRUE,
  ...
)
```

Arguments

video_ids	Character vector. One or more video IDs. Required.
include_content_details	Boolean. Include contentDetails (duration, definition, etc.) in response. Default: FALSE.
batch_size	Integer. Number of videos per API call when batching (max 50). Default: 50.
simplify	Boolean. Return simplified data frame for multiple videos. Default: TRUE.
...	Additional arguments passed to tuber_GET.

Value

For single video: list with elements id, viewCount, likeCount,dislikeCount, favoriteCount, commentCount. When include_content_details = TRUE, also includes duration, definition, dimension, licensedContent, projection. For multiple videos: data frame with one row per video (if simplify=TRUE) or list of results.

References

https://developers.google.com/youtube/v3/docs/videos/list#parameters

Examples

```
## Not run:

# Set API token via yt_oauth() first

# Single video
get_stats(video_ids="N708P-A45D0")

# Multiple videos (automatic batching)
video_ids <- c("N708P-A45D0", "M7FIvfx5J10", "kJQP7kiw5Fk")
stats_df <- get_stats(video_ids)

# Include video duration and other content details:
get_stats(video_ids="N708P-A45D0", include_content_details = TRUE)

# For unlisted videos, must authenticate as channel owner:
# yt_oauth("your_client_id", "your_client_secret")
# get_stats(video_ids="your_unlisted_video_id")
```

```
## End(Not run)
```

`get_subscriptions` *Get Subscriptions*

Description

Get Subscriptions

Usage

```
get_subscriptions(
  filter = NULL,
  part = "contentDetails",
  max_results = 50,
  for_channel_id = NULL,
  order = NULL,
  page_token = NULL,
  ...
)
```

Arguments

`filter`	string; Required. named vector of length 1 potential names of the entry in the vector: `channel_id`: ID of the channel. Required. No default. `subscription_id`: YouTube subscription ID
`part`	Part of the resource requested. Required. Character. A comma separated list of one or more of the following: `contentDetails`, `id`, `snippet`, `subscriberSnippet`. e.g. "id, snippet", "id", etc. Default: `contentDetails`.
`max_results`	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 will trigger additional requests and may increase API quota usage.
`for_channel_id`	Optional. String. A comma-separated list of channel IDs. Limits response to subscriptions matching those channels.
`order`	method that will be used to sort resources in the API response. Takes one of the following: alphabetical, relevance, unread
`page_token`	Specific page in the result set that should be returned. Optional. String.
`...`	Additional arguments passed to `tuber_GET`.

Value

named list of subscriptions

References

https://developers.google.com/youtube/v3/docs/subscriptions/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

get_subscriptions(filter = c(channel_id = "UChTJTbr5kf3hYazJZO-euHg"))

## End(Not run)
```

`get_super_chat_events` *Get Super Chat Events*

Description

Retrieves Super Chat events for a channel associated with the authenticated user. This endpoint requires OAuth 2.0 authentication and the channel must be approved for Super Chat.

Usage

```
get_super_chat_events(
  part = "snippet",
  hl = NULL,
  max_results = 50,
  page_token = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`part`	Parts to retrieve. Valid values are "snippet". Default is "snippet".
`hl`	Language used for text values. Optional.
`max_results`	Maximum number of items to return. Default is 50. Max is 50.
`page_token`	Specific page token to retrieve. Optional.
`simplify`	Whether to return a simplified data.frame. Default is TRUE.
`...`	Additional arguments passed to `tuber_GET`.

Value

A data.frame or list of Super Chat events.

References

https://developers.google.com/youtube/v3/live/docs/superChatEvents/list

Examples

```
## Not run:
# Set API token via yt_oauth() first

super_chats <- get_super_chat_events()

## End(Not run)
```

`get_video_details` *Get Video Details*

Description

Get details for one or more YouTube videos efficiently using batch processing.

Usage

```
get_video_details(
  video_ids,
  part = "snippet",
  simplify = TRUE,
  batch_size = 50,
  show_progress = NULL,
  auth = "token",
  ...
)
```

Arguments

`video_ids`	Character vector of video IDs to retrieve
`part`	Character vector of parts to retrieve. See `Details` for options.
`simplify`	Logical. If TRUE, returns a data frame. If FALSE, returns raw list. Default: TRUE.
`batch_size`	Number of videos per API call (max 50). Default: 50.
`show_progress`	Whether to show progress for large batches. Default: TRUE for >10 videos.
`auth`	Authentication method: "token" (OAuth2) or "key" (API key). Default: "token".
`...`	Additional arguments passed to `tuber_GET`.

Details

Valid values for `part`: `contentDetails`, `fileDetails`, `id`,`liveStreamingDetails`, `localizations`, `paidProductPlacementDetails`, `player`,`processingDetails`, `recordingDetails`, `snippet`, `statistics`, `status`,`suggestions`, `topicDetails`.

Certain parts like `fileDetails`, `suggestions`, `processingDetails` are only available to video owners and require OAuth authentication.

The function automatically batches requests to minimize API quota usage: - 1 video = 1 API call - 100 videos = 2 API calls (batched in groups of 50)

Value

When `simplify = TRUE` (default): Data frame with video details (not available for owner-only parts). When `simplify = FALSE`: List with items containing video details.

The result includes metadata as attributes: - `api_calls_made`: Number of API calls made - `quota_used`: Estimated quota units consumed - `videos_requested`: Number of videos requested - `results_found`: Number of videos found

References

https://developers.google.com/youtube/v3/docs/videos/list

Examples

```
## Not run:
# Single video
details <- get_video_details("yJXTXN4xrI8")

# Multiple videos - automatically batched
video_ids <- c("yJXTXN4xrI8", "LDZX4ooRsWs", "kJQP7kiw5Fk")
details <- get_video_details(video_ids)

# Get as data frame
df <- get_video_details(video_ids, simplify = TRUE)

# Get specific parts
stats <- get_video_details(video_ids, part = c("statistics", "contentDetails"))

# Extract specific fields:
details <- get_video_details("yJXTXN4xrI8")
title <- details$items[[1]]$snippet$title
view_count <- details$items[[1]]$statistics$viewCount

## End(Not run)
```

`get_video_thumbnails` *Get video thumbnails information*

Description

Retrieves thumbnail URLs and metadata for videos.

Usage

```
get_video_thumbnails(video_id, size = NULL, simplify = TRUE, auth = "key", ...)
```

Arguments

video_id	Video ID or vector of video IDs
size	Thumbnail size: "default", "medium", "high", "standard", "maxres"
simplify	Whether to return a simplified data frame
auth	Authentication method: "token" (OAuth2) or "key" (API key)
...	Additional arguments passed to tuber_GET

Value

List or data frame with thumbnail information

Examples

```
## Not run:
# Get all thumbnail sizes for a video
thumbs <- get_video_thumbnails("dQw4w9WgXcQ")

# Get only high resolution thumbnails
thumbs_hd <- get_video_thumbnails("dQw4w9WgXcQ", size = "high")

# Get thumbnails for multiple videos
thumbs_batch <- get_video_thumbnails(c("dQw4w9WgXcQ", "M7FIvfx5J10"))

## End(Not run)
```

---

handle_api_error *Handle YouTube API errors with context-specific messages*

Description

Handle YouTube API errors with context-specific messages

Usage

```
handle_api_error(
  error_response,
  context_msg = "",
  video_id = NULL,
  channel_id = NULL
)
```

Arguments

error_response	The error response from the API
context_msg	Additional context for the error
video_id	Video ID if applicable for better error messages
channel_id	Channel ID if applicable for better error messages

Value

Stops execution with informative error message

`handle_network_error` *Handle network/connection errors with retry suggestions*

Description

Handle network/connection errors with retry suggestions

Usage

```
handle_network_error(error, context_msg = "")
```

Arguments

`error`	The original error
`context_msg`	Additional context for the error

Value

Stops execution with informative error message

`has_emoji` *Detect emojis in text*

Description

Checks whether text contains any emoji characters.

Usage

```
has_emoji(text)
```

Arguments

`text`	Character vector to check for emojis

Value

Logical vector indicating whether each element contains emojis

Examples

```
has_emoji("Hello world")
has_emoji("Hello world! \U0001F44B")
has_emoji(c("No emoji", "Has emoji \U0001F600", "Also none"))
```

helper-functions *Helper Functions for Common YouTube Analysis Tasks*

Description

High-level convenience functions that combine multiple API calls to provide common YouTube analytics and research functionality out of the box.

insert_channel_banner *Insert Channel Banner*

Description

Uploads a channel banner image to YouTube. The image must be a JPEG, PNG, or GIF. The maximum file size is 6MB. This returns a URL that you can then use with 'update_channel' (if implemented) or through the standard API to set the channel banner.

Usage

```
insert_channel_banner(file, channel_id = NULL, ...)
```

Arguments

file	Character. Path to the banner image file.
channel_id	Character. Optional. The channel to upload the banner for (needed if using service accounts).
...	Additional arguments passed to POST.

Value

A list containing the response from the API, including the 'url' for the banner.

References

https://developers.google.com/youtube/v3/docs/channelBanners/insert

Examples

```
## Not run:
# Set API token via yt_oauth() first

banner <- insert_channel_banner(file = "banner.jpg")
print(banner$content$url)

## End(Not run)
```

is_cacheable_endpoint *Check if endpoint should be cached*

Description

Check if endpoint should be cached

Usage

```
is_cacheable_endpoint(endpoint)
```

Arguments

endpoint	API endpoint name

Value

Logical indicating if endpoint is cacheable

is_static_query *Check if query parameters indicate static data*

Description

Check if query parameters indicate static data

Usage

```
is_static_query(endpoint, query)
```

Arguments

endpoint	API endpoint
query	Query parameters

Value

Logical indicating if this specific query is cacheable

list_abuse_report_reasons *List reasons that can be used to report abusive videos*

Description

List reasons that can be used to report abusive videos

Usage

```
list_abuse_report_reasons(part = "id, snippet", hl = "en-US", ...)
```

Arguments

part	Caption resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., "id, snippet", "id", etc. Default: snippet.
hl	Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
...	Additional arguments passed to tuber_GET.

Value

If no results, empty data.frame returned If part requested = "id, snippet" or "snippet", data.frame with 4 columns: etag, id, label, secReasons If part requested = "id", data.frame with 2 columns: etag, id

References

https://developers.google.com/youtube/v3/docs/videoAbuseReportReasons/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_abuse_report_reasons()
list_abuse_report_reasons(part="id")
list_abuse_report_reasons(part="snippet")

## End(Not run)
```

list_captions *List Captions for YouTube Video*

Description

List Captions for YouTube Video

Usage

```
list_captions(video_id, query = NULL, auth = "token", ...)
```

Arguments

video_id	ID of the YouTube video
query	Fields for ‘query‘ in ‘GET‘
auth	Authentication method: "token" (OAuth2) or "key" (API key)
...	Additional arguments to send to tuber_GET

Value

A list containing caption information

Examples

```
## Not run:
video_id <- "M7FIvfx5J10"
list_captions(video_id)

## End(Not run)
```

list_caption_tracks *List Captions of a Video*

Description

List Captions of a Video

Usage

```
list_caption_tracks(
  part = "snippet",
  video_id = NULL,
  lang = "en",
  id = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

part	Caption resource requested. Required. Comma separated list of one or more of the following: id, snippet. e.g., "id, snippet", "id" Default: snippet.
video_id	ID of the video whose captions are requested. Required. No default.
lang	Language of the caption; required; default is English ("en")
id	comma-separated list of IDs that identify the caption resources that should be retrieved; optional; string
simplify	Boolean. Default is TRUE. When TRUE, and part is snippet, a data.frame is returned
...	Additional arguments passed to tuber_GET.

Value

list of caption tracks. When simplify is TRUE, a data.frame is returned with following columns: videoId, lastUpdated, trackKind, language, name,audioTrackType, isCC,isLarge, isEasyReader, isDraft, isAutoSynced, status, id (caption id)

References

https://developers.google.com/youtube/v3/docs/captions/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_caption_tracks(video_id = "yJXTXN4xrI8")

## End(Not run)
```

---

list_channel_activities

List Channel Activity

Description

Returns a list of channel events that match the request criteria.

Usage

```
list_channel_activities(
  filter = NULL,
  part = "snippet",
  max_results = 50,
  page_token = NULL,
```

```
  published_after = NULL,
  published_before = NULL,
  region_code = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: ID of the channel. Required. No default.
part	specify which part do you want. It can only be one of the three: contentDetails, id, snippet. Default is snippet.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 will trigger additional requests and may increase API quota usage.
page_token	specific page in the result set that should be returned, optional
published_after	Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"
published_before	Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"
region_code	ISO 3166-1 alpha-2 country code, optional, see also list_regions
simplify	Data Type: Boolean. Default is TRUE. If TRUE and if part requested is contentDetails, the function returns a data.frame. Else a list with all the information returned.
...	Additional arguments passed to tuber_GET.

Value

named list If simplify is TRUE, a data.frame is returned with 18 columns: publishedAt, channelId, title, description, thumbnails.default.url,thumbnails.default.width, thumbnails.default.height,thumbr thumbnails.medium.width, thumbnails.medium.height,thumbnails.high.url, thumbnails.high.width,thumbna thumbnails.standard.url, thumbnails.standard.width,thumbnails.standard.height, channelTitle, type

References

https://developers.google.com/youtube/v3/docs/activities/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_channel_activities(filter = c(channel_id = "UCRw8bIz2wMLmfgAgWm903cA"))
list_channel_activities(filter = c(channel_id = "UCRw8bIz2wMLmfgAgWm903cA", regionCode="US"))
list_channel_activities(filter = c(channel_id = "UCMtFAi84ehTSYSE9XoHefig"),
                        published_before = "2016-02-10T00:00:00Z",
```

```
                        published_after = "2016-01-01T00:00:00Z")

## End(Not run)
```

`list_channel_members` *List Channel Members*

Description

Retrieves a list of members for a channel associated with the authenticated user. This endpoint requires OAuth 2.0 authentication and the channel must have memberships enabled.

Usage

```
list_channel_members(
  part = "snippet",
  max_results = 50,
  page_token = NULL,
  mode = "all_current",
  has_access_to_level = NULL,
  simplify = TRUE,
  ...
)
```

Arguments

`part`	Parts to retrieve. Valid values are "snippet". Default is "snippet".
`max_results`	Maximum number of items to return. Default is 50. Max is 1000.
`page_token`	Specific page token to retrieve. Optional.
`mode`	Filter for members. Valid values: "all_current", "newest". Default is "all_current".
`has_access_to_level`	Filter by a specific membership level ID. Optional.
`simplify`	Whether to return a simplified data.frame. Default is TRUE.
`...`	Additional arguments passed to `tuber_GET`.

Value

A data.frame or list of channel members.

References

https://developers.google.com/youtube/v3/docs/members/list

Examples

```
## Not run:
# Set API token via yt_oauth() first

members <- list_channel_members()

## End(Not run)
```

`list_channel_resources` *Returns List of Requested Channel Resources*

Description

Returns List of Requested Channel Resources

Usage

```
list_channel_resources(
  filter = NULL,
  part = "contentDetails",
  max_results = 50,
  page_token = NULL,
  hl = "en-US",
  simplify = TRUE,
  ...
)
```

Arguments

`filter`	string; Required. named vector with a single valid name potential names of the entry in the vector: `category_id`: YouTube guide category that returns channels associated with that category `username`: YouTube username that returns the channel associated with that username. Multiple usernames can be provided. `channel_id`: a comma-separated list of the YouTube channel ID(s) for the resource(s) that are being retrieved
`part`	a comma-separated list of channel resource properties that response will include a string. Required. One of the following: `auditDetails`, `brandingSettings`, `contentDetails`,`contentOwnerDetails`, `id`, `invideoPromotion`, `localizations`, `snippet`,`statistics`, `status`, `topicDetails`. Default is `contentDetails`.
`max_results`	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 will trigger additional requests and may increase API quota usage.
`page_token`	specific page in the result set that should be returned, optional
`hl`	Language used for text values. Optional. The default is `en-US`. For other allowed language codes, see `list_langs`.

simplify	Logical. If TRUE, returns a data frame. If FALSE, returns raw list. Default: TRUE.
...	Additional arguments passed to tuber_GET.

Value

If simplify = TRUE (default) or username is used in filter, a data frame. Otherwise returns a list.

References

https://developers.google.com/youtube/v3/docs/channels/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_channel_resources(filter = c(channel_id = "UCT5Cx1l4IS3wHkJXNyuj4TA"))
list_channel_resources(filter = c(username = "latenight"), part = "id")
list_channel_resources(filter = c(username = c("latenight", "PBS")),
                       part = "id")

## End(Not run)
```

---

list_channel_sections *List Channel Sections*

Description

Returns list of channel sections that channel id belongs to.

Usage

```
list_channel_sections(filter = NULL, part = "snippet", hl = NULL, ...)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: channel_id: Channel ID id: Section ID
part	specify which part do you want. It can only be one of the following: contentDetails, id, localizations, snippet, targeting. Default is snippet.
hl	language that will be used for text values, optional, default is en-US. See also list_langs
...	Additional arguments passed to tuber_GET.

Value

captions for the video from one of the first track

References

https://developers.google.com/youtube/v3/docs/activities/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_channel_sections(c(channel_id = "UCRw8bIz2wMLmfgAgWm903cA"))

## End(Not run)
```

---

list_channel_videos *Returns List of Requested Channel Videos*

Description

Iterate through the `max_results` number of playlists in channel and get the videos for each of the playlists.

Usage

```
list_channel_videos(
  channel_id = NULL,
  max_results = 50,
  page_token = NULL,
  hl = "en-US",
  ...
)
```

Arguments

`channel_id`	String. ID of the channel. Required.
`max_results`	Maximum number of videos returned. Integer. Default is 50. If the number is over 50, all the videos will be returned.
`page_token`	Specific page in the result set that should be returned. Optional.
`hl`	Language used for text values. Optional. Default is `en-US`. For other allowed language codes, see `list_langs`
`...`	Additional arguments passed to `tuber_GET`.

Value

list of data.frame with each list corresponding to a different playlist

References

https://developers.google.com/youtube/v3/docs/channels/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_channel_videos(channel_id = "UCXOKEdfOFxsHO_-Su3K8SHg")
list_channel_videos(channel_id = "UCXOKEdfOFxsHO_-Su3K8SHg", max_results = 10)

## End(Not run)
```

---

list_guidecats *Get list of categories that can be associated with YouTube channels*

Description

Get list of categories that can be associated with YouTube channels

Usage

```
list_guidecats(filter = NULL, hl = NULL, ...)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: region_code: Character. Required. Has to be a ISO 3166-1 alpha-2 code (see https://www.iso.org/obp/ui/#search) category_id: YouTube channel category ID
hl	Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
...	Additional arguments passed to tuber_GET.

Value

data.frame with 5 columns: region_code, channelId, title,etag, id

References

https://developers.google.com/youtube/v3/docs/guideCategories/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_guidecats(c(region_code = "JP"))

## End(Not run)
```

`list_langs` *List Languages That YouTube Currently Supports*

Description

List Languages That YouTube Currently Supports

Usage

```
list_langs(hl = NULL, ...)
```

Arguments

`hl`	Language used for text values. Optional. Default is `en-US`. For other allowed language codes, see `list_langs`.
`...`	Additional arguments passed to `tuber_GET`.

Value

data.frame with 3 columns: `hl` (two letter abbreviation), `name` (of the language), `etag`

References

https://developers.google.com/youtube/v3/docs/i18nLanguages/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_langs()

## End(Not run)
```

list_langs_cached *List supported languages with caching*

Description

List supported languages with caching

Usage

```
list_langs_cached(auth = "key", cache_ttl = 86400, ...)
```

Arguments

auth	Authentication method
cache_ttl	Cache time-to-live (default: 24 hours)
...	Additional arguments

Value

Languages data

list_my_videos *List My videos*

Description

List My videos

Usage

```
list_my_videos(...)
```

Arguments

...	additional arguments to pass to list_channel_videos

Value

data.frame with each list corresponding to a different playlist

Examples

```
## Not run:
  list_my_videos()

## End(Not run)
```

list_regions *List Content Regions That YouTube Currently Supports*

Description

List Content Regions That YouTube Currently Supports

Usage

```
list_regions(hl = NULL, ...)
```

Arguments

hl	Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
...	Additional arguments passed to tuber_GET.

Value

data.frame with 3 columns: gl (two letter abbreviation), name (of the region), etag

References

https://developers.google.com/youtube/v3/docs/i18nRegions/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_regions()

## End(Not run)
```

list_regions_cached *List supported regions with caching*

Description

List supported regions with caching

Usage

```
list_regions_cached(auth = "key", cache_ttl = 86400, ...)
```

Arguments

auth	Authentication method
cache_ttl	Cache time-to-live (default: 24 hours)
...	Additional arguments

Value

Regions data

---

list_videocats *List of Categories That Can be Associated with Videos*

Description

List of Categories That Can be Associated with Videos

Usage

```
list_videocats(filter = NULL, ...)
```

Arguments

filter	string; Required. named vector of length 1 potential names of the entry in the vector: region_code: Character. Required. Has to be a ISO 3166-1 alpha-2 code (see https://www.iso.org/obp/ui/#search) category_id: video category ID
...	Additional arguments passed to tuber_GET.

Value

data.frame with 6 columns: region_code, channelId, title, assignable, etag, id

References

https://developers.google.com/youtube/v3/docs/videoCategories/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_videocats(c(region_code = "JP"))
list_videocats() # Will throw an error asking for a valid filter with valid region_code

## End(Not run)
```

list_videocats_cached *Enhanced versions of static data functions with caching*

Description

These functions automatically cache responses to reduce API quota usage for data that changes infrequently. List video categories with caching

Usage

```
list_videocats_cached(region_code = "US", auth = "key", cache_ttl = 86400, ...)
```

Arguments

region_code	Region code for categories
auth	Authentication method
cache_ttl	Cache time-to-live (default: 24 hours for categories)
...	Additional arguments

Value

Video categories data

list_videos *List (Most Popular) Videos*

Description

List (Most Popular) Videos

Usage

```
list_videos(
  part = "contentDetails",
  max_results = 50,
  page_token = NULL,
  hl = NULL,
  region_code = NULL,
  video_category_id = NULL,
  ...
)
```

Arguments

part	Required. Comma separated string including one or more of the following: contentDetails,fileDetails, id, liveStreamingDetails, localizations, player,processingDetails, recordingDetails,snippet, statistics, status, suggestions, topicDetails. Default: contentDetails.
max_results	Maximum number of items that should be returned. Integer. Optional. Default is 50. Values over 50 will trigger multiple requests and may use additional API quota.
page_token	specific page in the result set that should be returned, optional
hl	Language used for text values. Optional. Default is en-US. For other allowed language codes, see list_langs.
region_code	Character. Required. Has to be a ISO 3166-1 alpha-2 code (see https://www.iso.org/obp/ui/#search).
video_category_id	the video category for which the chart should be retrieved. See also list_videocats.
...	Additional arguments passed to tuber_GET.

Value

data.frame with 5 columns: channelId, title, assignable, etag, id

References

https://developers.google.com/youtube/v3/docs/search/list

Examples

```
## Not run:

# Set API token via yt_oauth() first

list_videos()

## End(Not run)
```

---

post_comment *Post a Top-Level Comment*

Description

Posts a new top-level comment on a YouTube video or channel. Requires OAuth 2.0 authentication.

Usage

```
post_comment(video_id = NULL, channel_id = NULL, text, ...)
```

Arguments

video_id	Character. ID of the video to comment on. Either 'video_id' or 'channel_id' must be provided.
channel_id	Character. ID of the channel to comment on.
text	Character. The text of the comment.
...	Additional arguments passed to tuber_POST_json.

Value

A list containing the API response.

References

https://developers.google.com/youtube/v3/docs/commentThreads/insert

Examples

```
## Not run:
# Set API token via yt_oauth() first

post_comment(video_id = "yJXTXN4xrI8", text = "Great video!")

## End(Not run)
```

---

print.tuber_result *Print method for tuber results*

Description

Custom print method that shows key metadata alongside the result data

Usage

```
## S3 method for class 'tuber_result'
print(x, ...)
```

Arguments

x	A tuber_result object
...	Additional arguments passed to default print methods

quota_management *YouTube API Quota Management*

Description

Functions to track and manage YouTube API quota usage

read_sbv *Read SBV file*

Description

Read SBV file

Usage

```
read_sbv(file)
```

Arguments

file	The file name of the sbv file

Value

A data.frame with start/stop times and the text

Examples

```
if (yt_authorized()) {
vids <- list_my_videos()
res <- list_caption_tracks(video_id = vids$contentDetails.videoId[1])
cap <- get_captions(id = res$id, as_raw = FALSE)
tfile <- tempfile(fileext = ".sbv")
writeLines(cap, tfile)
x <- read_sbv(tfile)
if (requireNamespace("hms", quietly = TRUE)) {
  x$start <- hms::as_hms(x$start)
  x$stop <- hms::as_hms(x$stop)
}
}
```

remove_emojis *Remove emojis from text*

Description

Removes all emoji characters from text.

Usage

```
remove_emojis(text)
```

Arguments

text	Character vector to remove emojis from

Value

Character vector with emojis removed

Examples

```
remove_emojis("Hello \U0001F44B World!")
remove_emojis(c("No emoji", "Has \U0001F600 emoji"))
```

replace_emojis *Replace emojis in text*

Description

Replaces all emoji characters with a specified string.

Usage

```
replace_emojis(text, replacement = "")
```

Arguments

text	Character vector to process
replacement	String to replace emojis with. Default: "" (empty string)

Value

Character vector with emojis replaced

Examples

```
replace_emojis("Hello \U0001F44B World!", replacement = "[emoji]")
replace_emojis("Rate: \U0001F600\U0001F600\U0001F600", replacement = "*")
```

reply_to_comment *Reply to a Comment*

Description

Replies to an existing comment. Requires OAuth 2.0 authentication.

Usage

```
reply_to_comment(parent_id, text, ...)
```

Arguments

parent_id	Character. The ID of the comment being replied to.
text	Character. The text of the reply.
...	Additional arguments passed to tuber_POST_json.

Value

A list containing the API response.

References

https://developers.google.com/youtube/v3/docs/comments/insert

Examples

```
## Not run:
# Set API token via yt_oauth() first

reply_to_comment(parent_id = "Ugz...", text = "Thanks for watching!")

## End(Not run)
```

search_shorts *Search for shorts (YouTube Shorts)*

Description

Search specifically for YouTube Shorts videos.

Usage

```
search_shorts(
  query,
  max_results = 25,
  order = "relevance",
  region_code = NULL,
  published_after = NULL,
  published_before = NULL,
  simplify = TRUE,
  auth = "key",
  ...
)
```

Arguments

query	Search query
max_results	Maximum number of results (1-50)
order	Sort order: "date", "rating", "relevance", "title", "viewCount"
region_code	Region code for search
published_after	RFC 3339 formatted date-time (e.g., "2023-01-01T00:00:00Z")
published_before	RFC 3339 formatted date-time
simplify	Whether to return simplified data frame
auth	Authentication method: "token" (OAuth2) or "key" (API key)
...	Additional arguments passed to tuber_GET

Value

List or data frame with search results for shorts

Examples

```
## Not run:
# Search for recent shorts about cats
shorts <- search_shorts("cats", max_results = 25, order = "date")

# Search for popular shorts in a specific region
shorts_us <- search_shorts("music", region_code = "US", order = "viewCount")

## End(Not run)
```

set_comment_moderation_status *Set Comment Moderation Status*

Description

Sets the moderation status of one or more comments. Requires OAuth 2.0 authentication and owner privileges.

Usage

```
set_comment_moderation_status(
  comment_id,
  moderation_status,
  ban_author = FALSE,
  ...
)
```

Arguments

comment_id	Character vector. The IDs of the comments to update.
moderation_status	Character. Valid values are 'heldForReview', 'published', 'rejected'.
ban_author	Logical. Whether to ban the author from commenting on the channel. Optional.
...	Additional arguments passed to tuber_POST_json.

Value

A list containing the API response.

References

https://developers.google.com/youtube/v3/docs/comments/setModerationStatus

Examples

```
## Not run:
# Set API token via yt_oauth() first

set_comment_moderation_status(comment_id = "Ugz...", moderation_status = "rejected")

## End(Not run)
```

set_video_thumbnail *Set Video Thumbnail*

Description

Uploads a custom video thumbnail to YouTube and sets it for a video. Requires OAuth 2.0 authentication.

Usage

```
set_video_thumbnail(video_id, file, ...)
```

Arguments

video_id	Character. ID of the video to set the thumbnail for.
file	Character. Path to the thumbnail image file (JPG or PNG, max 2MB).
...	Additional arguments passed to POST.

Value

A list containing the response from the API.

References

https://developers.google.com/youtube/v3/docs/thumbnails/set

Examples

```
## Not run:
# Set API token via yt_oauth() first

set_video_thumbnail(video_id = "yJXTXN4xrI8", file = "thumbnail.jpg")

## End(Not run)
```

store_cached_response *Store response in cache*

Description

Store response in cache

Usage

```
store_cached_response(cache_key, data, ttl = NULL)
```

Arguments

cache_key	Cache key
data	Response data to cache
ttl	Time-to-live in seconds (NULL for default)

---

suggest_solution *Provide helpful suggestions for common user errors*

Description

Provide helpful suggestions for common user errors

Usage

```
suggest_solution(issue_type, details = "")
```

Arguments

issue_type	Type of issue encountered
details	Additional details for the suggestion

---

summary.tuber_result *Summary method for tuber results*

Description

Displays a summary of the tuber API result including metadata

Usage

```
## S3 method for class 'tuber_result'
summary(object, ...)
```

Arguments

object	A tuber_result object
...	Additional arguments (ignored)

---

tuber **tuber** *provides access to the YouTube API V3.*

Description

tuber provides access to the YouTube API V3 via RESTful calls.

tuber_cache_clear *Clear cache entries*

Description

Clear cache entries

Usage

```
tuber_cache_clear(pattern = NULL, older_than = NULL)
```

Arguments

pattern	Regular expression pattern to match cache keys (NULL for all)
older_than	Clear entries older than this many seconds

tuber_cache_config *Configure caching settings*

Description

Configure caching settings

Usage

```
tuber_cache_config(
  enabled = TRUE,
  default_ttl = 3600,
  max_size = 1000,
  cache_dir = NULL
)
```

Arguments

enabled	Whether to enable caching globally
default_ttl	Default time-to-live in seconds
max_size	Maximum number of cached items
cache_dir	Directory for persistent cache (NULL for memory only)

tuber_cache_info *Get current cache configuration*

Description

Get current cache configuration

Usage

```
tuber_cache_info()
```

Value

List with cache configuration

tuber_check *Request Response Verification*

Description

Request Response Verification

Usage

```
tuber_check(req)
```

Arguments

req	request

Value

in case of failure, a message

tuber_DELETE *DELETE*

Description

DELETE

Usage

```
tuber_DELETE(path, query, auth = "token", ...)
```

Arguments

path	path to specific API request URL
query	query list
auth	A character vector of the authentication method, either "token" (the default) or "key"
...	Additional arguments passed to DELETE.

Value

list

tuber_GET *GET*

Description

GET

Usage

```
tuber_GET(path, query, auth = "token", use_etag = TRUE, ...)
```

Arguments

path	path to specific API request URL
query	query list
auth	A character vector of the authentication method, either "token" (the default) or "key"
use_etag	Logical. Whether to use ETag for caching. Default is TRUE.
...	Additional arguments passed to GET.

Value

list

tuber_GET_cached *Cached version of tuber_GET with automatic caching*

Description

Cached version of tuber_GET with automatic caching

Usage

```
tuber_GET_cached(
  path,
  query,
  auth = "token",
  cache_ttl = NULL,
  force_refresh = FALSE,
  ...
)
```

Arguments

path	API endpoint path
query	Query parameters
auth	Authentication method
cache_ttl	Override default TTL for this call
force_refresh	Skip cache and force fresh API call
...	Additional arguments passed to tuber_GET

Value

API response (from cache or fresh call)

tuber_info *Display tuber function metadata*

Description

Shows the metadata attributes added to tuber function results for debugging and quota management.

Usage

```
tuber_info(result)
```

Arguments

result	A result object from a tuber function with metadata attributes

Examples

```
## Not run:
result <- get_video_details("dQw4w9WgXcQ")
tuber_info(result)

## End(Not run)
```

`tuber_POST` *POST*

Description

POST

Usage

```
tuber_POST(path, query, body = "", auth = "token", ...)
```

Arguments

`path`	path to specific API request URL
`query`	query list
`body`	passing image through body
`auth`	A character vector of the authentication method, either "token" (the default) or "key"
`...`	Additional arguments passed to `POST`.

Value

list

`tuber_POST_json` *POST encoded in json*

Description

POST encoded in json

Usage

```
tuber_POST_json(path, query, body = "", ...)
```

Arguments

path	path to specific API request URL
query	query list
body	passing image through body
...	Additional arguments passed to GET.

Value

list

---

tuber_PUT *PUT*

Description

PUT

Usage

```
tuber_PUT(path, query, body = "", auth = "token", ...)
```

Arguments

path	path to specific API request URL
query	query list
body	JSON body content for the PUT request
auth	A character vector of the authentication method, either "token" (the default) or "key"
...	Additional arguments passed to PUT.

Value

list

---

unicode_utils *Unicode and Text Processing Utilities*

Description

Functions for consistent text and Unicode handling across tuber, including emoji detection, extraction, and manipulation.

`update_video_metadata` *Update a YouTube Video's Metadata*

Description

This function updates the metadata of an existing YouTube video using the YouTube Data API.

Usage

```
update_video_metadata(
  video_id,
  title,
  category_id,
  description,
  privacy_status,
  made_for_kids,
  auth = "token"
)
```

Arguments

`video_id`	A character string specifying the ID of the video you want to update.
`title`	A character string specifying the new title for the video.
`category_id`	A character string specifying the new category ID for the video.
`description`	A character string specifying the new description for the video.
`privacy_status`	A character string specifying the new privacy status for the video ('public', 'private', or 'unlisted').
`made_for_kids`	A boolean specifying whether the video is self-declared as made for kids.
`auth`	Authentication method: "token" (OAuth2) or "key" (API key)

Value

A list containing the server response after the update attempt.

Examples

```
## Not run:
update_video_metadata(video_id = "YourVideoID",
                      title = "New Video Title",
                      category_id = "24",
                      description = "New Description",
                      privacy_status = "public",
                      made_for_kids = FALSE)

## End(Not run)
```

upload_caption *Upload Video Caption to Youtube*

Description

Upload Video Caption to Youtube

Usage

```
upload_caption(
  file,
  video_id,
  language = "en-US",
  caption_name,
  is_draft = FALSE,
  query = NULL,
  open_url = FALSE,
  ...
)
```

Arguments

file	Filename of the caption file with timing information (e.g., '.srt', '.vtt'). As of April 12, 2024, timing information is required for all caption uploads.
video_id	YouTube Video ID. Try list_my_videos for examples.
language	character string of 'BCP47' language type. See https://www.rfc-editor.org/rfc/bcp/bcp47.txt for language specification
caption_name	character vector of the name for the caption.
is_draft	logical indicating whether the caption track is a draft.
query	Fields for 'query' in 'POST'
open_url	Should the video be opened using browseURL
...	Additional arguments to send to POST

Value

A list of the response object from the POST, content, and the URL of the video

Note

See https://developers.google.com/youtube/v3/docs/captions#resource for full specification

Examples

```
## Not run:
xx = list_my_videos()
video_id = xx$contentDetails.videoId[1]
video_id = as.character(video_id)
language = "en-US"

## End(Not run)
```

`upload_video` *Upload Video to Youtube*

Description

Upload Video to Youtube

Usage

```
upload_video(
  file,
  snippet = NULL,
  status = list(privacyStatus = "public"),
  query = NULL,
  open_url = FALSE,
  ...
)
```

Arguments

`file`	Filename of the video locally
`snippet`	Additional fields for the video, including 'description' and 'title'. See https://developers.google.com/youtube/v3/docs/videos#resource for other fields. Coerced to a JSON object
`status`	Additional fields to be put into the `status` input. options for 'status' are 'license' (which should hold: 'creativeCommon', or 'youtube'), 'privacyStatus', 'publicStatsViewable', 'publishAt'.
`query`	Fields for 'query' in 'POST'
`open_url`	Should the video be opened using `browseURL`
`...`	Additional arguments to send to `tuber_POST` and therefore `POST`

Value

A list of the response object from the `POST`, content, and the URL of the uploaded

Note

The information for ‘status‘ and ‘snippet‘ are at https://developers.google.com/youtube/v3/docs/videos#resource but the subset of these fields to pass in are located at: https://developers.google.com/youtube/v3/docs/videos/insert The ‘part‘‘ parameter serves two purposes in this operation. It identifies the properties that the write operation will set, this will be automatically detected by the names of ‘body‘. See https://developers.google.com/youtube/v3/docs/videos/insert#usage

Examples

```
## Not run:
snippet = list(
title = "Test Video",
description = "This is just a random test.",
tags = c("r language", "r programming", "data analysis")
)
status = list(privacyStatus = "private")

## End(Not run)
```

`validate_channel_id` *Validate YouTube channel ID format*

Description

Validate YouTube channel ID format

Usage

```
validate_channel_id(channel_id, name = "channel_id")
```

Arguments

`channel_id`	Channel ID to validate
`name`	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_language_code

Validate language codes

Description

Validate language codes

Usage

```
validate_language_code(language_code, name = "language_code")
```

Arguments

language_code	Language code to validate (ISO 639-1 or BCP-47)
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_part_parameter

Validate YouTube API part parameters

Description

Validate YouTube API part parameters

Usage

```
validate_part_parameter(part, endpoint, name = "part")
```

Arguments

part	Part parameter value(s)
endpoint	API endpoint name for context-specific validation
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_playlist_id *Validate YouTube playlist ID format*

Description

Validate YouTube playlist ID format

Usage

```
validate_playlist_id(playlist_id, name = "playlist_id")
```

Arguments

playlist_id	Playlist ID to validate
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_region_code *Validate region codes*

Description

Validate region codes

Usage

```
validate_region_code(region_code, name = "region_code")
```

Arguments

region_code	Region code to validate (ISO 3166-1 alpha-2)
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_rfc3339_date *Validate RFC 3339 date format for YouTube API*

Description

Validate RFC 3339 date format for YouTube API

Usage

```
validate_rfc3339_date(date_string, name)
```

Arguments

date_string	Date string to validate
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

validate_video_id *Validate YouTube video ID format*

Description

Validate YouTube video ID format

Usage

```
validate_video_id(video_id, name = "video_id")
```

Arguments

video_id	Video ID to validate
name	Parameter name for error messages

Value

Invisible NULL if valid, stops execution if invalid

warn_deprecated *Warn about deprecated functionality with migration guidance*

Description

Warn about deprecated functionality with migration guidance

Usage

```
warn_deprecated(old_function, new_function, version = "next major version")
```

Arguments

old_function	Name of deprecated function
new_function	Name of replacement function
version	Version when deprecation will become an error

with_retry *Exponential backoff retry logic for API calls*

Description

Implements exponential backoff with jitter for retrying failed API calls

Usage

```
with_retry(
  expr,
  max_retries = 3,
  base_delay = 1,
  max_delay = 60,
  backoff_factor = 2,
  jitter = TRUE,
  retry_on = function(e) is_transient_error(e),
  on_retry = NULL
)
```

Arguments

expr	Expression to evaluate (usually an API call)
max_retries	Maximum number of retry attempts
base_delay	Base delay in seconds for first retry
max_delay	Maximum delay in seconds
backoff_factor	Multiplier for delay between retries

jitter	Whether to add random jitter to prevent thundering herd
retry_on	Function that takes an error and returns TRUE if should retry
on_retry	Function called on each retry attempt with attempt number and error

Value

Result of successful expression evaluation

---

yt_get_quota_usage *Get Current Quota Usage*

Description

Returns the current estimated quota usage for the day

Usage

```
yt_get_quota_usage()
```

Value

List with quota_used, quota_limit, quota_remaining, and reset_time

Examples

```
## Not run:
quota_status <- yt_get_quota_usage()
cat("Used:", quota_status$quota_used, "/", quota_status$quota_limit)

## End(Not run)
```

---

yt_key *Manage YouTube API key*

Description

These functions manage your YouTube API key and package key in `.Renviron`.

Usage

```
yt_get_key(decrypt = FALSE)
yt_set_key(key, type)
```

Arguments

decrypt	A boolean vector specifying whether to decrypt the supplied key with ‘httr2::secret_decrypt()‘. Defaults to ‘FALSE‘. If ‘TRUE‘, requires the environment variable ‘TUBER_KEY‘ to be set in ‘.Renviron‘.
key	A character vector specifying a YouTube API key.
type	A character vector specifying the type of API key to set. One of ’api’ (the default, stored in ‘YOUTUBE_KEY‘) or ’package’. Package keys are stored in ‘TUBER_KEY‘ and are used to decrypt API keys, for use in continuous integration and testing.

Value

‘yt_get_key()‘ returns a character vector with the YouTube API key stored in ‘.Renviron‘. If this value is not stored in ‘.Renviron‘, the functions return ‘NULL‘.

When the ‘type‘ argument is set to ’api’, ‘yt_set_key()‘ assigns a YouTube API key to ‘YOUTUBE_KEY‘ in ‘.Renviron‘ and invisibly returns ‘NULL‘. When the ‘type‘ argument is set to ’package’, ‘yt_set_key()‘ assigns a package key to ‘TUBER_KEY‘ in ‘.Renviron‘ and invisibly returns ‘NULL‘.

Examples

```
## Not run:
## for interactive use
yt_get_key()

list_channel_videos(
  channel_id = "UCDgj5-mFohWZ5irWSFMFcng",
  max_results = 3,
  part = "snippet",
  auth = "key"
)

## for continuous integration and testing
yt_set_key(httr2::secret_make_key(), type = "package")
x <- httr2::secret_encrypt("YOUR_YOUTUBE_API_KEY", "TUBER_KEY")
yt_set_key(x, type = "api")
yt_get_key(decrypt = TRUE)

list_channel_videos(
  channel_id = "UCDgj5-mFohWZ5irWSFMFcng",
  max_results = 3,
  part = "snippet",
  auth = "key"
)

## End(Not run)
```

`yt_oauth` *Set up Authorization*

Description

The function looks for `.httr-oauth` in the working directory. If it doesn't find it, it expects an application ID and a secret. If you want to remove the existing `.httr-oauth`, set remove_old_oauth to TRUE. By default, it is set to FALSE. The function launches a browser to allow you to authorize the application

Usage

```
yt_oauth(
  app_id = NULL,
  app_secret = NULL,
  scope = "ssl",
  token = ".httr-oauth",
  ...
)
```

Arguments

`app_id`	client id; required; no default
`app_secret`	client secret; required; no default
`scope`	Character. `ssl`, `basic`, `own_account_readonly`, `upload_and_manage_own_videos`, `partner`, and `partner_audit`. Required. `ssl` and `basic` are basically interchangeable. Default is `ssl`.
`token`	path to file containing the token. If a path is given, the function will first try to read from it. Default is `.httr-oauth` in the local directory. So if there is such a file, the function will first try to read from it.
`...`	Additional arguments passed to `oauth2.0_token`

Details

If a browser cannot be opened, pass `use_oob = TRUE` to `yt_oauth()` so authentication can be completed using an out-of-band code. Delete the `.httr-oauth` file in the working directory to force re-authentication.

Value

sets the google_token option and also saves `.httr_oauth` in the working directory (find out the working directory via `getwd()`)

References

https://developers.google.com/youtube/v3/docs/

https://developers.google.com/youtube/v3/guides/auth/client-side-web-apps for different scopes

Examples

```
 ## Not run:
yt_oauth(paste0("998136489867-5t3tq1g7hbovoj46dreqd6k5kd35ctjn",
                ".apps.googleusercontent.com"),
         "MbOSt6cQhhFkwETXKur-L9rN")

## End(Not run)
```

`yt_reset_quota` *Reset Quota Counter*

Description

Reset the quota counter (typically done automatically at midnight UTC)

Usage

```
yt_reset_quota()
```

`yt_search` *Search YouTube*

Description

Search for videos, channels and playlists. (By default, the function searches for videos.)

Usage

```
yt_search(
  term = NULL,
  max_results = 50,
  channel_id = NULL,
  channel_type = NULL,
  type = "video",
  event_type = NULL,
  location = NULL,
  location_radius = NULL,
  published_after = NULL,
  published_before = NULL,
```

```
  video_definition = "any",
  video_caption = "any",
  video_license = "any",
  video_syndicated = "any",
  region_code = NULL,
  relevance_language = "en",
  video_type = "any",
  simplify = TRUE,
  get_all = TRUE,
  page_token = NULL,
  max_pages = Inf,
  ...
)
```

Arguments

term	Character. Search term; required; no default For using Boolean operators, see the API documentation. Here's some of the relevant information: "Your request can also use the Boolean NOT (-) and OR (|) operators to exclude videos or to find videos that are associated with one of several search terms. For example, to search for videos matching either "boating" or "sailing", set the q parameter value to boating|sailing. Similarly, to search for videos matching either "boating" or "sailing" but not "fishing", set the q parameter value to boating|sailing -fishing"
max_results	Maximum number of items that should be returned in total. Integer. Optional. Can be between 1 and 500. Default is 50. If `get_all = TRUE`, multiple API calls are made until this many results are collected (subject to YouTube limits). Requesting a large number of results will consume more API quota. Search results are constrained to a maximum of 500 videos if type is video and we have a value of `channel_id`.
channel_id	Character. Only return search results from this channel; Optional.
channel_type	Character. Optional. Takes one of two values: `'any'`, `'show'`. Default is `'any'`
type	Character. Optional. Takes one of three values: `'video'`, `'channel'`, `'playlist'`. Default is `'video'`.
event_type	Character. Optional. Takes one of three values: `'completed'`, `'live'`, `'upcoming'`
location	Character. Optional. Latitude and Longitude within parentheses, e.g. "(37.42307,-122.08427)"
location_radius	Character. Optional. e.g. "1500m", "5km", "10000ft", "0.75mi"
published_after	Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"
published_before	Character. Optional. RFC 339 Format. For instance, "1970-01-01T00:00:00Z"
video_definition	Character. Optional. Takes one of three values: `'any'` (return all videos; Default), `'high'`, `'standard'`

video_caption Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'closedCaption', 'none'. Type must be set to video.

video_license Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'creativeCommon' (return videos with Creative Commons license), 'youtube' (return videos with standard YouTube license).

video_syndicated
Character. Optional. Takes one of two values: 'any' (return all videos; Default), 'true' (return only syndicated videos)

region_code Character. Required. Has to be a ISO 3166-1 alpha-2 code (see https://www.iso.org/obp/ui/#search).

relevance_language
Character. Optional. The relevance_language argument instructs the API to return search results that are most relevant to the specified language. The parameter value is typically an ISO 639-1 two-letter language code. However, you should use the values zh-Hans for simplified Chinese and zh-Hant for traditional Chinese. Please note that results in other languages will still be returned if they are highly relevant to the search query term.

video_type Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'episode' (return episode of shows), 'movie' (return movies)

simplify Boolean. Return a data.frame if TRUE. Default is TRUE. If TRUE, it returns a list that carries additional information.

get_all get all results, iterating through all the results pages. Default is TRUE. Result is a data.frame. Optional.

page_token specific page in the result set that should be returned, optional

max_pages Maximum number of pages to retrieve when get_all is TRUE. Default is Inf (no page limit). Setting a lower value can reduce API quota usage.

... Additional arguments passed to tuber_GET.

Value

data.frame with 16 elements: video_id, publishedAt,channelId, title, description,thumbnails.default.url, thumbnails.default.width, thumbnails.default.height,thumbnails.medium.url,thumbnails.medium.width, thumbnails.medium.height, thumbnails.high.url,thumbnails.high.width,thumbnails.high.height, channelTitle, liveBroadcastContent The returned data.frame also has the following attributes: total_results: The total number of results reported by the API actual_results: The actual number of rows returned api_limit_reached: Whether the YouTube API result limit was reached

References

https://developers.google.com/youtube/v3/docs/search/list

Examples

```
## Not run:

# Set API token via yt_oauth() first
```

```
yt_search(term = "Barack Obama")
yt_search(term = "Barack Obama", published_after = "2016-10-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-09-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-03-01T00:00:00Z",
                                 published_after = "2016-02-01T00:00:00Z")
yt_search(term = "Barack Obama", published_before = "2016-02-10T00:00:00Z",
                                 published_after = "2016-01-01T00:00:00Z")

# To check how many results were found vs. how many were returned:
results <- yt_search(term = "drone videos")
attr(results, "total_results")  # Total number reported by YouTube
attr(results, "actual_results") # Number actually returned
attr(results, "api_limit_reached") # Whether API limit was reached

## End(Not run)
```

`yt_set_quota_limit` *Set Quota Limit*

Description

Set the daily quota limit (default is 10,000 units)

Usage

```
yt_set_quota_limit(limit)
```

Arguments

`limit`	Integer. Daily quota limit in units

Examples

```
## Not run:
# If you have a higher quota limit
yt_set_quota_limit(50000)

## End(Not run)
```

yt_token *Check if authentication token is in options*

Description

Check if authentication token is in options

Usage

```
yt_token()

yt_authorized()

yt_check_token()
```

Value

A Token2.0 class

yt_topic_search *Search YouTube by Topic It uses the Freebase list of topics*

Description

Search YouTube by Topic It uses the Freebase list of topics

Usage

```
yt_topic_search(topic = NULL, ...)
```

Arguments

topic	topic being searched for; required; no default
...	Additional arguments passed to tuber_GET.

Value

a list

Examples

```
 ## Not run:

# Set API token via yt_oauth() first

yt_topic_search(topic = "Barack Obama")

## End(Not run)
```

`[.tuber_result` *Subset method for tuber results*

Description

Preserves tuber metadata attributes when subsetting

Usage

```
## S3 method for class 'tuber_result'
x[...]
```

Arguments

`x`	A tuber_result object
`...`	Arguments passed to the underlying subset method

Index