Package 'tuber'

November 5, 2026

Title Client for the YouTube Data API

Version 3.0.0

Date 2026-11-04

Language en-US

Description Search public YouTube data and retrieve channels, videos, playlists, comments, captions, live broadcasts, and reference data. Authenticated methods support common uploads, playlist changes, comment moderation, and media updates. See the YouTube Data API documentation at <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, httr2 (>= 1.2.0), jsonlite, mime, rlang (>= 1.1.0), stringi, tibble, tidyr, tidyselect, utils

Suggests config, future, ggplot2, hms, knitr (>= 1.11), memoise, progress, promises, rmarkdown, spelling, testthat (>= 3.1.7), withr, xml2

VignetteBuilder knitr

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Maintainer Gaurav Sood <gsood07@gmail.com>

Repository CRAN

Date/Publication 2026-11-05 06:00:02 UTC

Contents

add_video_to_playlist . . . 3
analyze_channel . . . 4
analyze_trends . . . 5
bulk_video_analysis . . . 6
change_playlist_title . . . 7
compare_channels . . . 8
count_emojis . . . 9
create_playlist . . . 9
delete_caption . . . 10
delete_channel_section . . . 11
delete_comment . . . 11
delete_playlist . . . 12
delete_playlist_item . . . 13
delete_video . . . 13
download_caption . . . 14
extended-endpoints . . . 15
extract_emojis . . . 15
get_all_channel_video_stats . . . 16
get_all_comments . . . 17
get_channel_details . . . 17
get_playlist_item_ids . . . 19
get_playlist_video_ids . . . 20
get_video_broadcast_timing . . . 20
get_video_details . . . 21
get_video_stats . . . 23
get_video_thumbnails . . . 24
has_emoji . . . 24
helper-functions . . . 25
insert_channel_banner . . . 25
list_abuse_report_reasons . . . 26
list_captions . . . 27
list_channel_activities . . . 28
list_channel_members . . . 29
list_channel_sections . . . 30
list_channel_videos . . . 31
list_comments . . . 32
list_comment_threads . . . 33
list_languages . . . 34
list_live_broadcasts . . . 35
list_live_chat_messages . . . 36
list_my_videos . . . 37
list_playlists . . . 38
list_playlist_items . . . 39
list_popular_videos . . . 40
list_regions . . . 41
list_subscriptions . . . 42

list_super_chat_events . . . 43
list_video_categories . . . 44
post_comment . . . 45
quota_management . . . 46
read_sbv . . . 46
remove_emojis . . . 47
replace_emojis . . . 47
reply_to_comment . . . 48
set_comment_moderation_status . . . 48
set_video_thumbnail . . . 49
tuber . . . 50
tuber_cache_clear . . . 50
tuber_cache_config . . . 51
tuber_cache_info . . . 51
tuber_info . . . 52
update_video_metadata . . . 52
upload_caption . . . 54
upload_video . . . 55
yt_get_quota_usage . . . 57
yt_key . . . 57
yt_oauth . . . 58
yt_reset_quota . . . 59
yt_search . . . 60
yt_set_quota_limit . . . 63
yt_token . . . 63

Index **65**

`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.

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)

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)

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)

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 a Playlist Title*

Description

Changes a playlist title without deleting the existing description or default language. YouTube replaces every mutable property in the ‘snippet‘ part during an update, so this function retrieves and preserves the other snippet properties first.

Usage

```
change_playlist_title(
  playlist_id,
  new_title,
  on_behalf_of_content_owner = NULL
)
```

Arguments

`playlist_id`	YouTube playlist ID.
`new_title`	New playlist title.
`on_behalf_of_content_owner`	Optional YouTube content-owner ID. This is only available to authorized YouTube content partners.

Value

The updated playlist resource.

References

<https://developers.google.com/youtube/v3/docs/playlists/update>

Examples

```
## Not run:
change_playlist_title("PLAYLIST_ID", "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

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'.

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_caption *Delete a Caption Track*

Description

Delete a Caption Track

Usage

```
delete_caption(caption_id)
```

Arguments

caption_id	Caption-track ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_caption("y3ElXcEME3lSISz6izkWVT5GvxjPu8pA")

## End(Not run)
```

delete_channel_section *Delete a Channel Section*

Description

Delete a Channel Section

Usage

```
delete_channel_section(section_id)
```

Arguments

section_id	Channel-section ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_channel_section("SECTION_ID")

## End(Not run)
```

delete_comment *Delete a Comment*

Description

Delete a Comment

Usage

```
delete_comment(comment_id)
```

Arguments

comment_id	Comment ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_comment("COMMENT_ID")

## End(Not run)
```

---

delete_playlist *Delete a Playlist*

Description

Delete a Playlist

Usage

```
delete_playlist(playlist_id)
```

Arguments

playlist_id	Playlist ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_playlist("PLAYLIST_ID")

## End(Not run)
```

delete_playlist_item *Delete a Playlist Item*

Description

Delete a Playlist Item

Usage

```
delete_playlist_item(playlist_item_id)
```

Arguments

playlist_item_id	Playlist-item ID, not the video ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_playlist_item("PLAYLIST_ITEM_ID")

## End(Not run)
```

delete_video *Delete a Video*

Description

Delete a Video

Usage

```
delete_video(video_id)
```

Arguments

video_id	Video ID.

Value

The empty API response, invisibly.

References

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

Examples

```
## Not run:
delete_video("VIDEO_ID")

## End(Not run)
```

`download_caption` *Download a Caption Track*

Description

Downloads one caption track in its original format and language unless a translation language or output format is requested. YouTube requires OAuth authorization and permission to access the video's captions.

Usage

```
download_caption(caption_id, language = NULL, format = NULL, as_raw = TRUE)
```

Arguments

`caption_id`	Caption-track ID returned by [list_captions()].
`language`	Optional translation language code.
`format`	Optional output format: ‘"sbv"‘, ‘"scc"‘, ‘"srt"‘, ‘"ttml"‘, or ‘"vtt"‘.
`as_raw`	If ‘TRUE‘, return a raw vector; otherwise return one character string.

Value

A raw vector when ‘as_raw = TRUE‘; otherwise a character scalar.

References

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

Examples

```
## Not run:
download_caption("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,
  auth = if (mine) "token" else "key"
)
```

Arguments

`channel_id`	Character. Id of the channel
`mine`	Boolean. TRUE if you want to fetch stats of your own channel. Default is FALSE.
`auth`	Authentication method, '"key"' or '"token"'. 'mine = TRUE' requires OAuth.

Value

A data frame with one row per video and the columns 'video_id', 'title', 'published_at', 'description', 'channel_id', 'channel_title', 'view_count', 'like_count', 'comment_count' (numeric) and 'url'. A channel with no uploads returns the same columns and zero rows. An unknown 'channel_id' is an error.

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")

## End(Not run)
```

get_all_comments *Get All Video Comments, Including Replies*

Description

Retrieves top-level comments and, when a thread response contains only a reply preview, follows ‘comments.list‘ pagination to retrieve the remaining replies.

Usage

```
get_all_comments(video_id, max_results = NULL, auth = "key")
```

Arguments

video_id	Video ID.
max_results	Optional maximum total number of comments and replies. ‘NULL‘ retrieves all available comments.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame with one row per top-level comment or reply and snake-case column names.

References

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

Examples

```
## Not run:
get_all_comments("a-UQz7fqR3w", max_results = 100)

## End(Not run)
```

get_channel_details *Get Channel Details*

Description

Retrieves channels by ID, legacy username, or the authenticated user’s channel. Supply exactly one filter.

Usage

```
get_channel_details(
  channel_ids = NULL,
  usernames = NULL,
  mine = FALSE,
  part = c("snippet", "statistics"),
  language = NULL,
  simplify = TRUE,
  batch_size = 50,
  auth = if (mine) "token" else "key"
)

get_my_channel(
  part = c("snippet", "statistics"),
  language = NULL,
  simplify = TRUE
)
```

Arguments

channel_ids	Optional character vector of channel IDs.
usernames	Optional character vector of legacy YouTube usernames.
mine	If 'TRUE', retrieve the authenticated user's channel.
part	Character vector of channel resource parts.
language	Optional language code for localized text.
simplify	If 'TRUE', return a data frame; otherwise return a channels list response with all collected items.
batch_size	Number of channel IDs per request, at most 50.
auth	Authentication method, '"key"' or '"token"'. 'mine = TRUE' requires OAuth.

Value

A data frame when 'simplify = TRUE'; otherwise a channels-list response.

References

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

Examples

```
## Not run:
get_channel_details(channel_ids = "UCT5Cx1l4IS3wHkJXNyuj4TA")
get_channel_details(usernames = c("GoogleDevelopers", "PBS"))
get_my_channel()

## End(Not run)
```

`get_playlist_item_ids` *Get Playlist-Item IDs*

Description

Get Playlist-Item IDs

Usage

```
get_playlist_item_ids(
  playlist_id = NULL,
  playlist_item_ids = NULL,
  video_id = NULL,
  max_results = 50,
  page_token = NULL,
  auth = "key"
)
```

Arguments

`playlist_id`	Optional playlist ID.
`playlist_item_ids`	Optional character vector of playlist-item IDs. Supply exactly one of ‘playlist_id‘ or ‘playlist_item_ids‘.
`video_id`	Optional video ID used to filter items in ‘playlist_id‘.
`max_results`	Maximum total number of items to return.
`page_token`	Optional page token at which to start.
`auth`	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A character vector of playlist-item IDs.

Examples

```
## Not run:
get_playlist_item_ids(playlist_id = "PLrEnWoR732-CN09YykVof2lxdI3MLOZda")

## End(Not run)
```

get_playlist_video_ids

Get Video IDs from a Playlist

Description

Get Video IDs from a Playlist

Usage

```
get_playlist_video_ids(
  playlist_id,
  max_results = 50,
  page_token = NULL,
  auth = "key"
)
```

Arguments

playlist_id	Playlist ID.
max_results	Maximum total number of video IDs to return.
page_token	Optional page token at which to start.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A character vector of video IDs.

Examples

```
## Not run:
get_playlist_video_ids("PLrEnWoR732-CN09YykVof2lxdI3MLOZda")

## End(Not run)
```

get_video_broadcast_timing

Get video live-broadcast timing

Description

Retrieves scheduling and actual start/end information exposed in a video’s ‘liveStreamingDetails‘. The YouTube Data API does not expose a reliable flag that distinguishes premieres from other scheduled broadcasts.

Usage

```
get_video_broadcast_timing(video_ids, simplify = TRUE, auth = "key")
```

Arguments

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

Value

A data frame when 'simplify = TRUE'; otherwise the raw videos-list response.

Examples

```
## Not run:
timing <- get_video_broadcast_timing("dQw4w9WgXcQ")

timings <- get_video_broadcast_timing(c("video1", "video2", "video3"))

## 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 = "key"
)
```

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, ‘"key"‘ (the default) or ‘"token"‘.

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): a data frame whose columns mirror the requested API parts. Since ‘part‘ is user-selectable, these columns retain YouTube’s field names rather than the fixed snake-case schemas used by ‘list_*()‘ functions. Owner-only parts cannot be simplified. 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"))

# Preserve the nested API resource when that is easier to inspect:
details <- get_video_details("yJXTXN4xrI8", simplify = FALSE)
title <- details$items[[1]]$snippet$title

## End(Not run)
```

`get_video_stats` *Get Video Statistics*

Description

Retrieves statistics for one or more videos and always returns one row per video found.

Usage

```
get_video_stats(
  video_ids,
  include_content_details = FALSE,
  batch_size = 50,
  auth = "key"
)
```

Arguments

`video_ids`	Character vector of YouTube video IDs.
`include_content_details`	Include duration, definition, dimension, licensed-content, and projection fields.
`batch_size`	Number of video IDs per API request, up to 50.
`auth`	Authentication method, ‘"token"‘ or ‘"key"‘.

Value

A data frame with snake-case column names and one row per video. Count columns are numeric and missing statistics are returned as ‘NA‘.

References

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

Examples

```
## Not run:
get_video_stats("N708P-A45D0")
get_video_stats(c("N708P-A45D0", "M7FIvfx5J10"), auth = "key")

## End(Not run)
```

get_video_thumbnails *Get video thumbnails information*

Description

Retrieves thumbnail URLs and metadata for videos.

Usage

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

Arguments

video_ids	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)

Value

A tidy data frame with one row per video and thumbnail size when ‘simplify = TRUE‘; otherwise the raw videos-list response.

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)
```

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 or PNG. The maximum file size is 6 MB. 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, on_behalf_of_content_owner = NULL)
```

Arguments

file	Character. Path to the banner image file.
on_behalf_of_content_owner	Optional YouTube content-owner ID. This is only available to authorized YouTube content partners.

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)
```

---

list_abuse_report_reasons

List Video-Abuse Report Reasons

Description

List Video-Abuse Report Reasons

Usage

```
list_abuse_report_reasons(
  part = c("id", "snippet"),
  language = "en-US",
  auth = "key"
)
```

Arguments

part	Character vector of resource parts.
language	Language code for localized labels.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame with one row per primary report reason.

References

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

Examples

```
## Not run:
list_abuse_report_reasons()

## End(Not run)
```

`list_captions` *List Caption Tracks*

Description

Lists caption-track metadata for a video. The caption text itself is returned by [download_caption()]. YouTube requires OAuth authorization for this method.

Usage

```
list_captions(video_id, caption_ids = NULL, part = "snippet", simplify = TRUE)
```

Arguments

`video_id`	YouTube video ID.
`caption_ids`	Optional character vector of caption-track IDs to select.
`part`	Character vector of caption resource parts.
`simplify`	If 'TRUE', return a data frame; otherwise return the raw API response.

Value

A data frame when 'simplify = TRUE'; otherwise a caption-list response.

References

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

Examples

```
## Not run:
list_captions(video_id = "M7FIvfx5J10")

## End(Not run)
```

`list_channel_activities` *List Channel Activities*

Description

List Channel Activities

Usage

```
list_channel_activities(
  channel_id,
  part = c("snippet", "contentDetails"),
  max_results = 50,
  page_token = NULL,
  published_after = NULL,
  published_before = NULL,
  region_code = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

`channel_id`	Channel ID.
`part`	Character vector of activity resource parts.
`max_results`	Maximum total number of activities to return.
`page_token`	Optional page token at which to start.
`published_after`	Optional RFC 3339 lower timestamp bound.
`published_before`	Optional RFC 3339 upper timestamp bound.
`region_code`	Optional ISO 3166-1 alpha-2 content-region code.
`simplify`	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
`auth`	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise an activities-list response.

References

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

Examples

```
## Not run:
list_channel_activities("UCRw8bIz2wMLmfgAgWm903cA", max_results = 100)

## 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,
  filter_by_member_channel_ids = NULL,
  simplify = TRUE
)
```

Arguments

`part`	Parts to retrieve. Valid values are "snippet". Default is "snippet".
`max_results`	Maximum total number of members to return.
`page_token`	Specific page token to retrieve. Optional.
`mode`	Member stream, ‘"all_current"‘ or ‘"updates"‘.
`has_access_to_level`	Filter by a specific membership level ID. Optional.
`filter_by_member_channel_ids`	Optional member channel IDs whose membership status should be checked. YouTube accepts at most 100 per call.
`simplify`	Whether to return a simplified data.frame. Default is TRUE.

Value

A data.frame or list of channel members.

References

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

Examples

```
## Not run:
yt_oauth("YOUR_CLIENT_ID", "YOUR_CLIENT_SECRET", scope = "channel_memberships")
members <- list_channel_members()

## End(Not run)
```

`list_channel_sections` *List Channel Sections*

Description

Returns channel sections matching exactly one supported YouTube filter.

Usage

```
list_channel_sections(
  channel_id = NULL,
  section_ids = NULL,
  mine = FALSE,
  part = c("snippet", "contentDetails"),
  simplify = TRUE,
  auth = if (mine) "token" else "key"
)
```

Arguments

`channel_id`	Channel whose sections should be returned.
`section_ids`	One or more channel-section IDs.
`mine`	Set to ‘TRUE‘ to return sections for the authenticated user’s channel.
`part`	Character vector of resource parts to return.
`simplify`	If ‘TRUE‘, return a data frame; otherwise return the raw API response.
`auth`	Authentication method, ‘"token"‘ or ‘"key"‘. ‘mine = TRUE‘ requires ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise a channel-section list response.

References

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

Examples

```
## Not run:
list_channel_sections(channel_id = "UCRw8bIz2wMLmfgAgWm903cA")
list_channel_sections(mine = TRUE, auth = "token")

## End(Not run)
```

`list_channel_videos` *Returns List of Requested Channel Videos*

Description

Retrieves items from a channel's uploads playlist.

Usage

```
list_channel_videos(
  channel_id,
  max_results = 50,
  page_token = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

`channel_id`	String. ID of the channel. Required.
`max_results`	Maximum total number of videos returned.
`page_token`	Specific page in the result set that should be returned. Optional.
`simplify`	If 'TRUE', return a data frame; otherwise return the raw playlist-items response.
`auth`	Authentication method, '"key"' or '"token"'.

Value

A data frame when 'simplify = TRUE'; otherwise a playlist-items response.

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_comments` *List Comments or Replies*

Description

Retrieves specific comments by ID or replies to one parent comment.

Usage

```
list_comments(
  comment_ids = NULL,
  parent_id = NULL,
  part = c("id", "snippet"),
  max_results = 100,
  text_format = "html",
  page_token = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

`comment_ids`	Optional character vector of comment IDs.
`parent_id`	Optional parent-comment ID. Supply exactly one of ‘comment_ids‘ or ‘parent_id‘.
`part`	Character vector of comment resource parts.
`max_results`	Maximum total number of replies to return. Ignored when ‘comment_ids‘ is supplied.
`text_format`	Comment text format, ‘"html"‘ or ‘"plainText"‘.
`page_token`	Optional page token at which to start.
`simplify`	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
`auth`	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise a comments-list response.

References

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

Examples

```
## Not run:
list_comments(comment_ids = "COMMENT_ID")
list_comments(parent_id = "PARENT_COMMENT_ID", max_results = 200)

## End(Not run)
```

---

list_comment_threads *List Comment Threads*

Description

Retrieves comment threads using one explicit YouTube filter.

Usage

```
list_comment_threads(
  video_id = NULL,
  channel_id = NULL,
  thread_ids = NULL,
  all_threads_for_channel_id = NULL,
  part = c("id", "snippet"),
  text_format = "html",
  max_results = 100,
  page_token = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

video_id	Optional video ID.
channel_id	Optional channel ID.
thread_ids	Optional character vector of comment-thread IDs.
all_threads_for_channel_id	Optional channel ID for all threads related to that channel.
part	Character vector of comment-thread resource parts.
text_format	Comment text format, ‘"html"‘ or ‘"plainText"‘.

max_results	Maximum total number of threads to return.
page_token	Optional page token at which to start.
simplify	If ‘TRUE‘, return one row per top-level comment; otherwise return the raw API response with all collected items.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise a comment-threads response.

References

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

Examples

```
## Not run:
list_comment_threads(video_id = "N708P-A45D0", max_results = 200)

## End(Not run)
```

---

list_languages *List Supported Languages*

Description

List Supported Languages

Usage

```
list_languages(language = NULL, auth = "key")
```

Arguments

language	Optional language code for localized names.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame with ‘language_code‘, ‘name‘, and ‘etag‘ columns.

References

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

Examples

```
## Not run:
list_languages()

## End(Not run)
```

`list_live_broadcasts` *List live broadcasts*

Description

Retrieves YouTube ‘liveBroadcast‘ resources owned by the authenticated user.

Usage

```
list_live_broadcasts(
  broadcast_ids = NULL,
  part = "snippet,status",
  status = NULL,
  mine = FALSE,
  broadcast_type = NULL,
  max_results = 50,
  page_token = NULL,
  simplify = TRUE
)
```

Arguments

`broadcast_ids`	Broadcast IDs. Supply exactly one of ‘broadcast_ids‘, ‘status‘, or ‘mine = TRUE‘.
`part`	Parts to retrieve
`status`	Filter by status: ‘"active"‘, ‘"all"‘, ‘"upcoming"‘, or ‘"completed"‘.
`mine`	Logical. List the authenticated user’s own broadcasts.
`broadcast_type`	Optional broadcast type: ‘"all"‘, ‘"event"‘, or ‘"persistent"‘.
`max_results`	Maximum number of broadcasts to return.
`page_token`	Page token at which to start.
`simplify`	Whether to return a simplified data frame

Value

List or data frame with live stream information

Examples

```
## Not run:
broadcasts <- list_live_broadcasts(status = "active")

broadcast <- list_live_broadcasts(
  broadcast_ids = "abc123",
  part = c("snippet", "status")
)

## End(Not run)
```

`list_live_chat_messages` *List 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

```
list_live_chat_messages(
  live_chat_id,
  part = "snippet,authorDetails",
  language = 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".
`language`	Optional language code for localized text.
`max_results`	Maximum total number of messages to return. YouTube serves live chat in pages of at least 200 messages, so when fewer than 200 are still wanted the surplus on the last page is dropped and the returned ‘nextPageToken‘ continues after that page, not after the last message.
`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.

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 <- list_live_chat_messages(live_chat_id = "Cg0KC...")

## End(Not run)
```

---

list_my_videos *List My Videos*

Description

Lists videos in the authenticated channel's uploads playlist.

Usage

```
list_my_videos(max_results = 50, page_token = NULL, simplify = TRUE)
```

Arguments

max_results	Maximum total number of videos to return.
page_token	Optional page token at which to start.
simplify	If 'TRUE', return a data frame; otherwise return the raw playlist-items response.

Value

A data frame when 'simplify = TRUE'; otherwise a playlist-items response.

Examples

```
## Not run:
list_my_videos(max_results = 100)

## End(Not run)
```

`list_playlists` *List Playlists*

Description

Retrieves playlists by channel, playlist ID, or authenticated ownership.

Usage

```
list_playlists(
  channel_id = NULL,
  playlist_ids = NULL,
  mine = FALSE,
  part = c("snippet", "contentDetails", "status"),
  max_results = 50,
  language = NULL,
  page_token = NULL,
  simplify = TRUE,
  auth = if (mine) "token" else "key"
)
```

Arguments

`channel_id`	Optional channel ID.
`playlist_ids`	Optional character vector of playlist IDs.
`mine`	If ‘TRUE‘, retrieve playlists owned by the authenticated user.
`part`	Character vector of playlist resource parts.
`max_results`	Maximum total number of playlists to return.
`language`	Optional language code for localized text.
`page_token`	Optional page token at which to start.
`simplify`	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
`auth`	Authentication method, ‘"key"‘ or ‘"token"‘. ‘mine = TRUE‘ requires OAuth.

Value

A data frame when ‘simplify = TRUE‘; otherwise a playlists-list response.

References

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

Examples

```
## Not run:
list_playlists(channel_id = "UCMtFAi84ehTSYSE9XoHefig")
list_playlists(playlist_ids = c("PLAYLIST_1", "PLAYLIST_2"))
list_playlists(mine = TRUE)

## End(Not run)
```

---

list_playlist_items *List Playlist Items*

Description

Retrieves items from one playlist or retrieves specific playlist items by their playlist-item IDs.

Usage

```
list_playlist_items(
  playlist_id = NULL,
  playlist_item_ids = NULL,
  video_id = NULL,
  part = c("contentDetails", "snippet", "status"),
  max_results = 50,
  page_token = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

playlist_id	Optional playlist ID.
playlist_item_ids	Optional character vector of playlist-item IDs. Supply exactly one of ‘playlist_id‘ or ‘playlist_item_ids‘.
video_id	Optional video ID used to filter items in ‘playlist_id‘.
part	Character vector of playlist-item resource parts.
max_results	Maximum total number of items to return.
page_token	Optional page token at which to start.
simplify	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise a playlist-items response.

References

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

Examples

```
## Not run:
list_playlist_items(
  playlist_id = "PLrEnWoR732-CN09YykVof2lxdI3MLOZda",
  max_results = 100
)

## End(Not run)
```

`list_popular_videos` *List Most-Popular Videos*

Description

Retrieves the ‘mostPopular‘ videos chart.

Usage

```
list_popular_videos(
  region_code = NULL,
  category_id = NULL,
  max_results = 50,
  part = c("snippet", "statistics"),
  language = NULL,
  page_token = NULL,
  simplify = TRUE,
  auth = "key"
)
```

Arguments

`region_code`	Optional ISO 3166-1 alpha-2 content-region code.
`category_id`	Optional video-category ID.
`max_results`	Maximum total number of videos to return.
`part`	Character vector of video resource parts.
`language`	Optional language code for localized text.
`page_token`	Optional page token at which to start.
`simplify`	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
`auth`	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame when ‘simplify = TRUE‘; otherwise a videos-list response.

References

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

Examples

```
## Not run:
list_popular_videos(region_code = "US", max_results = 10)

## End(Not run)
```

---

list_regions *List Supported Content Regions*

Description

List Supported Content Regions

Usage

```
list_regions(language = NULL, auth = "key")
```

Arguments

language	Optional language code for localized names.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame with ‘region_code‘, ‘name‘, and ‘etag‘ columns.

References

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

Examples

```
## Not run:
list_regions()

## End(Not run)
```

`list_subscriptions` *List Subscriptions*

Description

Retrieves subscription resources using exactly one YouTube filter.

Usage

```
list_subscriptions(
  channel_id = NULL,
  subscription_ids = NULL,
  mine = FALSE,
  my_recent_subscribers = FALSE,
  my_subscribers = FALSE,
  for_channel_ids = NULL,
  part = c("snippet", "contentDetails"),
  order = "relevance",
  max_results = 50,
  page_token = NULL,
  simplify = TRUE,
  auth = if (mine || my_recent_subscribers || my_subscribers) {
     "token"
 } else {

      "key"
 }
)
```

Arguments

`channel_id`	Optional channel whose public subscriptions to retrieve.
`subscription_ids`	Optional character vector of subscription IDs.
`mine`	If 'TRUE', retrieve channels the authenticated user subscribes to.
`my_recent_subscribers`	If 'TRUE', retrieve the authenticated channel's recent subscribers.
`my_subscribers`	If 'TRUE', retrieve the authenticated channel's subscribers.
`for_channel_ids`	Optional channel IDs used to narrow matching subscriptions.
`part`	Character vector of subscription resource parts.
`order`	Sort order: '"alphabetical"', '"relevance"', or '"unread"'.
`max_results`	Maximum total number of subscriptions to return.
`page_token`	Optional page token at which to start.

simplify	If ‘TRUE‘, return a data frame; otherwise return the raw API response with all collected items.
auth	Authentication method, ‘"key"‘ or ‘"token"‘. Authenticated-user filters require OAuth.

Value

A data frame when ‘simplify = TRUE‘; otherwise a subscriptions-list response.

References

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

Examples

```
## Not run:
list_subscriptions(channel_id = "UChTJTbr5kf3hYazJZO-euHg")
list_subscriptions(mine = TRUE)

## End(Not run)
```

---

list_super_chat_events

List 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

```
list_super_chat_events(
  part = "snippet",
  language = NULL,
  max_results = 50,
  page_token = NULL,
  simplify = TRUE
)
```

Arguments

part	Parts to retrieve. Valid values are "snippet". Default is "snippet".
language	Optional language code for localized text.
max_results	Maximum total number of events to return.
page_token	Specific page token to retrieve. Optional.
simplify	Whether to return a simplified data.frame. Default is TRUE.

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 <- list_super_chat_events()

## End(Not run)
```

---

list_video_categories *List Video Categories*

Description

Lists video categories by region or by category ID.

Usage

```
list_video_categories(
  region_code = NULL,
  category_ids = NULL,
  language = NULL,
  auth = "key"
)
```

Arguments

region_code	Optional ISO 3166-1 alpha-2 content-region code.
category_ids	Optional character vector of video-category IDs.
language	Optional language code for localized text.
auth	Authentication method, ‘"key"‘ or ‘"token"‘.

Value

A data frame with one row per category and snake-case column names.

References

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

Examples

```
## Not run:
list_video_categories(region_code = "JP")
list_video_categories(category_ids = "10")

## End(Not run)
```

`post_comment` *Post a Top-Level Comment*

Description

Posts a new top-level comment on a YouTube video. Requires OAuth 2.0 authentication. YouTube no longer accepts comments on a channel itself, so a video is required.

Usage

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

Arguments

`video_id`	Character. ID of the video to comment on.
`channel_id`	Character. Optional ID of the channel that owns the video.
`text`	Character. The text of the comment.

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)
```

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 one row per caption, in file order, and the columns ‘text‘, ‘start‘ and ‘stop‘. A caption that spans several lines keeps its line breaks.

Examples

```
if (yt_authorized()) {
  vids <- list_my_videos()
  res <- list_captions(video_id = vids$video_id[[1]])
  cap <- download_caption(res$caption_id[[1]], 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.

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)
```

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_ids,
  moderation_status,
  ban_author = FALSE
)
```

Arguments

comment_ids	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.

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_ids = "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).

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)
```

---

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

Value

The number of entries cleared, invisibly.

Examples

```
tuber_cache_clear()
```

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)

Value

'NULL', invisibly. Called for its side effect on the session's cache settings.

Examples

```
tuber_cache_config(default_ttl = 600)
tuber_cache_info()$default_ttl

# Restore the defaults.
tuber_cache_config()
```

tuber_cache_info *Get current cache configuration*

Description

Get current cache configuration

Usage

```
tuber_cache_info()
```

Value

List with cache configuration

Examples

```
str(tuber_cache_info())
```

`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

Value

'result', invisibly, after printing its metadata; 'NULL', invisibly, when 'result' carries none.

Examples

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

## End(Not run)
```

`update_video_metadata` *Update Video Metadata*

Description

Updates selected mutable video fields while preserving the other fields in each requested YouTube resource part. At least one field must be supplied.

Usage

```
update_video_metadata(
  video_id,
  title = NULL,
  category_id = NULL,
  description = NULL,
  tags = NULL,
  default_language = NULL,
  privacy_status = NULL,
  made_for_kids = NULL,
  contains_synthetic_media = NULL,
  embeddable = NULL,
  license = NULL,
  public_stats_viewable = NULL,
  publish_at = NULL,
  on_behalf_of_content_owner = NULL
)
```

Arguments

video_id	YouTube video ID.
title	Optional title.
category_id	Optional video-category ID.
description	Optional description. Use ‘""‘ to clear it.
tags	Optional character vector of tags. Use ‘character()‘ to clear existing tags.
default_language	Optional default language.
privacy_status	Optional privacy status: ‘"private"‘, ‘"public"‘, or ‘"unlisted"‘.
made_for_kids	Optional self-declared made-for-kids setting.
contains_synthetic_media	Optional synthetic-media disclosure.
embeddable	Optional embeddable setting.
license	Optional license, ‘"creativeCommon"‘ or ‘"youtube"‘.
public_stats_viewable	Optional public-statistics setting.
publish_at	Optional RFC 3339 publication time. YouTube requires a private video that has never been published.
on_behalf_of_content_owner	Optional YouTube content-owner ID. This is only available to authorized YouTube content partners.

Value

The updated video resource.

References

<https://developers.google.com/youtube/v3/docs/videos/update>

Examples

```
## Not run:
update_video_metadata(
  video_id = "VIDEO_ID",
  title = "New Video Title",
  privacy_status = "unlisted"
)

## End(Not run)
```

`upload_caption` *Upload a Caption Track*

Description

Uploads a timed caption file using YouTube's resumable media-upload protocol. The request requires OAuth 2.0 authorization.

Usage

```
upload_caption(
  file,
  video_id,
  caption_name,
  language = "en-US",
  is_draft = FALSE,
  on_behalf_of_content_owner = NULL,
  open_url = FALSE,
  chunk_size = 8 * 1024^2,
  max_tries = 5
)
```

Arguments

`file`	Path to a caption file containing timing information.
`video_id`	YouTube video ID.
`caption_name`	Name of the caption track. YouTube limits names to 150 characters.
`language`	BCP 47 language tag for the caption track.
`is_draft`	Whether the caption track should remain a draft.
`on_behalf_of_content_owner`	Optional YouTube content-owner ID. This is only available to authorized YouTube content partners.

open_url	Whether to open the video's YouTube URL after a successful upload.
chunk_size	Bytes sent per request. Must be a multiple of 256 KB. Uploads resume from the last byte YouTube confirms, so a smaller chunk loses less work when a connection drops.
max_tries	Consecutive failed attempts to tolerate before giving up.

Value

A list containing the final HTTP response, parsed caption resource, and video URL.

References

<https://developers.google.com/youtube/v3/docs/captions/insert>

Examples

```
## Not run:
upload_caption(
  file = "captions.vtt",
  video_id = "dQw4w9WgXcQ",
  caption_name = "English"
)

## End(Not run)
```

---

upload_video *Upload Video to YouTube*

Description

Upload Video to YouTube

Usage

```
upload_video(
  file,
  snippet = NULL,
  status = list(privacyStatus = "public"),
  notify_subscribers = TRUE,
  on_behalf_of_content_owner = NULL,
  content_owner_channel_id = NULL,
  open_url = FALSE,
  chunk_size = 8 * 1024^2,
  max_tries = 5
)
```

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'.
notify_subscribers	Whether YouTube should notify subscribers about the new video.
on_behalf_of_content_owner	Optional YouTube content-owner ID. This is only available to authorized YouTube content partners.
content_owner_channel_id	Optional channel ID for a content partner upload. This must be supplied with 'on_behalf_of_content_owner'.
open_url	Should the video be opened using browseURL
chunk_size	Bytes sent per request. Must be a multiple of 256 KB. Uploads resume from the last byte YouTube confirms, so a smaller chunk loses less work when a connection drops.
max_tries	Consecutive failed attempts to tolerate before giving up.

Value

A list of the response object, 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)
```

yt_get_quota_usage *Get Current Quota Usage*

Description

Returns session-local estimated quota usage for the current quota day. Actual project usage is available in the Google Cloud Console.

Usage

```
yt_get_quota_usage()
```

Value

A data frame with one row per quota bucket and columns for estimated usage, configured limits, remaining quota, and reset time.

Examples

```
## Not run:
quota_status <- yt_get_quota_usage()
quota_status[quota_status$bucket == "data", ]

## End(Not run)
```

yt_key *Manage YouTube API key*

Description

These functions read and set the YouTube API key in the current R process.

Usage

```
yt_get_key()
yt_set_key(key)
```

Arguments

key	A character vector specifying a YouTube API key.

Value

‘yt_get_key()‘ returns ‘YOUTUBE_KEY‘ invisibly, or ‘NULL‘ when it is unset.

‘yt_set_key()‘ sets ‘YOUTUBE_KEY‘ for the current R process and invisibly returns the key. Put the variable in a user-level ‘.Renviron‘ file yourself if it must persist across sessions. In continuous integration, supply it from the service’s secret store as an environment variable.

Examples

```
## Not run:
yt_set_key("YOUR_YOUTUBE_API_KEY")
yt_get_key()

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

## End(Not run)
```

`yt_oauth` *Set up Authorization*

Description

The function reads a cached token when one exists. Otherwise, it opens the system browser and asks Google to authorize the application. By default, tokens are stored in the user's R cache directory rather than the project directory.

Usage

```
yt_oauth(
  app_id = NULL,
  app_secret = NULL,
  scope = "ssl",
  token = file.path(tools::R_user_dir("tuber", "cache"), "oauth-token.rds"),
  ...
)
```

Arguments

`app_id`	client id. Required the first time; optional once a token is cached.
`app_secret`	client secret. Required the first time; optional once a token is cached.
`scope`	Character. `ssl`, `basic`, `own_account_readonly`, `upload_and_manage_own_videos`, `channel_memberships`, `partner`, and `partner_audit`. Required. `ssl` and `basic` are basically interchangeable. Default is `ssl`.
`token`	Path to the token cache. The default is 'file.path(tools::R_user_dir("tuber", "cache"), "oauth-token.rds")'.
`...`	Additional arguments passed to `oauth_flow_auth_code`

Details

The cache file holds the token together with the OAuth client, including the client secret, because an access token lasts an hour and can only be refreshed by the client that minted it. Treat the file as a credential.

Value

The OAuth token, invisibly. The function also sets the 'google_token' option and saves the token at 'token'.

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(
  "YOUR-CLIENT-ID.apps.googleusercontent.com",
  "YOUR-CLIENT-SECRET"
)

## End(Not run)
```

---

yt_reset_quota *Reset Quota Counter*

Description

Reset the quota counter (typically done automatically at midnight Pacific Time, when YouTube's daily quota resets)

Usage

```
yt_reset_quota()
```

Value

'NULL', invisibly. Called for its side effect on the session's quota estimate.

Examples

```
yt_reset_quota()
yt_get_quota_usage()
```

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",
  order = "relevance",
  event_type = NULL,
  location = NULL,
  location_radius = NULL,
  published_after = NULL,
  published_before = NULL,
  video_definition = "any",
  video_duration = "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,
  auth = "key"
)
```

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'.

order Character. Sort order. One of 'date', 'rating','relevance', 'title', 'videoCount', 'viewCount'.

event_type Character. Optional. Takes one of three values: 'completed', 'live', 'upcoming'. Requires type = "video".

location Character. Optional. Latitude and Longitude within parentheses, e.g. "(37.42307,-122.08427)". Requires type = "video" and location_radius.

location_radius
Character. Optional. e.g. "1500m", "5km", "10000ft", "0.75mi"

published_after
Character. Optional. An RFC 3339 date-time, for instance "1970-01-01T00:00:00Z" or "1970-01-01T05:30:00+05:30".

published_before
Character. Optional. An RFC 3339 date-time, in the same format as published_after.

video_definition
Character. Optional. Takes one of three values: 'any' (return all videos; Default), 'high', 'standard'

video_duration Character. Optional. One of 'any', 'long','medium', 'short'. YouTube defines 'short' as less than four minutes; it does not identify the YouTube Shorts product.

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. Optional. An ISO 3166-1 alpha-2 country code.

relevance_language
Character. Default is "en". This default is tuber's choice: YouTube itself applies no language preference, so pass NULL to search without one. 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 FALSE, it returns the search-list response as a list.
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.
auth	Authentication method: ‘"token"‘ or ‘"key"‘.

Value

When ‘simplify = TRUE‘, a data frame with a resource ID column and snake-case metadata columns. Otherwise, a raw search-list response. 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, bucket = "data")
```

Arguments

limit	Integer. Daily quota limit for the selected bucket.
bucket	Quota bucket: ‘"data"‘, ‘"search"‘, or ‘"video_uploads"‘.

Value

The new limit, invisibly.

Examples

```
## Not run:
# If you have a higher quota limit
yt_set_quota_limit(50000, bucket = "data")

## 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

‘yt_token()‘ returns the httr2 token set by [yt_oauth()], or ‘NULL‘ when there is none. ‘yt_authorized()‘ returns ‘TRUE‘ when a token is set.

Examples

```
yt_authorized()
```

Index

add_video_to_playlist, 3
analyze_channel, 4
analyze_trends, 5

browseURL, *56*
bulk_video_analysis, 6

change_playlist_title, 7
compare_channels, 8
count_emojis, 9
create_playlist, 9

delete_caption, 10
delete_channel_section, 11
delete_comment, 11
delete_playlist, 12
delete_playlist_item, 13
delete_video, 13
download_caption, 14

extended-endpoints, 15
extract_emojis, 15

get_all_channel_video_stats, 16
get_all_comments, 17
get_channel_details, 17
get_my_channel *(*get_channel_details*)*, 17
get_playlist_item_ids, 19
get_playlist_video_ids, 20
get_video_broadcast_timing, 20
get_video_details, 21
get_video_stats, 23
get_video_thumbnails, 24

has_emoji, 24
helper-functions, 25

insert_channel_banner, 25

list_abuse_report_reasons, 26
list_captions, 27
list_channel_activities, 28
list_channel_members, 29
list_channel_sections, 30
list_channel_videos, 31
list_comment_threads, 33
list_comments, 32
list_languages, 34
list_live_broadcasts, 35
list_live_chat_messages, 36
list_my_videos, 37
list_playlist_items, 39
list_playlists, 38
list_popular_videos, 40
list_regions, 41
list_subscriptions, 42
list_super_chat_events, 43
list_video_categories, 44

oauth_flow_auth_code, *58*

post_comment, 45

quota_management, 46

read_sbv, 46
remove_emojis, 47
replace_emojis, 47
reply_to_comment, 48

set_comment_moderation_status, 48
set_video_thumbnail, 49

tuber, 50
tuber_cache_clear, 50
tuber_cache_config, 51
tuber_cache_info, 51
tuber_info, 52

update_video_metadata, 52
upload_caption, 54
upload_video, 55

yt_authorized *(yt_token)*, 63
yt_check_token *(yt_token)*, 63
yt_get_key *(yt_key)*, 57
yt_get_quota_usage, 57
yt_key, 57
yt_oauth, 58
yt_reset_quota, 59
yt_search, 60
yt_set_key *(yt_key)*, 57
yt_set_quota_limit, 63
yt_token, 63