Package ‘rStrava’

May 24, 2023

Type Package
Title Access the 'Strava' API
Version 1.2.0
Date 2023-05-24
Description Functions to access data from the 'Strava v3 API' <https://developers.strava.com/>.

BugReports https://github.com/fawda123/rStrava/issues
License CC0
Imports dplyr, geosphere, ggplot2, ggrepel, googleway, httr, magrittr, plyr, RCurl, rvest, tidyr, XML, xml2, purrr, tibble, V8
Suggests ggmap
Depends R (>= 3.5.0)
RoxygenNote 7.1.2
NeedsCompilation no
Author Marcus W. Beck [cre], Pedro Villarroel [aut], Daniel Padfield [aut], Lorenzo Gaborini [aut], Niklas von Maltzahn [aut]
Maintainer Marcus W. Beck <mbafs2012@gmail.com>
Repository CRAN
Date/Publication 2023-05-24 15:30:02 UTC

R topics documented:

achievement_fun .......................................................... 3
athlind_fun ................................................................. 3
athl_fun .......................................................... 4
chk_nopolyline .............................................................. 5
compile_activities ......................................................... 6
R topics documented:

compile_activity .................................................. 7
compile_activity_streams ........................................... 8
compile_club_activities ............................................. 9
compile_segment ..................................................... 10
compile_seg_effort .................................................. 11
compile_seg_efforts ................................................ 12
filter.actframe ..................................................... 13
get_activity ........................................................ 14
get_activity_list .................................................... 15
get_activity_streams ............................................... 16
get_athlete .......................................................... 17
get_basic ............................................................ 18
get_club ............................................................. 19
get_dists ............................................................ 20
get_efforts_list ..................................................... 21
get_elev_prof ......................................................... 22
get_explore .......................................................... 24
get_gear .............................................................. 25
get_heat_map .......................................................... 26
get_KOMs ............................................................ 29
get_laps ............................................................... 30
get_latlon ............................................................ 30
get_leaderboard ....................................................... 31
get_pages ............................................................. 32
get_segment .......................................................... 33
get_spdsplits ........................................................ 34
get_starred ........................................................... 35
get_streams .......................................................... 36
location_fun .......................................................... 37
monthly_fun .......................................................... 38
mutate.actframe ....................................................... 38
plot_spdsplits ........................................................ 39
ratelimit .............................................................. 40
recent_fun ............................................................. 41
seltime_fun ........................................................... 41
strava_oauth .......................................................... 42
units_fun ............................................................. 43
url_activities ........................................................ 44
url_athlete ............................................................ 44
url_clubs ............................................................. 45
url_gear .............................................................. 46
url_segment ........................................................... 46
url_streams ........................................................... 47

Index 48
achievement_fun

Get recent achievements

Description
Get recent achievements, used internally in athl_fun.

Usage
achievement_fun(prsd)

Arguments
prsd  parsed input list

Value
A data frame of recent achievements for the athlete. An empty list is returned if none found.

athlind_fun
Get data for a single athlete

Description
Get data for a single athlete by web scraping, does not require authentication.

Usage
athlind_fun(athl_num)

Arguments
athl_num  numeric athlete id used by Strava

Value
A list with elements for the athlete name, location, units of measurement, monthly data, recent activities, and achievements.
**athl_fun**

*Get data for an athlete*

**Description**

Get data for an athlete by web scraping, does not require authentication.

**Usage**

```r
athl_fun(athl_num, trace = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>athl_num</code></td>
<td>numeric vector of athlete id(s) used by Strava</td>
</tr>
<tr>
<td><code>trace</code></td>
<td>logical indicating if output is returned to console</td>
</tr>
</tbody>
</table>

**Details**

The athlete id is assigned to the user during registration with Strava and this must be known to use the function. Some users may have privacy settings that prevent public access to account information (a message indicating as such will be returned by the function). The function scrapes data using the following URL with the appended athlete id, e.g., [https://www.strava.com/athletes/2837007](https://www.strava.com/athletes/2837007). Opening the URL in a web browser can verify if the data can be scraped. Logging in to the Strava account on the website may also be required before using this function.

**Value**

A list for each athlete, where each element is an additional list with elements for the athlete name, location, units of measurement, monthly data, recent activities, and achievements. The list elements are named using the athlete id numbers. `NA` will be returned if the data for an athlete could not be accessed.

**Examples**

```r
## single athlete
athl_fun(2837007)

## multiple athletes
athl_fun(c(2837007, 2527465))
```
Remove activities with no geographic data

Usage

chk_nopolyline(act_data, ...)

## S3 method for class 'actframe'
chk_nopolyline(act_data, ...)

Arguments

act_data a data.frame returned by compile_activities
...

arguments passed to or from other methods

Details

This function is used internally within get_elev_prof and get_heat_map to remove activities that cannot be plotted because they have no geographic information. This usually applies to activities that were manually entered.

Value

act_data with rows removed where no polylines were available, the original dataset is returned if none were found. A warning is also returned indicating the row numbers that were removed if applicable.

Author(s)

Marcus Beck

Examples

### Not run:

# get my activities
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
myActs <- get_activity_list(stoken)
act_data <- compile_activities(myActs)
chk_nopolyline(act_data)

### End(Not run)
**compile_activities** converts a list of activities into a dataframe

**Description**

converts a list of activities into a dataframe

**Usage**

```
compile_activities(actlist, acts = NULL, id = NULL, units = "metric")
```

**Arguments**

- `actlist` an activities list returned by `get_activity_list`
- `acts` numeric indicating which activities to compile starting with most recent, defaults to all
- `id` optional numeric vector to specify the id(s) of the activity/activities to plot, `acts` is ignored if provided
- `units` chr string indicating metric or imperial

**Details**

each activity has a value for every column present across all activities, with NAs populating empty values

**Value**

An activities frame object (`actframe` that includes a data frame for the data and attributes for the distance, speed, and elevation units

**Author(s)**

Daniel Padfield

**See Also**

- `compile_club_activities` for compiling an activities list for club activities

**Examples**

```
## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

my_acts <- get_activity_list(stoken)

acts_data <- compile_activities(my_acts)
```
# show attributes
attr(acts_data, 'unit_type')
attr(acts_data, 'unit_vals')

## End(Not run)

## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
acts <- get_activity_list(stoken)
compile_activity(acts[1])
## End(Not run)

### compile_activity
convert a single activity list into a dataframe

#### Description
convert a single activity list into a dataframe

#### Usage
compile_activity(x, columns)

#### Arguments
- **x**: a list containing details of a single Strava activity
- **columns**: a character vector of all the columns in the list of Strava activities. Produced automatically in `compile_activities`. Leave blank if running for a single activity list.

#### Details
used internally in `compile_activities`

#### Value
dataframe where every column is an item from a list. Any missing columns rom the total number of columns

#### Author(s)
Daniel Padfield

#### Examples
```r
## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
acts <- get_activity_list(stoken)
compile_activity(acts[1])
## End(Not run)
```
**compile_activity_streams**

*Convert a set of streams of a single activity into a dataframe*

## Description

Convert a set of streams of a single activity into a dataframe, with the retrieved columns.

## Usage

```r
compile_activity_streams(streams, id = NULL)
```

## Arguments

- **streams**
  - a list containing details of the Strava streams of a single activity (output of `get_streams`)
- **id**
  - if not missing, the activity id of the stream (will be appended to the data.frame, if non-empty)

## Details

used internally in `get_activity_streams`

## Value

data frame where every column is the stream data for the retrieved types.

## Author(s)

Lorenzo Gaborini

## Examples

```r
## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
act_id <- 351217692
streams <- get_streams(stoken, id = act_id, types = list('distance', 'latlng'))
compile_activity_streams(streams, id = act_id)
## End(Not run)
```
compile_club_activities

converts a list of club activities into a dataframe

Description
converts a list of club activities into a dataframe

Usage
compile_club_activities(actlist)

Arguments
actlist a club activities list returned by get_activity_list

Details
each activity has a value for every column present across all activities, with NAs populating empty values

Value
An data.frame of the compiled activities from actlist

Author(s)
Marcus Beck

Examples

## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
club_acts <- get_activity_list(stoken, id = 13502, club = TRUE)
acts_data <- compile_club_activities(club_acts)

## End(Not run)
compile_segment

Compile information on a segment

Description

Compile generation information on a segment

Usage

compile_segment(seglist)

Arguments

seglist a Strava segment list returned by get_segment

Details

compiles information for a segment

Value

dataframe of all information given in a call from get_segment

Examples

## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id,
app_secret, cache = TRUE))

# compile segment info
get_segment(stoken, id = 229781) %>% compile_segment

# compile top ten leaderboard for the segment
get_segment(stoken, id = 229781, request = "leaderboard") %>% compile_segment

# compile all efforts for the authenticated user on the segment
get_segment(stoken, id = 4483903, request = 'all_efforts') %>% compile_segment

# compile the starred segments for the user
get_segment(stoken, request = 'starred') %>% compile_segment

## End(Not run)
**compile seg effort**

**Compile the efforts of a segment**

**Description**

Cleans up the output of `get_efforts_list()` into a dataframe

**Usage**

```r
compile_seg_effort(x)
```

**Arguments**

- `x` A list object produced by `get_efforts_list`

**Details**

Used internally in `compile_seg_efforts`. Can be used on the output of `get_efforts_list` to compile the segment efforts of a single segment. Each call to `get_efforts_list` returns a large list. This function returns a subset of this information.

**Value**

A dataframe containing all of the efforts of a specific segment. The columns returned are `athlete.id`, `distance`, `elapsed_time`, `moving_time`, `name`, `start_date` and `start_date_local`.

**Author(s)**

Daniel Padfield

**Examples**

```r
## Not run:
# set token
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

# segments to get efforts from - use some parkruns
segment <- 2269028

efforts <- get_efforts_list(stoken, segment)

# compile efforts
efforts <- compile_seg_effort(efforts)

## End(Not run)
```
**compile_seg_efforts**

*Compile the efforts of multiple segments*

**Description**

Compiles the information of athletes from multiple segments

**Usage**

```r
compile_seg_efforts(segment_ids, stoken)
```

**Arguments**

- `segment_ids` A vector of segment ids from which to compile efforts
- `stoken` A `config` object created using the `strava_oauth` function

**Details**

Uses `get_elev_prof` and `compile_seg_effort` internally to compile efforts of multiple segments

**Value**

A dataframe of the details of each segment effort

**Author(s)**

Daniel Padfield

**Examples**

```r
## Not run:
# set token
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

# segments to get efforts from - use some parkruns
segments <- c(2269028, 5954625)

# compile segment efforts
segments %>% purrr::map_df(.f = compile_seg_efforts, stoken = my_token, .id = 'id')

## End(Not run)
```
filter(actframe)  

**Filter**

**Description**

This is a wrapper function to dplyr::filter which can be applied to an actframe object.

**Usage**

```r
## S3 method for class 'actframe'
filter(.data, ...)
```

**Arguments**

- `.data`: an actframe object
- `...`: Logical predicates defined in terms of the variables in `.data`

**Value**

an actframe object

**Examples**

```r
## Not run:
library(dplyr)

# get actframe, all activities
stoken <- httr::config(
  token = strava_oauth(
    app_name,
    app_client_id,
    app_secret,
    app_scope="activity:read_all"
  
)
}
my_acts <- get_activity_list(stoken)
act_data <- compile_activities(my_acts)

# mutate
act_data %>% filter(name %in% 'Morning Ride')

## End(Not run)
```
get_activity

Get detailed data of an activity

Description

Get detailed data of an activity, including segment efforts

Usage

get_activity(id, stoken)

Arguments

id
numeric for id of the activity

stoken
A `config` object created using the `strava_oauth` function

Details

Requires authentication stoken using the `strava_oauth` function and a user-created API on the Strava website.

The id for each activity can be viewed using results from `get_activity_list`.

Value

Data from an API request.

Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

get_activity(75861631, stoken)

## End(Not run)
```
**get_activity_list**

*Get an activities list*

**Description**

Get an activities list of the desired type (club, user)

**Usage**

```r
get_activity_list(stoken, id = NULL, before = NULL, after = NULL, club = FALSE)
```

**Arguments**

- **stoken**
  - A `config` object created using the `strava_oauth` function
- **id**
  - numeric for id of the activity or club if `club = TRUE`, leave blank to retrieve all activities
- **before**
  - date object for filtering activities before the indicated date
- **after**
  - date object for filtering activities after the indicated date
- **club**
  - logical if you want the activities of a club

**Details**

Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website. If retrieving activities using individual id values, the output list returned contains additional information from the API and the results have not been tested with the functions in this package. It is better practice to retrieve all activities (as in the example below), use `compile_activities`, and then filter by individual activities.

If retrieving club activities, the user for the API must be a member of the club.

**Value**

A list of activities for further processing or plotting.

**Examples**

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

get_activity_list(stoken)
## End(Not run)
```
get_activity_streams

Retrieve streams for activities, and convert to a dataframe

Description

Retrieve streams for activities, and convert to a dataframe.

Usage

get_activity_streams(act_data, ...)

## S3 method for class 'list'
get_activity_streams(
  act_data,
  stoken,
  acts = NULL,
  id = NULL,
  types = NULL,
  resolution = "high",
  series_type = "distance",
  ...
)

## S3 method for class 'actframe'
get_activity_streams(
  act_data,
  stoken,
  types = NULL,
  resolution = "high",
  series_type = "distance",
  ...
)

Arguments

- **act_data**: an list object returned by get_activity_list or a data.frame returned by compile_activities
- **...**: arguments passed to or from other methods
- **stoken**: A config object created using the strava_oauth function
- **acts**: numeric indicating which activities to compile starting with most recent, defaults to all
- **id**: optional numeric vector to specify the id(s) of the activity/activities to plot, acts is ignored if provided
- **types**: list indicating which streams to get for each activity, defaults to all available, see details.
### Description

Get basic athlete data for an athlete using an API request

### Usage

get_athlete(stoken, id = NULL)
get_basic

Get basic Strava data

Description
Get basic Strava data with requests that don’t require pagination

Usage
get_basic(url_, stoken, queries = NULL)

Arguments
- url_  
  string of url for the request to the API
- stoken  
  A config object created using the strava_oauth function
- queries  
  list of additional queries or parameters

Details
Requires authentication stoken using the strava_oauth function and a user-created API on the strava website.
get_club

Value
Data from an API request.

Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

# get basic user info
get_basic('https://strava.com/api/v3/athlete', stoken)

## End(Not run)
```

---

code

get_club  
Get club data

Description

Get club data for a given request

Usage

```r
get_club(stoken, id = NULL, request = NULL)
```

Arguments

- **stoken**: A `config` object created using the `strava_oauth` function
- **id**: numeric for id of the club, defaults to authenticated club of the athlete
- **request**: chr string, must be "members", "activities" or NULL for club details

Details

Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website.

Value

Data from an API request.
get_dists

Description
Get distance from longitude and latitude points

Usage
get_dists(lon, lat)

Arguments
lon chr string indicating name of longitude column in dat_in
lat chr string indicating name of latitude column in dat_in

Details
Used internally in get_elev_prof on objects returned by get_latlon

Value
A vector of distances with the length as the number of rows in dat_in

Author(s)
Daniel Padfield

Examples
## Not run:
# get activity data
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
my_acts <- get_activity_list(stoken)

# get the latest activity
acts_data <- compile_activities(my_acts)[1, ]
# get lat, lon
polyline <- acts_data$map.summary_polyline
latlon <- get_latlon(polyline, key = mykey)

# get distance
get_dists(latlon$lon, latlon$lat)

## End(Not run)

---

**get_efforts_list**  
*Get all the efforts in a segment if no queries are specified*

### Description
Get all the efforts in a segment if no queries are specified

### Usage
```r
get_efforts_list(
  stoken,  
  id,  
  athlete_id = NULL,  
  start_date_local = NULL,  
  end_date_local = NULL
)
```

### Arguments
- **stoken**: A `config` object created using the `strava_oauth` function
- **id**: numeric for id of the segment
- **athlete_id**: numeric for the athlete id for filtering the results
- **start_date_local**: the start date for filtering the results
- **end_date_local**: the end date for filtering the results

### Details
Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website.

### Value
Data from an API request.
Examples

## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id,
app_secret, cache = TRUE))

get_efforts_list(stoken, id = 229781)

## End(Not run)

---

get_elev_prof Create elevation profiles from activity data

Description

Create elevation profiles from activity data

Usage

get_elev_prof(act_data, ...)

## S3 method for class 'list'
get_elev_prof(
  act_data,
  acts = 1,
  id = NULL,
  key,
  total = FALSE,
  expand = 10,
  units = "metric",
  fill = "darkblue",
  ...
)

## S3 method for class 'actframe'
get_elev_prof(
  act_data,
  key,
  total = FALSE,
  expand = 10,
  fill = "darkblue",
  ...
)

## S3 method for class 'strframe'
get_elev_prof(act_data, total = FALSE, expand = 10, fill = "darkblue", ...)


get_elev_prof

Arguments

act_data an activities list object returned by get_activity_list or a data.frame returned by compile_activities

... arguments passed to or from other methods

acts numeric value indicating which elements of act_data to plot, defaults to most recent

id optional numeric vector to specify the id(s) of the activity/activities to plot, acts is ignored if provided

key chr string of Google API key for elevation data, passed to google_elevation, see details

total logical indicating if elevations are plotted as cumulative climbed by distance

expand a numeric multiplier for expanding the number of lat/lon points on straight lines. This can create a smoother elevation profile. Set expand = 1 to suppress this behavior.

units chr string indicating plot units as either metric or imperial, this has no effect if input data are already compiled with compile_activities

fill chr string of fill color for profile

Details

The Google API key is easy to obtain, follow instructions here: https://developers.google.com/maps/documentation/elevation/#api_key

Value

A ggplot of elevation profiles, faceted by activity id, date

Author(s)

Daniel Padfield, Marcus Beck

See Also

get_dists

Examples

## Not run:
# get my activities
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
my_acts <- get_activity_list(stoken)

# your unique key
mykey <- 'Get Google API key'
get_elev_prof(my_acts, acts = 1:2, key = mykey)

# compile first, change units
my_acts <- compile_activities(my_acts, acts = c(1:2), units = 'imperial')
get_elev_prof(my_acts, key = mykey)
## get_explore

Explore segments within a bounded area

**Description**

Explore segments within a bounded area

**Usage**

```r
get_explore(
  stoken,
  bounds,
  activity_type = "riding",
  max_cat = NULL,
  min_cat = NULL
)
```

**Arguments**

- **stoken**: A `config` object created using the `strava_oauth` function
- **bounds**: chr string representing the comma separated list of bounding box corners 'sw.lat,sw.lng,ne.lat,ne.lng' or 'south, west, north, east', see the example
- **activity_type**: chr string indicating activity type, "riding" or "running"
- **max_cat**: numeric indicating the maximum climbing category
- **min_cat**: numeric indicating the minimum climbing category

**Details**

Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website.

**Value**

Data from an API request.

**Examples**

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

bnds <- "37.821362, -122.505373, 37.842038, -122.465977"
```
Description

Get gear details from its identifier

Usage

gear_gear(id, stoken)

Arguments

id string, identifier of the equipment item
stoken A config object created using the strava_oauth function

Details

Requires authentication token using the strava_oauth function and a user-created API on the Strava website.

Value

Data from an API request.

Examples

## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

gear_gear("g2275365", stoken)

## End(Not run)
get_heat_map

Makes a heat map from your activity data

Description

Makes a heat map from your activity data

Usage

get_heat_map(act_data, ...)

## S3 method for class 'list'
get_heat_map(
  act_data,
  key,
  acts = 1,
  id = NULL,
  alpha = NULL,
  f = 1,
  add_elev = FALSE,
  as_grad = FALSE,
  distlab = TRUE,
  distval = 0,
  size = 0.5,
  col = "red",
  expand = 10,
  maptype = "terrain",
  source = "google",
  units = "metric",
  ...
)

## S3 method for class 'actframe'
get_heat_map(
  act_data,
  key,
  alpha = NULL,
  f = 1,
  add_elev = FALSE,
  as_grad = FALSE,
  distlab = TRUE,
  distval = 0,
  size = 0.5,
  col = "red",
  expand = 10,
  maptype = "terrain",
  source = "google",
  units = "metric",
  ...
get_heat_map

...)

## S3 method for class 'strframe'
get_heat_map(
  act_data,
  alpha = NULL,
  f = 1,
  filltype = c("elevation", "distance", "slope", "speed"),
  distlab = TRUE,
  distval = 0,
  size = 0.5,
  col = "red",
  expand = 10,
  maptype = "terrain",
  source = "google",
  ...
)

Arguments

act_data 
an activities list object returned by get_activity_list, an actframe returned
by compile_activities, or a strfame returned by get_activity_streams
... arguments passed to or from other methods
key 
chr string of Google API key for elevation data, passed to google_elevation
for polyline decoding, see details
acts 
numeric indicating which activities to plot based on index in the activities list,
defaults to most recent
id 
optional numeric vector to specify the id(s) of the activity/activities to plot, acts
is ignored if provided
alpha 
the opacity of the line desired. A single activity should be 1. Defaults to 0.5
f 
number specifying the fraction by which the range should be extended for the
bounding box of the activities, passed to make_bbox
add_elev 
logical indicating if elevation is overlayed by color shading on the activity lines
as_grad 
logical indicating if elevation is plotted as percent gradient, applies only if
add_elev = TRUE
distlab 
logical if distance labels are plotted along the route with geom_label_repel
distval 
numeric indicating rounding factor for distance labels which has direct control
on label density, see details
size 
numeric indicating width of activity lines
col 
chr string indicating either a single color of the activity lines if add_grad =
FALSE or a color palette passed to scale_fill_distiller if add_grad = TRUE
expand 
a numeric multiplier for expanding the number of lat/lon points on straight lines.
This can create a smoother elevation gradient if add_grad = TRUE. Set expand =
1 to suppress this behavior.
maptype  chr string indicating the base map type relevant for the source, passed to `get_map`
source    chr string indicating map source, passed to `get_map`
units     chr string indicating plot units as either metric or imperial, this has no effect if input data are already compiled with `compile_activities`
filltype  chr string specifying which stream variable to use for filling line segments, applies only to `strframe` objects, acceptable values are "elevation", "distance", "slope", or "speed"

Details

uses `get_latlon` to produce a dataframe of latitudes and longitudes to use in the map. Uses `ggmap` to produce the map and `ggplot2` to plot the route.

The Google API key for elevation is easy to obtain, follow instructions here: https://developers.google.com/maps/documentation/elevation/#api_key

A Google API key is needed if using any map services where `source = "google"`. The same key used for the elevation API can be used but must be registered externally with the `ggmap` package using `register_google()` before executing `get_heat_map()`. See the examples.

The `distval` argument is passed to the `digits` argument of `round`. This controls the density of the distance labels, e.g., 1 will plot all distances in sequence of 0.1, 0 will plot all distances in sequence of one, -1 will plot all distances in sequence of 10, etc.

Value

plot of activity on a Google map

Author(s)

Daniel Padfield, Marcus Beck

Examples

```r
# Not run:
# get my activities
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
my_acts <- get_activity_list(stoken)

# register Google maps API key
library(ggmap)
register_google('xxxxxxxxxxxx') # enter your key here

# default, requires Google key
mykey <- 'Get Google API key'
get_heat_map(my_acts, acts = 1, alpha = 1, key = mykey)

# plot elevation on locations, requires key
get_heat_map(my_acts, acts = 1, alpha = 1, key = mykey, add_elev = TRUE, col = 'Spectral', size = 2)

# compile first, change units
my_acts <- compile_activities(my_acts, acts = 156, units = 'imperial')
get_heat_map(my_acts, key = mykey, alpha = 1, add_elev = T, col = 'Spectral', size = 2,
```
get_KOMs

maptype = 'satellite')

## End(Not run)

---

get_KOMs

Get KOMs/QOMs/CRs of an athlete

Description

Get KOMs/QOMs/CRs of an athlete

Usage

get_KOMs(id, stoken)

Arguments

id | string or integer of athlete
stoken | A config object created using the strava_oauth function

Details

Requires authentication stoken using the strava_oauth function and a user-created API on the strava website.

Value

Data from an API request.

Examples

## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

get_KOMs(2837007, stoken)

## End(Not run)
### get_laps

**Retrieve the laps of an activity**

**Description**

Retrieve the laps of an activity

**Usage**

```r
get_laps(stoken, id)
```

**Arguments**

- **stoken**: A `config` object created using the `strava_oauth` function
- **id**: numeric for id of the activity with the laps to request

**Details**

Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website.

**Value**

Data from an API request.

**Examples**

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

get_laps(stoken, id = 351217692)

## End(Not run)
```

### get_latlon

**get latitude and longitude from Google polyline**

**Description**

get latitude and longitude from Google polyline

**Usage**

```r
get_latlon(polyline, key)
```
get_leaderboard

Arguments

polyline a map polyline returned for an activity from the API
key chr string of Google API key for elevation data, passed to google_elevation

Value
dataframe of latitude and longitudes with a column for the unique identifier

Author(s)

Daniel Padfield, Marcus Beck

Examples

```r
## Not run:
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

my_acts <- get_activity_list(stoken)
acts_data <- compile_activities(my_acts)

# get lat and lon for a single activity
polyline <- acts_data$map.summary_polyline[[1]]
get_latlon(polyline, key = mykey)

## End(Not run)
```

get_leaderboard Retrieve the leaderboard of a segment

Description

Retrieve the leaderboard of a segment

Usage

get_leaderboard(sttoken, id, nleaders = 10, All = FALSE)

Arguments

sttoken A config object created using the strava_oauth function
id numeric for id of the segment
nleaders numeric for number of leaders to retrieve
All logical to retrieve all of the list

Details

Requires authentication sttoken using the strava_oauth function and a user-created API on the strava website.
Value

Data from an API request.

Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id,
                          app_secret, cache = TRUE))

get_leaderboard(stoken, id = 229781)

## End(Not run)
```

---

**get_pages**

*Get several pages of one type of request*

**Description**

Get several pages of one type of request to the API

**Usage**

```r
get_pages(
  url_,
  stoken,
  per_page = 30,
  page_id = 1,
  page_max = 1,
  before = NULL,
  after = NULL,
  queries = NULL,
  All = FALSE
)
```

**Arguments**

- `url_`: string of url for the request to the API
- `stoken`: A `config` object created using the `strava_oauth` function
- `per_page`: numeric indicating number of items retrieved per page (maximum 200)
- `page_id`: numeric indicating page id
- `page_max`: numeric indicating maximum number of pages to return
- `before`: date object for filtering activities before the indicated date
- `after`: date object for filtering activities after the indicated date
- `queries`: list of additional queries to pass to the API
- `All`: logical if you want all possible pages within the ratelimit constraint
get_segment

Details

Requires authentication stoken using the strava_oauth function and a user-created API on the strava website.

Value

Data from an API request.

Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

# get basic user info
# returns 30 activities
get_pages('https://strava.com/api/v3/activities', stoken)

## End(Not run)
```

---

get_segment | Retrieve details about a specific segment

Description

Retrieve details about a specific segment

Usage

get_segment(stoken, id = NULL, request = NULL)

Arguments

- `stoken` A config object created using the strava_oauth function
- `id` numeric for id of the segment
- `request` chr string, must be "starred", "leaderboard", "all_efforts", or NULL for segment details

Details

Requires authentication stoken using the strava_oauth function and a user-created API on the strava website. The authenticated user must have an entry for a segment to return all efforts if request = "all_efforts". For request = "starred", set id = NULL.
get_spd_splits

Description

Allows the return of speed splits of multiple rides.

Usage

get_spd_splits(act_id, stoken, units = "metric")

Arguments

act_id      a vector of activity IDs. These are easily found in the data.frame returned by compile_activities
stoken      A config object created using the strava_oauth function
units       chr string indicating plot units as either metric or imperial

Examples

## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

# get segment info
get_segment(stoken, id = 229781)

# get top ten leaderboard for the segment
get_segment(stoken, id = 229781, request = "leaderboard")

# get all efforts for the authenticated user on the segment
get_segment(stoken, id = 4483903, request = 'all_efforts')

# get the starred segments for the user
get_segment(stoken, request = 'starred')

## End(Not run)
get_starred

Value

a data frame containing the splits of the activity or activities selected.

Author(s)

Marcus Beck

Examples

## Not run:
# get my activities
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
my_acts <- get_activity_list(stoken)

# compile activities
acts_data <- compile_activities(my_acts)

# get spdsplits for all activities
spd_splits <- purrr::map_df(acts_data$id, get_spdsplits, stoken = stoken,
  units = 'metric', .id = 'id')

## End(Not run)

get_starred

Retrieve a summary of the segments starred by an athlete

Description

Retrieve a summary of the segments starred by an athlete

Usage

get_starred(stoken, id = NULL)

Arguments

stoken A config object created using the strava_oauth function
id numeric for id of the athlete, defaults to authenticated athlete

Details

Requires authentication stoken using the strava_oauth function and a user-created API on the strava website.

Value

Data from an API request.
Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

gt_streams(stoken)

## End(Not run)
```

---

**get_streams**

*Retrieve a Strava data stream for a single activity*

Description

Retrieve a Strava data stream for a single activity. Internally called by `get_activity_streams`.

Usage

```r
gt_streams(
  stoken,    
id,       
request = "activities", 
types = NULL, 
resolution = NULL, 
series_type = NULL
)
```

Arguments

- **stoken**: A `config` object created using the `strava_oauth` function
- **id**: numeric for id of the request
- **request**: chr string defining the stream type, must be "activities", "segment_efforts", "segments"
- **types**: list of chr strings with any combination of "time" (seconds), "latlng", "distance" (meters), "altitude" (meters), "velocity_smooth" (meters per second), "heartrate" (bpm), "cadence" (rpm), "watts", "temp" (degrees Celsius), "moving" (boolean), or "grade_smooth" (percent)
- **resolution**: chr string for the data resolution to retrieve, can be "low", "medium", "high", defaults to all
- **series_type**: chr string for merging the data if resolution is not equal to "all". Accepted values are "distance" or "time". If omitted, no merging is performed.
Details

Requires authentication stoken using the `strava_oauth` function and a user-created API on the strava website. From the API documentation, ‘streams’ is the Strava term for the raw data associated with an activity.

Value

Data from an API request.

Examples

```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id,
app_secret, cache = TRUE))

get_streams(stoken, id = 351217692, types = list('distance', 'latlng'))

## End(Not run)
```

---

**location_fun**  
**Get athlete location**

Description

Get athlete location, used internally in `athl_fun`

Usage

```r
location_fun(prsd)
```

Arguments

- **prsd**: parsed input list

Value

A character string of the athlete location
monthly_fun

*Get distances for last twelve months*

**Description**

Get distances for last twelve months, used internally in `athl_fun`

**Usage**

`monthly_fun(prsd)`

**Arguments**

- `prsd` parsed input list

**Value**

A data frame of monthly summaries for the athlete, including distance, time, and elevation gain each month. A NA value is returned if no activity was observed in recent months.

---

mutate.actframe

*Mutate*

**Description**

This is a wrapper function to `dplyr::mutate` which can be applied to an `actframe` object

**Usage**

```r
## S3 method for class 'actframe'
mutate(.data, ...)
```

**Arguments**

- `.data` an `actframe` object
- `...` Name-value pairs of expressions. Use NULL to drop a variable.

**Value**

an `actframe` object
plot_spdsplits

Examples

```r
## Not run:
library(dplyr)

# get actframe, all activities
stoken <- httr::config(
  token = strava_oauth(
    app_name,  
    app_client_id,  
    app_secret,  
    app_scope="activity:read_all"
  )
)
my_acts <- get_activity_list(stoken)
act_data <- compile_activities(my_acts)

# mutate
act_data %>% mutate(is_run=type=="Run")
## End(Not run)
```

---

plot_spdsplits

Plot speed by splits

Description

Plot average speed by splits for a single activity

Usage

```r
plot_spdsplits(act_data, ...)
```

## S3 method for class 'list'
```r
plot_spdsplits(
  act_data,  
  stoken,  
  acts = 1,  
  id = NULL,  
  units = "metric",  
  fill = "darkblue",  
  ...  
)
```

## Default S3 method:
```r
plot_spdsplits(act_data, stoken, units = "metric", fill = "darkblue", ...)
```
Arguments

act_data  an activities list object returned by \texttt{get_activity_list} or a data.frame returned by \texttt{compile_activities}

...  arguments passed to other methods

stoken  A \texttt{config} object created using the \texttt{strava_oauth} function

acts  numeric indicating which activity to plot based on index in the activities list, defaults to most recent

id  optional numeric vector to specify the id(s) of the activity/activities to plot, \texttt{acts} is ignored if provided

units  chr string indicating plot units as either metric or imperial

fill  chr string of fill color for profile

Details

The average speed per split is plotted, including a dashed line for the overall average. The final split is typically not a complete km or mile.

Value

plot of average distance for each split value in the activity

Author(s)

Marcus Beck

Examples

```r
## Not run:
# get my activities
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))
my_acts <- get_activity_list(stoken)

# default
plot_spdsplits(my_acts, stoken, acts = 1)
## End(Not run)
```

---

ratelimit  \textit{Generate the ratelimit indicator}

Description

Checks the ratelimit values after the last request and stores the left requests in a global variable.

Usage

\texttt{ratelimit(req)}
recent_fun

Arguments

req

value returned from the GET function, used internally in other functions

Details

Requests to the Strava API are rate-limited. The default rate limit allows 600 requests every 15 minutes, with up to 30,000 requests per day. See the documentation at https://strava.github.io/api/#access.

Value

A variable for the current limits.

---

recent_fun

Get last three recent activities

Description

Get last three recent activities, used internally in athl_fun

Usage

recent_fun(prsd)

Arguments

prsd

parsed input list

Value

A data frame of recent activities for the athlete. An empty list is returned if none found.

---

seltime_fun

Format before and after arguments for API query

Description

Format before and after arguments for API query

Usage

seltime_fun(dtin, before = TRUE)

Arguments

dtin

Date object for before or after inputs

before

logical indicating if input is before
strava_oauth

Value
A numeric object as an epoch timestamp

Examples

# convert to epoch timestamp
seltime_fun(Sys.Date())

# back to original
as.POSIXct(seltime_fun(Sys.Date(), before = FALSE), tz = Sys.timezone(), origin = '1970-01-01')

strava_oauth
Generate Strava API authentication token

Description
Generate a token for the user and the desired scope. The user is sent to the strava authentication page if he/she hasn’t given permission to the app yet, else, is sent to the app webpage.

Usage

strava_oauth(
  app_name,
  app_client_id,
  app_secret,
  app_scope = "public",
  cache = FALSE
)

Arguments

app_name chr string for name of the app
app_client_id chr string for ID received when the app was registered
app_secret chr string for secret received when the app was registered
app_scope chr string for scope of authentication. Must be "read", "read_all", "profile:read_all", "profile:write", "activity:read", "activity:read_all" or "activity:write"
cache logical to cache the token

Details
The app_name, app_client_id, and app_secret are specific to the user and can be obtained by registering an app on the Strava API authentication page: http://strava.github.io/api/v3/oauth/. This requires a personal Strava account.

Value
A Token2.0 object returned by oauth2.0_token to be used with API function calls
## Examples

```r
## Not run:
applename <- 'myappname' # chosen by user
app_client_id <- 'myid' # an integer, assigned by Strava
app_secret <- 'xxxxxxxx' # an alphanumeric secret, assigned by Strava

# create the authentication token
stoken <- httr::config(
  token = strava_oauth(
    app_name,
    app_client_id,
    app_secret,
    app_scope="activity:read_all"
  )
)

# use authentication token
get_athlete(stoken, id = '2837007')

## End(Not run)
```

---

### units_fun  Get units of measurement

#### Description

Get units of measurement, used internally in `athl_fun`

#### Usage

```r
units_fun(prsd)
```

#### Arguments

- **prsd**
  - parsed input list

#### Value

A character vector indicating the units for distance used by the athlete
url_activities  
*Set the url of activities for different activity lists*

**Description**
Set the url of activities for different activity lists

**Usage**
url_activities(id = NULL, club = FALSE)

**Arguments**
- **id**  
  string for id of the activity or club if `club = TRUE`
- **club**  
  logical if you want the activities of a club

**Details**
This function concatenates appropriate strings so no authentication token is required. This is used internally by other functions.

**Value**
The set url.

**Examples**
```r
## Not run:
# create authentication token
# requires user created app name, id, and secret from Strava website
stoken <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

url_activities(2837007)
## End(Not run)
```

---

url_athlete  
*Set the url of the athlete to get data*

**Description**
Set the url of the athlete to get data using an ID

**Usage**
url_athlete(id = NULL)
Arguments

- `id`: str or integer of athlete id assigned by Strava, NULL will set the authenticated user URL.

Details

used by other functions

Value

A character string of the athlete URL used for API requests

---

url_clubs

*Set the url of the clubs for the different requests*

Description

Set the url of the clubs for the different requests

Usage

```r
url_clubs(id = NULL, request = NULL)
```

Arguments

- `id`: numeric for id of the club, defaults to authenticated club of the athlete
- `request`: chr string, must be "members", "activities" or NULL for club details

Details

Function is used internally within `get_club`

Value

A url string.

Examples

```r
url_clubs()
url_clubs(123, request = 'members')
```
url_gear

Set the url of the equipment item to get data

Description
Set the url of the equipment item to get data using an ID

Usage
url_gear(id)

Arguments
id string of gear id assigned by Strava

Details
used by other functions

Value
A character string of the gear URL used for API requests

url_segment

Set the url for the different segment requests

Description
Set the url for the different segment requests

Usage
url_segment(id = NULL, request = NULL)

Arguments
id numeric for id of the segment if request = "all_efforts" or "leaderboard",
or id of the athlete if request = "starred", or NULL if using request = "explore"
or "starred" of the authenticated user
request chr string, must be "starred", "all_efforts", "leaderboard", "explore" or NULL
for segment details

Details
Function is used internally within get_segment, get_starred, get_leaderboard, get_efforts_list,
and get_explore
**url_streams**

**Value**

A url string.

**Examples**

```
url_segment()
```

```
url_segment(id = 123, request = 'leaderboard')
```

---

**url_streams** *Set the url for stream requests*

**Description**

Set the url for stream requests

**Usage**

```
url_streams(id, request = "activities", types = list("latlng"))
```

**Arguments**

- `id`: numeric for id of the request
- `request`: chr string defining the stream type, must be "activities", "segment_efforts", "segments"
- `types`: list of chr strings with any combination of "time", "latlng", "distance", "altitude", "velocity_smooth", "heartrate", "cadence", "watts", "temp", "moving", or "grade_smooth"

**Details**

Function is used internally within get_streams. From the API documentation, 'streams' is the Strava term for the raw data associated with an activity.

**Value**

A url string.

**Examples**

```
url_streams(123)
```
Index

* notoken
  achievement_fun, 3
  athl_fun, 4
  athlind_fun, 3
  compile_seg_effort, 11
  get_dists, 20
  location_fun, 37
  monthly_fun, 38
  recent_fun, 41
  units_fun, 43

* token
  chk_nopolyline, 5
  compile_activities, 6
  compile_activity, 7
  compile_activity_streams, 8
  compile_club_activities, 9
  compile_seg_efforts, 12
  compile_segment, 10
  get_activity, 14
  get_activity_list, 15
  get_activity_streams, 16
  get_athlete, 17
  get_basic, 18
  get_club, 19
  get_efforts_list, 21
  get_elev_prof, 22
  get_explore, 24
  get_gear, 25
  get_heat_map, 26
  get_KOMs, 29
  get_laps, 30
  get_latlon, 30
  get_leaderboard, 31
  get_pages, 32
  get_segment, 33
  get_spdsplits, 34
  get_starred, 35
  get_streams, 36
  plot_spdsplits, 39
  ratelimit, 40
  strava_oauth, 42
  url_activities, 44
  url_clubs, 45
  url_segment, 46
  url_streams, 47

  achievement_fun, 3
  athl_fun, 3, 4, 37, 38, 41, 43
  athlind_fun, 3

  chk_nopolyline, 5
  compile_activities, 5, 6, 7, 15, 16, 23, 27, 28, 34, 40
  compile_activity, 7
  compile_activity_streams, 8
  compile_club_activities, 6, 9
  compile_seg_effort, 11, 12
  compile_seg_efforts, 11, 12
  compile_segment, 10, 34
  config, 12, 14–16, 18, 19, 21, 24, 25, 29–36, 40

  filter.actframe, 13

  geom_label_repel, 27
  GET, 41
  get_activity, 14
  get_activity_list, 6, 9, 14, 15, 16, 23, 27, 40
  get_activity_streams, 8, 16, 27, 36
  get_athlete, 17
  get_basic, 18
  get_club, 19, 45
  get_dists, 20, 23
  get_efforts_list, 11, 21, 46
  get_elev_prof, 5, 12, 20, 22
  get_explore, 24, 46
  get_gear, 25
  get_heat_map, 5, 26
INDEX

get_KOMs, 29
get_laps, 30
get_lation, 20, 28, 30
get_leaderboard, 31, 46
get_map, 28
get_pages, 32
get_segment, 10, 33, 46
get_spdsplits, 34
get_starred, 35, 46
get_streams, 8, 36, 47
google_elevation, 23, 27, 31

location_fun, 37

make_bbox, 27
monthly_fun, 38
mutate.actframe, 38

oauth2.0_token, 42

plot_spdsplits, 39

ratelimit, 40
recent_fun, 41

scale_fill_distiller, 27
seltime_fun, 41
strava_oauth, 12, 14–16, 18, 19, 21, 24, 25, 29–37, 40, 42

units_fun, 43
url_activities, 44
url_athlete, 44
url_clubs, 45
url_gear, 46
url_segment, 46
url_streams, 47