

# Package ‘rStrava’

January 12, 2026

**Type** Package

**Title** Access the 'Strava' API

**Version** 1.3.4

**Date** 2026-01-05

**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, ggspatial, googleway, htr,  
jsonlite, magrittr, maptiles, rvest, tidyr, tidyterra, XML,  
xml2, purrr, tibble

**Depends** R (>= 3.5.0)

**RoxygenNote** 7.3.3

**Encoding** UTF-8

**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](mailto:mbafs2012@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-01-12 17:10:09 UTC

## Contents

athlind_fun . . . . .	3
athl_fun . . . . .	3
chk_nopolyline . . . . .	4
compile_activities . . . . .	5

compile_activity . . . . .	6
compile_activity_streams . . . . .	7
compile_club_activities . . . . .	8
compile_segment . . . . .	9
compile_seg_effort . . . . .	10
compile_seg_efforts . . . . .	11
filter.actframe . . . . .	12
follow_fun . . . . .	13
get_activity . . . . .	13
get_activity_list . . . . .	14
get_activity_streams . . . . .	15
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 . . . . .	25
get_KOMs . . . . .	29
get_laps . . . . .	29
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 . . . . .	37
mutate.actframe . . . . .	38
plot_spdsplits . . . . .	39
recent_fun . . . . .	40
seltime_fun . . . . .	40
strava_oauth . . . . .	41
url_activities . . . . .	42
url_athlete . . . . .	43
url_clubs . . . . .	44
url_gear . . . . .	44
url_segment . . . . .	45
url_streams . . . . .	46

---

athlind_fun	<i>Get data for a single athlete</i>
-------------	--------------------------------------

---

**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, as character string

**Value**

A list with elements for the athlete's information.

---

athl_fun	<i>Get data for an athlete</i>
----------	--------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

athl\_num          numeric vector of athlete id(s) used by Strava, as character string

trace             logical indicating if output is returned to console

**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>. 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's information. The list elements are named using the athlete id numbers.

**Examples**

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

## Not run:
## multiple athletes
athl_fun(c('2837007', '2527465'))

## End(Not run)
```

---

chk_nopolyline	<i>Remove activities with no geographic data</i>
----------------	--

---

**Description**

Remove activities with no geographic data, usually manual entries

**Usage**

```
chk_nopolyline(act_data, ...)
```

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

**Arguments**

act_data	a data.frame returned by <a href="#">compile_activities</a>
...	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))  
my_acts <- get_activity_list(stoken)  
act_data <- compile_activities(my_acts)  
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 <a href="#">get_activity_list</a>
acts	numeric indicating which activities to compile starting with most recent, defaults to all
id	optional character 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 <- htrr::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)
```

---

compile_activity	<i>convert a single activity list into a dataframe</i>
------------------	--

---

**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 <a href="#">compile_activities</a> . 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

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

acts <- get_activity_list(token)

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

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

## Arguments

streams	a list containing details of the Strava streams of a single activity (output of <a href="#">get_streams</a> )
id	if not missing, the activity id of the stream (will be appended to the data.frame, if non-empty), as character vector

## 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

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

act_id <- '351217692'
streams <- get_streams(token, 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:
token <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))

club_acts <- get_activity_list(token, id = 13502, club = TRUE)

acts_data <- compile_club_activities(club_acts)

## End(Not run)
```



---

compile_segment	<i>Compile information on a segment</i>
-----------------	---

---

## 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

```
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

```
## 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

# get segment efforts
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

```
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

```
## 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	<i>Filter</i>
-----------------	---------------

---

### Description

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

### Usage

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

### Arguments

<code>.data</code>	an <code>actframe</code> object
<code>...</code>	Logical predicates defined in terms of the variables in <code>.data</code>

### Value

an `actframe` object

### Examples

```
## 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)
```

---

follow_fun	<i>Get athlete follow data</i>
------------	--------------------------------

---

**Description**

Get athlete follow data, used internally in [athl\\_fun](#)

**Usage**

```
follow_fun(prsd)
```

**Arguments**

prsd	parsed input list
------	-------------------

**Value**

A data frame of counts of followers and following for the athlete. An empty list is returned if none found.

---

get_activity	<i>Get detailed data of an activity</i>
--------------	---

---

**Description**

Get detailed data of an activity, including segment efforts

**Usage**

```
get_activity(id, stoken)
```

**Arguments**

id	character vector for id of the activity
stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> 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**

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

get_activity('75861631', token)

## End(Not run)
```

---

get_activity_list	<i>Get an activities list</i>
-------------------	-------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

token	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character vector 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 token 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

```
## 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 <code>config</code> object created using the <code>strava_oauth</code> function
acts	numeric indicating which activities to compile starting with most recent, defaults to all
id	optional character 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.
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" (default) or "time".

### Details

Each activity has a value for every column present across all activities, with NAs populating missing values.

For the `types` argument, the default is `type = NULL` which will retrieve all available stream types. The available stream types can be any of `time`, `latlng`, `distance`, `altitude`, `velocity_smooth`, `heartrate`, `cadence`, `watts`, `temp`, `moving`, or `grade_smooth`. To retrieve only a subset of the types, pass a list argument with the appropriate character strings to `type`, e.g., `type = list("time", "latlng", "distance")`.

Invalid HTTP requests (404 or 400 code) may sometimes occur for activities with incomplete data, e.g., stationary activities with no distance information. In such cases, changing the `'series_type'` and `'resolution'` arguments may be needed, e.g., `'series_type = "time"'` and `'resolution = "medium"'`.

### Value

A stream frame object (`strframe` that includes a data frame for the stream data along with the units

### Author(s)

Lorenzo Gaborini

### Examples

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

my_acts <- get_activity_list(stoken)

strms_data <- get_activity_streams(my_acts, stoken, acts = 1:2)

## End(Not run)
```



---

get_athlete	<i>Get basic data for an athlete</i>
-------------	--------------------------------------

---

## Description

Get basic athlete data for an athlete using an API request

## Usage

```
get_athlete(stoken, id = NULL)
```

## Arguments

stoken	A <code>config</code> object created using the <code>strava_oauth</code> function
id	string of athlete

## Details

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

## Value

A list of athlete information including athlete name, location, followers, etc. as described here: <https://strava.github.io/api/v3/athlete/>.

## 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_athlete(stoken, id = '2837007')  
  
## End(Not run)
```

---

get_basic	<i>Get basic Strava data</i>
-----------	------------------------------

---

### 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 <a href="#">config</a> object created using the <a href="#">strava_oauth</a> 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.

### 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 basic user info  
get_basic('https://www.strava.com/api/v3/athlete', stoken)  
  
## End(Not run)
```

---

get_club	<i>Get club data</i>
----------	----------------------

---

## Description

Get club data for a given request

## Usage

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

## Arguments

stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character vector 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.

## 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_club(stoken)  
  
## End(Not run)
```

---

`get_dists`*Get distance from longitude and latitude points*

---

**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` 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  
token <- httr::config(token = strava_oauth(app_name, app_client_id, app_secret, cache = TRUE))  
my_acts <- get_activity_list(token)  
  
# 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	<i>Get all the efforts in a segment if no queries are specified</i>
------------------	---

---

### Description

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

### Usage

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

### Arguments

stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character string for id of the segment
athlete_id	character string 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	<i>Create elevation profiles from activity data</i>
---------------	---

---

### 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", ...)
```

### Arguments

act_data	an activities list object returned by <a href="#">get_activity_list</a> or a data.frame returned by <a href="#">compile_activities</a>
...	arguments passed to or from other methods
acts	numeric value indicating which elements of act_data to plot, defaults to most recent
id	optional character 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 <a href="#">google_elevation</a> , 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 <a href="#">compile_activities</a>
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>

### 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)

## End(Not run)
```

---

`get_explore`*Explore segments within a bounded area*

---

### Description

Explore segments within a bounded area

### Usage

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

### Arguments

<code>stoken</code>	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
<code>bounds</code>	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
<code>activity_type</code>	chr string indicating activity type, "riding" or "running"
<code>max_cat</code>	numeric indicating the maximum climbing category
<code>min_cat</code>	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

```
## 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"  
get_explore(stoken, bnds)  
  
## End(Not run)
```



---

get_gear	<i>Get gear details from its identifier</i>
----------	---

---

**Description**

Get gear details from its identifier

**Usage**

```
get_gear(id, stoken)
```

**Arguments**

id	string, identifier of the equipment item
stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> 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_gear("2275365", stoken)  
  
## End(Not run)
```

---

get_heat_map	<i>Makes a heat map from your activity data</i>
--------------	---

---

**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,
  add_elev = FALSE,
  as_grad = FALSE,
  distlab = TRUE,
  distval = 0,
  size = 0.5,
  col = "red",
  expand = 10,
  maptype = "CartoDB.Positron",
  zoom = 14,
  units = "metric",
  ...
)

## S3 method for class 'actframe'
get_heat_map(
  act_data,
  key,
  alpha = NULL,
  add_elev = FALSE,
  as_grad = FALSE,
  distlab = TRUE,
  distval = 0,
  size = 0.5,
  col = "red",
  expand = 10,
  maptype = "CartoDB.Positron",
  zoom = 14,
  ...
)

## S3 method for class 'strframe'
get_heat_map(
  act_data,
  alpha = NULL,
  filltype = "elevation",
  distlab = TRUE,
  distval = 0,
  size = 0.5,
```

```

    col = "red",
    expand = 10,
    maptype = "CartoDB.Positron",
    zoom = 14,
    ...
)

```

## Arguments

act_data	an activities list object returned by <a href="#">get_activity_list</a> , an actframe returned by <a href="#">compile_activities</a> , or a strframe returned by <a href="#">get_activity_streams</a>
...	arguments passed to or from other methods
key	chr string of Google API key for elevation data, passed to <a href="#">google_elevation</a> 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 character 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
add_elev	logical indicating if elevation is shown 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
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 <a href="#">scale_fill_distiller</a> 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 provider for the basemap, see details
zoom	numeric indicating zoom factor for map tiles, higher numbers increase resolution
units	chr string indicating plot units as either metric or imperial, this has no effect if input data are already compiled with <a href="#">compile_activities</a>
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 `ggspatial` to produce the map and `ggplot2` to plot the route.

A Google API key is needed for the elevation data and must be included with function execution. The API key can be obtained following the instructions here: <https://developers.google.com/maps/documentation/elevation/#>

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.

The base map type is selected with the `maptype` argument. The `zoom` value specifies the resolution of the map. Use higher values to download map tiles with greater resolution, although this increases the download time. Acceptable options for `maptype` include "OpenStreetMap", "OpenStreetMap.DE", "OpenStreetMap.France", "OpenStreetMap.HOT", "OpenTopoMap", "Esri.WorldStreetMap", "Esri.DeLorme", "Esri.WorldTopoMap", "Esri.WorldImagery", "Esri.WorldTerrain", "Esri.WorldShadedRelief", "Esri.OceanBasemap", "Esri.NatGeoWorldMap", "Esri.WorldGrayCanvas", "CartoDB.Positron", "CartoDB.PositronNoLabels", "CartoDB.PositronOnlyLabels", "CartoDB.DarkMatter", "CartoDB.DarkMatterNoLabels", "CartoDB.DarkMatterOnlyLabels", "CartoDB.Voyager", "CartoDB.VoyagerNoLabels", or "CartoDB.VoyagerOnlyLabels".

## Value

A `ggplot` object showing a map with activity locations.

## Author(s)

Daniel Padfield, Marcus Beck

## Examples

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

# 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)

## End(Not run)
```

---

get_KOMs	<i>Get KOMs/QOMs/CRs of an athlete</i>
----------	--

---

**Description**

Get KOMs/QOMs/CRs of an athlete

**Usage**

```
get_KOMs(id, stoken)
```

**Arguments**

id	string of athlete id
stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> 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	<i>Retrieve the laps of an activity</i>
----------	---

---

**Description**

Retrieve the laps of an activity

**Usage**

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

**Arguments**

token            A [config](#) object created using the [strava\\_oauth](#) function  
 id                character for id of the activity with the laps to request

**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
token <- httr::config(token = strava_oauth(app_name, app_client_id,
app_secret, cache = TRUE))

get_laps(token, id = '351217692')

## End(Not run)
```

---

get_latlon	<i>get latitude and longitude from Google polyline</i>
------------	--

---

**Description**

get latitude and longitude from Google polyline

**Usage**

```
get_latlon(polyline, key)
```

**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

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

my_acts <- get_activity_list(token)
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	<i>Retrieve the leaderboard of a segment</i>
-----------------	--

---

## Description

Retrieve the leaderboard of a segment

## Usage

```
get_leaderboard(token, id, nleaders = 10, All = FALSE)
```

## Arguments

token	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character for id of the segment
nleaders	numeric for number of leaders to retrieve
All	logical to retrieve all of the list

## 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
token <- 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

```

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 <a href="#">config</a> object created using the <a href="#">strava_oauth</a> 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

### 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 basic user info
# returns 30 activities
get_pages('https://www.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 <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character 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.

**Value**

Data from an API request.

**See Also**

[compile\\_segment](#) for converting the list output to data.frame

## 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\_spdsplits

*Get speed splits in a dataframe*

---

## Description

Allows the return of speed splits of multiple rides.

## Usage

```
get_spdsplits(act_id, stoken, units = "metric")
```

## Arguments

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

## 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 <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character 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

```
## 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_starred(stoken)
```

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

---

get_streams	<i>Retrieve a Strava data stream for a single activity</i>
-------------	--

---

## Description

Retrieve a Strava data stream for a single activity. Internally called by [get\\_activity\\_streams](#).

## Usage

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

## Arguments

stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
id	character 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**

```
## 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	<i>Get athlete location</i>
--------------	-----------------------------

---

**Description**

Get athlete location, used internally in [athl\\_fun](#)

**Usage**

```
location_fun(prsd)
```

**Arguments**

prsd            parsed input list

**Value**

A character string of the athlete location

---

monthly_fun	<i>Get distance and time for current month</i>
-------------	--

---

**Description**

Get distance and time for current month, used internally in [athl\\_fun](#)

**Usage**

```
monthly_fun(prsd)
```

**Arguments**

prsd            parsed input list

**Value**

A data frame of the current monthly distance and time for the athlete. An empty list is returned if none found.

---

mutate.actframe	<i>Mutate</i>
-----------------	---------------

---

## Description

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

## Usage

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

## Arguments

<code>.data</code>	an <code>actframe</code> object
<code>...</code>	Name-value pairs of expressions. Use <code>NULL</code> to drop a variable.

## Value

an `actframe` object

## Examples

```
## 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	<i>Plot speed by splits</i>
----------------	-----------------------------

---

**Description**

Plot average speed by splits for a single activity

**Usage**

```
plot_spdsplits(act_data, ...)

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

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

**Arguments**

act_data	an activities list object returned by <a href="#">get_activity_list</a> or a data.frame returned by <a href="#">compile_activities</a>
...	arguments passed to other methods
stoken	A <a href="#">config</a> object created using the <a href="#">strava_oauth</a> function
acts	numeric indicating which activity to plot based on index in the activities list, defaults to most recent
id	optional character vector to specify the id(s) of the activity/activities to plot, 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**

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

# default
plot_spdsplits(my_acts, token, acts = 1)

## End(Not run)
```

---

recent_fun	<i>Get last three recent activities</i>
------------	---

---

**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	<i>Format before and after arguments for API query</i>
-------------	--

---

**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

**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	<i>Generata Strava API authentication token</i>
--------------	---

---

**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 = c("public", "read", "read_all", "profile:read_all", "profile:write",
    "activity:read", "activity:read_all", "activity:write"),
  cache = TRUE
)
```

**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 one of "public", "read", "read\_all",  
                      "profile:read\_all", "profile:write", "activity:read", "activity:read\_all" or "activ-  
                      ity:write"  
 cache               logical to cache the token, default is set

**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**

```
## Not run:
app_name <- 'myappname' # chosen by user
app_client_id <- 'myid' # an integer, assigned by Strava
app_secret <- 'xxxxxxx' # 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)
```

---

url_activities	<i>Set the url of activities for different activity lists</i>
----------------	---

---

**Description**

Set the url of activities for different activity lists

**Usage**

```
url_activities(id = NULL, club = FALSE)
```

**Arguments**

<code>id</code>	string for id of the activity or club if <code>club = TRUE</code>
<code>club</code>	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**

```
## Not run:  
# create authentication token  
# requires user created app name, id, and secret from Strava website  
token <- httr::config(token = strava_oauth(app_name, app_client_id,  
app_secret, cache = TRUE))  
  
url_activities('2837007')  
  
## End(Not run)
```

---

url_athlete	<i>Set the url of the athlete to get data</i>
-------------	---

---

**Description**

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

**Usage**

```
url_athlete(id = NULL)
```

**Arguments**

id	character 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



**Details**

used by other functions

**Value**

A character string of the gear URL used for API requests

---

url_segment	<i>Set the url for the different segment requests</i>
-------------	---

---

**Description**

Set the url for the different segment requests

**Usage**

```
url_segment(id = NULL, request = NULL)
```

**Arguments**

id	character 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](#)

**Value**

A url string.

**Examples**

```
url_segment()

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

---

url_streams	<i>Set the url for stream requests</i>
-------------	--

---

**Description**

Set the url for stream requests

**Usage**

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

**Arguments**

id	character 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

- [athl\\_fun](#), 3
- [athlind\\_fun](#), 3
- [compile\\_seg\\_effort](#), 10
- [follow\\_fun](#), 13
- [get\\_dists](#), 20
- [location\\_fun](#), 37
- [monthly\\_fun](#), 37
- [recent\\_fun](#), 40

## \* token

- [chk\\_nopolyline](#), 4
- [compile\\_activities](#), 5
- [compile\\_activity](#), 6
- [compile\\_activity\\_streams](#), 7
- [compile\\_club\\_activities](#), 8
- [compile\\_seg\\_efforts](#), 11
- [compile\\_segment](#), 9
- [get\\_activity](#), 13
- [get\\_activity\\_list](#), 14
- [get\\_activity\\_streams](#), 15
- [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](#), 25
- [get\\_KOMs](#), 29
- [get\\_laps](#), 29
- [get\\_latlon](#), 30
- [get\\_leaderboard](#), 31
- [get\\_pages](#), 32
- [get\\_segment](#), 33
- [get\\_spdsplits](#), 34
- [get\\_starred](#), 35
- [get\\_streams](#), 36
- [plot\\_spdsplits](#), 39
- [strava\\_oauth](#), 41

- [url\\_activities](#), 42
- [url\\_clubs](#), 44
- [url\\_segment](#), 45
- [url\\_streams](#), 46

- [athl\\_fun](#), 3, 13, 37, 40
- [athlind\\_fun](#), 3

- [chk\\_nopolyline](#), 4
- [compile\\_activities](#), 4, 5, 6, 14, 15, 22, 23, 27, 34, 39
- [compile\\_activity](#), 6
- [compile\\_activity\\_streams](#), 7
- [compile\\_club\\_activities](#), 6, 8
- [compile\\_seg\\_effort](#), 10, 11
- [compile\\_seg\\_efforts](#), 10, 11
- [compile\\_segment](#), 9, 33
- [config](#), 11, 13, 14, 16–19, 21, 24, 25, 29–36, 39

- [filter.actframe](#), 12
- [follow\\_fun](#), 13

- [get\\_activity](#), 13
- [get\\_activity\\_list](#), 5, 8, 13, 14, 15, 22, 27, 39
- [get\\_activity\\_streams](#), 7, 15, 27, 36
- [get\\_athlete](#), 17
- [get\\_basic](#), 18
- [get\\_club](#), 19, 44
- [get\\_dists](#), 20, 23
- [get\\_efforts\\_list](#), 10, 21, 45
- [get\\_elev\\_prof](#), 4, 11, 20, 22
- [get\\_explore](#), 24, 45
- [get\\_gear](#), 25
- [get\\_heat\\_map](#), 4, 25
- [get\\_KOMs](#), 29
- [get\\_laps](#), 29
- [get\\_latlon](#), 20, 28, 30
- [get\\_leaderboard](#), 31, 45

get\_pages, 32  
get\_segment, 9, 33, 45  
get\_spdsplits, 34  
get\_starred, 35, 45  
get\_streams, 7, 36, 46  
ggplot, 28  
google\_elevation, 23, 27, 30  
  
location\_fun, 37  
  
monthly\_fun, 37  
mutate.actframe, 38  
  
oauth2.0\_token, 42  
  
plot\_spdsplits, 39  
  
recent\_fun, 40  
  
scale\_fill\_distiller, 27  
seltime\_fun, 40  
strava\_oauth, 11, 13, 14, 16–19, 21, 24, 25,  
29–36, 39, 41  
  
url\_activities, 42  
url\_athlete, 43  
url\_clubs, 44  
url\_gear, 44  
url\_segment, 45  
url\_streams, 46