Package ‘CSGo’

October 12, 2022

**Title**  Collecting Counter Strike Global Offensive Data

**Version**  0.6.7

**Description**  An implementation of calls designed to collect and organize in an easy way the data from the Steam API specifically for the Counter-Strike Global Offensive Game (CS Go) <https://developer.valvesoftware.com/wiki/Steam_Web_API>.

**License**  MIT + file LICENSE

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.1.1

**URL**  https://github.com/adsoncostanzifilho/CSGo

**BugReports**  https://github.com/adsoncostanzifilho/CSGo/issues

**Imports**  fuzzyjoin, purrr, httr, stringr, jsonlite, magrittr, dplyr, extrafont, ggplot2, future, furrr

**Depends**  R (>= 3.5.0)

**Suggests**  knitr, rmarkdown

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Author**  Adson Costanzi [aut, cre] (<https://orcid.org/0000-0002-5210-2952>), Rodrigo Fontoura [aut] (<https://orcid.org/0000-0002-8156-3424>)

**Maintainer**  Adson Costanzi <adsoncostanzi32@gmail.com>

**Repository**  CRAN

**Date/Publication**  2021-05-07 18:50:02 UTC

**R topics documented:**

- csgo_api_ach ............................................................... 2
- csgo_api_friend .......................................................... 3
- csgo_api_profile ......................................................... 3
- csgo_api_stats ............................................................ 4
**Description**

This function will return all the CS Go Achievements of the user_id (input).

**Usage**

csgo_api_ach(api_key, user_id)

**Arguments**

- **api_key**: string with the key provided by the steam API.
  
  PS: If you don’t have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

- **user_id**: string with the steam user ID.
  
  Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '7656119826364899'.
  
  PS: The user should have a public status.

**Value**

data frame with all the CS Go achievements of the user ID.

**Examples**

```r
## Not run:
## It is necessary to fill the "api_key" parameter to run the example

df_ach <- csgo_api_ach(api_key = 'XXX', user_id = '7656119826364899')

## End(Not run)
```
csgo_api_friend

**CS Go Friends**

**Description**
This function will return all the CS Go friends of the user_id (input).

**Usage**
csgo_api_friend(api_key, user_id)

**Arguments**

- **api_key**
  string with the key provided by the steam API.
  PS: If you don't have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

- **user_id**
  string with the steam user ID.
  Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '76561198263364899'.
  PS: The user should have a public status.

**Value**
data frame with all the CS Go friends of the user ID.

**Examples**

```r
## Not run:
## It is necessary to fill the "api_key" parameter to run the example
df_friend <- csgo_api_friend(api_key = 'XXX', user_id = '76561198263364899')
## End(Not run)
```

csgo_api_profile

**CS Go User Profile**

**Description**
This function will return the CS Go Profile of the user_id (input).

**Usage**
csgo_api_profile(api_key, user_id, name = FALSE)
csgo_api_stats

Arguments

api_key  
string with the key provided by the steam API.
PS: If you don’t have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

user_id  
string OR list with the steam user ID.
Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '76561198263364899'.
PS: The user should have a public status.

name  
logical: if the user_id input is a name change it for TRUE. ex: 'kevinarndt'.
PS: The query by name DOES NOT ALLOW a list of user_id.

Value

data frame with all the CS Go friends of the user ID.

Examples

## Not run:
## It is necessary to fill the "api_key" parameter to run the example

df_profile <- csgo_api_profile(api_key = 'XXX', user_id = '76561198263364899')

df_profile <- csgo_api_profile(
  api_key = 'XXX',
  user_id = list('76561198263364899','76561197996007619')
)

df_profile <- csgo_api_profile(api_key = 'XXX', user_id = 'kevinarndt', name = TRUE)

## End(Not run)

---

**csgo_api_stats**  
CS Go Statistics

**Description**

This function will return all the CS Go Statistics of the user_id (input).

**Usage**

```
csgo_api_stats(api_key, user_id)
```
get_stats_friends

Arguments

api_key 
string with the key provided by the steam API.
PS: If you don’t have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

user_id 
string with the steam user ID.
Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '7656119826364899'.
PS: The user should have a public status.

Value

data frame with all the CS Go statistics of the user ID.

Examples

```
## Not run:
## It is necessary to fill the "api_key" parameter to run the example

df_stats <- csgo_api_stats(api_key = 'XXX', user_id = '7656119826364899')

## End(Not run)
```

get_stats_friends 
Get the Friends Statistics

Description

This function will return the complete CS Go Statistics for all public friends of the user_id (input).

Usage

```
get_stats_friends(api_key, user_id, n_return = "all")
```

Arguments

api_key 
string with the key provided by the steam API.
PS: If you don’t have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

user_id 
string with the steam user ID.
Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '7656119826364899'.
PS: The user should have a public status.

n_return 
numeric indicating the number of friends to return, to return all use n_return = "all" (the default is "all").
get_stats_user

Value

- a list of two data frames
  - friends_stats: data frame with all the CS Go statistics of all public friends of the user ID.
  - friends: data frame with all the CS Go friends of the user ID (public and non public).

Examples

```r
## Not run:
## It is necessary to fill the "api_key" parameter to run the example

# set the "plan" to collect the data in parallel!!!!
future::plan(future::multisession, workers = parallel::detectCores())

fr_list <- get_stats_friends(api_key = "XXX", user_id = "76561198263364899")
fr_list$friends_stats
fr_list$friends

## End(Not run)
```

get_stats_user  

Get the User Statistics

Description

This function will return the complete CS Go Statistics of the user_id (input).

Usage

```r
get_stats_user(api_key, user_id)
```

Arguments

- **api_key**: string with the key provided by the steam API.
  PS: If you don’t have a API key yet run vignette("auth", package = "CSGo") and follow the presented steps.

- **user_id**: string with the steam user ID.
  Steam ID is the NUMBER OR NAME at the end of your steam profile URL. ex: '76561198263364899'.
  PS: The user should have a public status.

Details

Similar to the csgo_api_stats function but it will return a clean data frame with category and description of each statistic.

Value

data frame with all the CS Go statistics (divided in categories and subcategories) of the user ID.
## Examples

```r
## Not run:
## It is necessary to fill the "api_key" parameter to run the example
df <- get_stats_user(api_key = 'XXX', user_id = '7656119826336489')
## End(Not run)
```

---

### map_pictures

**Maps Images**

**Description**

A dataset containing the pictures of each map.

**Usage**

```r
map_pictures
```

**Format**

A data frame with 34 rows and 2 variables:

- **map_name** Name of the map.
- **map_photo** The image address.

**Source**

Created by the author.

---

### scale_color_csgo

**CSGo color palette - color**

**Description**

A color palette (color) to be used with ggplot2

**Usage**

```r
scale_color_csgo(discrete = TRUE, ...)
```

**Arguments**

- **discrete** logical: if TRUE it will generate a discrete pallet otherwise a continuous palette
- **...** all available options of the `discrete_scale` function or `scale_color_gradientn` both from ggplot2
**Value**

scale_color object

**Examples**

```r
## Not run:
library(CSGo)
library(ggplot2)
library(dplyr)
library(showtext)

## Loading Google fonts (https://fonts.google.com/)
font_add_google("Quantico", "quantico")

df %>%
top_n(n = 10, wt = kills) %>%
ggplot(aes(x = name_match, size = shots)) +
geom_point(aes(y = kills_efficiency, color = "Kills Efficiency")) +
geom_point(aes(y = hits_efficiency, color = "Hits Efficiency")) +
geom_point(aes(y = hits_to_kill, color = "Hits to Kill")) +
ggtitle("Weapon Efficiency") +
ylab("Efficiency (%)") +
xlab("") +
labs(color = "Efficiency Type", size = "Shots") +
theme_csgo(
  text = element_text(family = "quantico"),
  panel.grid.major.x = element_line(size = .1, color = "black", linetype = 2)
) +
scale_color_csgo()

## End(Not run)
```

---

**scale_fill_csgo**

CSGo color palette - fill

**Description**

A color palette (fill) to be used with ggplot2

**Usage**

```r
scale_fill_csgo(discrete = TRUE, ...)
```

**Arguments**

- **discrete** logical: if TRUE it will generate a discrete palette otherwise a continuous palette
- ... all available options of the discrete_scale function or scale_fill_gradientn both from ggplot2
## Not run:
library(CSGo)
library(ggplot2)
library(dplyr)
library(showtext)

## Loading Google fonts (https://fonts.google.com/)
font_add_google("Quantico", "quantico")

df %>%
top_n(n = 10, wt = value) %>%
ggplot(aes(x = name_match, y = value, fill = name_match)) +
geom_col() +
ggtitle("KILLS BY WEAPON") +
ylab("Number of Kills") +
xlab("") +
labs(fill = "Weapon Name") +
theme_csgo(text = element_text(family = "quantico")) +
scale_fill_csgo()

## End(Not run)

---

**Categories and Descriptions of the Statistics Data**

### Description

A dataset containing the categories, descriptions and types of the statistics data pulled from the csgo_api_stats.

### Usage

`support`

### Format

A data frame with 133 rows and 4 variables:

- **name_match**: Name to match with the name statistics data.
- **category**: Category name of the statistic.
- **desc**: Statistic description.
- **type**: Statistic type. ...
theme_csgo

CSGo theme

Description

A CSGo theme to be used with ggplot2

Usage

tHEME_CSGO(...)

Arguments

... all available options of the theme function from ggplot2

Value

theme object

Examples

## Not run:
library(CSGo)
library(ggplot2)
library(dplyr)
library(showtext)

## Loading Google fonts (https://fonts.google.com/)
font_add_google("Quantico", "quantico")

df %>%
top_n(n = 10, wt = value) %>%
ggplot(aes(x = name_match, y = value, fill = name_match)) +
geom_col() +
ggtitle("KILLS BY WEAPON") +
ylab("Number of Kills") +
xlab("") +
labs(fill = "Weapon Name") +
theme_csgo(text = element_text(family = "quantico"))

## End(Not run)
weapon_pictures

<table>
<thead>
<tr>
<th>weapon_pictures</th>
<th>Weapon Images</th>
</tr>
</thead>
</table>

**Description**

A dataset containing the pictures of each map.

**Usage**

weapon_pictures

**Format**

A data frame with 34 rows and 2 variables:

- **weapon_name** Name of the weapon.
- **weapon_photo** The image address. ...

**Source**

Created by the author.
Index

* datasets
  map_pictures, 7
  support, 9
  weapon_pictures, 11

csgo_api_ach, 2

csgo_api_friend, 3
csgo_api_profile, 3
csgo_api_stats, 4

get_stats_friends, 5
get_stats_user, 6

map_pictures, 7

scale_color_csgo, 7
scale_fill_csgo, 8
support, 9

theme_csgo, 10

weapon_pictures, 11