Package ‘BAwiR’

February 5, 2020

Type Package
Title Analysis of Basketball Data
Version 1.2.1
Date 2020-02-04
Author Guillermo Vinue
Maintainer Guillermo Vinue <Guillermo.Vinue@uv.es>
Description Collection of tools to work with basketball data. Functions available are related to friendly web scraping and visualization. Data were obtained from <http://www.euroleague.net/>, <http://www.eurocupbasketball.com/> and <http://www.acb.com>, following the instructions of their respective robots.txt files, when available. Tools for visualization include a population pyramid, 2D plots, circular plots of players' percentiles, plots of players' monthly/yearly stats, team heatmaps, team shooting plots, team four factors plots, crosstables with the results of regular season games and maps of nationalities. Please see Vinue (2020) <doi:10.1089/big.2018.0124>.
License GPL (>= 2)
URL https://www.R-project.org, https://www.uv.es/vivigui,
https://www.uv.es/vivigui/AppEuroACB.html
Depends R (>= 3.4.0)
Imports Anthropometry, plyr, dplyr, ggplot2, ggthemes, grid, hrbrthemes, httr, lubridate, magrittr, purrr, reshape2, rvest, rworldmap, scales, stringi, stringr, tidyr, xml2
Suggests knitr
VignetteBuilder knitr
LazyData true
RoxygenNote 6.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2020-02-05 14:00:03 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAwiR-package</td>
<td>3</td>
</tr>
<tr>
<td>acb_games_1718</td>
<td>4</td>
</tr>
<tr>
<td>acb_players_1718</td>
<td>5</td>
</tr>
<tr>
<td>capit_two_words</td>
<td>5</td>
</tr>
<tr>
<td>do_add_adv_stats</td>
<td>6</td>
</tr>
<tr>
<td>do_EPS</td>
<td>7</td>
</tr>
<tr>
<td>do_four_factors_df</td>
<td>8</td>
</tr>
<tr>
<td>do_join_games_bio</td>
<td>9</td>
</tr>
<tr>
<td>do_map_nats</td>
<td>10</td>
</tr>
<tr>
<td>do_OE</td>
<td>10</td>
</tr>
<tr>
<td>do_scraping_games</td>
<td>11</td>
</tr>
<tr>
<td>do_scraping_rosters</td>
<td>13</td>
</tr>
<tr>
<td>do_stats</td>
<td>14</td>
</tr>
<tr>
<td>do_stats_teams</td>
<td>15</td>
</tr>
<tr>
<td>eurocup_games_1718</td>
<td>16</td>
</tr>
<tr>
<td>eurocup_players_1718</td>
<td>16</td>
</tr>
<tr>
<td>euroleague_games_1718</td>
<td>17</td>
</tr>
<tr>
<td>euroleague_players_1718</td>
<td>17</td>
</tr>
<tr>
<td>get_barplot_monthly_stats</td>
<td>18</td>
</tr>
<tr>
<td>get_bubble_plot</td>
<td>19</td>
</tr>
<tr>
<td>get_four_factors_plot</td>
<td>21</td>
</tr>
<tr>
<td>get_games_rosters</td>
<td>22</td>
</tr>
<tr>
<td>get_heatmap_bb</td>
<td>25</td>
</tr>
<tr>
<td>get_map_nats</td>
<td>26</td>
</tr>
<tr>
<td>get_pop_pyramid</td>
<td>27</td>
</tr>
<tr>
<td>get_shooting_plot</td>
<td>28</td>
</tr>
<tr>
<td>get_similar_players</td>
<td>29</td>
</tr>
<tr>
<td>get_similar_teams</td>
<td>30</td>
</tr>
<tr>
<td>get_stats_seasons</td>
<td>31</td>
</tr>
<tr>
<td>get_table_results</td>
<td>32</td>
</tr>
<tr>
<td>join_players_bio_age_acb</td>
<td>33</td>
</tr>
<tr>
<td>join_players_bio_age_euro</td>
<td>34</td>
</tr>
<tr>
<td>scraping_games_acb</td>
<td>35</td>
</tr>
<tr>
<td>scraping_games_euro</td>
<td>36</td>
</tr>
<tr>
<td>scraping_rosters_acb</td>
<td>38</td>
</tr>
<tr>
<td>scraping_rosters_euro</td>
<td>39</td>
</tr>
</tbody>
</table>

**Index** 41
Analysis of Basketball Data

Description

Collection of tools to work with basketball data. Functions available are related to friendly web scraping and visualization. Data were obtained from <http://www.euroleague.net/>, <http://www.eurocupbasketball.com/> and <http://www.acb.com>, following the instructions of their respective robots.txt files, when available. Tools for visualization include a population pyramid, 2D plots, circular plots of players’ percentiles, plots of players’ monthly/yearly stats, team heatmaps, team shooting plots, team four factors plots, cross-tables with the results of regular season games and maps of nationalities. Please see Vinue (2020) <doi:10.1089/big.2018.0124>.

Details

Package: BAwiR
Type: Package
Version: 1.2.1
Date: 2020-02-04
License: GPL-2
LazyLoad: yes
LazyData: yes

acb_players_1718: ACB players 2017-2018.
capit_two_words: Capitalize two-word strings.
do_add_adv_stats: Advanced statistics.
do_EPS: Efficient Points Scored (EPS).
do_four_factors_df: Four factors data frame.
do_join_games_bio: Join games and players’ info.
do_map_nats: Data frame for the nationalities map.
do_scraping_games: Player game finder data.
do_scraping_rosters: Players profile data.
do_stats: Accumulated or average statistics.
do_stats_teams: Accumulated and average statistics for teams.
get_barplot_monthly_stats: Barplots with monthly stats.
get_bubble_plot: Basketball bubble plot.
get_four_factors_plot: Four factors plot.
get_games_rosters: Get all games and rosters.
get_heatmap_bb: Basketball heatmap.
get_map_nats: Nationalities map.
get_pop_pyramid: ACB population pyramid.
get_shooting_plot: Shooting plot.
get_similar_players: Similar players to archetypoids.
get_similar_teams: Similar teams to archetypoids.
get_stats_seasons: Season-by-season stats.
get_table_results: League cross table.
join_players_bio_age_acb: Join ACB games and players’ info.
join_players_bio_age_euro: Join Euroleague and Eurocup games and players’ info.
scraping_games_acb: ACB player game finder data.
scraping_games_euro: Euroleague and Eurocup player game finder data.
scraping_rosters_acb: ACB players’ profile.
scraping_rosters_euro: Euroleague and Eurocup players’ profile.

Author(s)
Guillermo Vinue <Guillermo.Vinue@uv.es>

References

Description
Games of the first seventeen days of the ACB 2017-2018 season.

Usage
acb_games_1718

Format
Data frame with 3939 rows and 38 columns.

Source
http://www.acb.com/
acb_players_1718  ACB players 2017-2018

Description

Players corresponding to the games of the first seventeen days of the ACB 2017-2018 season.

Usage

acb_players_1718

Format

Data frame with 255 rows and 7 columns.

Source

http://www.acb.com/

capit_two_words  Capitalize two-word strings

Description

Ancillary function to capitalize the first letter of both words in a two-word string. This can be used for example to capitalize the teams names for the plots title.

Usage

capit_two_words(two_word_string)

Arguments

two_word_string

Two-word string.

Value

Vector with the two words capitalized.

Author(s)

Guillermo Vinue

Examples

capit_two_words("valencia basket")
do_add_adv_stats

---

| do_add_adv_stats | Advanced statistics |

**Description**

This function adds to the whole data frame the advanced statistics for every player in every game.

**Usage**

do_add_adv_stats(df)

**Arguments**

df  
Data frame with the games and the players info.

**Details**

The advanced statistics computed are as follows:

- GameSc: Game Score.
- PIE: Player Impact Estimate.
- EFGPerc: Effective Field Goal Percentage.
- ThreeRate: Three points attempted regarding the total field goals attempted.
- FRate: Free Throws made regarding the total field goals attempted.
- STL_TOV: Steal to Turnover Ratio.
- AST_TOV: Assist to Turnover Ratio.
- PPS: Points Per Shot.
- OE: Offensive Efficiency.
- EPS: Efficient Points Scored.

The detailed definition of some of these stats can be found at [https://www.basketball-reference.com/about/glossary.html](https://www.basketball-reference.com/about/glossary.html) and [https://stats.nba.com/help/glossary/](https://stats.nba.com/help/glossary/).

**Value**

Data frame.

**Author(s)**

Guillermo Vinue

**See Also**

do_OE, do_EPS
\texttt{do\_EPS}

\textbf{Examples}

\begin{verbatim}
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
\end{verbatim}

\begin{verbatim}
do\_EPS(df1)[1]
\end{verbatim}

\textbf{do\_EPS} \hspace{1cm} \textit{Efficient Points Scored (EPS)}

\textbf{Description}

A limitation of \texttt{do\_OE} is that it doesn’t rely on the quantity of the player’s offense production, that’s to say, whether the player provides a lot of offense or not. In addition, it does not give credit for free-throws. An extension of \texttt{do\_OE} has been defined: the Efficient Points Scored (EPS), which is the result of the product of OE and points scored. Points scored counts free-throws, two-point and three-point field goals. A factor $F$ is also added to put the adjusted total points on a points scored scale. With the factor $F$, the sum of the EPS scores for all players in a given season is equal to the sum of the league total points scored in that season.

\textbf{Usage}

\begin{verbatim}
do\_EPS(df)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{df} \hspace{1cm} Data frame with the games and the players info.
\end{itemize}

\textbf{Value}

EPS values.

\textbf{Author(s)}

Guillermo Vinue

\textbf{References}


\textbf{See Also}

\texttt{do\_OE, do\_add\_adv\_stats}

\textbf{Examples}

\begin{verbatim}
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
do\_EPS(df1)[1]
\end{verbatim}
do_four_factors_df

Description
This function computes team’s offense and defense four factors. The four factors are Effective Field Goal Percentage (EFGP), Turnover Percentage (TOVP), Offensive Rebound Percentage (ORBP) and Free Throws Rate (FTRate). They are well defined at [http://www.rawbw.com/~deano/articles/20040601_roboscout.htm](http://www.rawbw.com/~deano/articles/20040601_roboscout.htm) and [https://www.basketball-reference.com/about/factors.html](https://www.basketball-reference.com/about/factors.html). As a summary, EFGP is a measure of shooting efficiency; TOVP is the percentage of possessions where the team missed the ball, see [http://www.nba.com/thunder/news/stats101.html](http://www.nba.com/thunder/news/stats101.html) to read about the 0.44 coefficient; ORBP measures how many rebounds were offensive from the total of available rebounds; Finally, FTRate is a measure of both how often a team gets to the line and how often they make them.

Usage

do_four_factors_df(df_games, teams)

Arguments

df_games      Data frame with the games, players info, advanced stats and eventually recoded teams names.
teams         Teams names.

Details
Instead of defining the Offensive and Defensive Rebound Percentage as mentioned in the previous links, I have computed just the Offensive Rebound Percentage for the team and for its rivals. This makes easier to have four facets, one per factor, in the ggplot.

In order to establish the team rankings, we have to consider these facts: In defense (accumulated statistics of the opponent teams to the team of interest), the best team in each factor is the one that allows the smallest EFGP, the biggest TOVP, the smallest ORBP and the smallest FTRate, respectively.

In offense (accumulated statistics of the team of interest), the best team in each factor is the one that has the biggest EFGP, the smallest TOVP, the biggest ORBP and the biggest FTRate, respectively.

Value
A list with two data frames, df_rank and df_no_rank. Both have the same columns:

- Team: Team name.
- Type: Either Defense or Offense.
- EFGP, ORBP, TOVP and FTRate.

The df_rank data frame contains the team ranking label for each statistic between parentheses. Therefore, df_no_rank is used to create the ggplot with the numerical values and df_rank is used to add the ranking labels.
do_join_games_bio

Author(s)
Guillermo Vinue

See Also
get_four_factors_plot

Examples

```r
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df_four_factors <- do_four_factors_df(df1, "Valencia")
```

do_join_games_bio  Join games and players' info

Description
This function calls the needed ancillary functions to join the games played by all the players in the
desired competition (currently ACB, Euroleague and Eurocup) with their personal details.

Usage

```r
do_join_games_bio(competition, df_games, df_rosters)
```

Arguments

- competition: String. Options are "ACB", "Euroleague" and "Eurocup".
- df_games: Data frame with the games.
- df_rosters: Data frame with the biography of the roster players.

Value
Data frame.

Author(s)
Guillermo Vinue

See Also
join_players_bio_age_acb, join_players_bio_age_euro

Examples

```r
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
```
do_map_nats  

Data frame for the nationalities map

Description
This function prepares the data frame with the nationalities to be mapped with `get_map_nats`. It is used inside it.

Usage
```
do_map_nats(df_stats)
```

Arguments
- `df_stats`: Data frame with the statistics and the corrected nationalities.

Value
List with the following elements:
- `df_all`: Data frame with each country, its latitudes and longitudes and whether it must be coloured or not (depending on if there are players from that country).
- `countr_num`: Vector with the countries from where there are players and the number of them.
- `leng`: Number of countries in the world.

Author(s)
Guillermo Vinue

See Also
- `get_map_nats`

---

do_OE  

Offensive Efficiency (OE)

Description
Offensive Efficiency (OE) is a measure to evaluate the quality of offense produced. OE counts the total number of successful offensive possessions the player was involved in, regarding the player’s total number of potential ends of possession.

This measure is used in the definition of `do_EPS`.

Usage
```
do_OE(df)
```
Arguments

df Data frame with the games and the players info.

Value

OE values.

Note

When either both the numerator and denominator of the OE expression are 0 or just the denominator is 0, the function returns a 0.

Author(s)

Guillermo Vinue

References


See Also

do_EPS, do_add_adv_stats

Examples

df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
# Players with OE = 0:
# df1[55, c("Player.x", "FG", "AST", "FGA", "ORB", "TOV")]
# Player.x FG AST FGA ORB TOV
# Triguero, J. 0 0 0 0 0
# OE can be greater than 1, for example:
# df1[17, c("Player.x", "FG", "AST", "FGA", "ORB", "TOV")]
# Player.x FG AST FGA ORB TOV
# Diagne, Moussa 3 0 3 1 0
do_OE(df1[1,])

do_scraping_games Player game finder data

Description

This function calls the needed ancillary functions to scrape the player game finder data for the desired competition (currently, ACB, Euroleague and Eurocup).
do_scraping_games

Usage

do_scraping_games(competition, type_league, nums, year, verbose, accents, r_user)

Arguments

competition String. Options are "ACB", "Euroleague" and "Eurocup".
type_league String. If competition is ACB, to scrape ACB league games ("ACB"), Copa del Rey games ("CREY") or Supercopa games ("SCOPA").
ums Numbers corresponding to the website from which scraping.
year If competition is either Euroleague or Eurocup, the year when the season starts is needed. 2017 refers to 2017-2018 and so on.
verbose Should R report information on progress? Default TRUE.
accents If competition is ACB, should we keep the Spanish accents? The recommended option is to remove them, so default FALSE.
r_user Email to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Value

A data frame with the player game finder data for the competition selected.

Author(s)

Guillermo Vinue

See Also

scraping_games_acb, scraping_games_euro

Examples

## Not run:
# Not needed to scrape every time the package is checked, built and installed.
dfl <- do_scraping_games(competition = "ACB", type_league = "ACB", nums = 62001,
                          year = "2017-2018", verbose = TRUE, accents = FALSE,
                          r_user = "guillermo.vinue@uv.es")

dfl_eur <- do_scraping_games(competition = "Euroleague", nums = 1,
                           year = "2017", verbose = TRUE,
                           r_user = "guillermo.vinue@uv.es")

## End(Not run)
do_scraping_rosters   Players profile data

Description

This function calls the needed ancillary functions to scrape the players’ profile data for the desired competition (currently, ACB, Euroleague and Eurocup).

Usage

do_scraping_rosters(competition, pcode, verbose, accents, year, r_user)

Arguments

- **competition**: String. Options are "ACB", "Euroleague" and "Eurocup".
- **pcode**: Code corresponding to the player’s website to scrape.
- **verbose**: Should R report information on progress? Default TRUE.
- **accents**: If competition is ACB, should we keep the Spanish accents? The recommended option is to remove them, so default FALSE.
- **year**: If competition is either Euroleague or Eurocup, the year when the season starts is needed. 2017 refers to 2017-2018 and so on.
- **r_user**: Email to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Value

A data frame with the players’ information.

Author(s)

Guillermo Vinue

See Also

scraping_games_acb, scraping_rosters_euro

Examples

```r
## Not run:
# Not needed to scrape every time the package is checked, built and installed.
df_bio <- do_scraping_rosters(competition = "ACB", pcode = "56C",
verbatim = TRUE, accents = FALSE,
r_user = "guillermo.vinue@uv.es")

df_bio_eur <- do_scraping_rosters(competition = "Euroleague", pcode = "007969",
year = "2017", verbose = TRUE,
```
do_stats

Accumulated or average statistics

Description
This function computes either the total or the average statistics.

Usage

\[
do_{\text{stats}}(df_{\text{games}}, \text{type}_{\text{stats}} = \text{"Total"}, \text{season}, \text{competition}, \text{type}_{\text{season}})\]

Arguments

- \(df_{\text{games}}\): Data frame with the games, players info, advanced stats and eventually recoded teams names.
- \(\text{type}_{\text{stats}}\): String. In English, the options are "Total" and "Average" and in Spanish, the options are "Totales" and "Promedio".
- \(\text{season}\): String indicating the season, for example, 2017-2018.
- \(\text{competition}\): String. Options are "ACB", "Euroleague" and "Eurocup".
- \(\text{type}_{\text{season}}\): String with the round of competition, for example regular season or playoffs and so on.

Value
Data frame.

Author(s)
Guillermo Vinue

Examples

\[
\begin{align*}
\text{compet} & \leftarrow \text{"ACB"} \\
\text{df} & \leftarrow \text{do}_\text{join}(_\text{games}_\text{bio}(\text{compet, acb_games}_1718, \text{acb_players}_1718)} \\
\text{df1} & \leftarrow \text{do}_\text{add}_\text{adv}_\text{stats}(\text{df}) \\
\text{df2} & \leftarrow \text{do}_\text{stats}(\text{df1, "Total", "2017-2018", compet, "Regular Season"})
\end{align*}
\]
**do_stats_teams**  

*Accumulated and average statistics for teams*

---

**Description**

This function computes the total and average statistics for every team.

**Usage**

```r
do_stats_teams(df_games, season, competition, type_season)
```

**Arguments**

- `df_games`: Data frame with the games, players info, advanced stats and eventually recoded teams names.
- `season`: String indicating the season, for example, 2017-2018.
- `competition`: String. Options are "ACB", "Euroleague" and "Eurocup".
- `type_season`: String with the round of competition, for example regular season or playoffs and so on.

**Value**

A list with two elements:

- `df_team_total`: Data frame with the total statistics for every team.
- `df_team_mean`: Data frame with the average statistics for every team.

**Author(s)**

Guillermo Vinue

**Examples**

```r
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df$Compet <- compet
df_teams <- do_stats_teams(df, "2017-2018", "ACB", "Regular Season")
# Total statistics:
#df_teams$df_team_total
# Average statistics:
#df_teams$df_team_mean
```
Eurocup games 2017-2018

Description
Games of the ten days of regular season and the first three days of top 16 of the Eurocup 2017-2018 season.

Usage
eurocup_games_1718

Format
Data frame with 3604 rows and 38 columns.

Source
http://www.eurocupbasketball.com/

Eurocup players 2017-2018

Description
Players corresponding to the games of the ten days of regular season and the first three days of top 16 of the Eurocup 2017-2018 season.

Usage
eurocup_players_1718

Format
Data frame with 351 rows and 7 columns.

Source
http://www.eurocupbasketball.com/
**euroleague_games_1718**  
*Euroleague games 2017-2018*

**Description**
Games of the first nineteen days of the Euroleague 2017-2018 season.

**Usage**
euroleague_games_1718

**Format**
Data frame with 3932 rows and 38 columns.

**Source**
http://www.euroleague.net

---

**euroleague_players_1718**  
*Euroleague players 2017-2018*

**Description**
Players corresponding to the games of the first nineteen days of the Euroleague 2017-2018 season.

**Usage**
euroleague_players_1718

**Format**
Data frame with 245 rows and 7 columns.

**Source**
http://www.euroleague.net
get_barplot_monthly_stats

Barplots with monthly stats

Description
In all the available basketball websites, the stats are presented for the whole number of games played. This function represents a barplot with the players’ stats for each month, which is very useful to analyse the players’ evolution.

Usage
get_barplot_monthly_stats(df_stats, title, size_text = 2.5)

Arguments
- df_stats: Data frame with the statistics.
- title: Plot title.
- size_text: Label size for each bar. Default 2.5.

Value
Graphical device.

Author(s)
Guillermo Vinue

See Also
capit_two_words

Examples
```r
# Not run:
library(dplyr)
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)

months <- c(df %>% distinct(Month))$Month
months_order <- c("September", "October", "November", "December",
"January", "February", "March", "April", "May", "June")
months_plot <- match(months_order, months)
months_plot1 <- months_plot[!is.na(months_plot)]
months_plot2 <- months[months_plot1]

df3_m <- df1 %>%
```
```r
filter(Team == "Real_Madrid",
    Player.x == "Doncic, Luka") %>%
group_by(Month) %>%
do(do_stats(., "Average", "2017-2018", "ACB", "Regular Season")) %>%
ungroup() %>%
mutate(Month = factor(Month, levels = months_plot2)) %>%
arrange(Month)

stats <- c("GP", "MP", "PTS", "FGA", "FGPerc", "ThreePA",
    "ThreePPer", "FTA", "FTPerc",
    "TRB", "ORB", "AST", "TOV", "STL")

df3_m1 <- df3_m %>%
    select(1:5, stats, 46:50)
get_barplot_monthly_stats(df3_m1, paste("; ACB", "2017-2018", "Average", sep = "; "), 2.5)

# For all teams and players:
teams <- as.character(sort(unique(df1$Team)))

df3_m <- df1 %>%
    filter(Team == teams[13]) %>%
group_by(Month) %>%
do(do_stats(., "Average", "2017-2018", "ACB", "Regular Season")) %>%
ungroup() %>%
mutate(Month = factor(Month, levels = months_plot2)) %>%
arrange(Month)

df3_m1 <- df3_m %>%
    select(1:5, stats, 46:50)

for (i in unique(df3_m1$Name)) {
    print(i)
    print(get_barplot_monthly_stats(df3_m1 %>% filter(Name == i),
        paste("; ACB", "2017-2018", "Average", sep = "; "), 2.5))
}

## End(Not run)
```

---

**get_bubble_plot**

**Basketball bubble plot**

**Description**

This plot is a representation of the percentiles of all statistics for a particular player. The figure shows four cells. The first box contains the percentiles between 0 and 24. The second, between 25 and 49. The third, between 50 and 74 and the fourth, between 75 and 100. The percentiles are computed with the function `percentilsArchetypoid`. Boxes of the same percentile category are in the same color in the interests of easy understanding.
This type of visualization allows the user to analyze each player in a very simple way, since a general idea of those aspects of the game in which the player excels can be obtained.

Usage

```
get_bubble_plot(df_stats, player, descr_stats, size_text, size_text_x, size_legend)
```

Arguments

- `df_stats` Data frame with the statistics.
- `player` Player.
- `descr_stats` Description of the statistics for the legend.
- `size_text` Text size inside each box.
- `size_text_x` Stats labels size.
- `size_legend` Legend size.

Details

In the example shown below, it can be seen that Alberto Abalde has a percentile of x in free throws percentage. This means that the x percent of league players has a fewer percentage than him, while there is a (100-x) percent who has a bigger percentage.

Value

Graphical device.

Author(s)

This function has been created using the code from this website: [https://www.r-bloggers.com/visualizing-the-best/](https://www.r-bloggers.com/visualizing-the-best/)

See Also

- `percentilsArchetypoid`

Examples

```r
## Not run:
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df2 <- do_stats(df1, "Total", "2017-2018", compet, "Regular Season")
# When choosing a subset of stats, follow the order in which they appear in the data frame.
           "FTA", "FTPerc", "TRB", "ORB", "AST", "STL", "TOV")
df2_1 <- df2[, c(1:5, which(colnames(df2) %in% stats), 46:49)]
descr_stats <- c("Games played", "Minutes played", "Points", 
                 "Field goals attempted", "Field goals percentage", 
                 "3-point field goals attempted", "3-point percentage",
```
Description

Once computed the team’s factors and its rankings with `do_four_factors_df`, this function represents them.

Usage

```r
get_four_factors_plot(df_rank, df_no_rank, team, language)
```

Arguments

- `df_rank` Data frame with the team’s offense and defense four factors and its ranking labels.
- `df_no_rank` Data frame with the team’s offense and defense four factors.
- `team` Team name. Multiple teams can be chosen.
- `language` Language labels. Current options are ‘en’ for English and ‘es’ for Spanish.

Value

Graphical device.

Author(s)

Guillermo Vinue

See Also

`do_four_factors_df`
get_games_rosters

Get all games and rosters

Description
This function is to get all the games and rosters of the competition selected.

Usage
get_games_rosters(competition, type_league, nums, verbose = TRUE,
                 accents = FALSE, r_user, df0, df_bio0)

Arguments

- **competition**: String. Options are "ACB", "Euroleague" and "Eurocup".
- **type_league**: String. If competition is ACB, to scrape ACB league games ("ACB"), Copa del Rey games ("CREY") or Supercopa games ("SCOPA").
- **nums**: Numbers corresponding to the website from which scraping.
- **verbose**: Should R report information on progress? Default TRUE.
- **accents**: If competition is ACB, should we keep the Spanish accents? The recommended option is to remove them, so default FALSE.
- **r_user**: Email to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.
- **df0**: Data frame to save the games data.
- **df_bio0**: Data frame to save the rosters data.

Value
Data frame.

Author(s)
Guillermo Vinue
## Not run:
library(readr)

# 1. The first time, all the historical data until the last games played can be directly scraped.

# ACB seasons available and corresponding games numbers:
names(acb_nums) <- paste(as.character(1985:2017), as.character(1986:2018), sep = "-")
df0 <- data.frame() df_bio0 <- data.frame(CombinID = NA, Player = NA, Position = NA, Height = NA, Date_birth = NA, Nationality = NA, Licence = NA, Website_player = NA)

# All the games and players:
get_data <- get_games_rosters(competition = "ACB", type_league = "ACB",
nums = acb_nums, verbose = TRUE, accents = FALSE,
r_user = "guillermo.vinue@uv.es",
df0 = df0, df_bio0 = df_bio0)

acb_games <- get_data$df0
acb_players <- get_data$df_bio0
write_csv(acb_games, path = "acb_games.csv")
write_csv(acb_players, path = "acb_players.csv")

# 2. Then, in order to scrape new games as they are played, the df0 and df_bio0 objects are the historical games and rosters:
acb_nums <- list(62136:62153)
names(acb_nums) <- "2017-2018"
df0 <- read_csv("acb_games.csv", guess_max = 1e5)
df_bio0 <- read_csv("acb_players.csv", guess_max = 1e3)
get_data <- get_games_rosters(competition = "ACB", type_league = "ACB",
nums = acb_nums, verbose = TRUE, accents = FALSE,
r_user = "guillermo.vinue@uv.es",
df0 = df0, df_bio0 = df_bio0)

# -----

# ACB Copa del Rey seasons available and corresponding games numbers (rosters were already downloaded with the ACB league):
get_games_rosters

70001:70007, 71001:71007, 72001:72007, 73001:73007,
74001:74007, 75001:75007, 76001:76007, 77001:77007,
78001:78007, 79001:79007, 80001:80007, 81001:81007)

names(acb_crey_nums) <- paste(as.character(1985:2016), as.character(1986:2017), sep = "-")

df0 <- data.frame()
get_data <- get_games_rosters(competition = "ACB", type_league = "CREY",
    nums = acb_crey_nums, verbose = TRUE, accents = FALSE,
    r_user = "guillermo.vinue@uv.es",
    df0 = df0, df_bio0 = NULL)
acb_crey_games <- get_data$df0
write_csv(acb_crey_games, path = "acb_crey_games.csv")

# -----

# ACB Supercopa seasons available and corresponding games numbers (rosters were
# already downloaded with the ACB league):
acb_scopa_nums <- list(1001, 2001, 3001, 4001, 5001:5004, 6001:6004,
    7001:7003, 9001:9003, 10001:10003, 11001:11003,
    12001:12003, 13001:13003, 14001:14003, 15001:15003,
    16001:16003, 17001:17003, 18001:18003, 19001:19003)

# I haven’t found the data for the supercopa in Bilbao 2007 ; 8001:8003
# http://www.acb.com/fichas/SCOPA8001.php
names(acb_scopa_nums) <- c(paste(as.character(1984:1987), as.character(1985:1988), sep = "-"),

df0 <- data.frame()
get_data <- get_games_rosters(competition = "ACB", type_league = "SCOPA",
    nums = acb_scopa_nums, verbose = TRUE, accents = FALSE,
    r_user = "guillermo.vinue@uv.es",
    df0 = df0, df_bio0 = NULL)
acb_scopa_games <- get_data$df0
write_csv(acb_scopa_games, path = "acb_scopa_games.csv")

# -----

# Euroleague seasons available and corresponding games numbers:
euroleague_nums <- list(1:128,
    1:263, 1:250, 1:251, 1:253, 1:253, 1:188, 1:189,
    1:188, 1:188, 1:231, 1:231, 1:231, 1:229, 1:220,
    1:220, 1:275, 1:169)
names(euroleague_nums) <- 2017:2000

df0 <- data.frame()
df_bio0 <- data.frame(CombinID = NA, Player = NA, Position = NA,
    Height = NA, Date_birth = NA,
    Nationality = NA, Website_player = NA)
get_data <- get_games_rosters(competition = "Euroleague", nums = euroleague_nums,
    verbose = TRUE, r_user = "guillermo.vinue@uv.es",
    df0 = df0, df_bio0 = df_bio0)
euroleague_games <- get_data$df0
euroleague_players <- get_data$df_bio0
get_heatmap_bb

write_csv(euroleague_games, path = "euroleague_games.csv")
write_csv(euroleague_players, path = "euroleague_players.csv")

# -----

# Eurocup seasons available and corresponding games numbers:
eurocup_nums <- list(1:128,
                      1:151, 1:326, 1:149, 1:149, 1:239, 1:209, 1:150)
names(eurocup_nums) <- 2017:2002
df0 <- data.frame()
df_bio0 <- data.frame(CombinID = NA, Player = NA, Position = NA,
                      Height = NA, Date_birth = NA,
                      Nationality = NA, Website_player = NA)
get_data <- get_games_rosters(competition = "Eurocup", nums = eurocup_nums,
                               verbose = TRUE, r_user = "guillermo.vinue@uv.es",
                               df0 = df0, df_bio0 = df_bio0)
eurocup_games <- get_data$df0
eurocup_players <- get_data$df_bio0
write_csv(eurocup_games, path = "eurocup_games.csv")
write_csv(eurocup_players, path = "eurocup_players.csv")

## End(Not run)

get_heatmap_bb

Basketball heatmap

Description

The heatmap created with this function allows the user to easily represent the stats for each player. The more intense the color, the more the player highlights in the statistic considered. The plot can be ordered by any statistic. If all the statistics are represented, the offensive statistics are grouped in red, the defensive in green, the rest in purple and the advanced in pink. Otherwise, the default color is red.

Usage

get_heatmap_bb(df_stats, team, levels_stats = NULL, stat_ord, base_size = 9, title)

Arguments

df_stats Data frame with the statistics.
team Team.
levels_stats Statistics classified in several categories to plot. If this is NULL, all the statistics are included in the data frame. Otherwise, the user can define a vector with the variables to represent.
get_map_nats

stat_ord  To sort the heatmap on one particular statistic.

base_size  Sets the font size in the theme used. Default 9.

title    Plot title.

Value

Graphical device.

Author(s)

This function has been created using the code from these websites: https://learnr.wordpress.com/2010/01/26/ggplot2-quick-heatmap-plotting/ and http://stackoverflow.com/questions/13016022/ggplot2-heatmaps-using-different-gradients-for-categories/13016912

Examples

```r
## Not run:
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df2 <- do_stats(df1, "Total", "2017-2018", compet, "Regular Season")
teams <- as.character(rev(sort(unique(df2$Team))))
get_heatmap_bb(df2, teams[6], NULL, "MP", 9, paste(compet, "2017-2018", "Total", sep = " "))
## End(Not run)
```

get_map_nats

Nationalities map

Description

A world map is represented. The countries from where there are players in the competition selected are in green color.

Usage

```r
get_map_nats(df_stats)
```

Arguments

- `df_stats`: Data frame with the statistics and the corrected nationalities.

Value

Graphical device.
get_pop_pyramid

Author(s)
Guillermo Vinue

See Also
do_map_nats

Examples
## Not run:
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df2 <- do_stats(df1, "Total", "2017-2018", compet, "Regular Season")
get_map_nats(df2)

## End(Not run)

gap_pop_pyramid Population_pyramid

Description
This is the code to get a population pyramid with the number of both Spanish and foreigner players along the seasons for the ACB league. This aids in discussion of nationality imbalance.

Usage
gap_pop_pyramid(df, title, language)

Arguments
df Data frame that contains the ACB players’ nationality.
title Title of the plot
language String, "eng" for English labels; "esp" for Spanish labels.

Value
Graphical device.

Author(s)
Guillermo Vinue
Examples

```r
## Not run:
# Load the data_app_acb file with the ACB games
# from seasons 1985-1986 to 2017-2018:
load(url("http://www.uv.es/vivigui/softw/data_app_acb.RData"))
title <- "Number of Spanish and foreign players along the ACB seasons \nData from www.acb.com"
get_pop_pyramid(data_app_acb, title, "eng")

## End(Not run)
```

get_shooting_plot        Shooting plot

Description

This plot represents the number of shots attempted and scored by every player of the same team, together with the scoring percentage. The players are sorted by percentage.

Usage

```r
get_shooting_plot(df_stats, team, type_shot, min_att, title, language)
```

Arguments

- `df_stats`: Data frame with the statistics.
- `team`: Team.
- `type_shot`: Numeric with values 1-2-3: 1 refers to free throws, 2 refers to two point shots and 3 refers to three points shots.
- `min_att`: Minimum number of attempts by the player to be represented in the plot.
- `title`: Plot title.
- `language`: Language labels. Current options are 'en' for English and 'es' for Spanish.

Value

Graphical device.

Author(s)

Guillermo Vinue
get_similar_players

Examples

```r
## Not run:
compet <- "ACB"
df <- do_join_games_bio(compet, acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df2 <- do_stats(df1, "Total", "2017-2018", compet, "Regular Season")
get_shooting_plot(df2, "Valencia", 3, 1,
                   paste("Valencia", compet, "2017-2018", sep = " "), "en")

## End(Not run)
```

get_similar_players Similar players to archetypoids

Description

Similar players to the archetypoids computed with `archetypoids` according to a similarity threshold.

Usage

```r
get_similar_players(atype, threshold, alphas, cases, data, variables, compet, season)
```

Arguments

- `atype`: Number assigned to the archetypoid (1:length(cases)) from which searching the players who most resemble to it.
- `threshold`: Similarity threshold.
- `alphas`: Alpha values of all the players.
- `cases`: Archetypoids.
- `data`: Data frame with the statistics.
- `variables`: Statistics used to compute the archetypoids.
- `compet`: Competition.
- `season`: Season.

Value

Data frame with the features of the similar players.

Author(s)

Guillermo Vinue

See Also

`archetypoids`
Examples

```r
(s0 <- Sys.time())
library(Anthropometry)
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df1 <- do_add_adv_stats(df)
df2 <- do_stats(df1, "Total", "2017-2018", "ACB", "Regular Season")
df3 <- df2[which(df2$Position == "Guard")[1:31], c("MP", "PTS", "Name")]
preproc <- preprocessing(df3[,1:2], stand = TRUE, percAccom = 1)
set.seed(4321)
lass <- stepArchetypesRawData(preproc$data, 1:2, numRep = 20, verbose = FALSE)
res <- archetypoids(2, preproc$data, huge = 200, step = FALSE, ArchObj = lass,
  nearest = "cand_ns", sequ = TRUE)
cases <- anthrCases(res)
df3[cases,]
alphas <- round(res$alphas, 4)
df3_aux <- df2[which(df2$Position == "Guard")[1:31], ]
get_similar_players(1, 0.99, alphas, cases, df3_aux, c("MP", "PTS"),
  unique(df3_aux$Compet), unique(df3_aux$Season))
s1 <- Sys.time() - s0
s1
```

---

**get_similar_teams**

Similar teams to archetypoids

**Description**

Similar teams to the archetypoids computed with `archetypoids` according to a similarity threshold.

**Usage**

```r
get_similar_teams(etype, threshold, alphas, cases, data, variables)
```

**Arguments**

- `etype` Number assigned to the archetypoid (1:length(cases)) from which searching the players who most resemble to it.
- `threshold` Similarity threshold.
- `alphas` Alpha values of all the players.
- `cases` Archetypoids.
- `data` Data frame with the statistics.
- `variables` Statistics used to compute the archetypoids.

**Value**

Data frame with the features of the similar teams.
**get_stats_seasons**

**Author(s)**

Guillermo Vinue

**See Also**

archetypoids

**Examples**

```r
## Not run:
(s0 <- Sys.time())
library(Anthropometry)
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df$Compet <- "ACB"
df_teams <- do_stats_teams(df, "2017-2018", "ACB", "Regular Season")
df_team_total <- df_teams$df_team_total

df3 <- df_team_total[, c("PTS", "PTSrv", "Team")]
preproc <- preprocessing(df3[,1:2], stand = TRUE, percAccomm = 1)
set.seed(4321)
lass <- stepArchetypesRawData(preproc$data, 1:2, numRep = 20, verbose = FALSE)
res <- archetypoids(2, preproc$data, huge = 200, step = FALSE, ArchObj = lass,
                   nearest = "cand_ns", sequ = TRUE)
cases <- anthrCases(res)
df3[cases,]
alphas <- round(res$alphas, 4)

gt_similar_teams(1, 0.95, alphas, cases, df_team_total, c("PTS", "PTSrv"))
s1 <- Sys.time() - s0
s1

## End(Not run)
```

---

**get_stats_seasons**  
*Season-by-season stats*

**Description**

This function represents the average values of a set of statistics for certain players in every season where the players played. It gives an idea of the season-by-season performance.

**Usage**

```r
get_stats_seasons(df, competition, player, variabs, type_season, add_text, show_x_axis)
```
get_table_results

League cross table

Description

The league results are represented with a cross table.
**Usage**

get_table_results(df, competition, season)

**Arguments**

- df: Data frame with the games and the players info.
- competition: Competition.
- season: Season.

**Value**

List with these two elements:

- plot_teams: Graphical device with the cross table.
- wins_teams: Vector with the team wins.

**Author(s)**

Guillermo Vinue

**Examples**

```r
## Not run:
df <- do_join_games_bio("ACB", acb_games_1718, acb_players_1718)
df$Compet <- "ACB"

gg <- get_table_results(df, "ACB", "2017-2018")

gg$wins_teams
gg$plot_teams
## End(Not run)
```

---

join_players_bio_age_acb

*Join ACB games and players’ info*

**Description**

This function joins the ACB games with the players’ bio and computes the players’ age at each game.

**Usage**

join_players_bio_age_acb(df_games, df_rosters)
Arguments

- `df_games` Data frame with the games.
- `df_rosters` Data frame with the biography of the roster players.

Value

Data frame.

Author(s)

Guillermo Vinue

See Also

`do_join_games_bio`

Examples

```r
df <- join_players_bio_age_euro(acb_games_1718, acb_players_1718)
```

---

**join_players_bio_age_euro**

*Join Euroleague and Eurocup games and players’ info*

Description

This function joins the Euroleague/Eurocup games with the players’ bio and computes the players’ age at each game.

Usage

```r
join_players_bio_age_euro(df_games, df_rosters)
```

Arguments

- `df_games` Data frame with the games.
- `df_rosters` Data frame with the biography of the roster players.

Value

Data frame.

Author(s)

Guillermo Vinue
See Also
do_join_games_bio

Examples
df <- join_players_bio_age_euro(euroleague_games_1718, euroleague_players_1718)

scraping_games_acb

ACB player game finder data

Description
This function allows us to get all the player game finder data for all the desired ACB seasons available from: http://www.acb.com.

Usage
scraping_games_acb(type_league, nums, year, verbose = TRUE,
accents = FALSE, r_user = "guillermo.vinue@uv.es")

Arguments
type_league String. If competition is ACB, to scrape ACB league games ("ACB"), Copa del Rey games ("CREY") or Supercopa games ("SCOPA").
nums Numbers corresponding to the website to scrape.
year Season, e.g. 2017-2018.
verbose Should R report information on progress? Default TRUE.
accents Should we keep the Spanish accents? The recommended option is to remove them, so default FALSE.
r_user Email to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Details
The official website of the Spanish basketball league ACB presents the statistics of each game in a php website, such as: http://www.acb.com/fichas/LACB62090.php.
Note that for example http://www.acb.com/fichas/LACB60315.php doesn’t exist, so for these cases is where we can use the httr package.
In https://www.uv.es/vivigui/docs/acb_scraping.pdf a document is available with the exact numbers xxxxx related to http://www.acb.com/fichas/LACBxxxxx.php for some seasons.

Value
A data frame with the player game finder data.
scraping_games_euro

Note
In addition to use the email address to stay identifiable, the function also contains two headers regarding the R platform and version used.

Furthermore, even though in the robots.txt file at http://www.acb.com/robots.txt, there is no information about scraping limitations and all robots are allowed to have complete access, the function also includes the command Sys.sleep(2) to pause between requests for 2 seconds. In this way, we don’t bother the server with multiple requests and we do carry out a friendly scraping.

Author(s)
Guillermo Vinue

See Also
do_scraping_games

Examples
## Not run:
# Not needed to scrape every time the package is checked, built and installed.
df1 <- scraping_games_acb(type_league = "ACB", nums = 62001:62002, year = "2017-2018", 
  verbose = TRUE, accents = FALSE, 
  r_user = "guillermo.vinue@uv.es")

## End(Not run)

scraping_games_euro  Euroleague and Eurocup player game finder data

Description
This function allows us to get all the player game finder data for all the desired Euroleague and Eurocup seasons available from http://www.euroleague.net/main/results/ and http://www.eurocupbasketball.com/eurocup/games/results, respectively.

Usage
scraping_games_euro(competition, nums, year, verbose = TRUE, 
  r_user = "guillermo.vinue@uv.es")

Arguments

competition  String. Options are "Euroleague" and "Eurocup".
nums          Numbers corresponding to the website from which scraping.
year          Year when the season starts. 2017 refers to 2017-2018 and so on.
verbose       Should R report information on progress? Default TRUE.
r_user

Email to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Details

See the examples in get_games_rosters to see the game numbers to scrape in each season.

Value

A data frame with the player game finder data.

Note

In addition to use the email address to stay identifiable, the function also contains two headers regarding the R platform and version used.

Furthermore, in the robots.txt file located at http://www.euroleague.net/robots.txt and https://www.eurocupbasketball.com/robots.txt there is the Crawl-delay field which asks crawlers to pause between requests for 15 seconds. This is done by adding to the function the command Sys.sleep(15).

Author(s)

Guillermo Vinue

See Also

do_scraping_games

Examples

## Not run:
# Not needed to scrape every time the package is checked, built and installed.
# It takes 15 seconds as it is required in http://www.euroleague.net/robots.txt
df1 <- do_scraping_games(competition = "Euroleague", nums = 1:2,
year = "2017", verbose = TRUE, r_user =
"guillermo.vinue@uv.es")

## End(Not run)
scraping_rosters_acb

ACB players’ profile

Description

This function allows us to obtain the basic information of each player, including his birth date. Then, we will be able to compute the age that each player had in the date that he played each game. The website used to collect information is https://www.acb.com.

Usage

scraping_rosters_acb(pcode, verbose = TRUE, accents = FALSE, r_user = "guillermo.vinue@uv.es")

Arguments

pcode Code corresponding to the player’s website to scrape.
verbose Should R report information on progress? Default TRUE.
accents Should we keep the Spanish accents? The recommended option is to remove them, so default FALSE.
r_user Email user to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Details

Some players have a particular licence, which does not necessarily match with their nationality, in order not to be considered as a foreign player, according to the current ACB rules.

Value

Data frame with eight columns:

- CombinID: Unique ID to identify the players.
- Player: Player’s name.
- Position: Player’s position on the court.
- Height: Player’s height.
- Date_birth: Player’s birth date.
- Nationality: Player’s nationality.
- Licence: Player’s licence.
- Website_player: Website.
**scraping_rosters_euro**

**Note**

In addition to use the email address to stay identifiable, the function also contains two headers regarding the R platform and version used.

Furthermore, even though in the robots.txt file at [http://www.acb.com/robots.txt](http://www.acb.com/robots.txt), there is no information about scraping limitations and all robots are allowed to have complete access, the function also includes the command `Sys.sleep(2)` to pause between requests for 2 seconds. In this way, we don’t bother the server with multiple requests and we do carry out a friendly scraping.

**Author(s)**

Guillermo Vinue

**See Also**

do_scraping_rosters

**Examples**

```r
## Not run:
# Not needed to scrape every time the package is checked, built and installed.
df_bio <- scraping_rosters_acb("56C", verbose = TRUE, accents = FALSE, 
r_user = "guillermo.vinue@uv.es")
## End(Not run)
```

---

**scraping_rosters_euro  Euroleague and Eurocup players' profile**

**Description**

This function allows us to obtain the basic information of each Euroleague/Eurocup player, including his birth date. Then, we will be able to compute the age that each player had in the date that he played each game. The websites used to collect information are [https://www.euroleague.net](https://www.euroleague.net) and [https://www.eurocupbasketball.com](https://www.eurocupbasketball.com).

**Usage**

```r
scraping_rosters_euro(competition, pcode, year, verbose = TRUE, 
r_user = "guillermo.vinue@uv.es")
```

**Arguments**

- `competition`: String. Options are "Euroleague" and "Eurocup".
- `pcode`: Code corresponding to the player’s website to scrape.
- `year`: Year when the season starts. 2017 refers to 2017-2018 and so on.
- `verbose`: Should R report information on progress? Default TRUE.
scraping_rosters_euro

r_user

Email user to identify the user when doing web scraping. This is a polite way to do web scraping and to certify that the user is working as transparently as possible with a research purpose.

Value

Data frame with seven columns:

- CombinID: Unique ID to identify the players.
- Player: Player’s name.
- Position: Player’s position on the court.
- Height: Player’s height.
- Date_birth: Player’s birth date.
- Nationality Player’s nationality.
- Website_player: Website.

Note

In addition to use the email address to stay identifiable, the function also contains two headers regarding the R platform and version used.

Furthermore, in the robots.txt file located at http://www.euroleague.net/robots.txt and https://www.eurocupbasketball.com/robots.txt there is the Crawl-delay field which asks crawlers to pause between requests for 15 seconds. This is done by adding to the function the command Sys.sleep(15).

Author(s)

Guillermo Vinue

See Also

do_scraping_rosters

Examples

## Not run:
# Not needed to scrape every time the package is checked, built and installed.
# It takes 15 seconds as it is required in http://www.euroleague.net/robots.txt
df_bio <- scraping_rosters_euro("Euroleague", "005791", "2017", verbose = TRUE,
                                r_user = "guillermo.vinue@uv.es")

## End(Not run)
Index

*Topic BAwiR
  BAwiR-package, 3

*Topic datasets
  acb_games_1718, 4
  acb_players_1718, 5
  eurocup_games_1718, 16
  eurocup_players_1718, 16
  euroleague_games_1718, 17
  euroleague_players_1718, 17

acb_games_1718, 4
acb_players_1718, 5
archetypoids, 29–31

BAwiR-package, 3
capit_two_words, 5, 18
do_add_adv_stats, 6, 7, 11
do_EPS, 6, 7, 10, 11
do_four_factors_df, 8, 21
do_join_games_bio, 9, 34, 35
do_map_nats, 10, 27
do_OE, 6, 7, 10
do_scraping_games, 11, 36, 37
do_scraping_rosters, 13, 39, 40
do_stats, 14
do_stats_teams, 15
eurocup_games_1718, 16
eurocup_players_1718, 16
euroleague_games_1718, 17
euroleague_players_1718, 17

get_barplot_monthly_stats, 18
get_bubble_plot, 19
get_four_factors_plot, 9, 21
get_games_rosters, 22, 37
get_heatmap_bb, 25
get_map_nats, 10, 26
get_pop_pyramid, 27
get_shooting_plot, 28
get_similar_players, 29
get_similar_teams, 30
get_stats_seasons, 31
get_table_results, 32
join_players_bio_age_acb, 9, 33
join_players_bio_age_euro, 9, 34
percentilsArchetypoid, 19, 20
scraping_games_acb, 12, 13, 35
scraping_games_euro, 12, 36
scraping_rosters_acb, 38
scraping_rosters_euro, 13, 39