Package ‘fflr’

September 14, 2023

Title   Retrieve ESPN Fantasy Football Data
Version  2.2.0
Description Format the raw data from the ESPN fantasy football API
          Retrieve data on public leagues, rosters, athletes, and matches.
License MIT + file LICENSE
Depends  R (>= 2.10)
Imports  httr (>= 1.4.2), jsonlite (>= 1.7.2), stats, tibble (>= 3.1.3)
Suggests knitr (>= 1.34), rmarkdown (>= 2.11), spelling (>= 2.2),
          testthat (>= 3.0.0), xml2 (>= 1.3.2)
VignetteBuilder knitr
Config/testthat/edition  3
Encoding  UTF-8
Language  en-US
LazyData  true
RoxygenNote  7.2.3
NeedsCompilation no
Author    Kiernan Nicholls [aut, cre, cph]
Maintainer Kiernan Nicholls <kiernan@protonmail.com>
Repository CRAN
Date/Publication  2023-09-14 07:00:02 UTC

R topics documented:

acquisition_settings ...................................................... 3
all_players ............................................................... 3
best_roster ............................................................... 4
combine_history ........................................................ 5
draft_recap ............................................................... 6
draft_settings .......................................................... 6
topics documented:

- espn_games .................................................. 7
- ffl_id .......................................................... 8
- ffl_info ....................................................... 9
- ffl_seasons .................................................. 9
- finance_settings .......................................... 10
- league_info ............................................... 11
- league_members ........................................... 11
- league_messages ......................................... 12
- league_name ............................................... 13
- league_simulation ........................................ 14
- league_size ............................................... 14
- league_standings ......................................... 15
- league_status ............................................. 16
- league_teams ............................................. 17
- list_players ............................................... 18
- live_scoring .............................................. 20
- nfl_players ............................................... 20
- nfl_schedule ............................................. 21
- nfl_teams .................................................. 22
- opponent_ranks .......................................... 22
- player_acquire ........................................... 23
- player_info ............................................... 24
- player_news .............................................. 24
- player_outlook ........................................... 25
- pro_events ............................................... 26
- pro_schedule ............................................ 26
- pro_scores ............................................... 27
- recent_activity .......................................... 27
- roster_score ............................................. 28
- roster_settings ......................................... 29
- schedule_settings ....................................... 30
- scoring_settings ........................................ 30
- start_roster ............................................. 31
- stat_corrections ......................................... 32
- team_abbrev ............................................. 32
- team_roster .............................................. 33
- tidy_schedule ........................................... 34
- tidy_scores .............................................. 35
- trade_settings .......................................... 36
- transaction_counter .................................... 36

Index ............................................................ 38
acquisition_settings

League waiver settings

Description

The type, days, and details of a league waiver process.

Usage

acquisition_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults togetOption("fflr.leagueId"). Function fails if no ID is found.
- **leagueHistory**: logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
- **...**: Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to query alongside `view`.

Value

A data frame of waiver settings by season.

See Also

Other league settings functions: `draft_settings()`, `finance_settings()`, `league_info()`, `league_name()`, `league_size()`, `roster_settings()`, `schedule_settings()`, `scoring_settings()`, `trade_settings()`

Examples

acquisition_settings(leagueId = "42654852")

all_players

All fantasy players (deprecated)

Description

See `list_players()`.

Usage

all_players(...)
Arguments

... Arguments passed to the new `list_players()` function.

See Also

Other player functions: `list_players()`, `player_info()`, `player_news()`, `player_outlook()`, `recent_activity()`, `transaction_counter()`

Examples

```r
## Not run:
all_players()

## End(Not run)
```

---

**best_roster**

Sort the optimal fantasy roster

---

Description

Uses the roster settings for each league to find the best possible combinations of players to score the most fantasy points.

Usage

```r
best_roster(
leagueId = ffl_id(),
useScore = c("actualScore", "projectedScore"),
scoringPeriodId = NULL,
...
)
```

Arguments

- **leagueId** Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **useScore** One of "projectedScore" or "actualScore" (default).
- **scoringPeriodId** Integer week of NFL season. By default, NULL will use the current week (see `ffl_week()`). Scoring periods are always one week in length, whereas matchups might be longer.
- ... Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Details

If `scoringPeriodId` is the current week (the default), then actual scoring might be incomplete (see `projectedScore` argument).
combine_history

Value

A dataframe (or list) with optimal rosters.

See Also

Other roster functions: \texttt{roster\_score()}, \texttt{start\_roster()}, \texttt{team\_roster()}

Examples

\begin{verbatim}
best_roster(leagueId = "42654852", scoringPeriodId = 1)
\end{verbatim}

\begin{verbatim}
combine_history(tidy_scores, leagueId = "252353")
\end{verbatim}

Description

Runs a function \texttt{fun} twice, once with the \texttt{leagueHistory} set to \texttt{TRUE} and once set to \texttt{FALSE}. Combined the output of both runs into a single data frame.

Usage

\begin{verbatim}
combine_history(fun, ...)
\end{verbatim}

Arguments

\begin{verbatim}
fun A function with the \texttt{leagueHistory} argument.
...
Additional arguments passed to the function used in \texttt{fun}.
\end{verbatim}

Value

A data frame of combined outputs.

Examples

\begin{verbatim}
combine_history(tidy_scores, leagueId = "252353")
\end{verbatim}
### draft_recap

*Fantasy draft history*

**Description**

Return the sequential result of a fantasy draft pick, whether snake or salary cap format.

**Usage**

```r
draft_recap(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

**Arguments**

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **leagueHistory**: logical; Should the `leagueHistory` version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
- ... Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Value**

A data frame(s) of draft picks.

**See Also**

Other league functions: `league_members()`, `league_messages()`, `league_standings()`, `league_status()`, `league_teams()`, `tidy_schedule()`, `transaction_counter()`

**Examples**

```r
draft_recap(leagueId = "42654852")
```

---

### draft_settings

*League draft settings*

**Description**

The type, date, and pick order of a league draft.

**Usage**

```r
draft_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```
espn_games

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **leagueHistory**: logical; Should the `leagueHistory` version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
- **...**: Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Value

A data frame of league draft settings by season.

See Also

Other league settings functions: `acquisition_settings()`, `finance_settings()`, `league_info()`, `league_name()`, `league_size()`, `roster_settings()`, `schedule_settings()`, `scoring_settings()`, `trade_settings()`

Examples

```r
draft_settings(leagueId = "42654852")
```

Description

List all fantasy games

Usage

```r
espn_games()
```

Value

A tibble of fantasy games.

See Also

Other Game information: `ffl_info()`, `ffl_seasons()`

Examples

```r
espn_games()
```
ffl_id

Get ESPN fantasy league ID

Description

Retrieve league ID from global options, as an input, or from a URL.

Usage

```r
ffl_id(leagueId = getOption("fflr.leagueId"), overwrite = FALSE)
```

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **overwrite**: logical; If an `fflr.leagueId` option exists, should it be temporarily changed for your current session.

Details

Since many users request data from the same ESPN league when using this package, you can use this function to set, call, or extract the unique ESPN league ID. By default, this function uses `getOption("fflr.leagueId")` to look for a default league ID defined in your `options()`. If no such option exists, and one is provided to the `leagueId` argument, the option will be temporarily defined for your current session. If a URL starting with `http` is provided, the numeric league ID will be extracted, defined as the temporary option, and returned as a character string.

Value

A numeric `leagueId` as a character vector with length one.

Examples

```r
options(fflr.leagueId = "42654852")
ffl_id()
ffl_id(
  leagueId = "https://fantasy.espn.com/football/team?leagueId=42654852",
  overwrite = TRUE
)```
**ffl_info**  
*Get fantasy football information*

**Description**  
Information on the current fantasy football season, with functions to quickly access and modify certain information (like the current seasonId or scoringPeriodId).

**Usage**  
ffl_info()
ffl_year(offset = 0)
ffl_week(offset = 0)

**Arguments**  
offset  
Add negative or positive values.

**Value**  
A list of season information.

**See Also**  
Other Game information: espn_games(), ffl_seasons()

**Examples**  
str(ffl_info())
Sys.time()
ffl_year()
ffl_week(-1)

---

**ffl_seasons**  
*List past fantasy football seasons*

**Description**  
List past fantasy football seasons

**Usage**  
ffl_seasons()
finance_settings

Value
A tibble of fantasy football seasons.

See Also
Other Game information: espn_games(), ffl_info()

Examples
ffl_seasons()

finance_settings League finance settings

Description
The off-site fees assigned to various roster movies and transactions.

Usage
finance_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments
leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
...
Additional queries passed to httr::GET(). Arguments are converted to a named list and passed to query alongside view.

Value
A data frame of finance settings by season.

See Also
Other league settings functions: acquisition_settings(), draft_settings(), league_info(), league_name(), league_size(), roster_settings(), schedule_settings(), scoring_settings(), trade_settings()

Examples
finance_settings(leagueId = "42654852")
**league_info**  

*League information*

---

**Description**

Basic information on an ESPN fantasy football league, like the name, size, and season length.

**Usage**

```r
league_info(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

**Arguments**

- **leagueId**  
  Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.

- **leagueHistory**  
  logical; Should the `leagueHistory` version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

- `...`  
  Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Value**

A data frame of league information by season.

**See Also**

Other league settings functions: `acquisition_settings()`, `draft_settings()`, `finance_settings()`, `league_name()`, `league_size()`, `roster_settings()`, `schedule_settings()`, `scoring_settings()`, `trade_settings()`

**Examples**

```r
league_info(leagueId = "42654852")
```

---

**league_members**  

*Fantasy league teams*

---

**Description**

The teams in a league and their owners.

**Usage**

```r
league_members(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```
Arguments

leagueId  Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.

leagueHistory logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.

... Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Value

A dataframe (or list) with league members.

See Also

Other league functions: `draft_recap()`, `league_messages()`, `league_standings()`, `league_status()`, `league_teams()`, `tidy_schedule()`, `transaction_counter()`

Examples

```r
league_members(leagueId = "42654852")
```

---

Description

The emails, chats, notes, and messages sent by league members.

Usage

```r
league_messages(leagueId = ffl_id(), leagueHistory = FALSE, ...)```

Arguments

leagueId  Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.

leagueHistory logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.

... Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Value

A tibble of messages.
See Also

Other league functions: `draft_recap()`, `league_members()`, `league_standings()`, `league_status()`, `league_teams()`, `tidy_schedule()`, `transaction_counter()`

Examples

```r
league_messages(leagueId = "42654852")
```

divider

<table>
<thead>
<tr>
<th>league_name</th>
<th>League name</th>
</tr>
</thead>
</table>

Description

League name

Usage

```r
league_name(leagueId = ff1_id(), leagueHistory = FALSE, ...)
```

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **leagueHistory**: logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.
- **...**: Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Value

A character vector.

See Also

Other league settings functions: `acquisition_settings()`, `draft_settings()`, `finance_settings()`, `league_info()`, `league_size()`, `roster_settings()`, `schedule_settings()`, `scoring_settings()`, `trade_settings()`

Examples

```r
league_name(leagueId = "42654852")
```
league_simulation  
League standing simulation

Description

The ESPN algorithm simulates the entire season according to the projection and matchup schedule to calculate the probability of a team winning their division and making the playoffs.

Usage

league_simulation(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

leagueId  Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory  logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
...

Value

A data frame of simulated team standings.

Examples

league_simulation(leagueId = "42654852")

league_size  
League size

Description

League size

Usage

league_size(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

Value

A data frame of team standings.

See Also

Other league settings functions: acquisition_settings(), draft_settings(), finance_settings(), league_info(), league_name(), roster_settings(), schedule_settings(), scoring_settings(), trade_settings()

Examples

league_size(leagueId = "42654852")

league_standings League standings

Description

Return the current and projected standings, win streak, total wins, losses, and points scored for and against each team.

Usage

league_standings(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

Value

A data frame of team standings.
See Also

Other league functions: `draft_recap()`, `league_members()`, `league_messages()`, `league_status()`, `league_teams()`, `tidy_schedule()`, `transaction_counter()`

Examples

```r
league_status(leagueId = "42654852")
```
Description

The teams in a league and their owners.

Usage

league_teams(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **leagueHistory**: logical; Should the `leagueHistory` version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
- **...**: Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Details

If any team has multiple owners, the `memberId` column will be a list of unique owner member ID strings per team (see `league_members()`).

Value

A dataframe (or list) with league teams.

See Also

Other league functions: `draft_recap()`, `league_members()`, `league_messages()`, `league_standings()`, `league_status()`, `tidy_schedule()`, `transaction_counter()`

Examples

`league_teams(leagueId = "42654852")`
Find fantasy players

Description

Filter fantasy players by their position, availability, professional team, and/or injury status. Sort and limit the responses in the same way as is done in the ESPN Fantasy Football website.

Usage

```r
list_players(
  leagueId = ffl_id(),
  sort = "ROST",
  position = NULL,
  status = "AVAILABLE",
  injured = NULL,
  proTeam = NULL,
  scoreType = c("STANDARD", "PPR"),
  limit = 50
)
```

Arguments

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- **sort**: The column from which to sort the data. Options match those on the ESPN website:
  - "PLAYER" = Alphabetical by player name
  - "PROJ" = Projection is ESPN's projected fantasy score for a player's upcoming game.
  - "SCORE" = Actual score for scoringPeriodId
  - "OPRK" = Opponent Rank shows how a player's upcoming NFL opponent performs against that player's position. Low numbers mean it may be a tough opponent; high numbers an easier opponent.
  - "START" = Start Percentage shows the number of fantasy leagues a player is started in divided by the number of leagues he is eligible in. This helps indicate how the public views a player.
  - "ROST" = Rostered Percentage shows the number of fantasy leagues in which a player is on a roster divided by the total number of fantasy leagues. This helps indicate how the public views a player.
  - "CHANGE" = Plus/Minus shows the change in %ROST over the last week. This will help show which players are hot and cold at a given moment.
  - "PRK" = Position Rank shows how a player stacks up against other players at his position. No. 1 is best.
  - "FPTS" = Total fantasy points scored thus far in the season.
list_players

- "AVG" = Average fantasy points scored in each game started.
- "LAST" = Last shows the player’s fantasy score in his team’s last game.

**position**
Abbreviation of player positions to filter, NULL for all:
- "QB" = Quarterback
- "RB" = Running Back
- "WR" = Wide Receiver
- "TE" = Tight End
- "FLEX" = Running Backs, Wide Receivers and Tight Ends can be used in this position
- "D/ST" = Defense and Special Teams
- "K" = Kicker

**status**
Availability status of player, one or more from:
- "ALL"
- "AVAILABLE" (default)
- "FREEAGENT"
- "WAIVERS"
- "ONTEAM"

**injured**
Whether to return only injured or healthy players. Use NULL (default) for all players, TRUE for injured players, and FALSE for healthy players.

**proTeam**
The abbreviation or ID of the professional team from which players should be returned. See pro_teams() for a list of all possible team abbreviations.

**scoreType**
The type of scoring used: "STANDARD" or "PPR."

**limit**
The limit of players to return. Use "" or NULL to return all. Defaults to 50, which is the default limit used by ESPN. Removing the limit can make the request take a long time.

**Value**
A data frame of players.

**See Also**
Other player functions: all_players(), player_info(), player_news(), player_outlook(), recent_activity(), transaction_counter()

**Examples**
list_players("42654852", proTeam = "Mia", sort = "START", limit = 3)
live_scoring  
*Live matchup scoreboard*

**Description**

The current and projected score for each ongoing match.

**Usage**

```r
live_scoring(leagueId = ffl_id(), yetToPlay = FALSE, bonusWin = FALSE, ...)
```

**Arguments**

- `leagueId` Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- `yetToPlay` If `TRUE`, `pro_schedule()` and the "mRoster" view are called to determine how many starting players have yet to start playing.
- `bonusWin` If `TRUE`, a logical column `bonusWin` will be added containing `TRUE` values for teams who are projected to score in the top half of points this week. This is a way to project the "bonus win" optional setting added in 2022.
- `...` Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Value**

A data frame of scores by period.

**See Also**

Other scoring functions: `tidy_scores()`

**Examples**

```r
live_scoring(leagueId = "42654852", yetToPlay = FALSE)
```

---

nfl_players  
*2023 NFL Players*

**Description**

All available ESPN fantasy football players as of the 2023 season, week 1.

**Usage**

```r
nfl_players
```
Format

A data frame with 1,102 rows and 11 variables:

- `playerId`  Unique ESPN player ID
- `firstName` First name
- `lastName` Last name
- `proTeam` Professional NFL team
- `defaultPosition` Position: QB, RB, WR, TE, D/ST
- `jersey` Jersey number
- `weight` Weight in integer pounds
- `height` Height in integer inches
- `age` Current age in integer year
- `dateOfBirth` Date of birth
- `birthPlace` Place of birth
- `debutYear` Season debuted in league
- `draftSelection` Overall pick number in the NFL draft...

Source

http://sports.core.api.espn.com/v2/sports/football/leagues/nfl/seasons/2023/athletes/

2023 NFL Schedule

Description

The 2023 NFL season schedule by team, as of September 10th.

Usage

nfl_schedule

Format

A data frame with 544 rows and 6 variables:

- `seasonId`  Season year
- `scoringPeriodId` Scoring period
- `matchupId` Unique ID for professional matchup
- `proTeam` Professional team abbreviation
- `opponent` Professional team opponent
- `isHome` Whether this is the home team
- `date` Matchup start date and time...
### nfl_teams
**2023 NFL Teams**

**Description**

The 32 professional NFL teams as of the 2023 season.

**Usage**

nfl_teams

**Format**

A data frame with 33 rows and 6 columns:

- **proTeamId**: Unique team ID
- **abbrev**: Professional team abbreviation
- **location**: Professional team geographic location
- **name**: Professional team full nickname
- **byeWeek**: Bye week, no game played
- **conference**: NFL conference ...

**Source**


---

### opponent_ranks
**NFL team performance against positions**

**Description**

The average opposition team point differential by position.

**Usage**

opponent_ranks(leagueId = ffl_id())

**Arguments**

- **leagueId**: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
### player_acquire

**Value**

A data frame of team performance against position.

**Examples**

```r
opponent_ranks()
```

---

### player_acquire

**Description**

The date and method of each player's acquisition onto a fantasy roster.

**Usage**

```r
player_acquire(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

**Arguments**

- `leagueId` Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- `leagueHistory` logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.
- `...` Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Value**

A data frame of roster players with acquisition method and date.

**Examples**

```r
player_acquire(leagueId = "42654852")
```
## player_info
### Individual player information

**Description**

Individual player information

**Usage**

```
player_info(playerId)
```

**Arguments**

- `playerId` A single player ID number.

**Value**

A list or row of a single player’s information.

**See Also**

Other player functions: `all_players()`, `list_players()`, `player_news()`, `player_outlook()`, `recent_activity()`, `transaction_counter()`

**Examples**

```
player_info(playerId = 15847)
```

## player_news
### Player news

**Description**

The free and premium ESPN stories on given players. A maximum of 50 stories can be returned at a time.

**Usage**

```
player_news(playerId, parseHTML = FALSE)
```

**Arguments**

- `playerId` A single player ID number.
- `parseHTML` Should HTML stories be parsed with `xml2::read_html()`?

**Examples**

```
player_news(playerId = 15847, parseHTML = FALSE)
```
player_outlook

Value

A data frame of news stories.

See Also

Other player functions: all_players(), list_players(), player_info(), player_outlook(), recent_activity(), transaction_counter()

Examples

player_news(playerId = "15847")

---

player_outlook

Player outlooks

Description

All available weekly ESPN outlook writeups for NFL players.

Usage

player_outlook(leagueId = ffl_id(), limit = 50)

Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.

limit The limit of players to return. Use "" or NULL to return all. Defaults to 50, which is the default limit used by ESPN. Removing the limit can make the request take a long time.

Value

A data frame of player outlooks by scoring period.

See Also

Other player functions: all_players(), list_players(), player_info(), player_news(), recent_activity(), transaction_counter()

Examples

player_outlook()
**pro_events**  
*Professional games*

**Description**

Data on the status of NFL games, including scores and odds, kickoff time, and broadcast information.

**Usage**

pro_events()

**Value**

A data frame of NFL events.

**See Also**

Other professional football functions: pro_schedule()

**Examples**

pro_events()

---

**pro_schedule**  
*Professional schedule*

**Description**

The opponents each team faces every week in a regular season.

**Usage**

pro_schedule(seasonId = ffl_year())

**Arguments**

seasonId  
Season schedule (2004-present), defaults to ffl_year().

**Value**

Data frame of team opponents by week.

**See Also**

Other professional football functions: pro_events()
**pro_scores**

**Examples**

```
pro_schedule(seasonId = ffl_year(-2))
```

---

**pro_scores**      **Professional scores**

**Description**

The tidy data frame of scores by team.

**Usage**

```
pro_scores()
```

**Value**

A data frame of NFL scores.

**Examples**

```
pro_scores()
```

---

**recent_activity**      **Roster moves**

**Description**

The individual proposed and executed transactions, trades, and waiver moves.

**Usage**

```
recent_activity(
  leagueId = ffl_id(),
  leagueHistory = FALSE,
  scoringPeriodId = NULL,
  ...
)
```
Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.

leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

scoringPeriodId Integer week of NFL season. By default, NULL will use the current week (see `ffl_week()`). Scoring periods are always one week in length, whereas matchups might be longer.

... Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to query alongside view.

Details

As of November 2021, activity data related to trades coming from the API is flawed. The items list column containing the players involved in a trade will only contain data for rejected trades (with an executionType of "CANCEL"). For accepted and upheld trades, that items element is NULL or an empty list. This flaw comes from the API itself, not processing done by this package.

Value

A data frame of transactions and roster moves.

See Also

Other player functions: `all_players()`, `list_players()`, `player_info()`, `player_news()`, `player_outlook()`, `transaction_counter()`

Examples

```r
recent_activity(leagueId = "42654852", scoringPeriodId = 2)
```

`roster_score`  
`Sum of starting scores in a roster`

Description

For a given roster tibble, sum the starting scores.

Usage

```r
roster_score(roster, useScore = c("actualScore", "projectedScore"))
```

Arguments

roster A roster data frame from `team_roster()`.

useScore One of "projectedScore" or "actualScore" (default).
**roster_settings**

Value

A starting score as double.

See Also

Other roster functions: `best_roster()`, `start_roster()`, `team_roster()`

Examples

```r
roster_score(team_roster(leagueId = "42654852"))[[1]]
```

---

**roster_settings**

*League roster settings*

Description

The number of players and positions on a fantasy football roster.

Usage

```r
roster_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

Arguments

- `leagueId` Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- `leagueHistory` logical; Should the `leagueHistory` version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
- `...` Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

Value

A data frame of league roster settings by season.

See Also

Other league settings functions: `acquisition_settings()`, `draft_settings()`, `finance_settings()`, `league_info()`, `league_name()`, `league_size()`, `schedule_settings()`, `scoring_settings()`, `trade_settings()`

Examples

```r
roster_settings(leagueId = "42654852")
```
schedule_settings  League schedule settings

Description

The length of a fantasy season and the match periods for each week.

Usage

schedule_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)

Arguments

leagueId  Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory  logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
...  Additional queries passed to http::GET(). Arguments are converted to a named list and passed to query alongside view.

Value

A data frame of league schedule settings by season.

See Also

Other league settings functions: acquisition_settings(), draft_settings(), finance_settings(), league_info(), league_name(), league_size(), roster_settings(), scoring_settings(), trade_settings()

Examples

schedule_settings(leagueId = "42654852")

scoring_settings  League scoring settings

Description

The scoring system used and points awarded for various actions.

Usage

scoring_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)
Arguments

leagueId  Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.
leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.
...
Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to query alongside view.

Value

A data frame of league scoring settings by season.

See Also

Other league settings functions: acquisition_settings(), draft_settings(), finance_settings(), league_info(), league_name(), league_size(), roster_settings(), schedule_settings(), trade_settings()

Examples

scoring_settings(leagueId = "42654852")

start_roster

Description

The starting 9 man roster using standard roster slots. In the future this function may be adapted to take roster slots from roster_settings().

Usage

start_roster(roster)

Arguments

roster  A roster data frame from team_roster().

Value

A data frame of starters on a roster.

See Also

Other roster functions: best_roster(), roster_score(), team_roster()

Examples

start_roster(team_roster(leagueId = "42654852")[[1]])
**stat_corrections**  
*Stat corrections*

**Description**  
Weekly retroactive stat corrections by player.

**Usage**  

```r
stat_corrections(date = Sys.Date(), limit = 100)
```

**Arguments**

- **date**  
  A date in the scoring week to return. Defaults to system date.

- **limit**  
  The limit of corrections to return. Use "" or NULL to return all. Defaults to 100, which is the default limit used by ESPN. Removing the limit can make the request take a long time.

**Value**  
A data frame of stat corrections.

**Examples**

```r
stat_corrections(date = "2021-09-13")
```

---

**team_abbrev**  
*Convert team ID to abbreviation*

**Description**  
Convert team ID to abbreviation

**Usage**  

```r
team_abbrev(teamId, teams = league_teams(leagueId = ffl_id()))
```

**Arguments**

- **teamId**  
  A integer vector of team numbers to convert.

- **teams**  
  A table of teams, like that from `league_teams()`.

**Value**  
A factor vector of team abbreviations.
team_roster

Examples

team_abbrev(teamId = 2, teams = league_teams(leagueId = "42654852"))

---

Fantasy team rosters

Description

The roster of all teams in a league.

Usage

team_roster(
  leagueId = ffl_id(),
  leagueHistory = FALSE,
  scoringPeriodId = NULL,
  ...
)

Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr leagueId"). Function fails if no ID is found.

leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

scoringPeriodId Integer week of NFL season. By default, NULL will use the current week (see ffl_week()). Scoring periods are always one week in length, whereas matchups might be longer.

Additional queries passed to http::GET(). Arguments are converted to a named list and passed to query alongside view.

Value

A dataframe (or list) with league teams.

See Also

Other roster functions: best_roster(), roster_score(), start_roster()

Examples

team_roster(leagueId = "42654852", scoringPeriodId = 1)
tidy_schedule

Fantasy match schedule

Description

The opponents each team faces every week in a fantasy regular season. Returned in a tidy format
where each row is a single team with an indication of home-away status. There are two rows per
matchup, one for each team.

Usage

```r
tidy_schedule(leagueId = ffl_id(), leagueHistory = FALSE, ...)
tidy_matchups(...)
```

Arguments

- `leagueId`: Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`.
  Function fails if no ID is found.
- `leagueHistory`: logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list
  of results is returned, with one element for each historical year of the league.
- `...`: Additional queries passed to `httr::GET()`. Arguments are converted to a named
  list and passed to `query` alongside `view`.

Value

A data frame(s) of match opponents.

See Also

Other league functions: `draft_recap()`, `league_members()`, `league_messages()`, `league_standings()`,
`league_status()`, `league_teams()`, `transaction_counter()`

Examples

```r
tidy_schedule(leagueId = "42654852")
```
tidy_scores

*Fantasy matchup scores*

**Description**

The score of each team in a matchup or scoring period and the match outcome.

**Usage**

`tidy_scores(leagueId = ffl_id(), leagueHistory = FALSE, useMatchup = TRUE, ...)`

**Arguments**

- `leagueId` Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- `leagueHistory` logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.
- `useMatchup` logical; Whether scoring should be summarized by `matchupPeriodId` (default) or `scoringPeriodId`. The later always relates to a single week of the NFL season, while fantasy matchups might span several scoring periods, especially in the playoffs.
- `...` Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Details**

`expectedWins` are calculated by comparing a team score against all other scores for a given matchup period. This statistic expresses how a team would fair if the schedule was random. The highest scoring team is thus expected to earn 1 win and the lowest scoring team would expect to win 0 matchups.

**Value**

A tidy data frame of scores by team and matchup/scoring period.

**See Also**

Other scoring functions: `live_scoring()`

**Examples**

`tidy_scores(leagueId = "42654852", useMatchup = FALSE)`
### trade_settings

**League trade settings**

**Description**

The time each trade can stand, votes needed to veto, and season deadline.

**Usage**

```r
trade_settings(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

**Arguments**

- `leagueId` Numeric league ID or ESPN fantasy page URL. Defaults to `getOption("fflr.leagueId")`. Function fails if no ID is found.
- `leagueHistory` logical; Should the `leagueHistory` version of the API be called? If `TRUE`, a list of results is returned, with one element for each historical year of the league.
- `...` Additional queries passed to `httr::GET()`. Arguments are converted to a named list and passed to `query` alongside `view`.

**Value**

A data frame of league trade settings by season.

**See Also**

Other league settings functions: `acquisition_settings()`, `draft_settings()`, `finance_settings()`, `league_info()`, `league_name()`, `league_size()`, `roster_settings()`, `schedule_settings()`, `scoring_settings()`

**Examples**

```r
trade_settings(leagueId = "42654852")
```

### transaction_counter

**League transactions**

**Description**

Summary of transactions and roster changes made during a season by team.

**Usage**

```r
transaction_counter(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```

```r
budget_summary(leagueId = ffl_id(), leagueHistory = FALSE, ...)
```
transaction_counter

Arguments

leagueId Numeric league ID or ESPN fantasy page URL. Defaults to getOption("fflr.leagueId"). Function fails if no ID is found.

leagueHistory logical; Should the leagueHistory version of the API be called? If TRUE, a list of results is returned, with one element for each historical year of the league.

Additional queries passed to http::GET(). Arguments are converted to a named list and passed to query alongside view.

Value

A data frame of transaction counts by team.

See Also

Other league functions: draft_recap(), league_members(), league_messages(), league_standings(), league_status(), league_teams(), tidy_schedule()

Other player functions: all_players(), list_players(), player_info(), player_news(), player_outlook(), recent_activity()

Examples

transaction_counter(leagueId = "42654852")
Index

* Game information
  - espn_games, 7
  - ffl_info, 9
  - ffl_seasons, 9

* datasets
  - nfl_players, 20
  - nfl_schedule, 21
  - nfl_teams, 22

* league functions
  - draft_recap, 6
  - league_members, 11
  - league_messages, 12
  - league_standings, 15
  - league_status, 16
  - league_teams, 17
  - tidy_schedule, 34
  - transaction_counter, 36

* league settings functions
  - acquisition_settings, 3, 7, 10, 11, 13, 15, 29–31, 36
  - all_players, 3, 7, 9, 10, 11, 13, 15
  - best_roster, 4, 29, 31, 33
  - budget_summary (transaction_counter), 36
  - combine_history, 5
  - draft_recap, 6, 12, 13, 16, 17, 34, 37
  - draft_settings, 3, 6, 10, 11, 13, 15, 29–31, 36
  - espn_games, 7, 9, 10
  - ffl_id, 8
  - ffl_info, 7, 9, 10
  - ffl_seasons, 7, 9, 9
  - ffl_week (ffl_info), 9
  - ffl_week(), 4, 28, 33
  - ffl_year (ffl_info), 9
  - ffl_year(), 26
  - finance_settings, 3, 7, 10, 11, 13, 15, 29–31, 36
  - httr::GET(), 3, 4, 6, 7, 10–17, 20, 23, 28–31, 33–37
  - league_info, 3, 7, 10, 11, 13, 15, 29–31, 36
  - league_members, 6, 11, 13, 16, 17, 34, 37
  - league_members(), 17
  - league_messages, 6, 12, 12, 16, 17, 34, 37