Package ‘ffsimulator’

October 19, 2022

Title Simulate Fantasy Football Seasons

Version 1.2.2

Description Uses bootstrap resampling to run fantasy football season simulations supported by historical rankings and ‘nflfastR’ data, calculating optimal lineups, and returning aggregated results.

License MIT + file LICENSE

URL https://ffsimulator.ffverse.com,
https://github.com/ffverse/ffsimulator

BugReports https://github.com/ffverse/ffsimulator/issues

Depends R (>= 3.5.0)

Imports checkmate (>= 2.0.0), cli (>= 3.0.0), data.table (>= 1.14.0),
ffscrapr (>= 1.4.6), glue (>= 1.3.0), magrittr (>= 1.0.0),
nflreadr (>= 1.2.0), Rglpk (>= 0.6.0), rlang (>= 0.4.0),
tidytable (>= 0.6.4)

Suggests covr (>= 3.0.0), ggplot2 (>= 3.0.0), ggridges (>= 0.5.0),
knitr (>= 1.0), progressr (>= 0.8.0), rmarkdown (>= 2.6),
scales (>= 1.0.0), testthat (>= 3.0.0), vdiffr (>= 1.0.2)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Tan Ho [aut, cre, cph] (<https://orcid.org/0000-0001-8388-5155>)

Maintainer Tan Ho <tan@tanho.ca>

Repository CRAN

Date/Publication 2022-10-19 09:15:08 UTC
R topics documented:

autoplot.ff_simulation ................................................. 2
autoplot.ff_simulation_week ........................................... 3
espn_connect ............................................................... 4
ffs_add_replacement_level ............................................ 5
ffs_adp_outcomes ......................................................... 6
ffs_adp_outcomes_week ................................................... 7
ffs_build_schedules ..................................................... 8
ffs_copy_template ......................................................... 9
ffs_franchises ............................................................ 10
ffs_generate_projections ............................................... 10
ffs_generate_projections_week ....................................... 11
ffs_latest_rankings ...................................................... 12
ffs_optimise_lineups .................................................... 13
ffs_repeat_schedules ................................................... 15
ffs_rosters ............................................................... 15
ffs_schedule .............................................................. 16
ffs_score_rosters ........................................................ 17
ffs_starter_positions ................................................... 18
ffs_summarise_week ...................................................... 19
ff_connect ............................................................... 20
ff_scoringhistory ........................................................ 20
ff_simulate .............................................................. 21
ff_simulate_week ........................................................ 22
ff_starter_positions .................................................... 23
ff_wins_added ............................................................ 24
fleaflicker_connect ....................................................... 25
fp_injury_table ............................................................ 25
fp_rankings_history ...................................................... 26
fp_rankings_history_week ............................................... 26
mfl_connect .............................................................. 27
sleeper_connect .......................................................... 27

Index 28

autoplot.ff_simulation

Automatically Plot ff_simulation Object

Description

Creates automatic plots for wins, ranks, or points for an ff_simulation object as created by ff_simulate().
Usage

autoplot.ff_simulation(object, type = c("wins", "rank", "points"), ...)

## S3 method for class 'ff_simulation'
plot(x, ..., type = c("wins", "rank", "points"), y)

Arguments

object      a ff_simulation object as created by ff_simulate()
type        one of "wins", "rank", "points"
...         unused, required by autoplot generic
x           A ff_simulation object.
y           Ignored, required for compatibility with the plot() generic.

Details

Usage of this function/method requires the ggplot2 package and (for wins and points plots) the ggridges package.

Value

a ggplot object

See Also

vignette("basic") for example usage

Examples

simulation <- .ffs_cache("foureight_sim.rds")

ggplot2::autoplot(simulation) # default is type = "wins"
ggplot2::autoplot(simulation, type = "rank")
ggplot2::autoplot(simulation, type = "points")

Description

Creates automatic plots for wins, ranks, or points for an ff_simulation object as created by ff_simulate().
Usage

```r
autoplot.ff_simulation_week(object, type = c("luck", "points"), ...)
```

## S3 method for class 'ff_simulation_week'
plot(x, ..., type = c("luck", "points"), y)

Arguments

- `object`: a `ff_simulation` object as created by `ff_simulate()`
- `type`: one of "luck" or "points"
- `...`: unused, required by autoplot generic
- `x`: A `ff_simulation_week` object.
- `y`: Ignored, required for compatibility with the `plot()` generic.

Details

Usage of this function/method requires the `ggplot2` package and (for wins and points plots) the `ggridges` package.

Value

- a `ggplot` object

See Also

- vignette("basic") for example usage

Examples

```r
simulation <- .ffs_cache("foureight_sim_week.rds")

ggplot2::autoplot(simulation) # default is type = "luck"

ggplot2::autoplot(simulation, type = "points")
```

---

**espn_connect**

Connect to a league

Description

See `ffscrapr::espn_connect()` for details.

Value

- a connection object to be used with `ff_*` functions
ffs_add_replacement_level

Add replacement level players to each roster

Description

Add replacement level players to each roster

Usage

```r
ffs_add_replacement_level(
  rosters,
  latest_rankings,
  franchises,
  lineup_constraints,
  pos_filter = c("QB", "RB", "WR", "TE")
)
```

Arguments

- `rosters` a dataframe of rosters as created by `ffs_rosters()`
- `latest_rankings` a dataframe of latest rankings as created by `ff_latest_rankings()`
- `franchises` a dataframe of franchises as created by `ffs_franchises()`
- `lineup_constraints` a dataframe of lineup constraints as created by `ffs_starter_positions`
- `pos_filter` a character vector of positions to filter to, defaults to c("QB", "RB", "WR", "TE", "K")

Value

a dataframe of rosters with replacements

See Also

Other ffscrapr-imports: `ff_connect()`, `ff_scoringhistory()`, `ff_starter_positions()`, `fleaflicker_connect()`, `mfl_connect()`, `sleeper_connect()`
Description

The backbone of the ffsimulator resampling process is coming up with a population of weekly outcomes for every preseason positional rank. This function creates that dataframe by connecting historical FantasyPros.com rankings to nflfastR-based scoring data, as created by ffscrapr::ff_scoringhistory().

Usage

```r
ffs_adp_outcomes(
    scoring_history,
    gp_model = "simple",
    pos_filter = c("QB", "RB", "WR", "TE")
)
```

Arguments

- `scoring_history`:
  a scoring history table as created by ffscrapr::ff_scoringhistory()

- `gp_model`:
  either "simple" or "none" - simple uses the average games played per season for each position/adp combination, none assumes every game is played.

- `pos_filter`:
  a character vector: filter the positions returned to these specific positions, default: c("QB","RB","WR","TE")

Value

A dataframe with position, rank, probability of games played, and a corresponding nested list per row of all week score outcomes.

See Also

- fp_rankings_history for the included historical rankings
- fp_injury_table for the historical injury table
- vignette("custom") for usage details.

Examples

```r
# cached data
scoring_history <- .ffs_cache("mfl_scoring_history.rds")

ffs_adp_outcomes(scoring_history, gp_model = "simple")
ffs_adp_outcomes(scoring_history, gp_model = "none")
```
Description

The backbone of the ffsimulator resampling process is coming up with a population of weekly outcomes for every inseason weekly rank. This function creates that dataframe by connecting historical FantasyPros.com rankings to nflfastR-based scoring data, as created by ffscrepr::ff_scoringhistory().

Usage

ffs_adp_outcomes_week(scoring_history, pos_filter = c("QB", "RB", "WR", "TE"))

Arguments

scoring_history
a scoring history table as created by ffscrepr::ff_scoringhistory()

pos_filter
a character vector: filter the positions returned to these specific positions, default: c("QB", "RB", "WR", "TE")

Value

a dataframe with position, rank, probability of games played, and a corresponding nested list per row of all week score outcomes.

See Also

fp_rankings_history_week for the included historical rankings

Examples

# cached data
scoring_history <- .ffs_cache("mfl_scoring_history.rds")
ffs_adp_outcomes_week(scoring_history, pos_filter = c("QB", "RB", "WR", "TE"))
ffs_build_schedules  Generate fantasy schedules

Description

This function generates random head to head schedules for a given number of seasons, teams, and weeks.

Usage

```r
ffs_build_schedules(
  n_teams = NULL,
  n_seasons = 100,
  n_weeks = 14,
  franchises = NULL,
  seed = NULL
)
```

Arguments

- `n_teams` number of teams in simulation
- `n_seasons` number of seasons to simulate, default = 100
- `n_weeks` number of weeks per season, default = 14
- `franchises` optional: a dataframe of franchises as created by `ffs_franchises()` - over-rides the `n_teams` argument and will attach actual franchise IDs to the schedule output.
- `seed` an integer to control reproducibility

Details

It starts with the circle method for round robin scheduling, grows or shrinks the schedule to match the required number of weeks, and then shuffles both the order that teams are assigned in and the order that weeks are generated. This doesn't "guarantee" unique schedules, but there are `n_teams! x n_weeks!` permutations of the schedule so it's very very likely that the schedules are unique (3x10^18 possible schedules for a 12 team league playing 13 weeks).

Value

a dataframe of schedules

See Also

`vignette("custom")` for example usage
**ffs_copy_template**

**Examples**

```r
ffs_build_schedules(n_teams = 12, n_seasons = 1, n_weeks = 14)
```

---

**ffs_copy_template**  
*Copy simulation template to filename*

**Description**

Creates a simulation template file with all of the components of ff_simulate, ready for tinkering!

**Usage**

```r
ffs_copy_template(
  filename = "ff_simulation.R",
  template = c("season", "week"),
  overwrite = NULL
)
```

**Arguments**

- `filename` New file name, defaults to putting "ff_simulation.R" into your current directory
- `template` choice of template: one of "season" or "week"
- `overwrite` a logical (or NULL) - overwrite if existing file found?

**Value**

a success message signalling success/failure.

**Examples**

```r
tmp <- tempfile()
ffs_copy_template(tmp)
```
ffs_franchises  Get Franchises

**Description**

This function lightly wraps `ffscrapr::ff_franchises()` and adds `league_id`, which is a required column for `ffsimulator`.

**Usage**

`ffs_franchises(conn)`

**Arguments**

- `conn`  
  a connection object as created by `ffscrapr::ff_connect()` and friends.

**Value**

a dataframe of franchises that includes the `league_id` column

**See Also**

vignette("Custom Simulations") for more detailed example usage

**Examples**

```r
# cached examples
conn <- .ffs_cache("mfl_conn.rds")

try({  
  # prevents CRAN connectivity issues, not actually required in normal usage
  ffs_franchises(conn)
})
```

---

ffs_generate_projections  Generate Projections

**Description**

Runs the bootstrapped resampling of player week outcomes on the latest rankings and rosters for a given number of seasons and weeks per season.
Usage

```r
ffs_generate_projections(
  adp_outcomes, 
  latest_rankings, 
  n_seasons = 100, 
  weeks = 1:14, 
  rosters = NULL 
)
```

Arguments

- `adp_outcomes`: a dataframe of adp-based weekly outcomes, as created by `ffs_adp_outcomes()`
- `latest_rankings`: a dataframe of rankings, as created by `ffs_latest_rankings()`
- `n_seasons`: number of seasons, default is 100
- `weeks`: a numeric vector of weeks to simulate, defaults to 1:14
- `rosters`: a dataframe of rosters, as created by `ffs_rosters()` - optional, reduces computation to just rostered players

Value

a dataframe of weekly scores for each player in the simulation, approximately of length `n_seasons` x `n_weeks` x `latest_rankings`

See Also

- vignette("custom") for example usage

Examples

```r
# cached examples
adp_outcomes <- .ffs_cache("adp_outcomes.rds")
latest_rankings <- .ffs_cache("latest_rankings.rds")

ffs_generate_projections(adp_outcomes, latest_rankings)
```

Description

Runs the bootstrapped resampling of player week outcomes on the latest rankings and rosters for a given number of seasons and weeks per season.
Usage

```r
ffs_generate_projections_week(
  adp_outcomes,
  latest_rankings,
  n = 1000,
  rosters = NULL
)
```

Arguments

- `adp_outcomes` a dataframe of adp-based weekly outcomes, as created by `ffs_adp_outcomes()`
- `latest_rankings` a dataframe of rankings, as created by `ffs_latest_rankings()`
- `n` number of weeks to simulate
- `rosters` a dataframe of rosters, as created by `ffs_rosters()` - optional, reduces computation to just rostered players

Value

A dataframe of weekly scores for each player in the simulation, approximately of length `n_seasons x n_weeks x latest_rankings`

See Also

- vignette("custom") for example usage

Examples

```r
# cached examples
adp_outcomes_week <- .ffs_cache("adp_outcomes_week.rds")
latest_rankings_week <- .ffs_cache("latest_rankings_week.rds")

ffs_generate_projections_week(adp_outcomes_week, latest_rankings_week)
```

---

**ffs_latest_rankings**

*Download latest rankings from DynastyProcess GitHub*

Description

Fetches a copy of the latest FantasyPros redraft positional rankings data from DynastyProcess.com’s data repository.

Usage

```r
ffs_latest_rankings(type = c("draft", "week"))
```
Arguments

  type           one of "draft" or "week" - controls whether to pull preseason or inseason rankings.

Details

If you have any issues with the output of this data, please open an issue in the DynastyProcess data repository.

Value

  a dataframe with a copy of the latest FP rankings from DynastyProcess’s data repository

See Also

  https://github.com/dynastyprocess/data

  vignette("custom") for example usage

Examples

  try({ # try block to prevent CRAN-related issues
      ffs_latest_rankings()
    })

ffs_optimise_lineups

Optimise Lineups

Description

  Calculates optimal lineups for all franchises in the dataframe based on a table of lineup constraints.

Usage

  ffs_optimise_lineups(
    roster_scores,
    lineup_constraints,
    lineup_efficiency_mean = 0.775,
    lineup_efficiency_sd = 0.05,
    best_ball = FALSE,
    pos_filter = c("QB", "RB", "WR", "TE")
  )

  ffs_optimize_lineups(
    roster_scores,
  )
lineup_constraints,
lineup_efficiency_mean = 0.775,
lineup_efficiency_sd = 0.05,
best_ball = FALSE,
pos_filter = c("QB", "RB", "WR", "TE")
)

Arguments

roster_scores a dataframe as generated by ffs_score_rosters() - should contain columns like: projected_score, pos, and player_id

lineup_constraints a dataframe as generated by ffscrapr::ff_starter_positions() - should contain columns pos, min, max, and offense_starters

lineup_efficiency_mean the average lineup efficiency to use, defaults to 0.775

lineup_efficiency_sd the standard deviation of lineup efficiency, defaults to 0.05

best_ball a logical: FALSE will apply a lineup efficiency factor and TRUE uses optimal scores as actual scores, default = FALSE

pos_filter a character vector specifying which positions are eligible - defaults to c("QB", "RB", "WR", "TE")

Details

Lineup efficiency is the percentage of optimal/best-ball score that is used as the actual score - by default, the lineup efficiency for a team in non-best-ball settings is normally distributed around a mean of 77.5% and a standard deviation of 5%.

Value

a dataframe of what each team scored for each week

See Also

vignette("custom") for example usage

Examples

# cached examples
roster_scores <- .ffs_cache("roster_scores.rds")
lineup_constraints <- .ffs_cache("mfl_lineup_constraints.rds")

ffs_optimise_lineups(roster_scores, lineup_constraints)
### ffs_repeat_schedules

**Repeat fantasy schedules**

**Description**

This function repeats an actual ffs_schedule() by the appropriate number of seasons.

**Usage**

```r
ffs_repeat_schedules(actual_schedule, n_seasons)
```

**Arguments**

- `actual_schedule`: a schedule retrieved by ffs_schedule()
- `n_seasons`: number of seasons to simulate, default = 100

**Value**

a dataframe of schedules for the simulation

**See Also**

vignette("Custom Simulations") for example usage

**Examples**

```r
try({
  conn <- .ffs_cache("mfl_conn.rds") # cached connection
  actual_schedule <- ffs_schedule(conn)

  ffs_repeat_schedules(actual_schedule = actual_schedule, n_seasons = 10)
})
```

### ffs_rosters

**Get Rosters**

**Description**

This function lightly wraps ffscraper::ff_rosters() and adds fantasypros_id, which is a required column for ffsimulator.
Usage

```r
ffs_rosters(conn)
## S3 method for class 'mfl_conn'
ffs_rosters(conn)
## S3 method for class 'sleeper_conn'
ffs_rosters(conn)
## S3 method for class 'flea_conn'
ffs_rosters(conn)
## S3 method for class 'espn_conn'
ffs_rosters(conn)
```

Arguments

- `conn`: a connection object as created by `ffscrapr::ff_connect()` and friends.

Value

A dataframe of rosters that includes a fantasypros_id column.

See Also

`vignette("custom")` for more detailed example usage.

Examples

```r
# cached examples
conn <- .ffs_cache("mfl_conn.rds")
try({
  # prevents CRAN connectivity issues, not actually required in normal usage
  ffs_rosters(conn)
})
```

---

**ffs_schedule**

*Get Schedule*

Description

This function lightly wraps `ffscrapr::ff_schedule()` and adds league_id, which is a required column for ffsimulator, casts IDs to character, and drops actual games played so as to only simulate unplayed games.
Usage

```
ffs_schedule(conn)
```

Arguments

- `conn`: a connection object as created by `ffs::ff_connect()` and friends.

Value

A dataframe of schedule that includes the league_id column

See Also

- vignette("Custom Simulations") for more detailed example usage

Examples

```
# cached examples
try({
  conn <- .ffs_cache("mfl_conn.rds")
  ffs_schedule(conn)
})
```

---

## ffs_score_rosters

*Join Rosters to Projected Scores*

**Description**

Attaches projected scores to rosters (via an inner-join) and creates a positional ranking column.

**Usage**

```
ffs_score_rosters(projected_scores, rosters)
```

**Arguments**

- `projected_scores`: a dataframe of projected scores, as created by `ffs_generate_projections()`
- `rosters`: a dataframe of rosters, as created by `ffs_rosters()`

**Value**

A dataframe of roster-level projected scores
ffs_starter_positions

Get league starter positions

Description

This function lightly wraps ffscrapr::ff_starter_positions() and cleans up some abbreviations (PK -> K)

Usage

ffs_starter_positions(conn)

Arguments

conn a connection object as created by ffscrapr::ff_connect() and friends.

Value

A tidy dataframe of positional lineup rules, one row per position with minimum and maximum starters as well as total starter calculations.

Examples

# cached examples
try({
  # try block to prevent CRAN-related issues
  conn <- .ffs_cache("mfl_conn.rds")
  ffs_starter_positions(conn)
})
**ffs_summarise_week**

**Summarise simulation outputs**

**Description**

These functions are used to summarise the simulation outputs, typically by joining the optimal scores with a matching schedule.

**Usage**

\[
\text{ffs\_summarise\_week}(\text{optimal\_scores}, \text{schedules})
\]

\[
\text{ffs\_summarise\_season}(\text{summary\_week})
\]

\[
\text{ffs\_summarise\_simulation}(\text{summary\_season})
\]

\[
\text{ffs\_summarise\_inseason}(\text{summary\_week}, n)
\]

**Arguments**

- **optimal\_scores**: a dataframe of optimized lineups as created by `ffs\_optimize\_lineups()`
- **schedules**: a dataframe of schedules as created by `ffs\_build\_schedules()` or `ffs\_actual\_schedules()`
- **summary\_week**: a dataframe as created by `ffs\_summarise\_week()`
- **summary\_season**: a dataframe as created by `ffs\_summarise\_season()`
- **n**: number of weeks

**Value**

- **ffs\_summarise\_week**: a dataframe summarising team results by simulation week
- **ffs\_summarise\_season**: a dataframe summarising franchise results across each simulation season
- **ffs\_summarise\_simulation**: a dataframe summarising franchise results across the simulation
- **ffs\_summarise\_inseason**: a dataframe summarising franchise results for the inseason simulation

**See Also**

`vignette("custom")` for example usage
Examples

```r
# cached examples
optimal_scores <- .ff_cache("optimal_scores.rds")
schedules <- .ff_cache("schedules.rds")

summary_week <- ffs_summarise_week(optimal_scores, schedules)
summary_week
summary_season <- ffs_summarise_season(summary_week)
summary_season
summary_simulation <- ffs_summarise_simulation(summary_season)
summary_simulation
```

---

**ff_connect**

*Connect to a league*

**Description**

See `ffscrapr::ff_connect()` for details.

**Value**

a connection object to be used with `ff_*` functions

**See Also**

Other `ffscrapr`-imports: `espn_connect()`, `ff_scoringhistory()`, `ff_starter_positions()`, `fleaflicker_connect()`, `mfl_connect()`, `sleeper_connect()`

---

**ff_scoringhistory**

*Get league scoring history*

**Description**

See `ffscrapr::ff_scoringhistory` for details.

**Value**

A tidy dataframe of weekly fantasy scoring data, one row per player per week

**See Also**

Other `ffscrapr`-imports: `espn_connect()`, `ff_connect()`, `ff_starter_positions()`, `fleaflicker_connect()`, `mfl_connect()`, `sleeper_connect()`
Description

The main function of the package - uses bootstrap resampling to run fantasy football season simulations supported by historical rankings and nflfastR data, calculating optimal lineups, and returns aggregated results.

Usage

ff_simulate(
  conn,
  n_seasons = 100,
  n_weeks = 14,
  best_ball = FALSE,
  seed = NULL,
  gp_model = c("simple", "none"),
  base_seasons = 2012:2020,
  actual_schedule = FALSE,
  replacement_level = TRUE,
  pos_filter = c("QB", "RB", "WR", "TE", "K"),
  verbose = NULL,
  return = c("default", "all")
)

Arguments

conn an connection to a league made with ff_connect() and friends (required)
n_seasons number of seasons to simulate, default = 100
n_weeks number of weeks per season, default = 14
best_ball a logical: are weekly wins based on optimal lineups?
seed an integer to control reproducibility
gp_model select between "simple", "none" to apply a model for whether a player played in a given game, defaults to "simple"
base_seasons a numeric vector that selects seasons as base data, earliest available is 2012
actual_schedule a logical: use actual ff_schedule? default is FALSE
replacement_level a logical: use best available on waiver as replacement level? defaults to TRUE
pos_filter a character vector of positions to filter/run, default is c("QB","RB","WR","TE","K")
verbose a logical: print status messages? default is TRUE, configure with options(ffsimulator.verbose)
return one of c("default", "all") - what objects to return in the output list
ff_simulate_week

Value

an `ff_simulation` object which can be passed to `plot()` and contains the output data from the simulation.

See Also

`vignette("basic")` for example usage
`vignette("custom")` for examples on using the subfunctions for your own processes.

Examples

try(
  # try block to prevent CRAN-related issues
  conn <- mfl_connect(2021, 22627)
  ff_simulate(conn, n_seasons = 25)
)

---

ff_simulate_week  Simulate Fantasy Week

Description

This function simulates a single upcoming week using the same methodology as in the season-long simulation, `ff_simulate()`.

Usage

```r
ff_simulate_week(
  conn,
  n = 1000,
  best_ball = FALSE,
  seed = NULL,
  base_seasons = 2012:2020,
  actual_schedule = TRUE,
  replacement_level = FALSE,
  pos_filter = c("QB", "RB", "WR", "TE", "K"),
  verbose = NULL,
  return = c("default", "all")
)
```

Arguments

- **conn**: an connection to a league made with `ff_connect()` and friends (required)
- **n**: number of times to simulate the upcoming week, default is 1000
- **best_ball**: a logical: are weekly wins based on optimal lineups?
**ff_starter_positions**

Get league starter positions

---

**Description**

See ffscreapr::ff_starter_positions for details.

**Value**

A tidy dataframe of positional lineup rules, one row per position with minimum and maximum starters as well as total starter calculations.

**See Also**

Other ffscreapr-imports: `espn_connect()`, `ff_connect()`, `ff_scoringhistory()`, `fleaflicker_connect()`, `mfl_connect()`, `sleeper_connect()`
**ff_wins_added**

**Wins Added**

**Description**

(EXPERIMENTAL) This function adds a basic wins-added calculation for each player on every team, presenting the change in wins if that player was removed from the team as the net wins-over-replacement for that player. This can be a bit of a time/compute-expensive calculation.

**Usage**

`ff_wins_added(conn, ...)`

**Arguments**

- `conn`: an connection to a league made with `ff_connect()` and friends (required)
- arguments passed on to `ff_simulate`
- `n_seasons`: number of seasons to simulate, default = 100
- `n_weeks`: number of weeks per season, default = 14
- `best_ball`: a logical: are weekly wins based on optimal lineups?
- `seed`: an integer to control reproducibility
- `gp_model`: select between "simple", "none" to apply a model for whether a player played in a given game, defaults to "simple"
- `base_seasons`: a numeric vector that selects seasons as base data, earliest available is 2012
- `actual_schedule`: a logical: use actual ff_schedule? default is FALSE
- `replacement_level`: a logical: use best available on waiver as replacement level? defaults to TRUE
- `pos_filter`: a character vector of positions to filter/run, default is c("QB","RB","WR","TE","K")
- `verbose`: a logical: print status messages? default is TRUE, configure with options(ffsimulator.verbose)
- `return`: one of c("default", "all") - what objects to return in the output list

**Details**

Runs base simulation once (with the usual parameters available for `ff_simulate`), then for every player on every team (except replacement level players):

- remove them from that specific roster
- reoptimize the lineups just for that roster without the player to calculate what the score ends up being without the player
- summarise the new simulation
- return the delta in wins and points

Summarise wins added as the difference between the sim with the player and the sim without them
fleaflicker_connect

Connect to a league

Value

a dataframe summarising the net effect of each player on their team’s wins

Examples

try({
  # try block to prevent CRAN-related issues
  # n_seasons set so that the example runs more quickly
  ff_wins_added(mfl_connect(2021,54040), n_seasons = 5)
})

fp_injury_table

FP injury table

Description

This dataframe contains a column (prob_gp) for each positional ranking that describes the probability of a player with that preseason ADP playing in a given game. It is modelled from historical rankings data and the number of games played per season for a given positional rank.

Usage

fp_injury_table

Format

An object of class tbl_df (inherits from tbl.data.frame) with 692 rows and 3 columns.
### fp_rankings_history

**Historical draft position ranks**

**Description**

This dataframe has historical positional draft rankings for 2012-2020 QB/RB/WR/TE/PK and 2015-2020 DL/LB/DB, as gathered by the ffpros package.

**Usage**

`fp_rankings_history`

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 10749 rows and 10 columns.

### fp_rankings_history_week

**Historical position ranks**

**Description**

This dataframe has historical positional in-season rankings for 2012-2020 QB/RB/WR/TE/PK and 2015-2020 DL/LB/DB, as gathered by the ffpros package.

**Usage**

`fp_rankings_history_week`

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 76224 rows and 11 columns.
mfl_connect

Connect to a league

Description
See ffscrapr::mfl_connect() for details.

Value
a connection object to be used with ff_* functions

See Also
Other ffscrapr-imports: espn_connect(), ff_connect(), ff_scoringhistory(), ff_starter_positions(), fleadlacker_connect(), sleeper_connect()

sleeper_connect

Connect to a league

Description
See ffscrapr::sleeper_connect() for details.

Value
a connection object to be used with ff_* functions

See Also
Other ffscrapr-imports: espn_connect(), ff_connect(), ff_scoringhistory(), ff_starter_positions(), fleadjlacker_connect(), mfl_connect()
Index

* datasets
  fp_injury_table, 25
  fp_rankings_history, 26
  fp_rankings_history_week, 26
* ffscrap-imports
  espn_connect, 4
  ff_connect, 20
  ff_scoringhistory, 20
  ff_starter_positions, 23
  fleaflicker_connect, 25
  mfl_connect, 27
  sleeper_connect, 27
autoplot.ff_simulation, 2
autoplot.ff_simulation_week, 3
espn_connect, 4, 20, 23, 25, 27
ff_connect, 5, 20, 23, 25, 27
ff_scoringhistory, 5, 20, 23, 25, 27
ff_simulate, 21, 24
ff_simulate_week, 22
ff_starter_positions, 5, 20, 23, 25, 27
ff_wins_added, 24
ffs_add_replacement_level, 5
ffs_adp_outcomes, 6
ffs_adp_outcomes_week, 7
ffs_build_schedules, 8
ffs_copy_template, 9
ffs_franchises, 10
ffs_franchises(), 8
ffs_generate_projections, 10
ffs_generate_projections_week, 11
ffs_latest_rankings, 12
ffs_optimise_lineups, 13
ffs_optimize_lineups
  (ffs_optimise_lineups), 13
ffs_repeat_schedules, 15
ffs_rosters, 15
ffs_schedule, 16
ffs_score_rosters, 17
ffs_starter_positions, 18
ffs_summarise_inseason
  (ffs_summarise_week), 19
ffs_summarise_season
  (ffs_summarise_week), 19
ffs_summarise_simulation
  (ffs_summarise_week), 19
ffs_summarise_week, 19
ffs_summarize_inseason
  (ffs_summarise_week), 19
ffs_summarize_simulation
  (ffs_summarise_week), 19
ffs_summarize_season
  (ffs_summarise_week), 19
ffs_summarize_simulation
  (ffs_summarise_week), 19
ffs_summarize_week
  (ffs_summarise_week), 19
fleaflicker_connect, 5, 20, 23, 25, 27
fp_injury_table, 25
fp_rankings_history, 26
fp_rankings_history_week, 26
mfl_connect, 5, 20, 23, 25, 27
mfl_connect, 5, 20, 23, 25, 27
plot.ff_simulation
  (autoplot.ff_simulation), 2
plot.ff_simulation_week
  (autoplot.ff_simulation_week), 3
sleeper_connect, 5, 20, 23, 25, 27
sleeper_connect, 5, 20, 23, 25, 27