Package ‘ffsimulator’

July 20, 2021

Title  Simulate Fantasy Football Seasons
Version  1.0.0
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), dplyr (>= 1.0.0), ffscrapr (>= 1.4.3), glue (>= 1.3.0), httr (>= 1.4.0), magrittr (>= 1.0.0), purrr (>= 0.3.0), Rglpk (>= 0.6.0), rlang (>= 0.4.0), stringr (>= 1.4.0), tibble (>= 3.0.0), tidyr (>= 1.0.0)
Suggests  covr (>= 3.0.0), forcats (>= 0.5.0), furrr (>= 0.2.0), future (>= 1.0.0), ggplot2 (>= 3.0.0), ggridges (>= 0.5.0), knitr (>= 1.0), rmarkdown (>= 2.6), scales (>= 1.0.0), testthat (>= 3.0.0), vdiffr (>= 1.0.2)
VignetteBuilder  knitr
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.1
Config/testthat/edition  3
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  2021-07-20 07:50:04 UTC
R topics documented:

- autoplot.ff_simulation
- espn_connect
- ffs_adp_outcomes
- ffs_build_schedules
- ffs_generate_projections
- ffs_latest_rankings
- ffs_optimise_lineups
- ffs_rosters
- ffs_score_rosters
- ffs_summarise_week
- ff_connect
- ff_scoringhistory
- ff_simulate
- ff_starter_positions
- fleaflicker_connect
- fp_injury_table
- fp_rankings_history
- mfl_connect
- sleeper_connect

Index

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.
### espn_connect

**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 Simulations") for example usage

**Examples**

```r
simulation <- .fscache("foureight_sim.rds")

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

**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 \texttt{ffscrapr::ff_scoringhistory()}.

Usage

\begin{verbatim}
ffs_adp_outcomes(scoring_history, injury_model = "simple")
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{scoring_history} a scoring history table as created by \texttt{ffscrapr::ff_scoringhistory()}
  \item \texttt{injury_model} either "simple" or "none" - simple uses the average games played per season for each position/adp combination, none assumes every game is played.
\end{itemize}

Value

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

See Also

\begin{itemize}
  \item \texttt{fp_rankings_history} for the included historical rankings
  \item \texttt{fp_injury_table} for the historical injury table
  \item \texttt{vignette("Custom Simulation")} for usage details.
\end{itemize}

Examples

\begin{verbatim}
#cached data
scoring_history <- .ffs_cache("mfl_scoring_history.rds")

ffs_adp_outcomes(scoring_history, injury_model = "simple")
ffs_adp_outcomes(scoring_history, injury_model = "none")
\end{verbatim}
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, n_seasons = 100, n_weeks = 14, 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
- `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 Simulations") for example usage

**Examples**

```r
ffs_build_schedules(n_teams = 12, n_seasons = 1, n_weeks = 14)
```
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,
  n_weeks = 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
- `n_weeks`: weeks per season, default is 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 Simulations")` 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)
```
**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**
`ffs_latest_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](https://github.com/dynastyprocess/data)
- vignette("Custom Simulation") for example usage

**Examples**

`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

```r
ffs_optimise_lineups(
  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 = 0.775,  # the average lineup efficiency to use, defaults to 0.775
  lineup_efficiency_sd = 0.05,  # the standard deviation of lineup efficiency, defaults to 0.05
  best_ball = FALSE,  # a logical: FALSE will apply a lineup efficiency factor and TRUE uses optimal scores as actual scores, default = FALSE
  parallel = FALSE  # a logical: TRUE will run the optimization in parallel, requires the furrr and future packages as well as setting future::plan() in advance/externally. Default FALSE.
)
```

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
- **parallel**: a logical: TRUE will run the optimization in parallel, requires the furrr and future packages as well as setting future::plan() in advance/externally. Default FALSE.

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 Simulations") 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_rosters**

*Get Rosters*

**Description**

This function lightly wraps `ffscrapr::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 Simulations") for more detailed example usage

Examples

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

See Also

vignette("Custom Simulations") for example usage

Examples

# cached examples
projected_scores <- .ffs_cache("projected_scores.rds")
rosters <- .ffs_cache("mfl_rosters.rds")

ffs_score_rosters(projected_scores, rosters)
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

```r
ffs_summarise_week(optimal_scores, schedules)
ffs_summarise_season(summary_week)
ffs_summarise_simulation(summary_season)
ffs_summarize_week(optimal_scores, schedules)
ffs_summarize_season(summary_week)
ffs_summarize_simulation(summary_season)
```

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()`
- `summary_week`: a dataframe as created by `ffs_summarise_week()`
- `summary_season`: a dataframe as created by `ffs_summarise_season()`

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

See Also

- vignette("Custom Simulations") for example usage

Examples

```r
# cached examples
optimal_scores <- .ffs_cache("optimal_scores.rds")
schedules <- .ffs_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()
ff_simulate

Simulate Fantasy Seasons

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,
  injury_model = c("simple", "none"),
  base_seasons = 2012:2020,
  parallel = FALSE
)

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
injury_model select between "simple","none"
base_seasons a numeric vector that selects seasons as base data, earliest available is 2012
parallel a logical: use parallel processing for optimizing lineups, default is FALSE

Value

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

See Also

vignette("Basic Simulations") for example usage
vignette("Custom Simulations") for examples on using the subfunctions for your own processes.
Examples

```r
conn <- mfl_connect(2021, 22627)
ff_simulate(conn, n_seasons = 25)
```

---

**ff_starter_positions**  
*Get league starter positions*

**Description**

See `ffscrapr::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 `ffscrapr`-imports: `espn_connect()`, `ff_connect()`, `ff_scoringhistory()`, `fleaflicker_connect()`, `mfl_connect()`, `sleeper_connect()`

---

**fleaflicker_connect**  
*Connect to a league*

**Description**

See `ffscrapr::fleaflicker_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()`, `mfl_connect()`, `sleeper_connect()`
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 647 rows and 3 columns.

Description

This dataframe has historical positional 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 7503 rows and 10 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()`, `fleaflicker_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()`, `fleaflicker_connect()`, `mfl_connect()`
Index

* datasets
  fp_injury_table, 15
  fp_rankings_history, 15
* ffscrap-imports
  espn_connect, 3
  ff_connect, 12
  ff_scoringhistory, 12
  ff_starter_positions, 14
  fleaflicker_connect, 14
  mfl_connect, 16
  sleeper_connect, 16

autoplot.ff_simulation, 2

espn_connect, 3, 12, 14, 16

ff_connect, 3, 12, 12, 14, 16
ff_scoringhistory, 3, 12, 12, 14, 16
ff_simulate, 13
ff_starter_positions, 3, 12, 14, 14, 16
ffs_adp_outcomes, 4
ffs_build_schedules, 5
ffs_generate_projections, 6
ffs_latest_rankings, 7
ffs_optimise_lineups, 7
ffs_optimize_lineups
  (ffs_optimise_lineups), 7
ffs_rosters, 9
ffs_score_rosters, 10
ffs_summarise_season
  (ffs_summarise_week), 11
ffs_summarise_simulation
  (ffs_summarise_week), 11
ffs_summarise_week, 11
ffs_summarize_season
  (ffs_summarise_week), 11
ffs_summarize_simulation
  (ffs_summarise_week), 11
ffs_summarize_week
  (ffs_summarise_week), 11

mfl_connect, 3, 12, 14, 16, 16

plot.ff_simulation
  (autoplot.ff_simulation), 2

sleeper_connect, 3, 12, 14, 16, 16