Package 'nflfastR'

November 27, 2024

Type Package

Title Functions to Efficiently Access NFL Play by Play Data

Version 5.0.0

Description A set of functions to access National Football League play-by-play data from <https://www.nfl.com/>.

License MIT + file LICENSE

URL https://www.nflfastr.com/, https://github.com/nflverse/nflfastR

BugReports https://github.com/nflverse/nflfastR/issues

Depends R (>= 3.6.0)

Imports cli (>= 3.0.0), curl, data.table (>= 1.15.0), dplyr (>= 1.0.0), fastrmodels (>= 1.0.1), furrr, future, glue, janitor, lifecycle, mgcv, nflreadr (>= 1.2.0), progressr (>= 0.6.0), rlang (>= 0.4.7), stringr (>= 1.4.0), tibble (>= 3.0), tidyr (>= 1.0.0), tidyselect (>= 1.1.0), xgboost (>= 1.1)

Suggests DBI, gsisdecoder, nflseedR (>= 1.0.2), purrr (>= 0.3.0), rmarkdown, RSQLite, testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Sebastian Carl [aut], Ben Baldwin [cre, aut], Lee Sharpe [ctb], Maksim Horowitz [ctb], Ron Yurko [ctb], Samuel Ventura [ctb], Tan Ho [ctb], John Edwards [ctb]

Maintainer Ben Baldwin <bbaldwin206@gmail.com>

Repository CRAN

Date/Publication 2024-11-26 23:10:02 UTC

Contents

`nflfastR-package` *nflfastR: Functions to Efficiently Access NFL Play by Play Data*

Description

A set of functions to access National Football League play-by-play data from https://www.nfl.com/.

Parallel Processing and Progress Updates in nflfastR

Preface:

Prior to nflfastR v4.0, parallel processing could be activated with an argument pp in the relevant functions and progress updates were always shown. Both of these methods are bad practice and were therefore removed in nflfastR v4.0

The next sections describe how to make nflfastR work in parallel processes and show progress updates if the user wants to.

More Speed Using Parallel Processing:

Nearly all nflfastR functions support parallel processing using `furrr::future_map()` if it is enabled by a call to `future::plan()` prior to the function call. Please see the documentation of the functions for detailed information.

As an example, the following code block will resolve all function calls in the current session using multiple sessions in the background and load play-by-play data for the 2018 through 2020 seasons or build them freshly for the 2018 and 2019 Super Bowls:

```
future::plan("multisession")
load_pbp(2018:2020)
build_nflfastR_pbp(c("2018_21_NE_LA", "2019_21_SF_KC"))
```

We recommend choosing a default parallel processing method and saving it as an environment variable in the R user profile to make sure all futures will be resolved with the chosen method by default. This can be done by following the below given steps.

First, run the following line and the file `.Renviron` should be opened automatically. If you haven't saved any environment variables yet, this will be an empty file.

```
usethis::edit_r_environ()
```

In the opened file `.Renviron` add the next line, then save the file and restart your R session. Please note that this example sets "multisession" as default. For most users this should be the appropriate plan but please make sure it truly is.

```
R_FUTURE_PLAN="multisession"
```

After the session is freshly restarted please check if the above method worked by running the next line. If the output is `FALSE` you successfully set up a default non-sequential `future::plan()`. If the output is `TRUE` all functions will behave like they were called with `purrr::map()` and NOT in multisession.

```
inherits(future::plan(), "sequential")
```

For more information on possible plans please see the future package Readme.

For more information on `.Renviron` please see this book chapter.

Get Progress Updates while Functions are Running:

Most nflfastR functions are able to show progress updates using `progressr::progressor()` if they are turned on before the function is called. There are at least two basic ways to do this by either activating progress updates globally (for the current session) with

```
progressr::handlers(global = TRUE)
```

or by piping the function call into `progressr::with_progress()`:

```
load_pbp(2018:2020) %>%
  progressr::with_progress()
```

Just like in the previous section, it is possible to activate global progression handlers by default. This can be done by following the below given steps.

First, run the following line and the file `.Rprofile` should be opened automatically. If you haven't saved any code yet, this will be an empty file.

```
usethis::edit_r_profile()
```

In the opened file `.Rprofile` add the next line, then save the file and restart your R session. All code in this file will be executed when a new R session starts. The part `if (require("progressr"))` makes sure this will only run if the package progressr is installed to avoid crashing R sessions.

```
if (requireNamespace("progressr", quietly = TRUE)) progressr::handlers(global = TRUE)
```

After the session is freshly restarted please check if the above method worked by running the next line. If the output is `TRUE` you successfully activated global progression handlers for all sessions.

```
progressr::handlers(global = NA)
```

For more information how to work with progress handlers please see progressr::progressr.

For more information on `.Rprofile` please see this book chapter.

Author(s)

Maintainer: Ben Baldwin <bbaldwin206@gmail.com>

Authors:

- Sebastian Carl <mrcaseb@gmail.com>

Other contributors:

- Lee Sharpe [contributor]
- Maksim Horowitz <maksim.horowitz@gmail.com> [contributor]
- Ron Yurko <ryurko@stat.cmu.edu> [contributor]
- Samuel Ventura <samventura22@gmail.com> [contributor]
- Tan Ho [contributor]
- John Edwards <edwards1860@gmail.com> [contributor]

See Also

Useful links:

- https://www.nflfastr.com/
- https://github.com/nflverse/nflfastR
- Report bugs at https://github.com/nflverse/nflfastR/issues

add_qb_epa *Compute QB epa*

Description

Compute QB epa

Usage

```
add_qb_epa(pbp, ...)
```

Arguments

pbp	is a Data frame of play-by-play data scraped using `fast_scraper()`.
...	Additional arguments passed to a message function (for internal use).

Details

Add the variable 'qb_epa', which gives QB credit for EPA for up to the point where a receiver lost a fumble after a completed catch and makes EPA work more like passing yards on plays with fumbles

add_xpass *Add expected pass columns*

Description

Build columns from the expected dropback model. Will return `NA` on data prior to 2006 since that was before NFL started marking scrambles. Must be run on a dataframe that has already had `clean_pbp()` run on it. Note that the functions `build_nflfastR_pbp()` and the database function `update_db()` already include this function.

Usage

```
add_xpass(pbp, ...)
```

Arguments

pbp	is a Data frame of play-by-play data scraped using `fast_scraper()`.
...	Additional arguments passed to a message function (for internal use).

Value

The input Data Frame of the parameter pbp with the following columns added:

xpass Probability of dropback scaled from 0 to 1.

pass_oe Dropback percent over expected on a given play scaled from 0 to 100.

add_xyac *Add expected yards after completion (xyac) variables*

Description

Add expected yards after completion (xyac) variables

Usage

```
add_xyac(pbp, ...)
```

Arguments

pbp	is a Data frame of play-by-play data scraped using `fast_scraper()`.
...	Additional arguments passed to a message function (for internal use).

Details

Build columns that capture what we should expect after the catch.

Value

The input Data Frame of the parameter 'pbp' with the following columns added:

xyac_epa Expected value of EPA gained after the catch, starting from where the catch was made. Zero yards after the catch would be listed as zero EPA.

xyac_success Probability play earns positive EPA (relative to where play started) based on where ball was caught.

xyac_fd Probability play earns a first down based on where the ball was caught.

xyac_mean_yardage Average expected yards after the catch based on where the ball was caught.

xyac_median_yardage Median expected yards after the catch based on where the ball was caught.

build_nflfastR_pbp *Build a Complete nflfastR Data Set*

Description

`build_nflfastR_pbp` is a convenient wrapper around 6 nflfastR functions:

- `fast_scraper()`
- `clean_pbp()`
- `add_qb_epa()`
- `add_xyac()`

- add_xpass()
- decode_player_ids()

Please see either the documentation of each function or the nflfastR Field Descriptions website to learn about the output.

Usage

```
build_nflfastR_pbp(
  game_ids,
  dir = getOption("nflfastR.raw_directory", default = NULL),
  ...,
  decode = TRUE,
  rules = TRUE
)
```

Arguments

game_ids	Vector of character ids or a data frame including the variable game_id (see details for further information).
dir	Path to local directory (defaults to option "nflfastR.raw_directory") where nflfastR searches for raw game play-by-play data. See save_raw_pbp() for additional information.
...	Additional arguments passed to the scraping functions (for internal use)
decode	If TRUE, the function decode_player_ids() will be executed.
rules	If FALSE, printing of the header and footer in the console output will be suppressed.

Details

To load valid game_ids please use the package function fast_scraper_schedules().

Value

An nflfastR play-by-play data frame like it can be loaded from https://github.com/nflverse/nflverse-data.

See Also

For information on parallel processing and progress updates please see nflfastR.

Examples

```
# Build nflfastR pbp for the 2018 and 2019 Super Bowls
try({# to avoid CRAN test problems
build_nflfastR_pbp(c("2018_21_NE_LA", "2019_21_SF_KC"))
})

# It is also possible to directly use the
```

```
# output of `fast_scraper_schedules` as input
try({# to avoid CRAN test problems
library(dplyr, warn.conflicts = FALSE)
fast_scraper_schedules(2020) %>%
  slice_tail(n = 3) %>%
  build_nflfastR_pbp()
 })
```

`calculate_expected_points` *Compute expected points*

Description

for provided plays. Returns the data with probabilities of each scoring event and EP added. The following columns must be present: season, home_team, posteam, roof (coded as 'open', 'closed', or 'retractable'), half_seconds_remaining, yardline_100, ydstogo, posteam_timeouts_remaining, defteam_timeouts_remaining

Usage

```
calculate_expected_points(pbp_data)
```

Arguments

`pbp_data`	Play-by-play dataset to estimate expected points for.

Details

Computes expected points for provided plays. Returns the data with probabilities of each scoring event and EP added. The following columns must be present:

- season
- home_team
- posteam
- roof (coded as 'outdoors', 'dome', or 'open'/'closed'/NA (retractable))
- half_seconds_remaining
- yardline_100
- down
- ydstogo
- posteam_timeouts_remaining
- defteam_timeouts_remaining

Value

The original pbp_data with the following columns appended to it:

ep expected points.

no_score_prob probability of no more scoring this half.

opp_fg_prob probability next score opponent field goal this half.

opp_safety_prob probability next score opponent safety this half.

opp_td_prob probability of next score opponent touchdown this half.

fg_prob probability next score field goal this half.

safety_prob probability next score safety this half.

td_prob probability text score touchdown this half.

Examples

```
try({# to avoid CRAN test problems
library(dplyr)
data <- tibble::tibble(
"season" = 1999:2019,
"home_team" = "SEA",
"posteam" = "SEA",
"roof" = "outdoors",
"half_seconds_remaining" = 1800,
"yardline_100" = c(rep(80, 17), rep(75, 4)),
"down" = 1,
"ydstogo" = 10,
"posteam_timeouts_remaining" = 3,
"defteam_timeouts_remaining" = 3
)

nflfastR::calculate_expected_points(data) %>%
  dplyr::select(season, yardline_100, td_prob, ep)
})
```

`calculate_series_conversion_rates`

Compute Series Conversion Information from Play by Play

Description

A "Series" begins on a 1st and 10 and each team attempts to either earn a new 1st down (on offense) or prevent the offense from converting a new 1st down (on defense). Series conversion rate represents how many series have been either converted to a new 1st down or ended in a touchdown. This function computes series conversion rates on offense and defense from nflverse play-by-play data along with other series results. The function automatically removes series that ended in a QB kneel down.

Usage

```
calculate_series_conversion_rates(pbp, weekly = FALSE)
```

Arguments

pbp	Play-by-play data as returned by load_pbp(), build_nflfastR_pbp(), or fast_scraper().
weekly	If TRUE, returns week-by-week stats, otherwise, season-by-season stats in argument pbp.

Value

A data frame of series information including the following columns:

season The NFL season

team NFL team abbreviation

week Week if weekly is TRUE

off_n The number of series the offense played (excludes QB kneel downs, kickoffs, extra point/two point conversion attempts, non-plays, and plays that do not list a "posteam")

off_scr The rate at which a series ended in either new 1st down or touchdown while the offense was on the field

off_scr_1st The rate at which an offense earned a 1st down or scored a touchdown on 1st down

off_scr_2nd The rate at which an offense earned a 1st down or scored a touchdown on 2nd down

off_scr_3rd The rate at which an offense earned a 1st down or scored a touchdown on 3rd down

off_scr_4th The rate at which an offense earned a 1st down or scored a touchdown on 4th down

off_1st The rate of series that ended in a new 1st down while the offense was on the field (does not include offensive touchdown)

off_td The rate of series that ended in an offensive touchdown while the offense was on the field

off_fg The rate of series that ended in a field goal attempt while the offense was on the field

off_punt The rate of series that ended in a punt while the offense was on the field

off_to The rate of series that ended in a turnover (including on downs), in an opponent score, or at the end of half (or game) while the offense was on the field

def_n The number of series the defense played (excludes QB kneel downs, kickoffs, extra point/two point conversion attempts, non-plays, and plays that do not list a "posteam")

def_scr The rate at which a series ended in either new 1st down or touchdown while the defense was on the field

def_scr_1st The rate at which a defense allowed a 1st down or touchdown on 1st down

def_scr_2nd The rate at which a defense allowed a 1st down or touchdown on 2nd down

def_scr_3rd The rate at which a defense allowed a 1st down or touchdown on 3rd down

def_scr_4th The rate at which a defense allowed a 1st down or touchdown on 4th down

def_1st The rate of series that ended in a new 1st down while the defense was on the field (does not include offensive touchdown)

def_td The rate of series that ended in an offensive touchdown while the defense was on the field

def_fg The rate of series that ended in a field goal attempt while the defense was on the field

def_punt The rate of series that ended in a punt while the defense was on the field

def_to The rate of series that ended in a turnover (including on downs), in an opponent score, or at the end of half (or game) while the defense was on the field

Examples

```
try({# to avoid CRAN test problems
  pbp <- nflfastR::load_pbp(2021)

  weekly <- calculate_series_conversion_rates(pbp, weekly = TRUE)
  dplyr::glimpse(weekly)

  overall <- calculate_series_conversion_rates(pbp, weekly = FALSE)
  dplyr::glimpse(overall)
})
```

`calculate_standings` *Compute Division Standings and Conference Seeds from Play by Play*

Description

This function calculates division standings as well as playoff seeds per conference based on either nflverse play-by-play data or nflverse schedule data.

Usage

```
calculate_standings(
  nflverse_object,
  tiebreaker_depth = 3,
  playoff_seeds = NULL
)
```

Arguments

`nflverse_object`

Data object of class `nflverse_data`. Either schedules as returned by `fast_scraper_schedules()` or `nflreadr::load_schedules()`. Or play-by-play data as returned by `load_pbp()`, `build_nflfastR_pbp()`, or `fast_scraper()`.

`tiebreaker_depth`

A single value equal to 1, 2, or 3. The default is 3. The value controls the depth of tiebreakers that shall be applied. The deepest currently implemented tiebreaker is strength of schedule. The following values are valid:

tiebreaker_depth = 1 Break all ties with a coinflip. Fastest variant.

tiebreaker_depth = 2 Apply head-to-head and division win percentage tiebreakers. Random if still tied.

tiebreaker_depth = 3 Apply all tiebreakers through strength of schedule. Random if still tied.

`playoff_seeds` Number of playoff teams per conference. If `NULL` (the default), the function will try to split `nflverse_object` into seasons prior 2020 (6 seeds) and 2020ff (7 seeds). If set to a numeric, it will be used for all seasons in `nflverse_object`!

Value

A tibble with NFL regular season standings

Examples

```
try({# to avoid CRAN test problems
  # load nflverse data both schedules and pbp
  scheds <- fast_scraper_schedules(2014)
  pbp <- load_pbp(c(2018, 2021))

  # calculate standings based on pbp
  calculate_standings(pbp)

  # calculate standings based on schedules
  calculate_standings(scheds)
})
```

---

calculate_stats *Calculate NFL Stats*

Description

Compute various NFL stats based off nflverse Play-by-Play data.

Usage

```
calculate_stats(
  seasons = nflreadr::most_recent_season(),
  summary_level = c("season", "week"),
  stat_type = c("player", "team"),
  season_type = c("REG", "POST", "REG+POST")
)
```

Arguments

`seasons` A numeric vector of 4-digit years associated with given NFL seasons - defaults to latest season. If set to TRUE, returns all available data since 1999.

`summary_level` Summarize stats by `"season"` or `"week"`.

`stat_type` Calculate `"player"` level stats or `"team"` level stats.

season_type	One of "REG", "POST", or "REG+POST". Filters data to regular season ("REG"), post season ("POST") or keeps all data. Only applied if summary_level == "season".

Value

A tibble of player/team stats summarized by season/week.

See Also

nfl_stats_variables for a description of all variables.

https://www.nflfastr.com/articles/stats_variables.html for a searchable table of the stats variable descriptions.

Examples

```
try({# to avoid CRAN test problems
stats <- calculate_stats(2023, "season", "player")
dplyr::glimpse(stats)
})
```

---

calculate_win_probability

Compute win probability

Description

for provided plays. Returns the data with probabilities of winning the game. The following columns must be present: receive_h2_ko (1 if game is in 1st half and possession team will receive 2nd half kickoff, 0 otherwise), home_team, posteam, half_seconds_remaining, game_seconds_remaining, spread_line (how many points home team was favored by), down, ydstogo, yardline_100, posteam_timeouts_remaining, defteam_timeouts_remaining

Usage

```
calculate_win_probability(pbp_data)
```

Arguments

pbp_data	Play-by-play dataset to estimate win probability for.

Details

Computes win probability for provided plays. Returns the data with spread and non-spread-adjusted win probabilities. The following columns must be present:

- receive_2h_ko (1 if game is in 1st half and possession team will receive 2nd half kickoff, 0 otherwise)
- score_differential
- home_team
- posteam
- half_seconds_remaining
- game_seconds_remaining
- spread_line (how many points home team was favored by)
- down
- ydstogo
- yardline_100
- posteam_timeouts_remaining
- defteam_timeouts_remaining

Value

The original pbp_data with the following columns appended to it:

wp win probability.

vegas_wp win probability taking into account pre-game spread.

Examples

```
try({# to avoid CRAN test problems
library(dplyr)
data <- tibble::tibble(
"receive_2h_ko" = 0,
"home_team" = "SEA",
"posteam" = "SEA",
"score_differential" = 0,
"half_seconds_remaining" = 1800,
"game_seconds_remaining" = 3600,
"spread_line" = c(1, 3, 4, 7, 14),
"down" = 1,
"ydstogo" = 10,
"yardline_100" = 75,
"posteam_timeouts_remaining" = 3,
"defteam_timeouts_remaining" = 3
)

nflfastR::calculate_win_probability(data) %>%
  dplyr::select(spread_line, wp, vegas_wp)
})
```

clean_pbp *Clean Play by Play Data*

Description

Clean Play by Play Data

Usage

```
clean_pbp(pbp, ...)
```

Arguments

pbp	is a Data frame of play-by-play data scraped using `fast_scraper()`.
...	Additional arguments passed to a message function (for internal use).

Details

Build columns that capture what happens on all plays, including penalties, using string extraction from play description. Loosely based on Ben's nflfastR guide (https://www.nflfastr.com/articles/beginners_guide.html) but updated to work with the RS data, which has a different player format in the play description; e.g. 24-M.Lynch instead of M.Lynch. The function also standardizes team abbreviations so that, for example, the Chargers are always represented by 'LAC' regardless of which year it was. Starting in 2022, play-by-play data was missing gsis player IDs of rookies. This functions tries to fix as many as possible.

Value

The input Data Frame of the parameter 'pbp' with the following columns added:

success Binary indicator wheter epa > 0 in the given play.

passer Name of the dropback player (scrambles included) including plays with penalties.

passer_jersey_number Jersey number of the passer.

rusher Name of the rusher (no scrambles) including plays with penalties.

rusher_jersey_number Jersey number of the rusher.

receiver Name of the receiver including plays with penalties.

receiver_jersey_number Jersey number of the receiver.

pass Binary indicator if the play was a pass play (sacks and scrambles included).

rush Binary indicator if the play was a rushing play.

special Binary indicator if the play was a special teams play.

first_down Binary indicator if the play ended in a first down.

aborted_play Binary indicator if the play description indicates "Aborted".

play Binary indicator: 1 if the play was a 'normal' play (including penalties), 0 otherwise.

passer_id ID of the player in the 'passer' column.

rusher_id ID of the player in the 'rusher' column.

receiver_id ID of the player in the 'receiver' column.

name Name of the 'passer' if it is not 'NA', or name of the 'rusher' otherwise.

fantasy Name of the rusher on rush plays or receiver on pass plays.

fantasy_id ID of the rusher on rush plays or receiver on pass plays.

fantasy_player_name Name of the rusher on rush plays or receiver on pass plays (from official stats).

fantasy_player_id ID of the rusher on rush plays or receiver on pass plays (from official stats).

jersey_number Jersey number of the player listed in the 'name' column.

id ID of the player in the 'name' column.

out_of_bounds = 1 if play description contains "ran ob", "pushed ob", or "sacked ob"; = 0 otherwise.

home_opening_kickoff = 1 if the home team received the opening kickoff, 0 otherwise.

See Also

For information on parallel processing and progress updates please see nflfastR.

`decode_player_ids` *Decode the player IDs in nflfastR play-by-play data*

Description

Takes all columns ending with `'player_id'` as well as the variables `'passer_id'`, `'rusher_id'`, `'fantasy_id'`, `'receiver_id'`, and `'id'` of an nflfastR play-by-play data set and decodes the player IDs to the commonly known GSIS ID format 00-00xxxxx.

The function uses by default the high efficient decode_ids of the package `gsisdecoder`. In the unlikely event that there is a problem with this function, an nflfastR internal decoder can be used with the option `fast = FALSE`.

The 2022 play by play data introduced new player IDs that can't be decoded with gsisdecoder. In that case, IDs are joined through nflreadr::load_players.

Usage

```
decode_player_ids(pbp, ..., fast = TRUE)
```

Arguments

`pbp`	is a Data frame of play-by-play data scraped using `fast_scraper()`.
`...`	Additional arguments passed to a message function (for internal use).
`fast`	If `TRUE` the IDs will be decoded with the high efficient function decode_ids. If `FALSE` an nflfastR internal function will be used for decoding (it is generally not recommended to do this, unless there is a problem with decode_ids which can take several days to fix on CRAN.)

Value

The input data frame of the parameter pbp with decoded player IDs.

Examples

```
# Decode data frame consisting of some names and ids
decode_player_ids(data.frame(
  name = c("P.Mahomes", "B.Baldwin", "P.Mahomes", "S.Carl", "J.Jones"),
  id = c(
    "32013030-2d30-3033-3338-3733fa30c4fa",
    NA_character_,
    "00-0033873",
    NA_character_,
    "32013030-2d30-3032-3739-3434d4d3846d"
  )
))
```

---

fast_scraper *Get NFL Play by Play Data*

Description

Load and parse NFL play-by-play data and add all of the original nflfastR variables. As nflfastR now provides multiple functions which add information to the output of this function, it is recommended to use build_nflfastR_pbp instead.

Usage

```
fast_scraper(
  game_ids,
  dir = getOption("nflfastR.raw_directory", default = NULL),
  ...,
  in_builder = FALSE
)
```

Arguments

game_ids	Vector of character ids or a data frame including the variable game_id (see details for further information).
dir	Path to local directory (defaults to option "nflfastR.raw_directory") where nflfastR searches for raw game play-by-play data. See save_raw_pbp() for additional information.
...	Additional arguments passed to the scraping functions (for internal use)
in_builder	If TRUE, the final message will be suppressed (for usage inside of build_nflfastR_pbp).

Details

To load valid game_ids please use the package function `fast_scraper_schedules` (the function can directly handle the output of that function)

Value

Data frame where each individual row represents a single play for all passed game_ids containing the following detailed information (description partly extracted from nflscrapR):

play_id Numeric play id that when used with game_id and drive provides the unique identifier for a single play.

game_id Ten digit identifier for NFL game.

old_game_id Legacy NFL game ID.

home_team String abbreviation for the home team.

away_team String abbreviation for the away team.

season_type 'REG' or 'POST' indicating if the game belongs to regular or post season.

week Season week.

posteam String abbreviation for the team with possession.

posteam_type String indicating whether the posteam team is home or away.

defteam String abbreviation for the team on defense.

side_of_field String abbreviation for which team's side of the field the team with possession is currently on.

yardline_100 Numeric distance in the number of yards from the opponent's endzone for the posteam.

game_date Date of the game.

quarter_seconds_remaining Numeric seconds remaining in the quarter.

half_seconds_remaining Numeric seconds remaining in the half.

game_seconds_remaining Numeric seconds remaining in the game.

game_half String indicating which half the play is in, either Half1, Half2, or Overtime.

quarter_end Binary indicator for whether or not the row of the data is marking the end of a quarter.

drive Numeric drive number in the game.

sp Binary indicator for whether or not a score occurred on the play.

qtr Quarter of the game (5 is overtime).

down The down for the given play.

goal_to_go Binary indicator for whether or not the posteam is in a goal down situation.

time Time at start of play provided in string format as minutes:seconds remaining in the quarter.

yrdln String indicating the current field position for a given play.

ydstogo Numeric yards in distance from either the first down marker or the endzone in goal down situations.

ydsnet Numeric value for total yards gained on the given drive.

desc Detailed string description for the given play.

play_type String indicating the type of play: pass (includes sacks), run (includes scrambles), punt, field_goal, kickoff, extra_point, qb_kneel, qb_spike, no_play (timeouts and penalties), and missing for rows indicating end of play.

yards_gained Numeric yards gained (or lost) by the possessing team, excluding yards gained via fumble recoveries and laterals.

shotgun Binary indicator for whether or not the play was in shotgun formation.

no_huddle Binary indicator for whether or not the play was in no_huddle formation.

qb_dropback Binary indicator for whether or not the QB dropped back on the play (pass attempt, sack, or scrambled).

qb_kneel Binary indicator for whether or not the QB took a knee.

qb_spike Binary indicator for whether or not the QB spiked the ball.

qb_scramble Binary indicator for whether or not the QB scrambled.

pass_length String indicator for pass length: short or deep.

pass_location String indicator for pass location: left, middle, or right.

air_yards Numeric value for distance in yards perpendicular to the line of scrimmage at where the targeted receiver either caught or didn't catch the ball.

yards_after_catch Numeric value for distance in yards perpendicular to the yard line where the receiver made the reception to where the play ended.

run_location String indicator for location of run: left, middle, or right.

run_gap String indicator for line gap of run: end, guard, or tackle

field_goal_result String indicator for result of field goal attempt: made, missed, or blocked.

kick_distance Numeric distance in yards for kickoffs, field goals, and punts.

extra_point_result String indicator for the result of the extra point attempt: good, failed, blocked, safety (touchback in defensive endzone is 1 point apparently), or aborted.

two_point_conv_result String indicator for result of two point conversion attempt: success, failure, safety (touchback in defensive endzone is 1 point apparently), or return.

home_timeouts_remaining Numeric timeouts remaining in the half for the home team.

away_timeouts_remaining Numeric timeouts remaining in the half for the away team.

timeout Binary indicator for whether or not a timeout was called by either team.

timeout_team String abbreviation for which team called the timeout.

td_team String abbreviation for which team scored the touchdown.

td_player_name String name of the player who scored a touchdown.

td_player_id Unique identifier of the player who scored a touchdown.

posteam_timeouts_remaining Number of timeouts remaining for the possession team.

defteam_timeouts_remaining Number of timeouts remaining for the team on defense.

total_home_score Score for the home team at the end of the play.

total_away_score Score for the away team at the end of the play.

posteam_score Score the posteam at the start of the play.

defteam_score Score the defteam at the start of the play.

score_differential Score differential between the posteam and defteam at the start of the play.

posteam_score_post Score for the posteam at the end of the play.

defteam_score_post Score for the defteam at the end of the play.

score_differential_post Score differential between the posteam and defteam at the end of the play.

no_score_prob Predicted probability of no score occurring for the rest of the half based on the expected points model.

opp_fg_prob Predicted probability of the defteam scoring a FG next.

opp_safety_prob Predicted probability of the defteam scoring a safety next.

opp_td_prob Predicted probability of the defteam scoring a TD next.

fg_prob Predicted probability of the posteam scoring a FG next.

safety_prob Predicted probability of the posteam scoring a safety next.

td_prob Predicted probability of the posteam scoring a TD next.

extra_point_prob Predicted probability of the posteam scoring an extra point.

two_point_conversion_prob Predicted probability of the posteam scoring the two point conversion.

ep Using the scoring event probabilities, the estimated expected points with respect to the possession team for the given play.

epa Expected points added (EPA) by the posteam for the given play.

total_home_epa Cumulative total EPA for the home team in the game so far.

total_away_epa Cumulative total EPA for the away team in the game so far.

total_home_rush_epa Cumulative total rushing EPA for the home team in the game so far.

total_away_rush_epa Cumulative total rushing EPA for the away team in the game so far.

total_home_pass_epa Cumulative total passing EPA for the home team in the game so far.

total_away_pass_epa Cumulative total passing EPA for the away team in the game so far.

air_epa EPA from the air yards alone. For completions this represents the actual value provided through the air. For incompletions this represents the hypothetical value that could've been added through the air if the pass was completed.

yac_epa EPA from the yards after catch alone. For completions this represents the actual value provided after the catch. For incompletions this represents the difference between the hypothetical air_epa and the play's raw observed EPA (how much the incomplete pass cost the posteam).

comp_air_epa EPA from the air yards alone only for completions.

comp_yac_epa EPA from the yards after catch alone only for completions.

total_home_comp_air_epa Cumulative total completions air EPA for the home team in the game so far.

total_away_comp_air_epa Cumulative total completions air EPA for the away team in the game so far.

total_home_comp_yac_epa Cumulative total completions yac EPA for the home team in the game so far.

total_away_comp_yac_epa Cumulative total completions yac EPA for the away team in the game so far.

total_home_raw_air_epa Cumulative total raw air EPA for the home team in the game so far.

total_away_raw_air_epa Cumulative total raw air EPA for the away team in the game so far.

total_home_raw_yac_epa Cumulative total raw yac EPA for the home team in the game so far.

total_away_raw_yac_epa Cumulative total raw yac EPA for the away team in the game so far.

wp Estimated win probabiity for the posteam given the current situation at the start of the given play.

def_wp Estimated win probability for the defteam.

home_wp Estimated win probability for the home team.

away_wp Estimated win probability for the away team.

wpa Win probability added (WPA) for the posteam.

vegas_wpa Win probability added (WPA) for the posteam: spread_adjusted model.

vegas_home_wpa Win probability added (WPA) for the home team: spread_adjusted model.

home_wp_post Estimated win probability for the home team at the end of the play.

away_wp_post Estimated win probability for the away team at the end of the play.

vegas_wp Estimated win probabiity for the posteam given the current situation at the start of the given play, incorporating pre-game Vegas line.

vegas_home_wp Estimated win probability for the home team incorporating pre-game Vegas line.

total_home_rush_wpa Cumulative total rushing WPA for the home team in the game so far.

total_away_rush_wpa Cumulative total rushing WPA for the away team in the game so far.

total_home_pass_wpa Cumulative total passing WPA for the home team in the game so far.

total_away_pass_wpa Cumulative total passing WPA for the away team in the game so far.

air_wpa WPA through the air (same logic as air_epa).

yac_wpa WPA from yards after the catch (same logic as yac_epa).

comp_air_wpa The air_wpa for completions only.

comp_yac_wpa The yac_wpa for completions only.

total_home_comp_air_wpa Cumulative total completions air WPA for the home team in the game so far.

total_away_comp_air_wpa Cumulative total completions air WPA for the away team in the game so far.

total_home_comp_yac_wpa Cumulative total completions yac WPA for the home team in the game so far.

total_away_comp_yac_wpa Cumulative total completions yac WPA for the away team in the game so far.

total_home_raw_air_wpa Cumulative total raw air WPA for the home team in the game so far.

total_away_raw_air_wpa Cumulative total raw air WPA for the away team in the game so far.

total_home_raw_yac_wpa Cumulative total raw yac WPA for the home team in the game so far.

total_away_raw_yac_wpa Cumulative total raw yac WPA for the away team in the game so far.

punt_blocked Binary indicator for if the punt was blocked.

first_down_rush Binary indicator for if a running play converted the first down.

first_down_pass Binary indicator for if a passing play converted the first down.

first_down_penalty Binary indicator for if a penalty converted the first down.

third_down_converted Binary indicator for if the first down was converted on third down.

third_down_failed Binary indicator for if the posteam failed to convert first down on third down.

fourth_down_converted Binary indicator for if the first down was converted on fourth down.

fourth_down_failed Binary indicator for if the posteam failed to convert first down on fourth down.

incomplete_pass Binary indicator for if the pass was incomplete.

touchback Binary indicator for if a touchback occurred on the play.

interception Binary indicator for if the pass was intercepted.

punt_inside_twenty Binary indicator for if the punt ended inside the twenty yard line.

punt_in_endzone Binary indicator for if the punt was in the endzone.

punt_out_of_bounds Binary indicator for if the punt went out of bounds.

punt_downed Binary indicator for if the punt was downed.

punt_fair_catch Binary indicator for if the punt was caught with a fair catch.

kickoff_inside_twenty Binary indicator for if the kickoff ended inside the twenty yard line.

kickoff_in_endzone Binary indicator for if the kickoff was in the endzone.

kickoff_out_of_bounds Binary indicator for if the kickoff went out of bounds.

kickoff_downed Binary indicator for if the kickoff was downed.

kickoff_fair_catch Binary indicator for if the kickoff was caught with a fair catch.

fumble_forced Binary indicator for if the fumble was forced.

fumble_not_forced Binary indicator for if the fumble was not forced.

fumble_out_of_bounds Binary indicator for if the fumble went out of bounds.

solo_tackle Binary indicator if the play had a solo tackle (could be multiple due to fumbles).

safety Binary indicator for whether or not a safety occurred.

penalty Binary indicator for whether or not a penalty occurred.

tackled_for_loss Binary indicator for whether or not a tackle for loss on a run play occurred.

fumble_lost Binary indicator for if the fumble was lost.

own_kickoff_recovery Binary indicator for if the kicking team recovered the kickoff.

own_kickoff_recovery_td Binary indicator for if the kicking team recovered the kickoff and scored a TD.

qb_hit Binary indicator if the QB was hit on the play.

rush_attempt Binary indicator for if the play was a run.

pass_attempt Binary indicator for if the play was a pass attempt (includes sacks).

sack Binary indicator for if the play ended in a sack.

touchdown Binary indicator for if the play resulted in a TD.

pass_touchdown Binary indicator for if the play resulted in a passing TD.

rush_touchdown Binary indicator for if the play resulted in a rushing TD.

return_touchdown Binary indicator for if the play resulted in a return TD.

extra_point_attempt Binary indicator for extra point attempt.

two_point_attempt Binary indicator for two point conversion attempt.

field_goal_attempt Binary indicator for field goal attempt.

kickoff_attempt Binary indicator for kickoff.

punt_attempt Binary indicator for punts.

fumble Binary indicator for if a fumble occurred.

complete_pass Binary indicator for if the pass was completed.

assist_tackle Binary indicator for if an assist tackle occurred.

lateral_reception Binary indicator for if a lateral occurred on the reception.

lateral_rush Binary indicator for if a lateral occurred on a run.

lateral_return Binary indicator for if a lateral occurred on a return.

lateral_recovery Binary indicator for if a lateral occurred on a fumble recovery.

passer_player_id Unique identifier for the player that attempted the pass.

passer_player_name String name for the player that attempted the pass.

passing_yards Numeric yards by the passer_player_name, including yards gained in pass plays with laterals. This should equal official passing statistics.

receiver_player_id Unique identifier for the receiver that was targeted on the pass.

receiver_player_name String name for the targeted receiver.

receiving_yards Numeric yards by the receiver_player_name, excluding yards gained in pass plays with laterals. This should equal official receiving statistics but could miss yards gained in pass plays with laterals. Please see the description of `lateral_receiver_player_name` for further information.

rusher_player_id Unique identifier for the player that attempted the run.

rusher_player_name String name for the player that attempted the run.

rushing_yards Numeric yards by the rusher_player_name, excluding yards gained in rush plays with laterals. This should equal official rushing statistics but could miss yards gained in rush plays with laterals. Please see the description of `lateral_rusher_player_name` for further information.

lateral_receiver_player_id Unique identifier for the player that received the last(!) lateral on a pass play.

lateral_receiver_player_name String name for the player that received the last(!) lateral on a pass play. If there were multiple laterals in the same play, this will only be the last player who received a lateral. Please see https://github.com/mrcaseb/nfl-data/tree/master/data/lateral_yards for a list of plays where multiple players recorded lateral receiving yards.

lateral_receiving_yards Numeric yards by the `lateral_receiver_player_name` in pass plays with laterals. Please see the description of `lateral_receiver_player_name` for further information.

lateral_rusher_player_id Unique identifier for the player that received the last(!) lateral on a run play.

lateral_rusher_player_name String name for the player that received the last(!) lateral on a run play. If there were multiple laterals in the same play, this will only be the last player who received a lateral. Please see https://github.com/mrcaseb/nfl-data/tree/master/data/lateral_yards for a list of plays where multiple players recorded lateral rushing yards.

lateral_rushing_yards Numeric yards by the `lateral_rusher_player_name` in run plays with laterals. Please see the description of `lateral_rusher_player_name` for further information.

lateral_sack_player_id Unique identifier for the player that received the lateral on a sack.

lateral_sack_player_name String name for the player that received the lateral on a sack.

interception_player_id Unique identifier for the player that intercepted the pass.

interception_player_name String name for the player that intercepted the pass.

lateral_interception_player_id Unique indentifier for the player that received the lateral on an interception.

lateral_interception_player_name String name for the player that received the lateral on an interception.

punt_returner_player_id Unique identifier for the punt returner.

punt_returner_player_name String name for the punt returner.

lateral_punt_returner_player_id Unique identifier for the player that received the lateral on a punt return.

lateral_punt_returner_player_name String name for the player that received the lateral on a punt return.

kickoff_returner_player_name String name for the kickoff returner.

kickoff_returner_player_id Unique identifier for the kickoff returner.

lateral_kickoff_returner_player_id Unique identifier for the player that received the lateral on a kickoff return.

lateral_kickoff_returner_player_name String name for the player that received the lateral on a kickoff return.

punter_player_id Unique identifier for the punter.

punter_player_name String name for the punter.

kicker_player_name String name for the kicker on FG or kickoff.

kicker_player_id Unique identifier for the kicker on FG or kickoff.

own_kickoff_recovery_player_id Unique identifier for the player that recovered their own kickoff.

own_kickoff_recovery_player_name String name for the player that recovered their own kickoff.

blocked_player_id Unique identifier for the player that blocked the punt or FG.

blocked_player_name String name for the player that blocked the punt or FG.

tackle_for_loss_1_player_id Unique identifier for one of the potential players with the tackle for loss.

tackle_for_loss_1_player_name String name for one of the potential players with the tackle for loss.

tackle_for_loss_2_player_id Unique identifier for one of the potential players with the tackle for loss.

tackle_for_loss_2_player_name String name for one of the potential players with the tackle for loss.

qb_hit_1_player_id Unique identifier for one of the potential players that hit the QB. No sack as the QB was not the ball carrier. For sacks please see `sack_player` or `half_sack_*_player`.

qb_hit_1_player_name String name for one of the potential players that hit the QB. No sack as the QB was not the ball carrier. For sacks please see `sack_player` or `half_sack_*_player`.

qb_hit_2_player_id Unique identifier for one of the potential players that hit the QB. No sack as the QB was not the ball carrier. For sacks please see `sack_player` or `half_sack_*_player`.

qb_hit_2_player_name String name for one of the potential players that hit the QB. No sack as the QB was not the ball carrier. For sacks please see `sack_player` or `half_sack_*_player`.

forced_fumble_player_1_team Team of one of the players with a forced fumble.

forced_fumble_player_1_player_id Unique identifier of one of the players with a forced fumble.

forced_fumble_player_1_player_name String name of one of the players with a forced fumble.

forced_fumble_player_2_team Team of one of the players with a forced fumble.

forced_fumble_player_2_player_id Unique identifier of one of the players with a forced fumble.

forced_fumble_player_2_player_name String name of one of the players with a forced fumble.

solo_tackle_1_team Team of one of the players with a solo tackle.

solo_tackle_2_team Team of one of the players with a solo tackle.

solo_tackle_1_player_id Unique identifier of one of the players with a solo tackle.

solo_tackle_2_player_id Unique identifier of one of the players with a solo tackle.

solo_tackle_1_player_name String name of one of the players with a solo tackle.

solo_tackle_2_player_name String name of one of the players with a solo tackle.

assist_tackle_1_player_id Unique identifier of one of the players with a tackle assist.

assist_tackle_1_player_name String name of one of the players with a tackle assist.

assist_tackle_1_team Team of one of the players with a tackle assist.

assist_tackle_2_player_id Unique identifier of one of the players with a tackle assist.

assist_tackle_2_player_name String name of one of the players with a tackle assist.

assist_tackle_2_team Team of one of the players with a tackle assist.

assist_tackle_3_player_id Unique identifier of one of the players with a tackle assist.

assist_tackle_3_player_name String name of one of the players with a tackle assist.

assist_tackle_3_team Team of one of the players with a tackle assist.

assist_tackle_4_player_id Unique identifier of one of the players with a tackle assist.

assist_tackle_4_player_name String name of one of the players with a tackle assist.

assist_tackle_4_team Team of one of the players with a tackle assist.

tackle_with_assist Binary indicator for if there has been a tackle with assist.

tackle_with_assist_1_player_id Unique identifier of one of the players with a tackle with assist.

tackle_with_assist_1_player_name String name of one of the players with a tackle with assist.

tackle_with_assist_1_team Team of one of the players with a tackle with assist.

tackle_with_assist_2_player_id Unique identifier of one of the players with a tackle with assist.

tackle_with_assist_2_player_name String name of one of the players with a tackle with assist.

tackle_with_assist_2_team Team of one of the players with a tackle with assist.

pass_defense_1_player_id Unique identifier of one of the players with a pass defense.

pass_defense_1_player_name String name of one of the players with a pass defense.

pass_defense_2_player_id Unique identifier of one of the players with a pass defense.

pass_defense_2_player_name String name of one of the players with a pass defense.

fumbled_1_team Team of one of the first player with a fumble.

fumbled_1_player_id Unique identifier of the first player who fumbled on the play.

fumbled_1_player_name String name of one of the first player who fumbled on the play.

fumbled_2_player_id Unique identifier of the second player who fumbled on the play.

fumbled_2_player_name String name of one of the second player who fumbled on the play.

fumbled_2_team Team of one of the second player with a fumble.

fumble_recovery_1_team Team of one of the players with a fumble recovery.

fumble_recovery_1_yards Yards gained by one of the players with a fumble recovery.

fumble_recovery_1_player_id Unique identifier of one of the players with a fumble recovery.

fumble_recovery_1_player_name String name of one of the players with a fumble recovery.

fumble_recovery_2_team Team of one of the players with a fumble recovery.

fumble_recovery_2_yards Yards gained by one of the players with a fumble recovery.

fumble_recovery_2_player_id Unique identifier of one of the players with a fumble recovery.

fumble_recovery_2_player_name String name of one of the players with a fumble recovery.

sack_player_id Unique identifier of the player who recorded a solo sack.

sack_player_name String name of the player who recorded a solo sack.

half_sack_1_player_id Unique identifier of the first player who recorded half a sack.

half_sack_1_player_name String name of the first player who recorded half a sack.

half_sack_2_player_id Unique identifier of the second player who recorded half a sack.

half_sack_2_player_name String name of the second player who recorded half a sack.

return_team String abbreviation of the return team.

return_yards Yards gained by the return team.

penalty_team String abbreviation of the team with the penalty.

penalty_player_id Unique identifier for the player with the penalty.

penalty_player_name String name for the player with the penalty.

penalty_yards Yards gained (or lost) by the posteam from the penalty.

replay_or_challenge Binary indicator for whether or not a replay or challenge.

replay_or_challenge_result String indicating the result of the replay or challenge.

penalty_type String indicating the penalty type of the first penalty in the given play. Will be `NA` if `desc` is missing the type.

defensive_two_point_attempt Binary indicator whether or not the defense was able to have an attempt on a two point conversion, this results following a turnover.

defensive_two_point_conv Binary indicator whether or not the defense successfully scored on the two point conversion.

defensive_extra_point_attempt Binary indicator whether or not the defense was able to have an attempt on an extra point attempt, this results following a blocked attempt that the defense recovers the ball.

defensive_extra_point_conv Binary indicator whether or not the defense successfully scored on an extra point attempt.

safety_player_name String name for the player who scored a safety.

safety_player_id Unique identifier for the player who scored a safety.

season 4 digit number indicating to which season the game belongs to.

cp Numeric value indicating the probability for a complete pass based on comparable game situations.

cpoe For a single pass play this is 1 - cp when the pass was completed or 0 - cp when the pass was incomplete. Analyzed for a whole game or season an indicator for the passer how much over or under expectation his completion percentage was.

series Starts at 1, each new first down increments, numbers shared across both teams NA: kickoffs, extra point/two point conversion attempts, non-plays, no posteam

series_success 1: scored touchdown, gained enough yards for first down.

series_result Possible values: First down, Touchdown, Opp touchdown, Field goal, Missed field goal, Safety, Turnover, Punt, Turnover on downs, QB kneel, End of half

start_time Kickoff time in eastern time zone.

order_sequence Column provided by NFL to fix out-of-order plays. Available 2011 and beyond with source "nfl".

time_of_day Time of day of play in UTC "HH:MM:SS" format. Available 2011 and beyond with source "nfl".

stadium Game site name.

weather String describing the weather including temperature, humidity and wind (direction and speed). Doesn't change during the game!

nfl_api_id UUID of the game in the new NFL API.

play_clock Time on the playclock when the ball was snapped.

play_deleted Binary indicator for deleted plays.

play_type_nfl Play type as listed in the NFL source. Slightly different to the regular play_type variable.

special_teams_play Binary indicator for whether play is special teams play from NFL source. Available 2011 and beyond with source "nfl".

st_play_type Type of special teams play from NFL source. Available 2011 and beyond with source "nfl".

end_clock_time Game time at the end of a given play.

end_yard_line String indicating the yardline at the end of the given play consisting of team half and yard line number.

drive_real_start_time Local day time when the drive started (currently not used by the NFL and therefore mostly 'NA').

drive_play_count Numeric value of how many regular plays happened in a given drive.

drive_time_of_possession Time of possession in a given drive.

drive_first_downs Number of forst downs in a given drive.

drive_inside20 Binary indicator if the offense was able to get inside the opponents 20 yard line.

drive_ended_with_score Binary indicator the drive ended with a score.

drive_quarter_start Numeric value indicating in which quarter the given drive has started.

drive_quarter_end Numeric value indicating in which quarter the given drive has ended.

drive_yards_penalized Numeric value of how many yards the offense gained or lost through penalties in the given drive.

drive_start_transition String indicating how the offense got the ball.

drive_end_transition String indicating how the offense lost the ball.

drive_game_clock_start Game time at the beginning of a given drive.

drive_game_clock_end Game time at the end of a given drive.

drive_start_yard_line String indicating where a given drive started consisting of team half and yard line number.

drive_end_yard_line String indicating where a given drive ended consisting of team half and yard line number.

drive_play_id_started Play_id of the first play in the given drive.

drive_play_id_ended Play_id of the last play in the given drive.

fixed_drive Manually created drive number in a game.

fixed_drive_result Manually created drive result.

away_score Total points scored by the away team.

home_score Total points scored by the home team.

location Either 'Home' o 'Neutral' indicating if the home team played at home or at a neutral site.

result Equals home_score - away_score and means the game outcome from the perspective of the home team.

total Equals home_score + away_score and means the total points scored in the given game.

spread_line The closing spread line for the game. A positive number means the home team was favored by that many points, a negative number means the away team was favored by that many points. (Source: Pro-Football-Reference)

total_line The closing total line for the game. (Source: Pro-Football-Reference)

div_game Binary indicator for if the given game was a division game.

roof One of 'dome', 'outdoors', 'closed', 'open' indicating indicating the roof status of the stadium the game was played in. (Source: Pro-Football-Reference)

surface What type of ground the game was played on. (Source: Pro-Football-Reference)

temp The temperature at the stadium only for 'roof' = 'outdoors' or 'open'.(Source: Pro-Football-Reference)

wind The speed of the wind in miles/hour only for 'roof' = 'outdoors' or 'open'. (Source: Pro-Football-Reference)

home_coach First and last name of the home team coach. (Source: Pro-Football-Reference)

away_coach First and last name of the away team coach. (Source: Pro-Football-Reference)

stadium_id ID of the stadium the game was played in. (Source: Pro-Football-Reference)

game_stadium Name of the stadium the game was played in. (Source: Pro-Football-Reference)

See Also

For information on parallel processing and progress updates please see nflfastR.

`build_nflfastR_pbp()`, `save_raw_pbp()`

Examples

```
# Get pbp data for two games
try({# to avoid CRAN test problems
fast_scraper(c("2019_01_GB_CHI", "2013_21_SEA_DEN"))
})


# It is also possible to directly use the
# output of `fast_scraper_schedules` as input
try({# to avoid CRAN test problems
library(dplyr, warn.conflicts = FALSE)
fast_scraper_schedules(2020) %>%
  slice_tail(n = 3) %>%
  fast_scraper()
})
```

`fast_scraper_roster` *Load Team Rosters for Multiple Seasons*

Description

Load Rosters

Usage

```
fast_scraper_roster(...)
```

Arguments

...	Arguments passed on to `nflreadr::load_rosters` `seasons` a numeric vector of seasons to return, defaults to returning this year's data if it is March or later. If set to `TRUE`, will return all available data. Data available back to 1920. `file_type` One of `c("rds", "qs", "csv", "parquet")`. Can also be set globally with `options(nflreadr.prefer)`

Details

See `nflreadr::load_rosters` for details.

Value

A tibble of season-level roster data.

See Also

For information on parallel processing and progress updates please see nflfastR.

Examples

```
# Roster of the 2019 and 2020 seasons
try({# to avoid CRAN test problems
fast_scraper_roster(2019:2020)
})
```

`fast_scraper_schedules`

Load NFL Season Schedules

Description

This returns game/schedule information as maintained by Lee Sharpe.

Usage

```
fast_scraper_schedules(...)
```

Arguments

...	Arguments passed on to `nflreadr::load_schedules` `seasons` a numeric vector of seasons to return, default `TRUE` returns all available data.

Details

See `nflreadr::load_schedules` for details.

Value

A tibble of game information for past and/or future games.

See Also

For information on parallel processing and progress updates please see nflfastR.

Examples

```
# Get schedules for the whole 2015 - 2018 seasons
try({# to avoid CRAN test problems
fast_scraper_schedules(2015:2018)
})
```

`field_descriptions` *nflfastR Field Descriptions*

Description

nflfastR Field Descriptions

Usage

```
field_descriptions
```

Format

A data frame including names and descriptions of all variables in an nflfastR dataset.

See Also

The searchable table on the nflfastR website

Examples

```
field_descriptions
```

load_pbp *Load Play By Play*

Description

Loads play by play seasons from the nflverse-data repository

Usage

```
load_pbp(...)
```

Arguments

...	Arguments passed on to `nflreadr::load_pbp`
	`seasons` A numeric vector of 4-digit years associated with given NFL seasons - defaults to latest season. If set to `TRUE`, returns all available data since 1999.
	`file_type` One of `c("rds", "qs", "csv", "parquet")`. Can also be set globally with `options(nflreadr.prefer)`

Value

The complete nflfastR dataset as returned by `nflfastR::build_nflfastR_pbp()` (see below) for all given `seasons`

See Also

https://nflreadr.nflverse.com/articles/dictionary_pbp.html for a web version of the data dictionary

`dictionary_pbp` for the data dictionary bundled as a package dataframe

https://www.nflfastr.com/reference/build_nflfastR_pbp.html for the nflfastR function `nflfastR::build_nflfastR_pbp()`

Issues with this data should be filed here: https://github.com/nflverse/nflverse-pbp

Examples

```
try({# to avoid CRAN test problems
pbp <- load_pbp(2019:2020)
dplyr::glimpse(pbp)
})
```

load_player_stats *Load Player Level Weekly Stats*

Description

Load Player Level Weekly Stats

Usage

```
load_player_stats(...)
```

Arguments

...	Arguments passed on to nflreadr::load_player_stats
seasons	a numeric vector of seasons to return, defaults to most recent season. If set to TRUE, returns all available data.
stat_type	one of "offense", "defense", or "kicking"
file_type	One of c("rds", "qs", "csv", "parquet"). Can also be set globally with options(nflreadr.prefer)

Value

A tibble of week-level player statistics that aims to match NFL official box scores.

See Also

The function calculate_player_stats() and the corresponding examples on the nflfastR website

Examples

```
try({# to avoid CRAN test problems
stats <- load_player_stats()
dplyr::glimpse(stats)
})
```

missing_raw_pbp *Compute Missing Raw PBP Data on Local Filesystem*

Description

Uses nflreadr::load_schedules() to load game IDs of finished games and compares these IDs to all files saved under dir. This function is intended to serve as input for save_raw_pbp().

Usage

```
missing_raw_pbp(
  dir = getOption("nflfastR.raw_directory", default = NULL),
  seasons = TRUE,
  verbose = TRUE
)
```

Arguments

dir	Path to local directory (defaults to option "nflfastR.raw_directory"). nflfastR will download the raw game files split by season into one sub directory per season.
seasons	a numeric vector of seasons to return, default TRUE returns all available data.
verbose	If TRUE, will print number of missing game files as well as oldest and most recent missing ID to console.

Value

A character vector of missing game IDs. If no files are missing, returns NULL invisibly.

See Also

save_raw_pbp()

Examples

```
try(
missing <- missing_raw_pbp(tempdir())
)
```

---

nfl_stats_variables *NFL Stats Variables*

Description

NFL Stats Variables

Usage

```
nfl_stats_variables
```

Format

A data frame explaining all variables returned by the function calculate_stats().

Examples

```
nfl_stats_variables
```

`report` *Get a Situation Report on System, nflverse Package Versions and Dependencies*

Description

This function gives a quick overview of the versions of R and the operating system as well as the versions of nflverse packages, options, and their dependencies. It's primarily designed to help you get a quick idea of what's going on when you're helping someone else debug a problem.

Usage

```
report(...)
```

Arguments

`...`	Arguments passed on to `nflreadr::nflverse_sitrep`
	`pkg` a character vector naming installed packages, or `NULL` (the default) meaning all nflverse packages. The function checks internally if all packages are installed and informs if that is not the case.
	`recursive` a logical indicating whether dependencies of `pkg` and their dependencies (and so on) should be included. Can also be a character vector listing the types of dependencies, a subset of `c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances")`. Character string `"all"` is shorthand for that vector, character string `"most"` for the same vector without `"Enhances"`, character string `"strong"` (default) for the first three elements of that vector.
	`redact_path` a logical indicating whether options that contain "path" in the name should be redacted, default = TRUE

Details

See `nflreadr::nflverse_sitrep` for details.

Examples

```
report(recursive = FALSE)
nflverse_sitrep(pkg = "nflreadr", recursive = TRUE)
```

save_raw_pbp *Download Raw PBP Data to Local Filesystem*

Description

The functions build_nflfastR_pbp() and fast_scraper() support loading raw pbp data from local file systems instead of Github servers. This function is intended to help setting this up. It loads raw pbp data and saves it in the given directory split by season in subdirectories.

Usage

```
save_raw_pbp(
  game_ids,
  dir = getOption("nflfastR.raw_directory", default = NULL)
)
```

Arguments

game_ids	A vector of nflverse game IDs.
dir	Path to local directory (defaults to option "nflfastR.raw_directory"). nflfastR will download the raw game files split by season into one sub directory per season.

Value

The function returns a data frame with one row for each downloaded file and the following columns:

- success if the HTTP request was successfully performed, regardless of the response status code. This is FALSE in case of a network error, or in case you tried to resume from a server that did not support this. A value of NA means the download was interrupted while in progress.
- status_code the HTTP status code from the request. A successful download is usually 200 for full requests or 206 for resumed requests. Anything else could indicate that the downloaded file contains an error page instead of the requested content.
- resumefrom the file size before the request, in case a download was resumed.
- url final url (after redirects) of the request.
- destfile downloaded file on disk.
- error if success == FALSE this column contains an error message.
- type the Content-Type response header value.
- modified the Last-Modified response header value.
- time total elapsed download time for this file in seconds.
- headers vector with http response headers for the request.

See Also

build_nflfastR_pbp(), missing_raw_pbp()

Examples

```
# CREATE LOCAL TEMP DIRECTORY
local_dir <- tempdir()

# LOAD AND SAVE A GAME TO TEMP DIRECTORY
save_raw_pbp("2021_20_BUF_KC", dir = local_dir)

# REMOVE THE DIRECTORY
unlink(file.path(local_dir, 2021))
```

---

stat_ids *NFL Stat IDs and their Meanings*

Description

NFL Stat IDs and their Meanings

Usage

```
stat_ids
```

Format

A data frame including NFL stat IDs, names and descriptions used in an nflfastR dataset.

Source

http://www.nflgsis.com/gsis/Documentation/Partners/StatIDs.html

Examples

```
stat_ids
```

---

teams_colors_logos *NFL Team names, colors and logo urls.*

Description

NFL Team names, colors and logo urls.

Usage

```
teams_colors_logos
```

Format

A data frame with 36 rows and 10 variables containing NFL team level information, including franchises in multiple cities:

team_abbr Team abbreviation

team_name Complete Team name

team_id Team id used in the roster function

team_nick Nickname

team_conf Conference

team_division Division

team_color Primary color

team_color2 Secondary color

team_color3 Tertiary color

team_color4 Quaternary color

team_logo_wikipedia Url to Team logo on wikipedia

team_logo_espn Url to higher quality logo on espn

team_wordmark Url to team wordmarks

team_conference_logo Url to AFC and NFC logos

team_league_logo Url to NFL logo

The primary and secondary colors have been taken from nfl.com with some modifications for better team distinction and most recent team color themes. The tertiary and quaternary colors are taken from Lee Sharpe's teamcolors.csv who has taken them from the `teamcolors` package created by Ben Baumer and Gregory Matthews. The Wikipeadia logo urls are taken from Lee Sharpe's logos.csv Team wordmarks from nfl.com

Examples

```
teams_colors_logos
```

`update_db` *Update or Create a nflfastR Play-by-Play Database*

Description

`update_db` updates or creates a database with `nflfastR` play by play data of all completed games since 1999.

Usage

```
update_db(
  dbdir = getOption("nflfastR.dbdirectory", default = "."),
  dbname = "pbp_db",
  tblname = "nflfastR_pbp",
  force_rebuild = FALSE,
  db_connection = NULL
)
```

Arguments

dbdir	Directory in which the database is or shall be located. Can also be set globally with options(nflfastR.dbdirectory)
dbname	File name of an existing or desired SQLite database within dbdir
tblname	The name of the play by play data table within the database
force_rebuild	Hybrid parameter (logical or numeric) to rebuild parts of or the complete play by play data table within the database (please see details for further information)
db_connection	A DBIConnection object, as returned by DBI::dbConnect() (please see details for further information)

Details

This function creates and updates a data table with the name tblname within a SQLite database (other drivers via db_connection) located in dbdir and named dbname. The data table combines all play by play data for every available game back to the 1999 season and adds the most recent completed games as soon as they are available for nflfastR.

The argument force_rebuild is of hybrid type. It can rebuild the play by play data table either for the whole nflfastR era (with force_rebuild = TRUE) or just for specified seasons (e.g. force_rebuild = c(2019, 2020)). Please note the following behavior:

- force_rebuild = TRUE: The data table with the name tblname will be removed completely and rebuilt from scratch. This is helpful when new columns are added during the Off-Season.
- force_rebuild = c(2019, 2020): The data table with the name tblname will be preserved and only rows from the 2019 and 2020 seasons will be deleted and re-added. This is intended to be used for ongoing seasons because the NFL fixes bugs in the underlying data during the week and we recommend rebuilding the current season every Thursday during the season.

The parameter db_connection is intended for advanced users who want to use other DBI drivers, such as MariaDB, Postgres or odbc. Please note that the arguments dbdir and dbname are dropped in case a db_connection is provided but the argument tblname will still be used to write the data table into the database.

Index