Package ‘nflreadr’

September 5, 2023

Title Download 'nflverse' Data

Version 1.4.0

Description A minimal package for downloading data from 'GitHub'
repositories of the 'nflverse' project.

License MIT + file LICENSE

URL https://nflreadr.nflverse.com,
https://github.com/nflverse/nflreadr

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

Depends R (>= 3.6.0)

Imports cachem (>= 1.0.0), cli (>= 3.0.0), curl (>= 4.3.0), data.table
(>= 1.14.0), glue (>= 1.4.0), memoise (>= 2.0.0), methods,
rappdirs (>= 0.3.0), rlang (>= 0.4.10), tools, utils

Suggests arrow (>= 6.0.0), covr (>= 3.0.0), DT (>= 0.15.0), fs (>= 1.5.0), gh (>= 1.0.0), knitr (>= 1.0.0), piggyback (>= 0.1.2),
progressr (>= 0.8.0), qs (>= 0.24.0), rmarkdown (>= 2.6.0),
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Tan Ho [aut, cre, cph]

Maintainer Tan Ho <tan@tanho.ca>
R topics documented:
clean_homeaway .............................................. 3
clean_player_names ........................................... 4
clean_team_abbrs ............................................. 5
clear_cache .................................................... 6
csv_from_url .................................................. 6
dictionary_combine ......................................... 10
dictionary_contracts ........................................ 11
dictionary_depth_charts ...................................... 11
dictionary_draft_picks ....................................... 12
dictionary_espn_qbr ......................................... 12
dictionary_ff_opportunity .................................... 13
dictionary_ff_playerids ...................................... 13
dictionary_ff_rankings ....................................... 14
dictionary_fpn_charting ..................................... 14
dictionary_injuries ........................................... 15
dictionary_nextgen_stats ................................... 15
dictionary_participation ...................................... 16
dictionary_pbp ................................................ 16
dictionary_pfr_passing ....................................... 17
dictionary_player_stats ..................................... 17
dictionary_player_stats_def ................................. 18
dictionary_rosters ......................................... 18
dictionary_schedules ........................................ 19
dictionary_snap_counts .................................... 19
dictionary_trades ........................................... 20
get_current_week ............................................ 20
join_coalesce ................................................ 21
load_combine ................................................ 22
load_contracts .............................................. 23
load_depth_charts .......................................... 24
load_draft_picks ........................................... 25
load_espn_qbr .............................................. 26
load_ff_opportunity ......................................... 27
load_ff_playerids .......................................... 28
load_ff_rankings ........................................... 28
load_from_url ............................................... 29
load_fpn_charting ......................................... 30
load_injuries ............................................... 31
load_nextgen_stats ....................................... 32
load_officials ............................................. 33
load_participation ......................................... 34
load_pbp ...................................................... 35
clean_homeaway

Clean Home/Away in dataframes into Team/Opponent dataframes

Description

This function converts dataframes with "home_" and "away_" prefixed columns to "team_" and "opponent_", and doubles the rows. This makes sure that there’s one row for each team (as opposed to one row for each game).

Usage

```
clean_homeaway(dataframe, invert = NULL)
```

Arguments

- **dataframe**: dataframe
- **invert**: a character vector of columns that gets inverted when referring to the away team (e.g. home_spread = 1 gets converted to away_spread = -1)

Value

A dataframe with one row per team (twice as long as the input dataframe)
Examples

# a small example dataframe
s <- data.frame(
  game_type = c("CON", "CON", "SB"),
  away_team = c("TB", "BUF", "KC"),
  away_score = c(31L, 24L, 9L),
  home_team = c("GB", "KC", "TB"),
  home_score = c(26L, 38L, 31L),
  location = c("Home", "Home", "Neutral"),
  result = c(-5L, 14L, 22L),
  spread_line = c(3, 3, -3)
)
clean_homeaway(s, invert = c("result","spread_line"))

____________________________________________________________________
clean_player_names Create Player Merge Names

Description

Applies some name-cleaning heuristics to facilitate joins. These heuristics may include:

- removing periods and apostrophes
- removing common suffixes, such as Jr, Sr, II, III, IV
- converting to lowercase
- using ffscrapr::dp_name_mapping to do common name substitutions, such as Mitch Trubisky to Mitchell Trubisky

Usage

clean_player_names(
  player_name,
  lowercase = FALSE,
  convert_lastfirst = TRUE,
  use_name_database = TRUE
)

Arguments

player_name a character vector of player names
lowercase defaults to FALSE - if TRUE, converts to lowercase
convert_lastfirst defaults to TRUE - converts names from "Last, First" to "First Last"
use_name_database uses internal name database to do common substitutions (Mitchell Trubisky to Mitch Trubisky etc)
**clean_team_abbrs**

**Description**

This function standardizes NFL team abbreviations to nflverse defaults. This helps for joins and plotting, especially with the new nflplotR package!

**Usage**

```r
clean_team_abbrs(abbr, current_location = TRUE, keep_non_matches = TRUE)
```

**Arguments**

- `abbr` a character vector of abbreviations
- `current_location`
  - If TRUE (the default), the abbreviation of the most recent team location will be used.
- `keep_non_matches`
  - If TRUE (the default) an element of `abbr` that can’t be matched to any of the internal mapping vectors will be kept as is. Otherwise it will be replaced with NA.

**Value**

A character vector with the length of `abbr` and cleaned team abbreviations if they are included in `team_abbr_mapping` or `team_abbr_mapping_norelocate` (depending on the value of `current_location`). Non matches may be replaced with NA (depending on the value of `keep_non_matches`).

**Details**

Equivalent to the operation done by `ffscrapr::dp_clean_names()` and uses the same player name database.

**Value**

a character vector of cleaned names

**Examples**

```r
clean_player_names(c("A.J. Green", "Odell Beckham Jr. ", "Le'Veon Bell Sr."))
clean_player_names(c("Trubisky, Mitch", "Atwell, Chatarius", "Elliott, Zeke", "Elijah Moore"), convert_lastfirst = TRUE)
```
`csv_from_url`  

**Load .csv/.csv.gz file from a remote connection**

**Description**  
This is a thin wrapper on data.table::fread, but memoised & cached for twenty four hours.

**Usage**  
`csv_from_url(...)`
Arguments

Arguments passed on to `data.table::fread`

**input** A single character string. The value is inspected and deferred to either `file=` (if no `\n` present), `text=` (if at least one `\n` is present) or `cmd=` (if no `\n` is present, at least one space is present, and it isn’t a file name). Exactly one of `input=`, `file=`, `text=`, or `cmd=` should be used in the same call.

**file** File name in working directory, path to file (passed through `path.expand` for convenience), or a URL starting `http://`, `file://`, etc. Compressed files with extension `.gz` and `.bz2` are supported if the `R.utils` package is installed.

**text** The input data itself as a character vector of one or more lines, for example as returned by `readLines()`.

**cmd** A shell command that pre-processes the file; e.g. `fread(cmd=paste("grep",word,"filename"))`. See Details.

**sep** The separator between columns. Defaults to the character in the set `[,,\t |;:]` that separates the sample of rows into the most number of lines with the same number of fields. Use `NULL` or `""` to specify no separator; i.e. each line a single character column like `base::readLines` does.

**sep2** The separator *within* columns. A list column will be returned where each cell is a vector of values. This is much faster using less working memory than `strsplit` afterwards or similar techniques. For each column `sep2` can be different and is the first character in the same set above `[,,\t |;]`, other than `sep`, that exists inside each field outside quoted regions in the sample. NB: `sep2` is not yet implemented.

**nrows** The maximum number of rows to read. Unlike `read.table`, you do not need to set this to an estimate of the number of rows in the file for better speed because that is already automatically determined by `fread` almost instantly using the large sample of lines. `nrows=0` returns the column names and typed empty columns determined by the large sample; useful for a dry run of a large file or to quickly check format consistency of a set of files before starting to read any of them.

**header** Does the first data line contain column names? Defaults according to whether every non-empty field on the first data line is type character. If so, or `TRUE` is supplied, any empty column names are given a default name.

**na.strings** A character vector of strings which are to be interpreted as `NA` values. By default, `"", ""` for columns of all types, including type character is read as `NA` for consistency. `""`, is unambiguous and read as an empty string. To read `,NA`, as `NA`, set `na.strings="NA"`. To read `,`, as blank string `"", set `na.strings=NULL`. When they occur in the file, the strings in `na.strings` should not appear quoted since that is how the string literal `,"NA"`, is distinguished from `,NA`, for example, when `na.strings="NA"`.

**stringsAsFactors** Convert all character columns to factors?

**verbose** Be chatty and report timings?

**skip** If 0 (default) start on the first line and from there finds the first row with a consistent number of columns. This automatically avoids irregular header information before the column names row. `skip>0` means ignore the first
skip rows manually. `skip="string"` searches for "string" in the file (e.g. a substring of the column names row) and starts on that line (inspired by `read.xls` in package gdata).

**select** A vector of column names or numbers to keep, drop the rest. `select` may specify types too in the same way as `colClasses`; i.e., a vector of `colname=type` pairs, or a list of `type=col(s)` pairs. In all forms of `select`, the order that the columns are specified determines the order of the columns in the result.

**drop** Vector of column names or numbers to drop, keep the rest.

**colClasses** As in `utils::read.csv`; i.e., an unnamed vector of types corresponding to the columns in the file, or a named vector specifying types for a subset of the columns by name. The default, `NULL` means types are inferred from the data in the file. Further, `data.table` supports a named list of vectors of column names or `numbers` where the list names are the class names; see examples. The list form makes it easier to set a batch of columns to be a particular class. When column numbers are used in the list form, they refer to the column number in the file not the column number after `select` or `drop` has been applied. If type coercion results in an error, introduces NaNs, or would result in loss of accuracy, the coercion attempt is aborted for that column with warning and the column’s type is left unchanged. If you really desire data loss (e.g. reading 3.14 as integer) you have to truncate such columns afterwards yourself explicitly so that this is clear to future readers of your code.

**integer64** "integer64" (default) reads columns detected as containing integers larger than $2^{31}$ as type `bit64::integer64`. Alternatively, "double"|"numeric" reads as `utils::read.csv` does; i.e., possibly with loss of precision and if so silently. Or, "character".

**dec** The decimal separator as in `utils::read.csv`. If not "." (default) then usually ",". See details.

**col.names** A vector of optional names for the variables (columns). The default is to use the header column if present or detected, or if not "V" followed by the column number. This is applied after `check.names` and before `key` and `index`.

**check.names** default is `FALSE`. If `TRUE` then the names of the variables in the `data.table` are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by `make.names`) so that they are, and also to ensure that there are no duplicates.

**encoding** default is "unknown". Other possible options are "UTF-8" and "Latin-1". Note: it is not used to re-encode the input, rather enables handling of encoded strings in their native encoding.

**quote** By default ("\""), if a field starts with a double quote, `fread` handles embedded quotes robustly as explained under Details. If it fails, then another attempt is made to read the field as is, i.e., as if quotes are disabled. By setting `quote="",` the field is always read as if quotes are disabled. It is not expected to ever need to pass anything other than "\"" to quote; i.e., to turn it off.

**strip.white** default is `TRUE`. Strips leading and trailing whitespaces of unquoted fields. If `FALSE`, only header trailing spaces are removed.
fill logical (default is FALSE). If TRUE then in case the rows have unequal length, blank fields are implicitly filled.

blank.lines.skip logical, default is FALSE. If TRUE blank lines in the input are ignored.

key Character vector of one or more column names which is passed to setkey. It may be a single comma separated string such as key="x,y,z", or a vector of names such as key=c("x","y","z"). Only valid when argument data.table=TRUE. Where applicable, this should refer to column names given in col.names.

index Character vector or list of character vectors of one or more column names which is passed to setindexv. As with key, comma-separated notation like index="x,y,z" is accepted for convenience. Only valid when argument data.table=TRUE. Where applicable, this should refer to column names given in col.names.

showProgress TRUE displays progress on the console if the ETA is greater than 3 seconds. It is produced in fread's C code where the very nice (but R level) txtProgressBar and tkProgressBar are not easily available.

data.table TRUE returns a data.table. FALSE returns a data.frame. The default for this argument can be changed with options(data.table.fread.datatable=FALSE).

nThread The number of threads to use. Experiment to see what works best for your data on your hardware.

logical01 If TRUE a column containing only 0s and 1s will be read as logical, otherwise as integer.

keepLeadingZeros If TRUE a column containing numeric data with leading zeros will be read as character, otherwise leading zeros will be removed and converted to numeric.

yaml If TRUE, fread will attempt to parse (using yaml.load) the top of the input as YAML, and further to glean parameters relevant to improving the performance of fread on the data itself. The entire YAML section is returned as parsed into a list in the yaml_metadata attribute. See Details.

autostart Deprecated and ignored with warning. Please use skip instead.

tmpdir Directory to use as the tmpdir argument for any tempfile calls, e.g. when the input is a URL or a shell command. The default is tempdir() which can be controlled by setting TMPDIR before starting the R session; see base::tempdir.

tz Relevant to datetime values which have no Z or UTC-offset at the end, i.e. unmarked datetime, as written by utils::write.csv. The default tz="UTC" reads unmarked datetime as UTC POSIXct efficiently. tz="" reads unmarked datetime as type character (slowly) so that as.POSIXct can interpret (slowly) the character datetimes in local timezone; e.g. by using "POSIXct" in colClasses=. Note that fwrite() by default writes datetime in UTC including the final Z and therefore fwrite's output will be read by fread consistently and quickly without needing to use tz= or colClasses=. If the TZ environment variable is set to "UTC" (or "" on non-Windows where unset vs "" is significant) then the R session's timezone is already UTC and tz="" will result in unmarked datetimes being read
as UTC POSIXct. For more information, please see the news items from v1.13.0 and v1.14.0.

**Value**

A dataframe as created by `data.table::fread()`

**Examples**

```r
try({
  # prevents cran errors
  csv_from_url("https://github.com/nflverse/nflverse-data/releases/download/test/combines.csv")
})
```

---

**dictionary_combine**  
*Data Dictionary: Combine*

**Description**

A dataframe containing the data dictionary for `load_combine()`

**Usage**

`dictionary_combine`

**Format**

An object of class `data.frame` with 18 rows and 3 columns.

**See Also**

`vignette("Data Dictionary - Combine")`

[https://nflreadr.nflverse.com/articles/dictionary_combine.html](https://nflreadr.nflverse.com/articles/dictionary_combine.html)
**dictionary_contracts**  
*Data Dictionary: Contracts*

---

**Description**

A dataframe containing the data dictionary for `load_contracts()`

**Usage**

```r
dictionary_contracts
```

**Format**

An object of class `data.frame` with 15 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Contracts")  
https://nflreadr.nflverse.com/articles/dictionary_contracts.html

---

**dictionary_depth_charts**  
*Data Dictionary: Depth Charts*

---

**Description**

A dataframe containing the data dictionary for `load_depth_charts()`

**Usage**

```r
dictionary_depth_charts
```

**Format**

An object of class `data.frame` with 13 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Depth Charts")  
https://nflreadr.nflverse.com/articles/dictionary_depth_charts.html
dictionary_draft_picks

Data Dictionary: Draft Picks

Description
A dataframe containing the data dictionary for `load_draft_picks()`

Usage
dictionary_draft_picks

Format
An object of class `data.frame` with 36 rows and 3 columns.

See Also
vignette("Data Dictionary - Draft Picks")
https://nflreadr.nflverse.com/articles/dictionary_draft_picks.html

dictionary_espn_qbr

Data Dictionary: ESPN QBR

Description
A dataframe containing the data dictionary for `load_espn_qbr()`

Usage
dictionary_espn_qbr

Format
An object of class `data.frame` with 23 rows and 3 columns.

See Also
vignette("Data Dictionary - ESPN QBR")
https://nflreadr.nflverse.com/articles/dictionary_espn_qbr.html
dictionary_ff_opportunity

Data Dictionary: Expected Fantasy Points

Description
A dataframe containing the data dictionary for load_ff_opportunity()

Usage
dictionary_ff_opportunity

Format
An object of class data.frame with 218 rows and 4 columns.

See Also
vignette("Data Dictionary - Expected Fantasy Points")
https://nflreadr.nflverse.com/articles/dictionary_ff_opportunity.html

dictionary_ff_playerids

Data Dictionary: Fantasy Player IDs

Description
A dataframe containing the data dictionary for load_ff_playerids()

Usage
dictionary_ff_playerids

Format
An object of class data.frame with 35 rows and 3 columns.

See Also
vignette("Data Dictionary - FF Player IDs")
https://nflreadr.nflverse.com/articles/dictionary_ff_playerids.html
Data Dictionary: Fantasy Football Rankings

Description
A dataframe containing the data dictionary for load_ff_rankings()

Usage
dictionary_ff_rankings

Format
An object of class data.frame with 25 rows and 3 columns.

See Also
vignette("Data Dictionary - FF Rankings")
https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html

Data Dictionary: FTN Charting Data

Description
A dataframe containing the data dictionary for load_ftn_charting()

Usage
dictionary_ftn_charting

Format
An object of class data.frame with 28 rows and 5 columns.

See Also
vignette("Data Dictionary - FTN Charting")
https://nflreadr.nflverse.com/articles/dictionary_ftn_charting.html
Other ftn_charting: load_ftn_charting()
**dictionary_injuries**  
*Data Dictionary: Injuries*

**Description**
A dataframe containing the data dictionary for `load_injuries()`

**Usage**
dictionary_injuries

**Format**
An object of class `data.frame` with 16 rows and 3 columns.

**See Also**
`vignette("Data Dictionary - Injuries")`  
https://nflreadr.nflverse.com/articles/dictionary_injuries.html

---

**dictionary_nextgen_stats**  
*Data Dictionary: Next Gen Stats*

**Description**
A dataframe containing the data dictionary for `load_nextgen_stats()`

**Usage**
dictionary_nextgen_stats

**Format**
An object of class `data.frame` with 51 rows and 3 columns.

**See Also**
`vignette("Data Dictionary - Next Gen Stats")`  
https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html
**dataDictionary_pbp**

**dictionary_participation**

*Data Dictionary: Participation*

**Description**

A dataframe containing the data dictionary for `load_participation()`

**Usage**

`dictionary_participation`

**Format**

An object of class `data.frame` with 14 rows and 3 columns.

**See Also**

`vignette("Data Dictionary - Participation")`

[https://nflreadr.nflverse.com/articles/dictionary_participation.html](https://nflreadr.nflverse.com/articles/dictionary_participation.html)

**dictionary_pbp**

*Data Dictionary: Play by Play*

**Description**

A dataframe containing the data dictionary for `load_pbp()`

**Usage**

`dictionary_pbp`

**Format**

An object of class `data.frame` with 372 rows and 3 columns.

**See Also**

`vignette("Data Dictionary - PBP")`

[https://nflreadr.nflverse.com/articles/dictionary_pbp.html](https://nflreadr.nflverse.com/articles/dictionary_pbp.html)
dictionary_pfr_passing

Data Dictionary: PFR Passing

Description

A dataframe containing the data dictionary for `load_pfr_passing()`

Usage

dictionary_pfr_passing

Format

An object of class `data.frame` with 28 rows and 3 columns.

See Also

https://nflreadr.nflverse.com/articles/dictionary_pfr_passing.html
vignette("Data Dictionary - PFR Passing")

dictionary_player_stats

Data Dictionary: Player Stats

Description

A dataframe containing the data dictionary for `load_player_stats()`

Usage

dictionary_player_stats

Format

An object of class `data.frame` with 48 rows and 2 columns.

See Also

vignette("Data Dictionary - Player Stats")
https://nflreadr.nflverse.com/articles/dictionary_player_stats.html
**dictionary_player_stats_def**

*Data Dictionary: Player Stats Defense*

**Description**

A dataframe containing the data dictionary for `load_player_stats()`

**Usage**

```r
dictionary_player_stats_def
```

**Format**

An object of class `data.frame` with 22 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Player Stats Defense")
https://nflreadr.nflverse.com/articles/dictionary_player_stats_def.html

---

**dictionary_rosters**

*Data Dictionary: Rosters*

**Description**

A dataframe containing the data dictionary for `load_rosters()`

**Usage**

```r
dictionary_rosters
```

**Format**

An object of class `data.frame` with 25 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Rosters")
https://nflreadr.nflverse.com/articles/dictionary_rosters.html
**dictionary_schedules**  
*Data Dictionary: Schedules*

**Description**

A dataframe containing the data dictionary for `load_schedules()`

**Usage**

dictionary_schedules

**Format**

An object of class `data.frame` with 45 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Schedules")

https://nflreadr.nflverse.com/articles/dictionary_schedules.html

---

**dictionary_snap_counts**  
*Data Dictionary: Snap Counts*

**Description**

A dataframe containing the data dictionary for `load_snap_counts()`

**Usage**

dictionary_snap_counts

**Format**

An object of class `data.frame` with 16 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Snap Counts")

https://nflreadr.nflverse.com/articles/dictionary_snap_counts.html
dictionary_trades  

**Data Dictionary: Trades**

**Description**

A dataframe containing the data dictionary for `load_trades()`

**Usage**

`dictionary_trades`

**Format**

An object of class `data.frame` with 11 rows and 3 columns.

**See Also**

vignette("Data Dictionary - Trades")

https://nflreadr.nflverse.com/articles/dictionary_trades.html

get_current_week  

**Get Current Week**

**Description**

A helper function that returns the upcoming NFL regular season week based on either the nflverse schedules file (as found in `load_schedules()`) or some date-based heuristics (number of weeks since the first Monday of September)

**Usage**

`get_current_week(use_date = FALSE)`

**Arguments**

- `use_date`  
  a logical to determine whether to use date-based heuristics to determine current week, default FALSE (i.e. uses schedule file)

**Details**

Note that the date heuristic will count a new week starting on Thursdays, while the schedule-based method will count a new week after the last game of the previous week, e.g. after MNF is completed. Tan and Ben argued for a while about this.

**Value**

current nfl regular season week as a numeric
Description

EXPERIMENTAL! This function joins two dataframes together by key, and then coalesces any columns that have shared names (i.e. fills in NAs). A utility function primarily used internally within nflverse to help build player IDs

Usage

```r
join_coalesce(
  x,
  y,
  by = NULL,
  type = c("left", "inner", "full"),
  ..., 
  by.x = NULL,
  by.y = NULL,
  sort = TRUE,
  incomparables = c(NA, NaN)
)
```

Arguments

- `x, y` dataframes. Will be coerced to data.table
- `by` keys to join on, as a plain or named character vector
- `type` one of "left" (all rows of x and matching rows of y), "inner" (matching rows of x and y), "full" (all rows of x and y)
- `...` other args passed to `merge.data.frame()`
by.x, by.y alternate form of keys to join on - if provided, will override by.

sort whether to sort output by the join keys

incomparables keys to NOT match on, i.e. NA should not match on NA.

Value
a data.table joining x and y dataframes together, with every column from both x and y and patching NA values in x with those in y.

Examples

x <- data.frame(id1 = c(NA_character_, letters[1:4]), a = c(1,NA,3,NA,5), b = 1:5 * 10)
y <- data.frame(id2 = c(letters[3:11],NA_character_), a = -(1:10), c = 1:10 * 100)

join_coalesce(x,y, by = c("id1"="id2"))
join_coalesce(x,y, by.x = "id1", by.y = "id2")
join_coalesce(x,y, by = c("id1"="id2"), type = "inner")
join_coalesce(x,y, by = c("id1"="id2"), type = "full")

load_combine

Description
Loads combine data since 2000 courtesy of PFR.

Usage

load_combine(
  seasons = TRUE,
  file_type = getOption("nflreadr.prefer", default = "rds")
)

Arguments

seasons a numeric vector of seasons to return, default TRUE returns all available data

file_type One of c("rds", "qs", "csv", "parquet"). Can also be set globally with options(nflreadr.prefer)

Value
A tibble of NFL combine data provided by Pro Football Reference.
**load_contracts**

See Also

Issues with this data should be filed here: https://github.com/nflverse/nflverse-data
https://nflreadr.nflverse.com/articles/dictionary_combine.html for a web version of the dictionary
dictionary_combine for the data dictionary as bundled within the package

Examples

```r
try({
  # prevents cran errors
  load_combine()
})
```

---

**load_contracts**

Load Historical Player Contracts from OverTheCap.com

Description

Loads player contracts from OverTheCap.com

Usage

```r
load_contracts(file_type = getOption("nflreadr.prefer", default = "rds"))
```

Arguments

- `file_type` One of "rds", "qs", "csv", or "parquet". Can also be set globally with options(nflreadr.prefer)

Value

A tibble of active and non-active NFL player contracts.

See Also

https://overthecap.com/contract-history for a web version of the data
https://nflreadr.nflverse.com/articles/dictionary_contracts.html for a web version of the dictionary
dictionary_contracts for the data dictionary as bundled within the package

Issues with this data should be filed here: https://github.com/nflverse/rotc
Examples

```r
try({
  # prevents cran errors
  load_contracts()
})
```

---

**load_depth_charts**  
*Load Weekly Depth Charts*

**Description**

Loads depth charts for each NFL team for each week back to 2001.

**Usage**

```r
load_depth_charts(
  seasons = most_recent_season(),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**Arguments**

- `seasons`  
a numeric vector specifying what seasons to return, if TRUE returns all available data. Defaults to latest season.

- `file_type`  
One of c("rds", "qs", "csv", "parquet"). Can also be set globally with `options(nflreadr.prefer)`

**Value**

A tibble of week-level depth charts for each team.

**See Also**

[https://nflreadr.nflverse.com/articles/dictionary_depth_charts.html](https://nflreadr.nflverse.com/articles/dictionary_depth_charts.html) for a web version of the dictionary

**dictionary_depth_charts** for the data dictionary as bundled within the package

Issues with this data should be filed here: [https://github.com/nflverse/nflverse-data](https://github.com/nflverse/nflverse-data)
load_draft_picks

Examples

```r
try({ # prevents cran errors
    load_depth_charts(2020)
})
```

---

**load_draft_picks**  
*Load Draft Picks from PFR*

---

**Description**

Loads every draft pick since 1980 courtesy of PFR.

**Usage**

```r
load_draft_picks(
    seasons = TRUE,
    file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**Arguments**

- `seasons` a numeric vector of seasons to return, default `TRUE` returns all available data
- `file_type` One of `c("rds", "qs", "csv", "parquet")`. Can also be set globally with `options(nflreadr.prefer)`

**Value**

A tibble of NFL draft picks provided by Pro Football Reference.

**See Also**

- [https://nflreadr.nflverse.com/articles/dictionary_draft_picks.html](https://nflreadr.nflverse.com/articles/dictionary_draft_picks.html) for the web data dictionary
- `dictionary_draft_picks` for the data dictionary as bundled within the package
- Issues with this data should be filed here: [https://github.com/nflverse/nflverse-data](https://github.com/nflverse/nflverse-data)

**Examples**

```r
try({ # prevents cran errors
    load_draft_picks()
})
```
load_espn_qbr  

Load ESPN's QBR

Description

Load ESPN's QBR

Usage

load_espn_qbr(
  league = c("nfl", "college"),
  seasons = most_recent_season(),
  summary_type = c("season", "weekly")
)

Arguments

- **league**: One of "nfl" or "college", defaults to "nfl"
- **seasons**: a numeric vector of seasons to return, data available since 2006. Defaults to latest season available. TRUE will select all seasons.
- **summary_type**: One of "season" or "weekly", defaults to season

Value

a tibble of season-level injury report data.

See Also

- [https://nflreadr.nflverse.com/articles/dictionary_espn_qbr.html](https://nflreadr.nflverse.com/articles/dictionary_espn_qbr.html) for a web version of the dictionary
- `dictionary_espn_qbr` for the data dictionary as bundled within the package

Issues with this data should be filed here: [https://github.com/nflverse/espnscrapeR-data](https://github.com/nflverse/espnscrapeR-data)

Examples

load_espn_qbr("nfl", 2020)
load_ff_opportunity

Load Expected Fantasy Points

Description

This function downloads precomputed expected points data from ffopportunity automated releases.

Usage

load_ff_opportunity(
  seasons = most_recent_season(),
  stat_type = c("weekly", "pbp_pass", "pbp_rush"),
  model_version = c("latest", "v1.0.0")
)

Arguments

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 "weekly", "pbp_pass", "pbp_rush"
model_version  one of "latest" or "v1.0.0"

Value

Precomputed expected fantasy points data from the ffopportunity automated releases.

See Also

https://ffopportunity.ffverse.com for more on the package, data, and modelling
https://nflreadr.ffverse.com/articles/dictionary_ff_opportunity.html for the web data dictionary
dictionary_ff_opportunity for the data dictionary bundled as a package data frame
Issues with this data should be filed here: https://github.com/ffverse/ffopportunity

Examples

try({ # prevents cran errors
  load_ff_opportunity()
  load_ff_opportunity(seasons = 2021, stat_type = "pbp_pass", model_version = "v1.0.0")
})
`load_ff_playerids`  *Load Fantasy Player IDs*

**Description**
Accesses DynastyProcess.com’s database of fantasy football player IDs, which help connect nflverse to various other platforms and IDs.

**Usage**
```r
load_ff_playerids()
```

**Value**
a dataframe of player IDs

**See Also**
- [https://nflreadr.nflverse.com/articles/dictionary_ff_playerids.html](https://nflreadr.nflverse.com/articles/dictionary_ff_playerids.html) for the web data dictionary
- Issues with this data should be filed here: [https://github.com/dynastyprocess/data](https://github.com/dynastyprocess/data)

**Examples**
```r
try({
  # prevents cran errors
  load_ff_playerids()
})
```

`load_ff_rankings`  *Load Latest FantasyPros Rankings*

**Description**
Accesses DynastyProcess.com’s repository of the latest FP expert consensus rankings - updated on a weekly basis.

**Usage**
```r
load_ff_rankings(type = c("draft", "week", "all"))
```

**Arguments**
- `type`  
  one of "draft" (preseason), "week" (this week, inseason), or "all" (full archive)
load_from_url

**Value**

a dataframe of expert consensus rankings

**See Also**

[https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html](https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html) for the web data dictionary

[https://www.fantasypros.com](https://www.fantasypros.com) for the source of data

Issues with this data should be filed here: [https://github.com/dynastyprocess/data](https://github.com/dynastyprocess/data)

**Examples**

```r
try({ # prevents cran errors
  load_ff_rankings()
})
```

---

### load_from_url

**Load any rds/csv/csv.gz/parquet/qs file from a remote URL**

**Description**

Load any rds/csv/csv.gz/parquet/qs file from a remote URL

**Usage**

```r
load_from_url(url, ..., seasons = TRUE, nflverse = FALSE)
```

**Arguments**

- `url` a vector of URLs to load into memory. If more than one URL provided, will row-bind them.
- `...` named arguments that will be added as attributes to the data, e.g. `nflverse_type = "pbp"`
- `seasons` a numeric vector of years that will be used to filter the dataframe's season column. If TRUE (default), does not filter.
- `nflverse` TRUE to add nflverse_data classing and attributes.

**Value**

a dataframe, possibly of type `nflverse_data`
load_ftn_charting

Load FTN Charting Data

Description

FTN Data manually charts plays and has graciously provided a subset of their charting data to be published via the nflverse. Data is available from the 2022 season onwards and is charted within 48 hours following each game. This data is released under the CC-BY-SA 4.0 Creative Commons license and attribution must be made to FTN Data via nflverse.

Usage

load_ftn_charting(seasons = most_recent_season())

Arguments

seasons

a numeric vector of seasons to return, defaults to most recent season. If set to TRUE, returns all available data. Data available from 2022 onwards.

Value

Precomputed expected fantasy points data from the ffoportunity automated releases.

Author(s)

FTN Data

Source

FTNData.com

See Also

https://www.ftndata.com
vignette("Data Dictionary - FTN Charting")
https://nflreadr.nflverse.com/articles/dictionary_ftn_charting.html for the web data dictionary

Other ftn_charting: dictionary_ftn_charting
Examples

try({ # prevents cran errors
    load_ftn_charting()
})

load_injuries  Load Injury Reports

Description

Data collected from an API for weekly injury report data.

Usage

load_injuries(
    seasons = most_recent_season(),
    file_type = getOption("nflreadr.prefer", default = "rds")
)

Arguments

seasons a numeric vector of seasons to return, data available since 2009. Defaults to latest season available.

file_type One of c("rds", "qs", "csv", "parquet"). Can also be set globally with options(nflreadr.prefer)

Value

a tibble of season-level injury report data.

See Also

https://nflreadr.nflverse.com/articles/dictionary_injuries.html for a web version of the dictionary
dictionary_injuries for the data dictionary as bundled within the package

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

Examples

try({ # prevents cran errors
    load_injuries(2020)
})
**load_nextgen_stats**  

*Load Player Level Weekly NFL Next Gen Stats*

**Description**

Loads player level weekly stats provided by NFL Next Gen Stats starting with the 2016 season. Three different stat types are available and the current season's data updates every night. NGS will only provide data for players above a minimum number of pass/rush/rec attempts.

**Usage**

```r
load_nextgen_stats(
  seasons = TRUE,
  stat_type = c("passing", "receiving", "rushing"),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**Arguments**

- **seasons**  
  a numeric vector specifying what seasons to return, if TRUE returns all available data

- **stat_type**  
  one of "passing", "receiving", or "rushing"

- **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 provided by NFL Next Gen Stats. Regular season summary is given for week == 0.

**See Also**

- [https://nextgenstats.nfl.com/stats/passing](https://nextgenstats.nfl.com/stats/passing) for stat_type = "passing"
- [https://nextgenstats.nfl.com/stats/receiving](https://nextgenstats.nfl.com/stats/receiving) for stat_type = "receiving"
- [https://nextgenstats.nfl.com/stats/rushing](https://nextgenstats.nfl.com/stats/rushing) for stat_type = "rushing"
- [https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html](https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html) for a web version of the data dictionary
- [dictionary_nextgen_stats](https://github.com/nflverse/nflverse-data) for the data dictionary as bundled within the package

Issues with this data should be filed here: [https://github.com/nflverse/nflverse-data](https://github.com/nflverse/nflverse-data)
load_officials

Examples

```r
try({
  # prevents cran errors
  load_nextgen_stats(stat_type = "passing")
  load_nextgen_stats(stat_type = "receiving")
  load_nextgen_stats(stat_type = "rushing")
})
```

load_officials  Load Officials

Description

Loads data on which officials are assigned to oversee a specific game. Data available from 2015 onwards.

Usage

```r
load_officials(
  seasons = TRUE,
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

Arguments

- `seasons` a numeric vector specifying what seasons to return, if TRUE returns all available data
- `file_type` One of c("rds", "qs", "csv", "parquet"). Can also be set globally with `options(nflreadr.prefer)`

Value

A tibble with one row per game per official.

See Also

Issues with this data should be filed here: [https://github.com/nflverse/nflreadr](https://github.com/nflverse/nflreadr) and it will be triaged appropriately.
load_participation

Examples

```r
try({ # prevents cran errors
    load_officials()
})
```

---

**load_participation**  
*Load Participation Data*

**Description**

Loads participation data from the nflverse-data repository

**Usage**

```r
load_participation(
    seasons = most_recent_season(),
    include_pbp = FALSE,
    file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**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 2016.

- `include_pbp`  
  a logical: download and join pbp to this data?

- `file_type`  
  One of c("rds", "qs", "csv", "parquet"). Can also be set globally with `options(nflreadr.prefer)`

**Value**

A dataframe of participation data, optionally merged with play by play

**Examples**

```r
try({ # prevents cran errors
    load_participation(seasons = 2020, include_pbp = TRUE)
})
```
**Description**

Loads play by play seasons from the nflverse-data repository

**Usage**

```r
load_pbp(
  seasons = most_recent_season(),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**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.

- **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](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](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](https://github.com/nflverse/nflverse-pbp)

**Examples**

```r
try({# prevents cran errors
  load_pbp(2019:2020)
})
```
load_pfr_advstats | Load Advanced Stats from PFR

**Description**

Loads player level season stats provided by Pro Football Reference starting with the 2018 season, primarily to augment existing nflverse data.

**Usage**

```r
load_pfr_advstats(
  seasons = most_recent_season(),
  stat_type = c("pass", "rush", "rec", "def"),
  summary_level = c("week", "season"),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

**Arguments**

- **seasons**: a numeric vector specifying what seasons to return, if TRUE returns all available data
- **stat_type**: one of "pass", "rush", "rec", "def"
- **summary_level**: one of "week" (default) or "season" - some data is only available at the season level
- **file_type**: One of c("rds", "qs", "csv", "parquet"). Can also be set globally with options(nflreadr.prefer)

**Value**

A tibble of player statistics provided by Pro Football Reference that supplements data in nflverse

**See Also**

- [https://nflreadr.nflverse.com/articles/dictionary_pfr_passing.html](https://nflreadr.nflverse.com/articles/dictionary_pfr_passing.html) for the web data dictionary

Issues with this data should be filed here: [https://github.com/nflverse/nflverse-data](https://github.com/nflverse/nflverse-data)

**Examples**

```r
try({  
  # prevents cran errors  
  load_pfr_advstats()  
})
```
load_players

**Load Players**

**Description**

Load a dataframe of player-level information, including IDs and other mostly-immutable data (birthdates, college, draft position etc.)

**Usage**

```r
load_players(file_type =getOption("nflreadr.prefer", default = "rds"))
```

**Arguments**

- **file_type**
  
  One of c("rds", "qs", "csv", "parquet"). Can also be set globally with `options(nflreadr.prefer)`

**Value**

A tibble with one row per player.

**See Also**

Issues with this data should be filed here: [https://github.com/nflverse/nflreadr](https://github.com/nflverse/nflreadr) and it will be triaged appropriately.

**Examples**

```r
try({ # prevents cran errors
    load_players()
})
```

load_player_stats

**Load Player Level Weekly Stats**

**Description**

Load Player Level Weekly Stats
Usage

```r
load_player_stats(
  seasons = most_recent_season(),
  stat_type = c("offense", "defense", "kicking"),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

Arguments

- `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

- [https://nflreadr.nflverse.com/articles/dictionary_player_stats.html](https://nflreadr.nflverse.com/articles/dictionary_player_stats.html) for a web version of the data dictionary
- `dictionary_player_stats` for the data dictionary

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

Examples

```r
try(
  # prevents cran errors
  load_player_stats()
  load_player_stats(stat_type = "kicking")
)
```

---

**load_rosters** *Load Rosters*

Description

Load Rosters
load_rosters_weekly

Usage

load_rosters_weekly(
    seasons = most_recent_season(roster = TRUE),
    file_type = getOption("nflreadr.prefer", default = "rds")
)

Arguments

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)

Value

A tibble of season-level roster data.

See Also

https://nflreadr.nflverse.com/articles/dictionary_rosters.html for a web version of the data dictionary
dictionary_rosters for the data dictionary as a dataframe
Issues with this data should be filed here: https://github.com/nflverse/nflverse-data

Examples

try({ # prevents cran errors
    load_rosters(2020)
})
load_schedules

Load Game/Schedule Data

Description

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

Usage

load_schedules(seasons = TRUE)

Arguments

seasons a numeric vector of seasons to return, default TRUE returns all available data.

Value

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

See Also

Issues with this data should be filed here: https://github.com/nflverse/nflverse-data
Load Snap Counts from PFR

Description

Loads game level snap counts stats provided by Pro Football Reference starting with the 2012 season.

Usage

```r
load_snap_counts(
  seasons = most_recent_season(),
  file_type = getOption("nflreadr.prefer", default = "rds")
)
```

Arguments

- `seasons` a numeric vector specifying what seasons to return, if `TRUE` returns all available data
- `file_type` One of c("rds", "qs", "csv", "parquet"). Can also be set globally with `options(nflreadr.prefer)`

Value

A tibble of game-level snap counts provided by Pro Football Reference.

See Also

- [https://nflreadr.nflverse.com/articles/dictionary_snap_counts.html](https://nflreadr.nflverse.com/articles/dictionary_snap_counts.html) for the web data dictionary
- `dictionary_snap_counts` for the data dictionary as bundled within the package
- Issues with this data should be filed here: [https://github.com/nflverse/nflverse-pfr](https://github.com/nflverse/nflverse-pfr)
Examples

```r
try({ # prevents CRAN errors
    load_snap_counts()
})
```

---

**load_teams**  
*Load NFL Team Graphics, Colors, and Logos*

**Description**  
Load team graphics, colors, and logos - useful for plots!

**Usage**  
```r
load_teams(current = TRUE)
```

**Arguments**  
- `current`  
  If TRUE (the default), returns a standardized list of current teams only, with abbreviations as per `team_abbr_mapping`.

**Value**  
A tibble of team-level image URLs and hex color codes.

**See Also**  
Issues with this data should be filed here: [https://github.com/nflverse/nflverse-pbp](https://github.com/nflverse/nflverse-pbp)

**Examples**

```r
try({ # prevents cran errors
    load_teams()
})
```
load_trades  

Description
This returns a table of historical trades as maintained by Lee Sharpe.

Usage
load_trades(seasons = TRUE)

Arguments
seasons a numeric vector of seasons to return, default TRUE returns all available data.

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

See Also
https://nflreadr.nflverse.com/articles/dictionary_trades.html for a web version of the dictionary
dictionary_trades for the data dictionary as bundled within the package
Issues with this data should be filed here: https://github.com/nflverse/nfldata

Examples

load_trades(2020)

most_recent_season  

Description
A helper function to choose the most recent season available for a given dataset

Usage
most_recent_season(roster = FALSE)
get_latest_season(roster = FALSE)
get_current_season(roster = FALSE)
Arguments

roster a TRUE/FALSE flag: if TRUE, returns the current year if March 15th or later. if FALSE, returns the current year if September 1st or later. Otherwise returns current year minus 1.

Value

most recent season (a four digit numeric)

See Also

Other Date utils: get_current_week()
Examples

try({
  ## could also set options like
  # options(nflreadr.download_path = tempdir(), nflreadr.prefer = "parquet")

  nflverse_download(combine, contracts, folder_path = tempdir(), file_type = "parquet")

  list.files(tempdir(),pattern = ".parquet$") # check that files were downloaded!
})

---

nflverse_game_id  Compute nflverse Game Identifiers

Description

Compute nflverse Game Identifiers

Usage

nflverse_game_id(season, week, away, home)

Arguments

- **season**: 4 digit season between 1999 and the output of `most_recent_season()`
- **week**: Numeric or character giving the week, between 1 and 22.
- **home, away**: Valid NFL team abbreviation as it can be found in `team_abbr_mapping`

Value

A character vector

Examples

nflverse_game_id(2022, 2, "LAC", "KC")
nflverse_releases  List all available nflverse releases

Description
This function lists all nflverse data releases that are available in the nflverse-data repo. Release names can be used for downloads in `nflverse_download()`.

Usage

```r	nflverse_releases(.token = "default")
```

Arguments

- `.token` a GitHub API token, "default" uses `gh::gh_token()`

Value

A dataframe containing release names, release descriptions, and other relevant release information.

Examples

```r
try( # avoids cran failures, can skip in normal usage
nflverse_releases()
)
```

nflverse_sitrep  Get a Situation Report on System, nflverse/ffverse 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/ffverse 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

nflverse_sitrep(
  pkg = c("nflreadr", "nflfastR", "nflseedR", "nfl4th", "nflplotR", "nflverse"),
  recursive = TRUE,
  redact_path = TRUE
)

ffverse_sitrep(
  pkg = c("ffscrapr", "ffsimulator", "ffpros", "ffopportunity"),
  recursive = TRUE,
  redact_path = TRUE
)

.sitrep(
  pkg,
  recursive = TRUE,
  redact_path = TRUE,
)

Arguments

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
dev_repos  Developmental cran-like repos to check, e.g. r-universe repos

Examples

try({
  nflverse_sitrep()
  ffverse_sitrep()
  .sitrep("cachem")
})
parquet_from_url  Load .parquet file from a remote connection

Description
Retrieves a parquet file from URL. This function is cached.

Usage
parquet_from_url(url)

Arguments
url  a character url

Value
a dataframe as parsed by arrow::read_parquet()

Examples

try({
  parquet_from_url(
    "https://github.com/nflverse/nflverse-data/releases/download/player_stats/player_stats.parquet"
  )
})

player_name_mapping  Alternate player name mappings

Description
A named character vector mapping common alternate names, re-exported from ffscrappr.

Usage
player_name_mapping

Format
A named character vector

name attribute  The "alternate" name.
value attribute  The "correct" name.
progressively

Details

You can suggest additions to this table by opening an issue in ffscrapr.

Examples

```r
player_name_mapping[c("Chatarius Atwell", "Robert Kelley")]
```

<table>
<thead>
<tr>
<th>progressively</th>
<th>Progressively</th>
</tr>
</thead>
</table>

Description

This function helps add progress-reporting to any function - given function `f()` and progressor `p()`, it will return a new function that calls `f()` and then (on exiting) will call `p()` after every iteration. This is inspired by purrr’s `safely`, `quietly`, and `possibly` function decorators.

Usage

```r
progressively(f, p = NULL)
```

Arguments

- `f` a function to add progressor functionality to.
- `p` a function such as one created by `progressr::progressor()` - also accepts purrr-style lambda functions.

Value

a function that does the same as `f` but it calls `p()` after iteration.

See Also

[https://nflreadr.nflverse.com/articles/exporting_nflreadr.html](https://nflreadr.nflverse.com/articles/exporting_nflreadr.html) for vignette on exporting nflreadr in packages

Examples

```r
try({ # prevents cran errors
  lapply(urls, progressively(read.csv, ~cli::cli_progress_step('Loading...')))})
```
qs_from_url <- function(urls){
  p <- progressr::progressor(along = urls)
  lapply(urls, progressively(read.csv, p))
}

progressr::with_progress(read_rosters())

qs_from_url

Load .qs file from a remote connection

Description
Load .qs file from a remote connection

Usage
qs_from_url(url)

Arguments
url a character url

Value
a dataframe as parsed by qs::qdeserialized

Examples
try({
  qs_from_url(
    "https://github.com/nflverse/nflverse-data/releases/download/player_stats/player_stats.qs"
  )
})
**Description**

This function allows you to retrieve data from a URL into raw format, which can then be passed into the appropriate file-reading function. Data is memoised/cached for 24 hours.

**Usage**

```r
raw_from_url(url)
```

**Arguments**

- `url`: a character url

**Value**

a raw vector

**Examples**

```r
try({
  # prevents CRAN errors
  head(raw_from_url("https://github.com/nflverse/nflverse-data/releases/download/test/combines.rds"), 50)
})
```

---

**rds_from_url**

Load .rds file from a remote connection

**Description**

Load .rds file from a remote connection

**Usage**

```r
rds_from_url(url)
```

**Arguments**

- `url`: a character url
### team_abbr_mapping

**Value**

a dataframe as created by `readRDS()`

**Examples**

```r
try({
  rds_from_url("https://github.com/nflverse/nflverse-data/releases/download/test/combines.rds")
})
```

<table>
<thead>
<tr>
<th>team_abbr_mapping</th>
<th>Alternate team abbreviation mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Description**

A named character vector mapping common alternate team abbreviations.

**Usage**

```r
team_abbr_mapping
```

**Format**

A named character vector

- **name attribute** The "alternate" name.
- **value attribute** The "correct" name.

**Details**

You can suggest additions to this table by opening an issue in nflreadr.

**See Also**

- `team_abbr_mapping_norelocate` for the same thing but relocations stay in their original cities.

**Examples**

```r
team_abbr_mapping[c("STL", "OAK","CRD","BLT", "CLV")]
```
### team_abbr_mapping_norelocate

**Alternate team abbreviation mappings, no relocation**

<table>
<thead>
<tr>
<th>Description</th>
<th>A named character vector mapping common alternate team abbreviations, but does not follow relocations to their current city.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td><code>team_abbr_mapping_norelocate</code></td>
</tr>
<tr>
<td>Format</td>
<td>A named character vector</td>
</tr>
<tr>
<td></td>
<td><strong>name attribute</strong> The &quot;alternate&quot; name.</td>
</tr>
<tr>
<td></td>
<td><strong>value attribute</strong> The &quot;correct&quot; name.</td>
</tr>
<tr>
<td>Details</td>
<td>You can suggest additions to this table by <a href="https://github.com/NFLData/nflreadr">opening an issue in nflreadr</a>.</td>
</tr>
<tr>
<td>Examples</td>
<td><code>team_abbr_mapping_norelocate[c(&quot;STL&quot;, &quot;OAK&quot;, &quot;CRD&quot;, &quot;BLT&quot;, &quot;CLV&quot;)]</code></td>
</tr>
</tbody>
</table>
Index

* Date utils
  - get_current_week, 20
  - most_recent_season, 43

* datasets
  - dictionary_combine, 10
  - dictionary_contracts, 11
  - dictionary_depth_charts, 11
  - dictionary_draft_picks, 12
  - dictionary_espn_qbr, 12
  - dictionary_ff_opportunity, 13
  - dictionary_ff_playerids, 13
  - dictionary_ff_rankings, 14
  - dictionary_ftn_charting, 14
  - dictionary_injuries, 15
  - dictionary_nextgen_stats, 15
  - dictionary_participation, 16
  - dictionary_pbp, 16
  - dictionary_pfr_passing, 17
  - dictionary_player_stats, 17
  - dictionary_player_stats_def, 18
  - dictionary_rosters, 18
  - dictionary_schedules, 19
  - dictionary_snap_counts, 19
  - dictionary_trades, 20
  - player_name_mapping, 48
  - team_abbr_mapping, 52
  - team_abbr_mapping_norelocate, 53

* ftn_charting
  - dictionary_ftn_charting, 14
  - load_ftn_charting, 30
  - .clear_cache(.clear_cache), 6
  - .sitrep(.nflverse_sitrep), 46

- arrow::read_parquet(), 48
- base::tempdir, 9
- clean_homeaway, 3
- clean_player_names, 4
- clean_team_abbrs, 5
- clear_cache, 6
- csv_from_url, 6
- data.table::.fread, 7
- data.table::.fread(), 10
- dictionary_combine, 10, 23
- dictionary_contracts, 11, 23
- dictionary_depth_charts, 11, 24
- dictionary_draft_picks, 12, 25
- dictionary_espn_qbr, 12, 26
- dictionary_ff_opportunity, 13, 27
- dictionary_ff_playerids, 13
- dictionary_ff_rankings, 14
- dictionary_ftn_charting, 14, 30
- dictionary_injuries, 15, 31
- dictionary_nextgen_stats, 15, 32
- dictionary_participation, 16
- dictionary_pbp, 16, 35
- dictionary_pfr_passing, 17
- dictionary_player_stats, 17, 38
- dictionary_player_stats_def, 18
- dictionary_rosters, 18, 39
- dictionary_schedules, 19, 41
- dictionary_snap_counts, 19, 41
- dictionary_trades, 20, 43
- fverse_sitrep(nflverse_sitrep), 46

- get_current_season
- (most_recent_season), 43
- get_current_week, 20, 44
- get_latest_season(most_recent_season), 43

- join_coalesce, 21
- load_combine, 22
- load_combine(), 10
- load_contracts, 23
- load_contracts(), 11
- load_depth_charts, 24

54
INDEX

load_depth_charts(), 11
load_draft_picks, 25
load_draft_picks(), 12
load_espn_qbr, 26
load_espn_qbr(), 12
load_ff_opportunity, 27
load_ff_opportunity(), 13
load_ff_playerids, 28
load_ff_playerids(), 13
load_ff_rankings, 28
load_ff_rankings(), 14
load_from_url, 29
load_ftn_charting, 14, 30
load_ftn_charting(), 14
load_injuries, 31
load_injuries(), 15
load_nextgen_stats, 32
load_nextgen_stats(), 15
load_officials, 33
load_participation, 34
load_participation(), 16
load_pbp, 35
load_pbp(), 16
load_pfr_advstats, 36
load_pfr_passing(), 17
load_player_stats, 37
load_player_stats(), 17, 18
load_players, 37
load_rosters, 38
load_rosters(), 18
load_rosters_weekly, 39
load_schedules, 40
load_schedules(), 19
load_snap_counts, 41
load_snap_counts(), 19
load_teams, 42
load_trades, 43
load_trades(), 20
make.names, 8
most_recent_season, 21, 43
most_recent_season(), 45
nflverse_download, 44
nflverse_download(), 46
nflverse_game_id, 45
nflverse_releases, 46
nflverse_releases(), 44
nflverse_sitrep, 46
parquet_from_url, 48
path.expand, 7
player_name_mapping, 48
progressively, 49
qs::qdeserialize(), 50
qs_from_url, 50
raw_from_url, 51
rds_from_url, 51
readRDS(), 52
setindexv, 9
setkey, 9
team_abbr_mapping, 5, 42, 45, 52
team_abbr_mapping_norelocate, 5, 53
utils::read.csv, 8
utils::write.csv, 9
yaml.load, 9