Package ‘nflreadr’

March 17, 2022

Title Download ‘nflverse’ Data

Version 1.2.0

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), qs (>= 0.24.0), rappdirs (>= 0.3.0), Rcpp (>= 1.0.7), RcppParallel (>= 5.1.4), rlang (>= 0.4.0), utils

Suggests covr (>= 3.0.0), DT (>= 0.15.0), knitr (>= 1.0.0), progressr
(>= 0.8.0), rmarkdown (>= 2.6.0), testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

NeedsCompilation no

Author Tan Ho [aut, cre, cph] (<https://orcid.org/0000-0001-8388-5155>),
Sebastian Carl [aut],
John Edwards [ctb],
Ben Baldwin [ctb],
Thomas Mock [ctb],
Lee Sharpe [ctb]

Maintainer Tan Ho <tan@tanho.ca>

Repository CRAN

Date/Publication 2022-03-17 13:20:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>.clear_cache</td>
<td>3</td>
</tr>
<tr>
<td>clean_homeaway</td>
<td>3</td>
</tr>
<tr>
<td>clean_player_names</td>
<td>4</td>
</tr>
<tr>
<td>clean_team_abbrs</td>
<td>5</td>
</tr>
<tr>
<td>csv_from_url</td>
<td>6</td>
</tr>
<tr>
<td>dictionary_combine</td>
<td>9</td>
</tr>
<tr>
<td>dictionary_depth_charts</td>
<td>10</td>
</tr>
<tr>
<td>dictionary_draft_picks</td>
<td>10</td>
</tr>
<tr>
<td>dictionaryESPN_qbr</td>
<td>11</td>
</tr>
<tr>
<td>dictionary_ff_opportunity</td>
<td>11</td>
</tr>
<tr>
<td>dictionary_ff_playerids</td>
<td>12</td>
</tr>
<tr>
<td>dictionary_ff_rankings</td>
<td>12</td>
</tr>
<tr>
<td>dictionary_injuries</td>
<td>13</td>
</tr>
<tr>
<td>dictionary_nextgen_stats</td>
<td>13</td>
</tr>
<tr>
<td>dictionary_pbp</td>
<td>14</td>
</tr>
<tr>
<td>dictionary_pfr_passing</td>
<td>14</td>
</tr>
<tr>
<td>dictionary_player_stats</td>
<td>15</td>
</tr>
<tr>
<td>dictionary_rosters</td>
<td>15</td>
</tr>
<tr>
<td>dictionary_schedules</td>
<td>16</td>
</tr>
<tr>
<td>dictionary_snap_counts</td>
<td>16</td>
</tr>
<tr>
<td>dictionary_trades</td>
<td>17</td>
</tr>
<tr>
<td>load_combine</td>
<td>17</td>
</tr>
<tr>
<td>load_depth_charts</td>
<td>18</td>
</tr>
<tr>
<td>load_draft_picks</td>
<td>19</td>
</tr>
<tr>
<td>loadESPN_qbr</td>
<td>19</td>
</tr>
<tr>
<td>load_ff_opportunity</td>
<td>20</td>
</tr>
<tr>
<td>load_ff_playerids</td>
<td>21</td>
</tr>
<tr>
<td>load_ff_rankings</td>
<td>22</td>
</tr>
<tr>
<td>load_injuries</td>
<td>22</td>
</tr>
<tr>
<td>load_nextgen_stats</td>
<td>23</td>
</tr>
<tr>
<td>load_pbp</td>
<td>24</td>
</tr>
<tr>
<td>load_pfr_advstats</td>
<td>25</td>
</tr>
<tr>
<td>load_player_stats</td>
<td>26</td>
</tr>
<tr>
<td>load_rosters</td>
<td>27</td>
</tr>
<tr>
<td>load_schedules</td>
<td>28</td>
</tr>
<tr>
<td>load_snap_counts</td>
<td>28</td>
</tr>
<tr>
<td>load_teams</td>
<td>29</td>
</tr>
<tr>
<td>load_trades</td>
<td>30</td>
</tr>
<tr>
<td>most_recent_season</td>
<td>30</td>
</tr>
<tr>
<td>nflverse_sitrep</td>
<td>31</td>
</tr>
<tr>
<td>player_name_mapping</td>
<td>32</td>
</tr>
<tr>
<td>progressively</td>
<td>33</td>
</tr>
<tr>
<td>qs_from_url</td>
<td>34</td>
</tr>
<tr>
<td>raw_from_url</td>
<td>34</td>
</tr>
<tr>
<td>rds_from_url</td>
<td>35</td>
</tr>
<tr>
<td>team_abbr_mapping</td>
<td>35</td>
</tr>
</tbody>
</table>
clear_cache

Description
This function clears the memoised cache of all functions memoised by nflreadr.

Usage
.clear_cache()

Value
A success message after clearing the cache.

Examples
.clear_cache()

clean_homeaway

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataframe</td>
<td>dataframe</td>
</tr>
<tr>
<td>invert</td>
<td>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)</td>
</tr>
</tbody>
</table>

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

```r
# creating a small example dataframe!
cols <- c("season", "week", "home_team", "home_score", "away_team", "away_score", "result", "spread_line")
x <- as.data.frame(load_schedules(2020))
x <- utils::head(x[cols])

# how the data looks like
x

clean_homeaway(x, 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

```r
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

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)
```

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

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abbr</code></td>
<td>a character vector of abbreviations</td>
</tr>
<tr>
<td><code>current_location</code></td>
<td>If TRUE (the default), the abbreviation of the most recent team location will be used.</td>
</tr>
<tr>
<td><code>keep_non_matches</code></td>
<td>If TRUE (the default) an element of <code>abbr</code> that can't be matched to any of the internal mapping vectors will be kept as is. Otherwise it will be replaced with NA.</td>
</tr>
</tbody>
</table>

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`).
Examples

```r
x <- c("PIE", "LAR", "PIT", "CRD", "OAK", "SL")
# use current location and keep non matches
clean_team_abbrs(x)

# keep old location and replace non matches
clean_team_abbrs(x, current_location = FALSE, keep_non_matches = FALSE)
```

### 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

```r
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 `
` present), `text=` (if at least one `
` is present) or `cmd=` (if no `
` 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 NAs, 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 \texttt{utils::read.csv} does; i.e., possibly with loss of precision and if so silently. Or, "character".

dec The decimal separator as in \texttt{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 \texttt{key} and \texttt{index}.

check.names default is \texttt{FALSE}. If \texttt{TRUE} then the names of the variables in the \texttt{data.table} are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by \texttt{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, \texttt{fread} handles embedded quotes robustly as explained under Details. If it fails, then another attempt is made to read the field as \texttt{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 \texttt{TRUE}. Strips leading and trailing whitespaces of unquoted fields. If \texttt{FALSE}, only header trailing spaces are removed.

fill logical (default is \texttt{FALSE}). If \texttt{TRUE} then in case the rows have unequal length, blank fields are implicitly filled.

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

key Character vector of one or more column names which is passed to \texttt{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 \texttt{data.table=TRUE}. Where applicable, this should refer to column names given in \texttt{col.names}.

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

showProgress \texttt{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) \texttt{txtProgressBar} and \texttt{tkProgressBar} are not easily available.

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

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

logical01 If \texttt{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

csv_from_url("https://github.com/nflverse/nfldata/raw/master/data/games.csv")

dictionary_combine Data Dictionary: Combine

Description

A dataframe containing the data dictionary for load_combine()

Usage

dictionary_combine
dictionary_draft_picks

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

dictionary_depth_charts

Data Dictionary: Depth Charts

Description
A dataframe containing the data dictionary for load_depth_charts()

Usage
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 10 rows and 3 columns.
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 219 rows and 4 columns.

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

**See Also**

vignette("Data Dictionary -PBP")

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")
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

Data Dictionary: Rosters

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

Usage
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 27 rows and 2 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 12 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

load_combine
Load Combine Data from PFR

Description
Loads combine data since 2000 courtesy of PFR.

Usage
load_combine(seasons = TRUE)

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

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

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
load_depth_charts

Examples

load_combine()

load_depth_charts  Load Weekly Depth Charts

Description

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

Usage

load_depth_charts(seasons = most_recent_season())

Arguments

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

Value

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

See Also

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

Examples

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)
```

**Arguments**

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

**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/nfldata](https://github.com/nflverse/nfldata)

**Examples**

```r
load_draft_picks()
```

---

**load_espn_qbr**  
**Load ESPN’s QBR**

**Description**

Load ESPN’s QBR

**Usage**

```r
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 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

Examples

load_espn_qbr("nfl",2020)
load_ff_playerids

Value
Precomputed expected fantasy points data from the ffoportunity automated releases.

See Also
https://ffopportunity.ffverse.com for more on the package, data, and modelling
https://nflreadr.nflverse.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

```r
try({
  load_ff_opportunity()
  load_ff_opportunity(seasons = 2021, type = "pbp_pass", 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
load_ff_playerids()

Value
a dataframe of player IDs

See Also
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

Examples

```r
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)

**Value**

a dataframe of expert consensus rankings

**See Also**

- https://nflreadr.nflverse.com/articles/dictionary_ff_rankings.html for the web data dictionary
- https://www.fantasypros.com for the source of data

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

**Examples**

```r
load_ff_rankings()
```

### load_injuries  
**Load Injury Reports**

**Description**

Data collected from an API for weekly injury report data.

**Usage**

```r
load_injuries(seasons = most_recent_season(), file_type = NULL)
```

- **seasons**  
  most_recent_season()

- **file_type**  
  NULL

**Examples**

```r
load_injuries()
```
load_nextgen_stats

Arguments

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

file_type Deprecated: now uses rds by default.

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

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

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

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 "rds" or "qs". 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 for stat_type = "passing"
https://nextgenstats.nfl.com/stats/receiving for stat_type = "receiving"
https://nextgenstats.nfl.com/stats/rushing for stat_type = "rushing"
https://nflreadr.nflverse.com/articles/dictionary_nextgen_stats.html for a web version of the data dictionary
dictionary_nextgen_stats for the data dictionary as bundled within the package
Issues with this data should be filed here: https://github.com/nflverse/nflverse-data

Examples

load_nextgen_stats(stat_type = "passing")
load_nextgen_stats(stat_type = "receiving")
load_nextgen_stats(stat_type = "rushing")

load_pbp

Load Play By Play

Description

Loads play by play seasons from the nflverse-data repository

Usage

load_pbp(
  seasons = most_recent_season(),
  file_type = getOption("nflreadr.prefer", default = "qs")
)

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 "rds" or "qs". Can also be set globally with options(nflreadr.prefer)

Value

The complete nlfastR dataset as returned by nlfastR::build_nlfastR_pbp() (see below) for all given seasons
load_pfr_advstats

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/nflfastR-data

Examples

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

load_pfr_advstats(
  seasons = most_recent_season(),
  stat_type = c("pass", "rush", "rec", "def"),
  summary_level = c("week", "season")
)

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

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 for the web data dictionary

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

Examples

load_pfr_advstats()

---

load_player_stats  Load Player Level Weekly Stats

Description

Load Player Level Weekly Stats

Usage

load_player_stats(
  seasons = most_recent_season(),
  stat_type = c("offense", "kicking"),
  file_type = getOption("nflreadr.prefer", default = "qs")
)

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 or kicking
file_type  One of "rds" or "qs". 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 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/nflfastR-data
Examples

```r
load_player_stats()
load_player_stats(stat_type = "kicking")
```

---

**load_rosters**

*Load Rosters*

**Description**

Load Rosters

**Usage**

```r
load_rosters(seasons = most_recent_season(roster = TRUE))
```

**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.

**Value**

A tibble of season-level roster data.

**See Also**

- [https://nflreadr.nflverse.com/articles/dictionary_rosters.html](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](https://github.com/nflverse/nflverse-data)

**Examples**

```r
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
- [https://nflreadr.nflverse.com/articles/dictionary_schedules.html](https://nflreadr.nflverse.com/articles/dictionary_schedules.html) for a web version of the data dictionary
- `dictionary_schedules` for the data dictionary as a dataframe
- Issues with this data should be filed here: [https://github.com/nflverse/nfldata](https://github.com/nflverse/nfldata)

Examples
load_schedules(2020)

load_snap_counts  

Load Snap Counts from PFR

Description
Loads game level snap counts stats provided by Pro Football Reference starting with the 2013 season.

Usage
load_snap_counts(seasons = most_recent_season())
load_teams

Arguments

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

Value

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

See Also

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/pfr_scrapR

Examples

load_snap_counts()

load_teams

Load NFL Team Graphics, Colors, and Logos

Description

Loads team graphics, colors, and logos - useful for plots!

Usage

load_teams()

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/nflfastR-data

Examples

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)

Get Latest Season

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

Usage
most_recent_season(roster = FALSE)
Arguments

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

Value

season (a four digit numeric)

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 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

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

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

.sitrep(pkg, recursive = TRUE, header = "")
```

Arguments

pkgnflverse_sitrepGet a Situation Report on System, nflverse/ffverse Package Versions and Dependencies

args:

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.

header a string that is printed in the horizontal separation lines and used to differentiate between nflverse and ffverse output.
Examples

```r
try({
  nflverse_sitrep()
  ffverse_sitrep()
  .sitrep("cachem")
})
```

---

**player_name_mapping**

<table>
<thead>
<tr>
<th>player_name_mapping</th>
<th>Alternate player name mappings</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

`player_name_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 ffscrpr.

**Examples**

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

Usage

progressively(f, p = NULL)

Arguments

f  
a function to add progressr functionality to.

p  
a progressor function as created by progressr::progressor()

Details

This is inspired by purrr’s safely, quietly, and possibly function decorators.

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 for vignette on exporting nflreadr in packages

Examples

read_rosters <- function(){

  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**
```r
code
qs_from_url(url)
```

**Arguments**
- `url` a character url

**Value**
a dataframe as parsed by `qs::qdeserialize()`

**Examples**
```r
qs_from_url(
  "https://github.com/nflverse/nflverse-data/releases/download/player_stats/player_stats.qs"
)
```

---

raw_from_url  
*Load raw filedata from a remote connection*

**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, such as `arrow::read_parquet()`

**Usage**
```r
code
raw_from_url(url)
```

**Arguments**
- `url` a character url

**Value**
a raw vector
**rds_from_url**

Description

Load .rds file from a remote connection

**Usage**

rds_from_url(url)

**Arguments**

url a character url

**Value**

a dataframe as created by `readRDS()`

**Examples**

rds_from_url("https://github.com/nflverse/nfldata/raw/master/data/games.rds")

---

**team_abbr_mapping**

Alternate team abbreviation mappings

Description

A named character vector mapping common alternate team abbreviations.

**Usage**

team_abbr_mapping
team_abbr_mapping_norelocate

**Description**
A named character vector mapping common alternate team abbreviations, but does not follow relocations to their current city.

**Usage**
```r
team_abbr_mapping_norelocate
```

**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.
team_abbr_mapping_norelocate

Examples

team_abbr_mapping_norelocate[c("STL", "OAK", "CRD", "BLT", "CLV")]

Index

* datasets
  dictionary_combine, 9
dictionary_depth_charts, 10
dictionary_draft_picks, 10
dictionary_espn_qbr, 11
dictionary_ff_opportunity, 11
dictionary_ff_playerids, 12
dictionary_ff_rankings, 12
dictionary_injuries, 13
dictionary_nextgen_stats, 13
dictionary_pbp, 14
dictionary_pfr_passing, 14
dictionary_player_stats, 15
dictionary_rosters, 15
dictionary_schedules, 16
dictionary_snap_counts, 16
dictionary_trades, 17
player_name_mapping, 32
team_abbr_mapping, 35
team_abbr_mapping_norelocate, 36
.clear_cache, 3
.sitrep(nflverse_sitrep), 31

base::tempdir, 9

clean_homeaway, 3
clean_player_names, 4
clean_team_abbrs, 5
csv_from_url, 6
data.table::fread, 6
data.table::fread(), 9
dictionary_combine, 9, 17
dictionary_depth_charts, 10, 18
dictionary_draft_picks, 10, 19
dictionary_espn_qbr, 11, 20
dictionary_ff_opportunity, 11, 21
dictionary_ff_playerids, 12
dictionary_ff_rankings, 12
dictionary_injuries, 13, 23
dictionary_nextgen_stats, 13, 24
dictionary_pbp, 14, 25
dictionary_pfr_passing, 14
dictionary_player_stats, 15, 26
dictionary_rosters, 15, 27
dictionary_schedules, 16, 28
dictionary_snap_counts, 16, 29
dictionary_trades, 17, 30

ffverse_sitrep(nflverse_sitrep), 31

load_combine, 17
load_combine(), 9
load_depth_charts, 18
load_depth_charts(), 10
load_draft_picks, 19
load_draft_picks(), 10
load_espn_qbr, 19
load_espn_qbr(), 11
load_FF_opportunity, 20
load_FF_opportunity(), 11
load_FF_playerids, 21
load_FF_playerids(), 12
load_FF_rankings, 22
load_FF_rankings(), 12
load_injuries, 22
load_injuries(), 13
load_nextgen_stats, 23
load_nextgen_stats(), 13
load_pbp, 24
load_pbp(), 14
load_pfr_advstats, 25
load_pfr_passing, 14
load_player_stats, 26
load_player_stats(), 15
load_rosters, 27
load_rosters(), 15
load_schedules, 28
load_schedules(), 16
load_snap_counts, 28
load_snap_counts(), 16
load_teams, 29
load_trades, 30
load_trades(), 17
make.names, 8
most_recent_season, 30

nflverse_sitrep, 31
path.expand, 6
player_name_mapping, 32
progressively, 33

qs::qdeserialize(), 34
qs_from_url, 34

raw_from_url, 34
rds_from_url, 35
readRDS(), 35

setindexv, 8
setkey, 8

team_abbr_mapping, 5, 35
team_abbr_mapping_norelocate, 5, 36

utils::read.csv, 7
utils::write.csv, 9

yaml.load, 9