Package 'ffscrapr'

November 27, 2020

Type Package

Title API Client for Fantasy Football League Platforms

Version 1.2.0

Description Helps access various Fantasy Football APIs by handling authentication and rate-limiting, forming appropriate calls, and returning tidy dataframes which can be easily connected to other data sources.

License MIT + file LICENSE

URL https://ffscrapr.dynastyprocess.com, https://github.com/dynastyprocess/ffscrapr, https://api.myfantasyleague.com/2020/api_info, https://docs.sleeper.app, https://www.fleaflicker.com/api-docs/index.html

BugReports https://github.com/dynastyprocess/ffscrapr/issues

Depends R (>= 3.0.0)

Imports dplyr (>= 1.0.0), glue (>= 1.3.0), httr (>= 1.4.0), jsonlite (>= 1.6.0), lubridate (>= 1.5.0), magrittr (>= 1.5.0), memoise (>= 1.1.0), purrr (>= 0.3.0), rappdirs (>= 0.3.0), ratelimitr (>= 0.4.0), rlang (>= 0.4.0), stringr (>= 1.4.0), tibble (>= 3.0.0), tidyr (>= 1.0.0)

Suggests covr (>= 3.0.0), httptest (>= 3.0.0), knitr, rmarkdown, testthat (>= 2.1.0), checkmate

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Tan Ho [aut, cre]

Maintainer Tan Ho <tan@tanho.ca>

Repository CRAN

Date/Publication 2020-11-27 10:20:02 UTC

R topics documented:

`.ff_clear_cache` *Empty Function Cache*

Description

This function will reset the cache for any and all ffscrapr cached functions.

Usage

```
.ff_clear_cache()
```

dp_playerids *Import latest DynastyProcess player IDs*

Description

Fetches a copy of the latest DynastyProcess player IDs csv

Usage

```
dp_playerids()
```

Value

a tibble of player IDs

See Also

https://github.com/DynastyProcess/data

Examples

```
dp_playerids()
```

dp_values *Import latest DynastyProcess values*

Description

Fetches a copy of the latest DynastyProcess dynasty trade values sheets

Usage

```
dp_values(file = c("values.csv", "values-players.csv", "values-picks.csv"))
```

Arguments

file	one of c("values.csv","values-players.csv","values-picks.csv")

Value

a tibble of trade values from DynastyProcess

See Also

https://github.com/DynastyProcess/data

Examples

```
dp_values()
```

`ff_connect` *Connect to a League*

Description

This function creates a connection object which stores parameters and gets a login-cookie if available - it does so by passing arguments to the appropriate league-based handler.

Usage

```
ff_connect(platform = "mfl", league_id = NULL, ...)
```

Arguments

`platform`	one of MFL or Sleeper (Fleaflicker, ESPN, Yahoo in approximate priority order going forward)
`league_id`	league_id (currently assuming one league at a time)
`...`	other parameters passed to the connect function for each specific platform.

Value

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

See Also

`ff_connect`, `sleeper_connect`

Examples

```
ff_connect(platform = "mfl", season = 2019, league_id = 54040, rate_limit = FALSE)
```

`ff_draft` *Get Draft Results*

Description

This function gets a tidy dataframe of draft results for the current year. Can handle MFL devy drafts or startup drafts by specifying the custom_players argument

Usage

```
ff_draft(conn, ...)

## S3 method for class 'flea_conn'
ff_draft(conn, ...)

## S3 method for class 'mfl_conn'
ff_draft(conn, custom_players = FALSE, ...)

## S3 method for class 'sleeper_conn'
ff_draft(conn, ...)
```

Arguments

`conn`	a conn object created by `ff_connect()`
`...`	args for other methods
`custom_players`	MFL: TRUE or FALSE - retrieve custom players from the MFL database? (Allows for devy, placeholder picks, slightly slower)

Value

A tidy dataframe of draft results

Methods (by class)

- `flea_conn`: Fleaflicker: returns a table of drafts for the current year
- `mfl_conn`: MFL: returns a table of drafts for the current year - can handle devy/startup-rookie-picks by specifying custom_players (slower!)
- `sleeper_conn`: Sleeper: returns a dataframe of all drafts and draft selections, if available.

Examples

```
conn <- fleaflicker_connect(season = 2020, league_id = 206154)
ff_draft(conn)
```

```
ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_draft(ssb_conn)


jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_draft(jml_conn)
```

`ff_draftpicks` *Get Draft Picks*

Description

Returns all draft picks (current and future) that belong to a specific franchise and have not yet been converted into players (i.e. selected.)

Usage

```
ff_draftpicks(conn, ...)

## S3 method for class 'flea_conn'
ff_draftpicks(conn, franchise_id = NULL, ...)

## S3 method for class 'mfl_conn'
ff_draftpicks(conn, ...)

## S3 method for class 'sleeper_conn'
ff_draftpicks(conn, ...)
```

Arguments

`conn`	the list object created by `ff_connect()`
`...`	other arguments (currently unused)
`franchise_id`	A list of franchise IDs to pull, if NULL will return all franchise IDs

Value

Returns a dataframe with current and future draft picks for each franchise

Methods (by class)

- `flea_conn`: Fleaflicker: retrieves current and future draft picks, potentially for a specified team.
- `mfl_conn`: MFL: returns current and future picks
- `sleeper_conn`: Sleeper: retrieves current and future draft picks

Examples

```
conn <- fleaflicker_connect(2020, 206154)
ff_draftpicks(conn, franchise_id = 1373475)



dlf_conn <- mfl_connect(2020, league_id = 37920)
ff_draftpicks(conn = dlf_conn)



jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_draftpicks(jml_conn)
```

`ff_franchises` *Get League Franchises*

Description

Return franchise-level data (including divisions, usernames, etc) - available data may vary slightly based on platform.

Usage

```
ff_franchises(conn)

## S3 method for class 'flea_conn'
ff_franchises(conn)

## S3 method for class 'mfl_conn'
ff_franchises(conn)

## S3 method for class 'sleeper_conn'
ff_franchises(conn)
```

Arguments

`conn`	a conn object created by `ff_connect()`

Value

A tidy dataframe of franchises, complete with IDs

Methods (by class)

- flea_conn: Fleaflicker: returns franchise and division information.
- mfl_conn: MFL: returns franchise and division information.
- sleeper_conn: Sleeper: retrieves a list of franchise information, including user IDs and co-owner IDs.

Examples

```
conn <- fleaflicker_connect(season = 2020, league_id = 206154)
ff_franchises(conn)


ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_franchises(ssb_conn)


jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_franchises(jml_conn)
```

---

ff_league *Get League Summary*

Description

This function returns a tidy dataframe of common league settings, including details like "1QB" or "2QB/SF", scoring, best ball, team count, IDP etc. This is potentially useful in summarising the features of multiple leagues.

Usage

```
ff_league(conn)

## S3 method for class 'flea_conn'
ff_league(conn)

## S3 method for class 'mfl_conn'
ff_league(conn)

## S3 method for class 'sleeper_conn'
ff_league(conn)
```

Arguments

conn	the connection object created by `ff_connect()`

Value

A one-row summary of each league's main features.

Methods (by class)

- `flea_conn`: Flea: returns a summary of league features.
- `mfl_conn`: MFL: returns a summary of league features.
- `sleeper_conn`: Sleeper: returns a summary of league features.

Examples

```
conn <- fleaflicker_connect(2020, 206154)
ff_league(conn)



ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_league(ssb_conn)



jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_league(jml_conn)
```

`ff_playerscores` *Get Player Scoring History*

Description

This function returns a tidy dataframe of player scores based on league rules.

Unfortunately, Sleeper has deprecated their player stats endpoint from their supported/open API. Adding some nflfastr-based ideas to the pipeline for future iterations.

Usage

```
ff_playerscores(conn, ...)

## S3 method for class 'flea_conn'
ff_playerscores(conn, page_limit = NULL, ...)

## S3 method for class 'mfl_conn'
ff_playerscores(conn, season, week, ...)
```

```
## S3 method for class 'sleeper_conn'
ff_playerscores(conn, ...)
```

Arguments

conn	the list object created by ff_connect()
...	other arguments (currently unused)
page_limit	A numeric describing the number of pages to return - default NULL returns all available
season	the season of interest - generally only the most recent 2-3 seasons are available
week	a numeric or one of YTD (year-to-date) or AVG (average to date)

Value

A tibble of historical player scoring

Methods (by class)

- flea_conn: Fleaflicker: returns the season, season average, and standard deviation
- mfl_conn: MFL: returns the player fantasy scores for each week (not the actual stats)
- sleeper_conn: Sleeper: Deprecated their open API endpoint for player scores

Examples

```
conn <- fleaflicker_connect(2020, 312861)
x <- ff_playerscores(conn, page_limit = 2)
x


dlf_conn <- mfl_connect(2020, league_id = 37920)
ff_playerscores(conn = dlf_conn, season = 2019, week = "YTD")
```

---

ff_rosters *Get League Rosters*

Description

This function returns a tidy dataframe of team rosters

Usage

```
ff_rosters(conn, ...)

## S3 method for class 'flea_conn'
ff_rosters(conn, ...)

## S3 method for class 'mfl_conn'
ff_rosters(conn, custom_players = FALSE, ...)

## S3 method for class 'sleeper_conn'
ff_rosters(conn, ...)
```

Arguments

conn	a conn object created by ff_connect()
...	arguments passed to other methods (currently none)
custom_players	TRUE or FALSE - include custom players? defaults to FALSE

Value

A tidy dataframe of rosters, joined to basic player information and basic franchise information

Methods (by class)

- flea_conn: Fleaflicker: Returns roster data (minus age as of right now)
- mfl_conn: MFL: returns roster data
- sleeper_conn: Sleeper: Returns all roster data.

Examples

```
joe_conn <- ff_connect(platform = "fleaflicker", league_id = 312861, season = 2020)
ff_rosters(joe_conn)


ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_rosters(ssb_conn)



jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_rosters(jml_conn)
```

`ff_schedule` *Get Schedule*

Description

This function returns a tidy dataframe with one row for every team for every weekly matchup

Usage

```
ff_schedule(conn, ...)

## S3 method for class 'flea_conn'
ff_schedule(conn, week = 1:17, ...)

## S3 method for class 'mfl_conn'
ff_schedule(conn, ...)

## S3 method for class 'sleeper_conn'
ff_schedule(conn, ...)
```

Arguments

conn	a conn object created by `ff_connect()`
...	for other platforms
week	a numeric or numeric vector specifying which weeks to pull

Value

A tidy dataframe with one row per game per franchise per week

Methods (by class)

- `flea_conn`: Flea: returns schedule data, one row for every franchise for every week. Completed games have result data.
- `mfl_conn`: MFL: returns schedule data, one row for every franchise for every week. Completed games have result data.
- `sleeper_conn`: Sleeper: returns all schedule data

Examples

```
conn <- fleaflicker_connect(season = 2019, league_id = 206154)
x <- ff_schedule(conn, week = 2:4)



ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
```

```
ff_schedule(ssb_conn)


jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_schedule(jml_conn)
```

`ff_scoring` *Get League Scoring settings*

Description

This function returns a dataframe with detailed scoring settings for each league - broken down by event, points, and (if available) position.

Usage

```
ff_scoring(conn)

## S3 method for class 'flea_conn'
ff_scoring(conn)

## S3 method for class 'mfl_conn'
ff_scoring(conn)

## S3 method for class 'sleeper_conn'
ff_scoring(conn)
```

Arguments

conn	a conn object created by `ff_connect()`

Value

A tibble of league scoring rules for each position defined.

Methods (by class)

- `flea_conn`: Fleaflicker: returns scoring settings in a flat table, one row per position per rule.
- `mfl_conn`: MFL: returns scoring settings in a flat table, one row per position per rule.
- `sleeper_conn`: Sleeper: returns scoring settings in a flat table, one row per position per rule.

See Also

http://www03.myfantasyleague.com/2020/scoring_rules#rules

Examples

```
joe_conn <- ff_connect(platform = "fleaflicker", league_id = 312861, season = 2020)
ff_scoring(joe_conn)


ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_scoring(ssb_conn)


jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_scoring(jml_conn)
```

`ff_standings` *Get Standings*

Description

This function returns a tidy dataframe of season-long fantasy team stats, including H2H wins as well as points, potential points, and all-play.

Usage

```
ff_standings(conn, ...)

## S3 method for class 'flea_conn'
ff_standings(conn, include_allplay = TRUE, include_potentialpoints = TRUE, ...)

## S3 method for class 'mfl_conn'
ff_standings(conn, ...)

## S3 method for class 'sleeper_conn'
ff_standings(conn, ...)
```

Arguments

`conn`	a conn object created by `ff_connect()`
`...`	arguments passed to other methods (currently none)
`include_allplay`	TRUE/FALSE - return all-play win pct calculation? defaults to TRUE
`include_potentialpoints`	TRUE/FALSE - return potential points calculation? defaults to TRUE.

Value

A tidy dataframe of standings data

Methods (by class)

- `flea_conn`: Fleaflicker: returns H2H/points/all-play/best-ball data in a table.
- `mfl_conn`: MFL: returns H2H/points/all-play/best-ball data in a table.
- `sleeper_conn`: Sleeper: returns all standings and points data and manually calculates allplay results.

Examples

```
conn <- fleaflicker_connect(season = 2020, league_id = 206154)
x <- ff_standings(conn)



ssb_conn <- ff_connect(platform = "mfl", league_id = 54040, season = 2020)
ff_standings(ssb_conn)



jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
ff_standings(jml_conn)
```

---

ff_starters *Get Starting Lineups*

Description

This function returns a tidy dataframe with one row for every starter (and bench) for every week and their scoring, if available.

Usage

```
ff_starters(conn, ...)

## S3 method for class 'flea_conn'
ff_starters(conn, week = 1:17, ...)

## S3 method for class 'mfl_conn'
ff_starters(conn, week = "all", season = NULL, ...)

## S3 method for class 'sleeper_conn'
ff_starters(conn, week = 1:17, ...)
```

Arguments

conn	the list object created by ff_connect()
...	other arguments (currently unused)
week	a numeric or one of YTD (year-to-date) or AVG (average to date)
season	the season of interest - generally only the most recent 2-3 seasons are available

Value

A tidy dataframe with every player for every week, including a flag for whether they were started or not

Methods (by class)

- flea_conn: Fleaflicker: returns who was started as well as what they scored.
- mfl_conn: MFL: returns the player fantasy scores for each week (not the actual stats)
- sleeper_conn: Sleeper: returns only "who" was started, without any scoring/stats data. Only returns season specified in initial connection object.

Examples

```
conn <- fleaflicker_connect(season = 2020, league_id = 206154)
ff_starters(conn)



dlf_conn <- mfl_connect(2020, league_id = 37920)
ff_starters(conn = dlf_conn, week = 1:2)



jml_conn <- sleeper_connect(league_id = "522458773317046272", season = 2020)
jml_starters <- ff_starters(jml_conn)
```

---

ff_transactions *Get League Transactions*

Description

This function returns a tidy dataframe of transactions - generally one row per player per transaction per team. Each trade is represented twice, once per each team.

Usage

```
ff_transactions(conn, ...)

## S3 method for class 'flea_conn'
ff_transactions(conn, franchise_id = NULL, ...)

## S3 method for class 'mfl_conn'
ff_transactions(conn, custom_players = FALSE, ...)

## S3 method for class 'sleeper_conn'
ff_transactions(conn, week = 1:17, ...)
```

Arguments

conn	the list object created by ff_connect()
...	additional args for other methods
franchise_id	fleaflicker returns transactions grouped by franchise id, pass a list here to filter
custom_players	TRUE or FALSE - fetch custom players
week	A week filter for transactions - 1 returns all offseason transactions. Default 1:17 returns all transactions.

Value

A tidy dataframe of transaction data

Methods (by class)

- flea_conn: Fleaflicker: returns all transactions, including free agents, waivers, and trades.
- mfl_conn: MFL: returns all transactions, including auction, free agents, IR, TS, waivers, and trades.
- sleeper_conn: Sleeper: returns all transactions, including free agents, waivers, and trades.

Examples

```
conn <- fleaflicker_connect(season = 2020, league_id = 312861)
ff_transactions(conn)



dlf_conn <- mfl_connect(2019, league_id = 37920)
ff_transactions(dlf_conn)


jml_conn <- ff_connect(platform = "sleeper", league_id = "522458773317046272", season = 2020)
x <- ff_transactions(jml_conn, week = 1:17)
```

`ff_userleagues` *Get User Leagues*

Description

This function returns a tidy dataframe with one row for every league a user is in. This requries authentication cookies for MFL usage.

Usage

```
ff_userleagues(conn, ...)

## S3 method for class 'flea_conn'
ff_userleagues(conn = NULL, user_email = NULL, season = NULL, ...)

## S3 method for class 'mfl_conn'
ff_userleagues(conn, season = NULL, ...)

## S3 method for class 'sleeper_conn'
ff_userleagues(conn = NULL, user_name = NULL, season = NULL, ...)
```

Arguments

`conn`	a connection object created by `ff_connect()`
`...`	arguments that may be passed to other methods (for method consistency)
`user_email`	the username to look up - defaults to user created in conn if available
`season`	the season to look up leagues for
`user_name`	the username to look up - defaults to user created in conn if available

Value

A tidy dataframe with one row for every league a user is in

Methods (by class)

- `flea_conn`: flea: returns a listing of leagues for a given user_email
- `mfl_conn`: MFL: With username/password, it will return a list of user leagues.
- `sleeper_conn`: Sleeper: returns a listing of leagues for a given user_id or user_name

See Also

`fleaflicker_userleagues` to call this function for flea leagues without first creating a connection object.

`sleeper_userleagues` to call this function for Sleeper leagues without first creating a connection object.

`fleaflicker_connect` *Connect to Fleaflicker League*

Description

This function creates a connection object which stores parameters and a user ID if available.

Usage

```
fleaflicker_connect(
  season = NULL,
  league_id = NULL,
  user_email = NULL,
  user_agent = NULL,
  rate_limit = TRUE,
  rate_limit_number = NULL,
  rate_limit_seconds = NULL,
  ...
)
```

Arguments

`season`	Season to access on Fleaflicker - if missing, will guess based on system date (current year if March or later, otherwise previous year)
`league_id`	League ID
`user_email`	Optional - attempts to get user's user ID by email
`user_agent`	User agent to self-identify (optional)
`rate_limit`	TRUE by default - turn off rate limiting with FALSE
`rate_limit_number`	number of calls per `rate_limit_seconds`, suggested is under 1000 calls per 60 seconds
`rate_limit_seconds`	number of seconds as denominator for rate_limit
`...`	other arguments (for other methods, for R compat)

Value

a list that stores Fleaflicker connection objects

fleaflicker_getendpoint *GET any Fleaflicker endpoint*

Description

The endpoint names and HTTP parameters (i.e. argument names) are CASE SENSITIVE and should be passed in exactly as displayed on the Fleaflicker API reference page.

Usage

```
fleaflicker_getendpoint(endpoint, ...)
```

Arguments

endpoint	a string defining which endpoint to return from the API
...	Arguments which will be passed as "argumentname = argument" in an HTTP query parameter

Details

Check out the vignette for more details and example usage.

Value

A list object containing the query, response, and parsed content.

See Also

https://www.fleaflicker.com/api-docs/index.html

```
vignette("fleaflicker_getendpoint")
```

fleaflicker_players *fleaflicker players library*

Description

A cached table of Fleaflicker NFL players. Will store in memory for each session! (via memoise in zzz.R)

Usage

```
fleaflicker_players(conn, page_limit = NULL)
```

Arguments

conn	a conn object created by ff_connect()
page_limit	A number limiting the number of players to return, or NULL (default) returns all

Value

a dataframe containing all ~7000+ players in the Fleaflicker database

Examples

```
conn <- fleaflicker_connect(2020, 312861)
player_list <- fleaflicker_players(conn, page_limit = 2)
```

---

fleaflicker_userleagues

flea - Get User Leagues

Description

This function returns the leagues that a specific user is in. This variant can be used without first creating a connection object.

Usage

```
fleaflicker_userleagues(user_email, season = NULL)
```

Arguments

user_email	the username to look up
season	the season to return leagues from - defaults to current year based on heuristics

Value

a dataframe of leagues for the specified user

See Also

ff_userleagues

`mfl_connect` *Connect to MFL League*

Description

This function creates a connection object which stores parameters and gets a login-cookie if available

Usage

```
mfl_connect(
  season = NULL,
  league_id = NULL,
  APIKEY = NULL,
  user_name = NULL,
  password = NULL,
  user_agent = NULL,
  rate_limit = TRUE,
  rate_limit_number = NULL,
  rate_limit_seconds = NULL,
  ...
)
```

Arguments

`season`	Season to access on MFL - if missing, will guess based on system date (current year if March or later, otherwise previous year)
`league_id`	league_id Numeric ID parameter for each league, typically found in the URL
`APIKEY`	APIKEY - optional - allows access to private leagues. Key is unique for each league and accessible from Developer's API page (currently assuming one league at a time)
`user_name`	MFL user_name - optional - when supplied in conjunction with a password, will attempt to retrieve authentication token
`password`	MFL password - optional - when supplied in conjunction with user_name, will attempt to retrieve authentication token
`user_agent`	A string representing the user agent to be used to identify calls - may find improved rate_limits if verified token
`rate_limit`	TRUE by default, pass FALSE to turn off rate limiting
`rate_limit_number`	number of calls per `rate_limit_seconds`, suggested is 60 calls per 60 seconds
`rate_limit_seconds`	number of seconds as denominator for rate_limit
`...`	silently swallows up unused arguments

Value

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

See Also

`ff_connect`, `sleeper_connect`

Examples

```
mfl_connect(season = 2020, league_id = 54040)
mfl_connect(season = 2019, league_id = 54040, rate_limit = FALSE)
```

`mfl_getendpoint` *GET any MFL endpoint*

Description

Create a GET request to any MFL export endpoint.

Usage

```
mfl_getendpoint(conn, endpoint, ...)
```

Arguments

`conn`	the list object created by `mfl_connect()`
`endpoint`	a string defining which endpoint to return from the API
`...`	Arguments which will be passed as "argumentname = argument" in an HTTP query parameter

Details

This function will read the connection object and automatically pass in the rate-limiting, league ID (L), authentication cookie, and/or API key (APIKEY) if configured in the connection object.

The endpoint names and HTTP parameters (i.e. argument names) are CASE SENSITIVE and should be passed in exactly as displayed on the MFL API reference page.

Check out the vignette for more details and example usage.

Value

A list object containing the query, response, and parsed content.

See Also

`https://api.myfantasyleague.com/2020/api_info?STATE=details`

`vignette("mfl_getendpoint")`

mfl_players *MFL players library*

Description

A cached table of MFL players. Will store in memory for each session! (via memoise in zzz.R)

Usage

```
mfl_players(conn = NULL)
```

Arguments

conn	optionally, pass in a conn object generated by ff_connect to receive league-specific custom players

Value

a dataframe containing all ~2000+ players in the MFL database

Examples

```
player_list <- mfl_players()
dplyr::sample_n(player_list, 5)
```

sleeper_connect *Connect to Sleeper League*

Description

This function creates a connection object which stores parameters and a user ID if available.

Usage

```
sleeper_connect(
  season = NULL,
  league_id = NULL,
  user_name = NULL,
  user_agent = NULL,
  rate_limit = TRUE,
  rate_limit_number = NULL,
  rate_limit_seconds = NULL,
  ...
)
```

Arguments

season	Season to access on Sleeper - if missing, will guess based on system date (current year if March or later, otherwise previous year)
league_id	League ID (currently assuming one league at a time)
user_name	Sleeper user_name - optional - attempts to get user's user ID
user_agent	User agent to self-identify (optional)
rate_limit	TRUE by default - turn off rate limiting with FALSE
rate_limit_number	number of calls per rate_limit_seconds, suggested is under 1000 calls per 60 seconds
rate_limit_seconds	number of seconds as denominator for rate_limit
...	other arguments (for other methods)

Value

a list that stores Sleeper connection objects

---

sleeper_getendpoint *GET any SLEEPER endpoint*

Description

The endpoint names and HTTP parameters (i.e. argument names) are CASE SENSITIVE and should be passed in exactly as displayed on the Sleeper API reference page.

Usage

```
sleeper_getendpoint(endpoint, ...)
```

Arguments

endpoint	a string defining which endpoint to return from the API
...	Arguments which will be passed as "argumentname = argument" in an HTTP query parameter

Details

Check out the vignette for more details and example usage.

Value

A list object containing the query, response, and parsed content.

See Also

https://docs.sleeper.app

vignette("sleeper_getendpoint")

`sleeper_players` *Sleeper players library*

Description

A cached table of Sleeper NFL players. Will store in memory for each session! (via memoise in zzz.R)

Usage

```
sleeper_players()
```

Value

a dataframe containing all ~7000+ players in the Sleeper database

Examples

```
player_list <- sleeper_players()
```

`sleeper_userleagues` *Sleeper - Get User Leagues*

Description

This function returns the leagues that a specific user is in. This variant can be used without first creating a connection object.

Usage

```
sleeper_userleagues(user_name, season = NULL)
```

Arguments

`user_name`	the username to look up
`season`	the season to return leagues from - defaults to current year based on heuristics

Value

a dataframe of leagues for the specified user

See Also

ff_userleagues

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Index