Package 'ffscrapr'

August 17, 2020

Type Package

Title API Client for Fantasy Football League Platforms

Version 1.0.0

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

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

Depends R (>= 3.0.0)

Imports dplyr (>= 0.8.0), 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), 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)

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 2020-08-17 16:40:06 UTC

R topics documented:

`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 trade values

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 sheet

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

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, ...)
```

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. rate_limit Defaults to TRUE. Pass FALSE to turn off. rate_limit_number number of attempts to try in rate_limit_seconds. APIKEY MFL-specific - grants access to perform something as a user in a specific league user_name Collects user_name field for leagues that support it (MFL, Sleeper so far) password Collects password and attempts to fetch an authorization token which functions a lot like an APIKEY user_agent Identifies user scraping the data

Value

a connection object to be used with ff_* functions

See Also

mfl_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 table of the 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 'mfl_conn'
ff_draft(conn, custom_players = FALSE, ...)
```

Arguments

conn	a conn object created by ff_connect()
...	not sure if there'll be other params yet!
custom_players	TRUE or FALSE - retrieve custom players from the MFL database? (Devy, placeholder picks etc)

Value

A tibble of draft results

Examples

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

---

ff_draftpicks *Get Draft Picks*

Description

Get current and future draft picks that have not yet been selected/converted into players

Usage

```
ff_draftpicks(conn, ...)

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

Arguments

conn	the list object created by ff_connect()
...	other arguments (currently unused)

Value

a tibble detailing future draft picks

Examples

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

ff_franchises *Get League Franchises*

Description

Returns a tidy dataframe of franchises and any relevant information

Usage

```
ff_franchises(conn)

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

Arguments

conn	a conn object created by `ff_connect()`

Value

A tibble of franchises, complete with IDs

Examples

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

ff_league *Get League Summary*

Description

This function returns a tibble of common league settings, including details like "1QB" or "2QB/SF", best ball, team count etc

Usage

```
ff_league(conn)

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

Arguments

conn	the connection object created by `ff_connect()`

Value

A one-row tibble of league settings.

Examples

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

---

ff_playerscores *Get Player Scoring History*

Description

This function returns a tibble of player scores based on league rules

Usage

```
ff_playerscores(conn, season, week, ...)

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

Arguments

conn	the list object created by ff_connect()
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)
...	other arguments (currently unused)

Value

A tibble of historical player scoring

Examples

```
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 tibble of team rosters

Usage

```
ff_rosters(conn, ...)

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

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 tibble of rosters, joined to basic player information and basic franchise information

Examples

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

ff_schedule *Get Schedule*

Description

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

Usage

```
ff_schedule(conn, ...)

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

Arguments

conn	a conn object created by `ff_connect()`
...	additional args which might be used eventually

Value

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

Examples

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

---

ff_scoring *Get League Scoring settings*

Description

This function returns a dataframe with detailed scoring settings, broken down by position, event, range, and points.

Usage

```
ff_scoring(conn)

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

Arguments

conn	a conn object created by `ff_connect()`

Value

A tibble of league scoring rules for each position defined.

See Also

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

Examples

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

ff_standings *Get Standings*

Description

This function returns a tibble of season-long fantasy team stats

Usage

```
ff_standings(conn, ...)

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

Arguments

conn	a conn object created by ff_connect()
...	arguments passed to other methods (currently none)

Value

A tibble of standings

Examples

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

ff_transactions *Get League Transactions*

Description

This function returns a tibble of transactions

Usage

```
ff_transactions(conn, ...)

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

Arguments

conn	the list object created by ff_connect()
...	additional args
custom_players	TRUE or FALSE - fetch custom players

Value

A tidy dataframe of transaction data

Examples

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

---

forget *Clear memoised cache*

Description

See `memoise::forget` for details.

---

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 = 4,
  rate_limit_seconds = 5
)
```

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

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 = 100,
  rate_limit_seconds = 60
)
```

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 100 calls per 60 seconds
`rate_limit_seconds`	number of seconds as denominator for rate_limit

Value

a list that stores MFL connection objects

Index