

Package ‘yfR’

July 21, 2025

Title Downloads and Organizes Financial Data from Yahoo Finance

Version 1.1.2

Description

Facilitates download of financial data from Yahoo Finance <<https://finance.yahoo.com/>>, a vast repository of stock price data across multiple financial exchanges. The package offers a local caching system and support for parallel computation.

URL <https://github.com/ropensci/yfR>, <https://docs.ropensci.org/yfR/>

BugReports <https://github.com/ropensci/yfR/issues>

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stringr, tidyr, lubridate, furrr, purrr, future, tibble, zoo, cli, readr, rvest, dplyr, quantmod (>= 0.4.20), magrittr, methods, pingr, tidyselect, glue, httr, jsonlite, janitor

License MIT + file LICENSE

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, covr, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Marcelo Perlin [aut, cre],

Nic Crane [rev] (Nic reviewed the package (v. 0.0.5) for rOpenSci, see <<https://github.com/ropensci/software-review/issues/523>>), Alexander Fischer [rev] (Alexander reviewed the package (v. 0.0.5) for rOpenSci, see <<https://github.com/ropensci/software-review/issues/523>>)

Maintainer Marcelo Perlin <marceloperlin@gmail.com>

Repository CRAN

Date/Publication 2025-05-19 18:00:02 UTC

Contents

yf_cachefolder_get	2
yf_collection_get	2
yf_convert_to_wide	3
yf_get	4
yf_get_available_collections	7
yf_get_dividends	7
yf_index_composition	8
yf_index_list	9
yf_live_prices	9
Index	10

yf_cachefolder_get	Returns the default folder for caching
--------------------	--

Description

By default, yfR uses a temp dir to store files.

Usage

```
yf_cachefolder_get()
```

Value

a path (string)

Examples

```
print(yf_cachefolder_get())
```

yf_collection_get	Downloads a collection of data from Yahoo Finance
-------------------	---

Description

This function will use a set collection of YF data, such as index components and will download all data from Yahoo Finance using [yf_get](#).

Usage

```

yf_collection_get(
  collection,
  first_date = Sys.Date() - 30,
  last_date = Sys.Date(),
  do_parallel = FALSE,
  do_cache = TRUE,
  cache_folder = yf_cachefolder_get(),
  ...
)

```

Arguments

collection	A collection to fetch data (e.g. "SP500", "IBOV", "FTSE"). See function yf_get_available_collections for finding all available collections
first_date	The first date of query (Date or character as YYYY-MM-DD)
last_date	The last date of query (Date or character as YYYY-MM-DD)
do_parallel	Flag for using parallel or not (default = FALSE). Before using parallel, make sure you call function <code>future::plan()</code> first. See https://furrr.futureverse.org/ for more details.
do_cache	Use cache system? (default = TRUE)
cache_folder	Where to save cache files? (default = <code>yfR::yf_cachefolder_get()</code>)
...	Other arguments passed to yf_get

Value

A data frame with financial prices from collection

Examples

```

df_yf <- yf_collection_get(collection = "IBOV",
                           first_date = Sys.Date() - 30,
                           last_date = Sys.Date()
)

```

yf_convert_to_wide	<i>Transforms a long (stacked) data frame into a list of wide data frames</i>
--------------------	---

Description

Transforms a long (stacked) data frame into a list of wide data frames

Usage

```
yf_convert_to_wide(df_in)
```

Arguments

df_in dataframe in the long format (probably the output of yf_get())

Value

A list with dataframes in the wide format (each element is a different column)

Examples

```
my_f <- system.file("extdata/example_data_yfR.rds", package = "yfR")
df_tickers <- readRDS(my_f)

print(df_tickers)

l_wide <- yf_convert_to_wide(df_tickers)
l_wide
```

yf_get

Download financial data from Yahoo Finance

Description

Based on a ticker (id of a stock) and time period, this function will download stock price data from Yahoo Finance and organizes it in the long format. Yahoo Finance <<https://finance.yahoo.com/>> provides a vast repository of stock price data around the globe. It cover a significant number of markets and assets, being used extensively in academic research and teaching. In the website you can lookup the ticker of a company.

Usage

```
yf_get(
  tickers,
  first_date = Sys.Date() - 30,
  last_date = Sys.Date(),
  thresh_bad_data = 0.75,
  bench_ticker = "^GSPC",
  type_return = "arit",
  freq_data = "daily",
  how_to_aggregate = "last",
  do_complete_data = FALSE,
  do_cache = TRUE,
  cache_folder = yf_cachefolder_get(),
  do_parallel = FALSE,
  be_quiet = FALSE
)
```

Arguments

<code>tickers</code>	A single or vector of tickers. If not sure whether the ticker is available, search for it in YF < https://finance.yahoo.com/ >.
<code>first_date</code>	The first date of query (Date or character as YYYY-MM-DD)
<code>last_date</code>	The last date of query (Date or character as YYYY-MM-DD)
<code>thresh_bad_data</code>	A percentage threshold for defining bad data. The dates of the benchmark ticker are compared to each asset. If the percentage of non-missing dates with respect to the benchmark ticker is lower than <code>thresh_bad_data</code> , the function will ignore the asset (default = 0.75)
<code>bench_ticker</code>	The ticker of the benchmark asset used to compare dates. My suggestion is to use the main stock index of the market from where the data is coming from (default = ^GSPC (SP500, US market))
<code>type_return</code>	Type of price return to calculate: 'arit' - arithmetic (default), 'log' - log returns.
<code>freq_data</code>	Frequency of financial data: 'daily' (default), 'weekly', 'monthly', 'yearly'
<code>how_to_aggregate</code>	Defines whether to aggregate the data using the first observations of the aggregating period or last ('first', 'last'). For example, if <code>freq_data</code> = 'yearly' and <code>how_to_aggregate</code> = 'last', the last available day of the year will be used for all aggregated values such as <code>price_adjusted</code> . (Default = "last")
<code>do_complete_data</code>	Return a complete/balanced dataset? If TRUE, all missing pairs of ticker-date will be replaced by NA or closest price (see input <code>do_fill_missing_prices</code>). Default = FALSE.
<code>do_cache</code>	Use cache system? (default = TRUE)
<code>cache_folder</code>	Where to save cache files? (default = <code>yfR::yf_cachefolder_get()</code>)
<code>do_parallel</code>	Flag for using parallel or not (default = FALSE). Before using parallel, make sure you call function <code>future::plan()</code> first. See < https://furry.futureverse.org/ > for more details.
<code>be_quiet</code>	Flag for not printing statements (default = FALSE)

Value

A dataframe with the financial data for working days, when markets are open. All price data is **measured** at the unit of the financial exchange. For example, price data for META (NYSE/US) is measured in dollars, while price data for PETR3.SA (B3/BR) is measured in Reais (Brazilian currency).

The return dataframe contains the following columns:

- ticker** The requested tickers (ids of stocks)
- ref_date** The reference day (this can also be year/month/week when using argument `freq_data`)
- price_open** The opening price of the day/period
- price_high** The highest price of the day/period

price_close The close/last price of the day/period

volume The financial volume of the day/period

price_adjusted The stock price adjusted for corporate events such as splits, dividends and others – this is usually what you want/need for studying stocks as it represents the actual financial performance of stockholders

ret_adjusted_prices The arithmetic or log return (see input `type_return`) for the adjusted stock prices

ret_prices The arithmetic or log return (see input `type_return`) for the closing stock prices

cumret_adjusted_prices The accumulated arithmetic/log return for the period (starts at 100%)

The cache system

The yfR's cache system is basically a bunch of rds files that are saved every time data is imported from YF. It indexes all data by ticker and time period. Whenever a user asks for a dataset, it first checks if the ticker/time period exists in cache and, if it does, loads the data from the rds file.

By default, a temporary folder is used (see function `yf_cache_folder_get`, which means that all cache files are session-persistent. In practice, whenever you restart your R/RStudio session, all cache files are lost. This is a choice I've made due to the fact that merging adjusted stock price data after corporate events (dividends/splits) is a mess and prone to errors. This only happens for stock price data, and not indices data.

If you really need a persistent cache folder, which is Ok for indices data, simply set a path with argument `cache_folder` (see warning section).

Warning

Be aware that when using cache system in a local folder (and not the default `tempdir()`), the aggregate prices series might not match if a split or dividends event happens in between cache files.

Examples

```
tickers <- c("TSLA", "MMM")

first_date <- Sys.Date() - 30
last_date <- Sys.Date()

df_yf <- yf_get(
  tickers = tickers,
  first_date = first_date,
  last_date = last_date
)

print(df_yf)
```

yf_get_available_collections	<i>Returns available collections</i>
------------------------------	--------------------------------------

Description

Returns available collections

Usage

```
yf_get_available_collections(print_description = FALSE)
```

Arguments

print_description	Logical (TRUE/FALSE) - flag for printing description of available indices/collections
-------------------	---

Value

A string vector with available collections

Examples

```
print(yf_get_available_collections())
```

yf_get_dividends	<i>Get Yahoo Finance Dividends from a single stock</i>
------------------	--

Description

This function will use the json api to retrieve dividends from Yahoo finance.

Usage

```
yf_get_dividends(ticker, first_date = Sys.Date() - 365, last_date = Sys.Date())
```

Arguments

ticker	a single ticker symbol
first_date	The first date of query (Date or character as YYYY-MM-DD)
last_date	The last date of query (Date or character as YYYY-MM-DD)

Value

a tibble with dividends

Examples

```
yf_get_dividends(ticker = "PETR4.SA")
```

yf_index_composition	<i>Get current composition of stock indices</i>
----------------------	---

Description

Get current composition of stock indices

Usage

```
yf_index_composition(  
  mkt_index,  
  do_cache = TRUE,  
  cache_folder = yf_cachefolder_get(),  
  force_fallback = FALSE  
)
```

Arguments

mkt_index	the index (e.g. IBOV, SP500, FTSE)
do_cache	Use cache system? (default = TRUE)
cache_folder	Where to save cache files? (default = yfR::yf_cachefolder_get())
force_fallback	Logical (TRUE/FALSE). Forces the function to use the fallback system

Value

A dataframe with the index composition (column might vary)

Examples

```
df_sp500 <- yf_index_composition("SP500")
```

yf_index_list	<i>Get available indices in package</i>
---------------	---

Description

This function will return all available market indices that are registered in the package.

Usage

```
yf_index_list(print_description = FALSE)
```

Arguments

print_description
Logical (TRUE/FALSE) - flag for printing description of available indices/collections

Value

A vector of mkt indices

Examples

```
indices <- yf_index_list()
indices
```

yf_live_prices	<i>Yahoo Finance Live Prices</i>
----------------	----------------------------------

Description

This function will use the json api to retrieve live prices from Yahoo finance.

Usage

```
yf_live_prices(ticker)
```

Arguments

ticker a single ticker symbol

Value

a tibble with live prices

Examples

```
yfR::yf_live_prices("PETR4.SA")
```

Index

yf_cachefolder_get, [2](#), [6](#)
yf_collection_get, [2](#)
yf_convert_to_wide, [3](#)
yf_get, [2](#), [3](#), [4](#)
yf_get_available_collections, [3](#), [7](#)
yf_get_dividends, [7](#)
yf_index_composition, [8](#)
yf_index_list, [9](#)
yf_live_prices, [9](#)