Package 'yahoofinancer'

May 21, 2026

Type Package

Title Fetch Data from Yahoo Finance API

Version 0.5.0

Description Obtain historical and near real time data related to stocks, index and currencies from the Yahoo Finance API. This package is community maintained and is not officially supported by 'Yahoo'. The accuracy of data is only as correct as provided on <https://finance.yahoo.com/>.

Depends R(>= 3.4)

Imports curl, dplyr, httr, jsonlite, lubridate, magrittr, purrr, R6

Suggests covr, httptest, httptest2, testthat (>= 3.0.0)

Config/testthat/edition 3

License MIT + file LICENSE

Encoding UTF-8

URL https://yahoofinancer.rsquaredacademy.com/, https://github.com/rsquaredacademy/yahoofinancer

BugReports https://github.com/rsquaredacademy/yahoofinancer/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Aravind Hebbali [aut, cre]

Maintainer Aravind Hebbali <hebbali.aravind@gmail.com>

Repository CRAN

Date/Publication 2026-05-21 08:40:02 UTC

Contents

`currency_converter` *Currency converter*

Description

Retrieve current conversion rate between two currencies as well as historical rates.

Usage

```
currency_converter(
  from = "EUR",
  to = "USD",
  start = NULL,
  end = NULL,
  period = "ytd",
  interval = "1d"
)
```

Arguments

`from`	Currency to convert from.
`to`	Currency to convert to.
`start`	Specific starting date. `String` or `date` object in `yyyy-mm-dd` format.
`end`	Specific ending date. `String` or `date` object in `yyyy-mm-dd` format.
`period`	Length of time. Defaults to `'ytd'` Valid values are: • `'1d'` • `'5d'` • `'1mo'` • `'3mo'` • `'6mo'` • `'1y'` • `'2y'` • `'5y'` • `'10y'` • `'ytd'` • `'max'`
`interval`	Time between data points. Defaults to `'1d'` Valid values are:

- '1h'
- '1d'
- '5d'
- '1wk'
- '1mo'
- '3mo'

Value

A data.frame.

Examples

```
currency_converter('GBP', 'USD', '2022-07-01', '2022-07-10')
currency_converter('GBP', 'USD', period = '1mo', interval = '1d')
```

---

get_currencies *Currencies*

Description

List of currencies Yahoo Finance supports.

Usage

```
get_currencies()
```

Value

Symbol, short and long name of the currencies.

Examples

```
get_currencies()
```

get_market_summary *Market Summary*

Description

Summary info of relevant exchanges for specific country.

Usage

```
get_market_summary(country = "US")
```

Arguments

country	Name of the country.

Value

A data.frame.

Examples

```
get_market_summary(country = 'US')
```

get_trending *Trending securities*

Description

List of trending securities for specific country.

Usage

```
get_trending(country = "US", count = 10)
```

Arguments

country	Name of the country.
count	Number of securities.

Value

Securities trending in the country.

Examples

```
get_trending()
```

Indice-class *R6 Class Representing an Index*

Description

Base class for getting all data related to indices from Yahoo Finance API.

Format

An R6 class object

Super class

`yahoofinancer::YahooFinanceBase` -> `Index`

Active bindings

`index` Deprecated. Returns `self$symbol`.

Methods

Public methods:

- `Index$new()`
- `Index$set_index()`
- `Index$clone()`

Method `new()`**:** Create a new Index object

Usage:

```
Index$new(symbol = NA, index = NA)
```

Arguments:

`symbol` Symbol

`index` Deprecated. Use `symbol` instead.

Returns: A new 'Index' object

Examples:

```
nifty_50 <- Index$new('^NSEI')
```

Method `set_index()`**:** Set a new index.

Usage:

```
Index$set_index(symbol = NA, index = NA)
```

Arguments:

`symbol` New symbol

`index` Deprecated. Use `symbol` instead.

Examples:

```
indice <- Index$new('^NSEI')
indice$set_index('^NDX')
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Index$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `Index$new`
## ------------------------------------------------

nifty_50 <- Index$new('^NSEI')

## ------------------------------------------------
## Method `Index$set_index`
## ------------------------------------------------

indice <- Index$new('^NSEI')
indice$set_index('^NDX')
```

`Ticker-class` *R6 Class Representing a Ticker*

Description

Base class for getting all data related to ticker from Yahoo Finance API.

Format

An R6 class object

Super class

`yahoofinancer::YahooFinanceBase` -> `Ticker`

Active bindings

`valuation_measures` Retrieves valuation measures

Methods

Public methods:

- `Ticker$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Ticker$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Tickers` *Multi-Symbol Ticker Data Aggregator*

Description

An R6 class to manage and aggregate data from multiple Yahoo Finance tickers simultaneously. It wraps the `Ticker` class to provide a unified, long-format data interface suitable for bulk analysis and visualization.

Details

The `Tickers` class automates the process of iterating over a vector of symbols. It handles per-symbol errors gracefully via an internal `aggregate_data` helper, ensuring that a failure in one symbol does not prevent the collection of data for others.

Most properties return a `data.frame` where the first column is `symbol`, facilitating easy filtering and joining in tidy workflows.

Public fields

`symbols` A unique character vector of symbols being tracked.

`ticker_objs` A named list of underlying `Ticker` R6 objects.

Active bindings

`recommendations` Related symbols suggested by Yahoo Finance and their relevance scores.

`valuation_measures` Quarterly valuation statistics including PE and Enterprise Value.

`technical_insights` Technical indicator snapshots (e.g., RSI, Moving Averages).

`regular_market_price` The current market price for each symbol.

`regular_market_time` The timestamp of the last market trade.

`regular_market_volume` The current trading volume.

`regular_market_day_high` The highest price during the current trading session.

`regular_market_day_low` The lowest price during the current trading session.

`previous_close` The closing price of the previous trading day.

`fifty_two_week_high` The highest price over the last 52 weeks.

`fifty_two_week_low` The lowest price over the last 52 weeks.

`currency` The currency code (e.g., "USD") for the symbols.

`exchange_name` The short name of the stock exchange.

`full_exchange_name` The full name of the stock exchange.

`first_trade_date` The Unix timestamp of the first recorded trade.

`timezone` The timezone code (e.g., "EDT").

`exchange_timezone_name` The full name of the exchange's timezone.

Methods

Public methods:

- `Tickers$new()`
- `Tickers$get_history()`
- `Tickers$aggregate_data()`
- `Tickers$clone()`

Method `new()`**:** Create a new Tickers object.

Usage:

```
Tickers$new(symbols)
```

Arguments:

`symbols` A character vector of Yahoo Finance ticker symbols.

Returns: A new `Tickers` object.

Method `get_history()`**:** Retrieve historical market data for all symbols.

Usage:

```
Tickers$get_history(period = "1y", interval = "1d", start = NULL, end = NULL)
```

Arguments:

`period` The duration of history to fetch; "1d", "5d", "1mo", "1y", "max" etc.

`interval` The frequency of data points; "1m", "2m", "5m", "1h", "1d", "1wk" etc.

`start` Date or timestamp for the start of the period.

`end` Date or timestamp for the end of the period.

Returns: A tidy `data.frame` containing historical prices and volumes.

Method `aggregate_data()`**:** Internal helper to execute a method across all symbols and combine results.

Usage:

```
Tickers$aggregate_data(fn)
```

Arguments:

`fn` A function or anonymous function that takes a `Ticker` object.

Returns: A combined `data.frame` or `NULL`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Tickers$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
# Initialize for a set of tech stocks
stocks <- Tickers$new(c("AAPL", "MSFT", "GOOGL"))

# 1. Get 1 month of historical daily prices
hist_data <- stocks$get_history(period = "1mo", interval = "1d")
head(hist_data)

# 2. Get current market prices for the group
current_prices <- stocks$regular_market_price

# 3. View recommended related symbols and scores
recs <- stocks$recommendations

# 4. Get technical insights (RSI, Moving Averages)
tech <- stocks$technical_insights

## End(Not run)
```

`validate` *Symbol validation*

Description

Validate symbols before retrieving data.

Usage

```
validate(symbol = NA, index = NA)
```

Arguments

`symbol`	Ticker, index or fund name.
`index`	Deprecated. Use `symbol` instead.

Examples

```
validate("aapl")
validate("aapls")
```

Index