Package 'BatchGetSymbols'

October 12, 2019

Title Downloads and Organizes Financial Data for Multiple Tickers

Version 2.5.4

Description Makes it easy to download a large number of trade data from Yahoo Finance <https://finance.yahoo.com/>.

Date 2019-10-12

Depends R (>= 3.4.0), rvest, dplyr

Imports stringr, curl, quantmod, XML, tidyr, lubridate, scales, furrr, purrr, future, tibble, zoo

License GPL-2

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, testthat, ggplot2

VignetteBuilder knitr

NeedsCompilation no

Author Marcelo Perlin [aut, cre]

Maintainer Marcelo Perlin <marceloperlin@gmail.com>

Repository CRAN

Date/Publication 2019-10-12 12:40:03 UTC

R topics documented:

BatchGetSymbols 2
calc.ret 3
df.fill.na 4
fix.ticker.name 5
get.clean.data 5
GetFTSE100Stocks 6
GetIbovStocks 7
GetSP500Stocks 7
myGetSymbols 8
reshape.wide 9

Index **11**

`BatchGetSymbols` *Function to download financial data*

Description

This function is designed to make batch downloads of financial data using `getSymbols`. Based on a set of tickers and a time period, the function will download the data for each ticker and return a report of the process, along with the actual data in the long dataframe format. The main advantage of the function is that it automatically recognizes the source of the dataset from the ticker and structures the resulting data from different sources in the long format. A caching system is also presente, making it very fast.

Usage

```
BatchGetSymbols(tickers, first.date = Sys.Date() - 30,
  last.date = Sys.Date(), thresh.bad.data = 0.75,
  bench.ticker = "^GSPC", type.return = "arit", freq.data = "daily",
  do.complete.data = FALSE, do.fill.missing.prices = TRUE,
  do.cache = TRUE, cache.folder = "BGS_Cache", do.parallel = FALSE,
  be.quiet = FALSE)
```

Arguments

`tickers`	A vector of tickers. If not sure whether the ticker is available, check the websites of google and yahoo finance. The source for downloading the data can either be Google or Yahoo. The function automatically selects the source webpage based on the input ticker.
`first.date`	The first date to download data (date or char as YYYY-MM-DD)
`last.date`	The last date to download data (date or char as YYYY-MM-DD)
`thresh.bad.data`	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 thresh.bad.data, the function will ignore the asset (default = 0.75)
`bench.ticker`	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))
`type.return`	Type of price return to calculate: 'arit' (default) - aritmetic, 'log' - log returns.
`freq.data`	Frequency of financial data ('daily', 'weekly', 'monthly', 'yearly')
`do.complete.data`	Return a complete/balanced dataset? If TRUE, all missing pairs of ticker-date will be replaced by NA or closest price (see input do.fill.missing.prices). Default = FALSE.

`do.fill.missing.prices`
: Finds all missing prices and replaces them by their closest price with preference for the previous price. This ensures a balanced dataset for all assets, without any NA. Default = TRUE.

`do.cache`
: Use caching system? (default = TRUE)

`cache.folder`
: Where to save cache files? (default = 'BGS_Cache')

`do.parallel`
: Flag for using parallel or not (default = FALSE). Before using parallel, make sure you call function future::plan() first.

`be.quiet`
: Logical for printing statements (default = FALSE)

Value

A list with the following items:

df.control A dataframe containing the results of the download process for each asset

df.tickers A dataframe with the financial data for all valid tickers

Warning

Do notice that adjusted prices are not available from google finance. When using this source, the function will output NA values for this column.

See Also

getSymbols

Examples

```
tickers <- c('FB','MMM')

first.date <- Sys.Date()-30
last.date <- Sys.Date()

l.out <- BatchGetSymbols(tickers = tickers,
                         first.date = first.date,
                        last.date = last.date, do.cache=FALSE)

print(l.out$df.control)
print(l.out$df.tickers)
```

---

calc.ret *Function to calculate returns from a price and ticker vector*

Description

Created so that a return column is added to a dataframe with prices in the long (tidy) format.

Usage

```
calc.ret(P, tickers = rep("ticker", length(P)), type.return = "arit")
```

Arguments

P	Price vector
tickers	Ticker of symbols (usefull if working with long dataframe)
type.return	Type of price return to calculate: 'arit' (default) - aritmetic, 'log' - log returns.

Value

A vector of returns

Examples

```
P <- c(1,2,3)
R <- calc.ret(P)
```

---

df.fill.na *Replaces NA values in dataframe for closest price*

Description

Helper function for BatchGetSymbols. Replaces NA values and returns fixed dataframe.

Usage

```
df.fill.na(df.in)
```

Arguments

df.in	DAtaframe to be fixed

Value

A fixed dataframe.

Examples

```
df <- data.frame(price.adjusted = c(NA, 10, 11, NA, 12, 12.5, NA ), volume = c(1,10, 0, 2, 0, 1, 5))

df.fixed.na <- df.fill.na(df)
```

fix.ticker.name *Fix name of ticker*

Description

Removes bad symbols from names of tickers. This is useful for naming files with cache system.

Usage

```
fix.ticker.name(ticker.in)
```

Arguments

ticker.in	A bad ticker name

Value

A good ticker name

Examples

```
bad.ticker <- '^GSPC'
good.ticker <- fix.ticker.name(bad.ticker)
good.ticker
```

get.clean.data *Get clean data from yahoo/google*

Description

Get clean data from yahoo/google

Usage

```
get.clean.data(tickers, src = "yahoo", first.date, last.date)
```

Arguments

tickers	A vector of tickers. If not sure whether the ticker is available, check the websites of google and yahoo finance. The source for downloading the data can either be Google or Yahoo. The function automatically selects the source webpage based on the input ticker.
src	Source of data (yahoo or google)
first.date	The first date to download data (date or char as YYYY-MM-DD)
last.date	The last date to download data (date or char as YYYY-MM-DD)

Value

A dataframe with the cleaned data

Examples

```
df.sp500 <- get.clean.data('^GSPC',
                           first.date = as.Date('2010-01-01'),
                           last.date = as.Date('2010-02-01'))
```

`GetFTSE100Stocks` *Function to download the current components of the FTSE100 index from Wikipedia*

Description

This function scrapes the stocks that constitute the FTSE100 index from the wikipedia page at <https://en.wikipedia.org/wiki/FTSE_100_Index#List_of_FTSE_100_companies>.

Usage

```
GetFTSE100Stocks(do.cache = TRUE, cache.folder = "BGS_Cache")
```

Arguments

`do.cache`	Use caching system? (default = TRUE)
`cache.folder`	Where to save cache files? (default = 'BGS_Cache')

Value

A dataframe that includes a column with the list of tickers of companies that belong to the FTSE100 index

Examples

```
## Not run:
df.FTSE100 <- GetFTSE100Stocks()
print(df.FTSE100$tickers)

## End(Not run)
```

`GetIbovStocks` *Function to download the current components of the Ibovespa index from Bovespa website*

Description

This function scrapes the stocks that constitute tSP500 index from the wikipedia page at http://bvmf.bmfbovespa.com.br/indic br.

Usage

```
GetIbovStocks(do.cache = TRUE, cache.folder = "BGS_Cache",
  max.tries = 10)
```

Arguments

`do.cache`	Use caching system? (default = TRUE)
`cache.folder`	Where to save cache files? (default = 'BGS_Cache')
`max.tries`	Maximum number of attempts to download the data

Value

A dataframe that includes a column with the list of tickers of companies that belong to the Ibovespa index

Examples

```
## Not run:
df.ibov <- GetIbovStocks()
print(df.ibov$tickers)

## End(Not run)
```

`GetSP500Stocks` *Function to download the current components of the SP500 index from Wikipedia*

Description

This function scrapes the stocks that constitute the SP500 index from the wikipedia page at https://en.wikipedia.org/wiki/List_

Usage

```
GetSP500Stocks(do.cache = TRUE, cache.folder = "BGS_Cache")
```

Arguments

do.cache	Use caching system? (default = TRUE)
cache.folder	Where to save cache files? (default = 'BGS_Cache')

Value

A dataframe that includes a column with the list of tickers of companies that belong to the SP500 index

Examples

```
## Not run:
df.SP500 <- GetSP500Stocks()
print(df.SP500$tickers)

## End(Not run)
```

---

myGetSymbols *An improved version of function* getSymbols *from quantmod*

Description

This is a helper function to BatchGetSymbols and it should normaly not be called directly. The purpose of this function is to download financial data based on a ticker and a time period. The main difference from getSymbols is that it imports the data as a dataframe with proper named columns and saves data locally with the caching system.

Usage

```
myGetSymbols(ticker, i.ticker, length.tickers, src = "yahoo", first.date,
  last.date, do.cache = TRUE, cache.folder = file.path(tempdir(),
  "BGS_Cache"), df.bench = NULL, be.quiet = FALSE, thresh.bad.data)
```

Arguments

ticker	A single ticker to download data
i.ticker	A index for the stock that is downloading (for cat() purposes)
length.tickers	total number of stocks being downloaded (also for cat() purposes)
src	The source of the data ('google' or'yahoo')
first.date	The first date to download data (date or char as YYYY-MM-DD)
last.date	The last date to download data (date or char as YYYY-MM-DD)
do.cache	Use caching system? (default = TRUE)
cache.folder	Where to save cache files? (default = 'BGS_Cache')
df.bench	Data for bechmark ticker

be.quiet Logical for printing statements (default = FALSE)

thresh.bad.data
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 thresh.bad.data, the function will ignore the asset (default = 0.75)

Value

A dataframe with the financial data

See Also

getSymbols for the base function

Examples

```
ticker <- 'FB'

first.date <- Sys.Date()-30
last.date <- Sys.Date()

## Not run:
df.ticker <- myGetSymbols(ticker,
                          first.date = first.date,
                          last.date = last.date)

## End(Not run)
```

---

reshape.wide *Transforms a dataframe in the long format to a list of dataframes in the wide format*

Description

Transforms a dataframe in the long format to a list of dataframes in the wide format

Usage

```
reshape.wide(df.tickers)
```

Arguments

df.tickers Dataframe in the long format

Value

A list with dataframes in the wide format

Examples

```
my.f <- system.file( 'extdata/ExampleData.rds', package = 'BatchGetSymbols' )
df.tickers <- readRDS(my.f)
l.wide <- reshape.wide(df.tickers)
l.wide
```

Index

BatchGetSymbols, 2, *8*

calc.ret, 3

df.fill.na, 4

fix.ticker.name, 5

get.clean.data, 5
GetFTSE100Stocks, 6
GetIbovStocks, 7
GetSP500Stocks, 7
getSymbols, *2*, *3*, *8*, *9*

myGetSymbols, 8

reshape.wide, 9