Package 'GetHFData'

April 8, 2019

Title Download and Aggregate High Frequency Trading Data from Bovespa

Version 1.7

Date 2019-04-08

Description Downloads and aggregates high frequency trading data for Brazilian instruments directly from Bovespa ftp site <ftp://ftp.bmf.com.br/MarketData/>.

Depends R (>= 3.3.0)

Imports stringr,stats,RCurl, lubridate, readr, utils, curl,dplyr

License GPL-2

BugReports https://github.com/msperlin/GetHFData/issues

URL https://github.com/msperlin/GetHFData/

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, testthat, ggplot2

VignetteBuilder knitr

NeedsCompilation no

Author Marcelo Perlin [aut, cre],
Henrique Ramos [ctb]

Maintainer Marcelo Perlin <marceloperlin@gmail.com>

Repository CRAN

Date/Publication 2019-04-08 17:33:42 UTC

R topics documented:

`add.order` *Adds an order to the LOB*

Description

Adds an order to the LOB

Usage

```
add.order(my.lob, order.in, silent = TRUE)
```

Arguments

`my.lob`	A LOB (order book)
`order.in`	An order from the data
`silent`	Should the function print progress ? (TRUE (default) or FALSE)

Value

An LOB with the new order

Examples

```
# no example (internal)
```

`ghfd_build_lob` *Building LOB (limit order book) from orders*

Description

Building LOB (limit order book) from orders

Usage

```
ghfd_build_lob(df.orders, silent = TRUE)
```

Arguments

df.orders	A dataframe, output from ghfd_GetHFData
silent	Should the function print progress ? (TRUE (default) or FALSE)

Value

A dataframe with information about LOB

Examples

```
## Not run:
library(GetHFData)
first.time <- '11:00:00'
last.time <- '17:00:00'
first.date <- as.Date('2015-11-03')
last.date <- as.Date('2015-11-03')
type.output <- 'raw'
type.data <- 'orders'
type.market = 'equity-odds'

df.out <- ghfd_get_HF_data(my.assets =my.assets,
                           type.market = type.market,
                           type.data = type.data,
                           first.date = first.date,
                           last.date = last.date,
                           first.time = first.time,
                           last.time = last.time,
                           type.output = type.output)

df.lob <- ghfd_build_lob(df.out)

## End(Not run)
```

---

ghfd_download_file *Downloads a single file from Bovespa ftp*

Description

This function will take as input a ftp addresss, the name of the downloaded file in the local drive, and it will download the corresponding file. Returns TRUE if it worked and FALSE otherwise.

Usage

```
ghfd_download_file(my.ftp, out.file, dl.dir = "Dl Files",
  max.dl.tries = 10)
```

Arguments

my.ftp	A complete, including file name, ftp address to download the file from
out.file	Name of downloaded file with HFT data from Bovespa
dl.dir	The folder to download the zip files (default = 'ftp files')
max.dl.tries	Maximum attempts to download the files from ftp

Value

TRUE if sucessfull, FALSE if not

Examples

```
my.ftp <- 'ftp://ftp.bmf.com.br/MarketData/Bovespa-Opcoes/NEG_OPCOES_20151229.zip'
out.file <- 'temp.zip'

## Not run:
ghfd_download_file(my.ftp = my.ftp, out.file=out.file)

## End(Not run)
```

---

ghfd_get_available_tickers_from_file

Function to get available tickers from downloaded zip file

Description

This function will read the zip file downloaded from Bovespa and output a numeric vector where the names of the elements represents the different tickers and the numeric values as the number of trades for each ticker

Usage

```
ghfd_get_available_tickers_from_file(out.file)
```

Arguments

out.file	Name of downloaded file with HFT data from Bovespa

Value

A dataframe with the number of trades for each ticker found in file

Examples

```
## get file from package (usually this would be been downloaded from the ftp)
out.file <- system.file("extdata", 'NEG_OPCOES_20151126.zip', package = "GetHFData")

df.tickers <- ghfd_get_available_tickers_from_file(out.file)

print(head(df.tickers))
```

---

ghfd_get_available_tickers_from_ftp

Function to get available tickers from ftp

Description

This function will read the Bovespa ftp for a given market/date and output a numeric vector where the names of the elements represents the different tickers and the numeric values as the number of trades for each ticker

Usage

```
ghfd_get_available_tickers_from_ftp(my.date = "2015-11-03",
  type.market = "equity", type.data = "trades", dl.dir = "ftp files",
  max.dl.tries = 10)
```

Arguments

my.date	A single date to check tickers in ftp (e.g. '2015-11-03')
type.market	The type of market to download data from ('equity', 'equity-odds','options', 'BMF').
type.data	The type of financial data to download and aggregate ('trades' or 'orders').
dl.dir	The folder to download the zip files (default = 'ftp files')
max.dl.tries	Maximum attempts to download the files from ftp

Value

A data.frame with the tickers, number of found trades and file name

Examples

```
## Not run:
 df.tickers <- ghfd_get_available_tickers_from_ftp(my.date = '2015-11-03',
 type.market = 'BMF')

 print(head(df.tickers))

## End(Not run)
```

ghfd_get_ftp_contents *Gets the contents of Bovespa ftp*

Description

This function will access the Bovespa ftp and return a vector with all files related to trades (all others are ignored)

Usage

```
ghfd_get_ftp_contents(type.market = "equity", max.dl.tries = 10,
  type.data = "trades")
```

Arguments

type.market	The type of market to download data from ('equity', 'equity-odds','options', 'BMF').
max.dl.tries	Maximum attempts to download the files from ftp
type.data	The type of financial data to download and aggregate ('trades' or 'orders').

Value

A list with all files from the ftp that are related to executed trades

Examples

```
## Not run:
ftp.files <- ghfd_get_ftp_contents(type.market = 'equity')
print(ftp.files)

## End(Not run)
```

ghfd_get_HF_data *Downloads and aggregates high frequency trading data directly from the Bovespa ftp*

Description

This function downloads zip files containing trades from Bovespa's ftp (ftp://ftp.bmf.com.br/MarketData/) and imports it into R. See the vignette and examples for more details on how to use the function.

Usage

```
ghfd_get_HF_data(my.assets = NULL, type.matching = "exact",
  type.market = "equity", type.data = "trades",
  first.date = "2016-01-01", last.date = "2016-01-05",
  first.time = NULL, last.time = NULL, type.output = "agg",
  agg.diff = "15 min", dl.dir = "ftp files", max.dl.tries = 10,
  clean.files = FALSE, only.dl = FALSE)
```

Arguments

my.assets	The tickers (symbols) of the derised assets to import data (e.g. c('PETR4', 'VALE5')). The function allow for partial patching (e.g. 'PETR' for all assets related to Petrobras). Default is set to NULL (download all available tickers)
type.matching	Type of matching for asset names in data ('exact' or 'partial')
type.market	The type of market to download data from ('equity', 'equity-odds','options', 'BMF').
type.data	The type of financial data to download and aggregate ('trades' or 'orders').
first.date	The first date of the imported data (e.g. '2016-01-01')
last.date	The last date of the imported data (e.g. '2016-01-05')
first.time	The first intraday period to import the data. All trades/orders before this time of day are ignored. As character, e.g. '10:00:00'.
last.time	The last intraday period to import the data. All trades/orders after this time of day are ignored. As character, e.g. '18:00:00'.
type.output	Defines the type of output of the data. The choice 'agg' outputs aggregated data for time intervals defined in agg.diff. The choice 'raw' outputs the raw, tick by tick/order by order, data from the zip files.
agg.diff	The time interval used in the aggregation of data. Only used for type.output='agg'. It should contain a integer followed by a time unit ('sec' or 'secs', 'min' or 'mins', 'hour' or 'hours', 'day' or 'days'). Example: agg.diff = '15 mins', agg.diff = '1 hour'.
dl.dir	The folder to download the zip files (default = 'ftp files')
max.dl.tries	Maximum attempts to download the files from ftp
clean.files	Logical. Should the files be removed after reading it? (TRUE or FALSE)
only.dl	Logical. Should the function only download the files? (TRUE or FALSE). This is usefull if you just want the file for later analysis

Value

A dataframe with the financial data in the raw format (tick by tick) or aggregated

Examples

```
my.assets <- 'ABEVA69'
type.market <- 'options'
```

```
first.date <- as.Date('2015-12-29')
last.date <- as.Date('2015-12-29')

## Not run:
df.out <- ghfd_get_HF_data(my.assets, type.market, first.date,  last.date)

## End(Not run)
```

---

ghfd_read_file *Reads zip file downloaded from Bovespa ftp (trades or orders)*

Description

Reads zip file downloaded from Bovespa ftp (trades or orders)

Usage

```
ghfd_read_file(out.file, my.assets = NULL, type.matching = "exact",
  type.data = "trades", first.time = "10:00:00",
  last.time = "17:00:00", type.output = "agg", agg.diff = "15 min")
```

Arguments

out.file	Name of zip file
my.assets	The tickers (symbols) of the derised assets to import data (e.g. c('PETR4', 'VALE5')). The function allow for partial patching (e.g. 'PETR' for all assets related to Petrobras). Default is set to NULL (download all available tickers)
type.matching	Type of matching for asset names in data ('exact' or 'partial')
type.data	The type of financial data to download and aggregate ('trades' or 'orders').
first.time	The first intraday period to import the data. All trades/orders before this time of day are ignored. As character, e.g. '10:00:00'.
last.time	The last intraday period to import the data. All trades/orders after this time of day are ignored. As character, e.g. '18:00:00'.
type.output	Defines the type of output of the data. The choice 'agg' outputs aggregated data for time intervals defined in agg.diff. The choice 'raw' outputs the raw, tick by tick/order by order, data from the zip files.
agg.diff	The time interval used in the aggregation of data. Only used for type.output='agg'. It should contain a integer followed by a time unit ('sec' or 'secs', 'min' or 'mins', 'hour' or 'hours', 'day' or 'days'). Example: agg.diff = '15 mins', agg.diff = '1 hour'.

Value

A dataframe with the raw (tick by tick/order by order) dataset

Examples

```
my.assets <- c('ABEVA20', 'PETRL78')

## getting data from local file (in practice it would be downloaded from ftp)
out.file <- system.file("extdata", 'NEG_OPCOES_20151126.zip', package = "GetHFData")

df.out <- ghfd_read_file(out.file, my.assets)
print(head(df.out))
```

`ghfd_read_file.orders` *Reads zip file downloaded from Bovespa ftp (orders) - INTERNAL USE*

Description

Reads zip file downloaded from Bovespa ftp (orders) - INTERNAL USE

Usage

```
ghfd_read_file.orders(out.file, my.assets = NULL, type.matching = NULL,
  first.time = "10:00:00", last.time = "17:00:00",
  type.output = "agg", agg.diff = "15 min")
```

Arguments

`out.file`	Name of zip file
`my.assets`	The tickers (symbols) of the derised assets to import data (e.g. c('PETR4', 'VALE5')). The function allow for partial patching (e.g. 'PETR' for all assets related to Petrobras). Default is set to NULL (download all available tickers)
`type.matching`	Type of matching for asset names in data ('exact' or 'partial')
`first.time`	The first intraday period to import the data. All trades/orders before this time of day are ignored. As character, e.g. '10:00:00'.
`last.time`	The last intraday period to import the data. All trades/orders after this time of day are ignored. As character, e.g. '18:00:00'.
`type.output`	Defines the type of output of the data. The choice 'agg' outputs aggregated data for time intervals defined in agg.diff. The choice 'raw' outputs the raw, tick by tick/order by order, data from the zip files.
`agg.diff`	The time interval used in the aggregation of data. Only used for type.output='agg'. It should contain a integer followed by a time unit ('sec' or 'secs', 'min' or 'mins', 'hour' or 'hours', 'day' or 'days'). Example: agg.diff = '15 mins', agg.diff = '1 hour'.

Value

A dataframe with trade data (aggregated or raw)

Examples

```
# no example
```

`ghfd_read_file.trades` *Reads zip file downloaded from Bovespa ftp (trades) - INTERNAL USE*

Description

Reads zip file downloaded from Bovespa ftp (trades) - INTERNAL USE

Usage

```
ghfd_read_file.trades(out.file, my.assets = NULL, type.matching = NULL,
  first.time = "10:00:00", last.time = "17:00:00",
  type.output = "agg", agg.diff = "15 min")
```

Arguments

`out.file`	Name of zip file
`my.assets`	The tickers (symbols) of the derised assets to import data (e.g. c('PETR4', 'VALE5')). The function allow for partial patching (e.g. 'PETR' for all assets related to Petrobras). Default is set to NULL (download all available tickers)
`type.matching`	Type of matching for asset names in data ('exact' or 'partial')
`first.time`	The first intraday period to import the data. All trades/orders before this time of day are ignored. As character, e.g. '10:00:00'.
`last.time`	The last intraday period to import the data. All trades/orders after this time of day are ignored. As character, e.g. '18:00:00'.
`type.output`	Defines the type of output of the data. The choice 'agg' outputs aggregated data for time intervals defined in agg.diff. The choice 'raw' outputs the raw, tick by tick/order by order, data from the zip files.
`agg.diff`	The time interval used in the aggregation of data. Only used for type.output='agg'. It should contain a integer followed by a time unit ('sec' or 'secs', 'min' or 'mins', 'hour' or 'hours', 'day' or 'days'). Example: agg.diff = '15 mins', agg.diff = '1 hour'.

Value

A dataframe with trade data (aggregated or raw)

Examples

```
# no example
```

organize.lob *Organizes LOB (internal function)*

Description

This internal recursive function organizes the lob by making sure that all prices and time are ordered. Every time that prices in the bid and ask matches, it will create a trade and modify the lob accordingly.

Usage

```
organize.lob(my.lob, silent = TRUE)
```

Arguments

my.lob	A LOB (order book)
silent	Should the function print progress ? (TRUE (default) or FALSE)

Value

An organized LOB

Examples

```
# no examples (internal)
```

print.lob *Prints the LOB*

Description

Prints the LOB

Usage

```
## S3 method for class 'lob'
print(my.lob, max.level = 3)
```

Arguments

my.lob	A LOB (order book)
max.level	Max level of lob to print

Value

nothing

Examples

```
# no example (internal)
```

`process.lob.from.df` *Process LOB from asset dataframe*

Description

Process LOB from asset dataframe

Usage

```
process.lob.from.df(asset.df, silent = T)
```

Arguments

`asset.df`	A dataframe with orders for a single asset
`silent`	Should the function print progress ? (TRUE (default) or FALSE)

Value

The lob for the single asset

Examples

```
# no example (internal)
```

Index