Package 'rusquant'

September 16, 2024

Type Package

Title Quantitative Trading Framework

Version 1.1.4

Date 2024-09-16

Depends quantmod,data.table,jsonlite,httr,xts

Imports XML,stringr,jose,stats,rvest,base64enc

Maintainer Vyacheslav Arbuzov <arbuzov1989@gmail.com>

Description

Collection of functions to retrieve financial data from various sources, including brokerage and exchange platforms, financial websites, and data providers. Includes functions to retrieve account information, portfolio information, and place/cancel orders from different brokers. Additionally, allows users to download historical data such as earnings, dividends, stock splits.

LazyLoad yes

License GPL-3

Encoding UTF-8

URL https://rusquant.ru

BugReports https://github.com/arbuzovv/rusquant/issues

RoxygenNote 7.2.3

NeedsCompilation no

Author Vyacheslav Arbuzov [cph, cre, aut],
Sergey Edunov [aut]

Repository CRAN

Date/Publication 2024-09-16 15:10:02 UTC

Contents

---

auth *Set creds for Datasource*

Description

Sets creds for the Datasource by storing it in an environment variable for the current R session. This token will be used by other functions in the package to authenticate API requests.

Usage

```
auth(src = "Moex", login, password)
```

Arguments

src	datasource name.
login	login of datasource
password	password of datasource

Value

Invisible NULL, side-effect function setting an environment variable.

Examples

```
## Not run:
  auth(login = "user@email.com",password = "mypassword")

## End(Not run)
```

---

cancelOrder   *Cancel an order on a broker/exchange platform*

Description

This function cancels an existing order on a specified broker or exchange platform

Usage

```
cancelOrder(
  src = "",
  api.key = "",
  orderId = "",
  clientId = "",
  board = "",
  live = TRUE,
  verbose = TRUE
)
```

Arguments

src	character string, the name of the broker/exchange platform, e.g. "tinkoff", "finam", "alor"
api.key	character string, the API key required for authentication
orderId	character string, the ID of the order to be cancelled
clientId	character string, the ID of the client account
board	character string, the name of the exchange board, required for some platforms
live	logical, whether to execute the order in a live environment, default is TRUE
verbose	logical, whether to print the HTTP response message, default is TRUE

Value

character string, the response message from the HTTP request

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Examples

```
cancelOrder(src = 'Finam',api.key = 'finam_token',orderId = 'otderID',clientId = 'your cliend id')
```

---

getAccounts   *Get account information from a brokerage or exchange*

Description

This function retrieves account information from a brokerage or exchange.

Usage

```
getAccounts(src = "tinkoff", api.key = "", verbose = FALSE)
```

Arguments

src	a character string specifying the brokerage or exchange. Can be one of "tinkoff" or "alor". Default is "tinkoff".
api.key	a character string representing the authorization key for the API.
verbose	a logical value indicating whether to print detailed information about the request/response. Default is FALSE.

Value

A list object with account information, or an error message if the request fails.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
# get account information from tinkoff
account_info <- getAccounts(src = "Tinkoff", api.key = "your_api_key")

# get account information from alor
account_info <- getAccounts(src = "Alor", api.key = "your_api_key")
```

getDividends   *Download dividends data*

Description

This function returns dividends data from Investing.com or Tinkoff broker.

Usage

```
getDividends(
  src = "investing",
  figi = "",
  api.key = "",
  from = Sys.Date() - 10,
  to = Sys.Date(),
  country = ""
)
```

Arguments

src	source of dividends information. Could be 'investing' or 'tinkoff'
figi	FIGI of the instrument to get dividends for (only for Tinkoff broker)
api.key	Tinkoff broker API key (only for Tinkoff broker)
from	start date of the dividends data. Default is 10 days ago
to	end date of the dividends data. Default is today
country	a character string with the country name to filter dividends data for (only for Investing.com)

Value

a list with dividends data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getDividends(from = Sys.Date(),to = Sys.Date()+10,country = "Australia")
```

---

getEarnings   *Download earnings data from investing.com*

Description

This function retrieves earnings data from the investing.com website for a specified time period

Usage

```
getEarnings(
  from = Sys.Date() - 5,
  to = Sys.Date() + 5,
  country = "United States"
)
```

Arguments

from	the start date in yyyy-mm-dd format (default is 10 days prior to current date)
to	the end date in yyyy-mm-dd format (default is the current date)
country	the country from which to get earnings data (default is United States)

Value

a data frame with earnings data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getEarnings(from = Sys.Date(),to = Sys.Date()+5,country='Belgium')
```

---

getEconomic   *Get economic data from Investing.com*

Description

This function retrieves economic data from the investing.com website for a specified time period

Usage

```
getEconomic(from = Sys.Date() - 10, to = Sys.Date(), country = "United States")
```

Arguments

from	A date in the format of YYYY-MM-DD. Defaults to 10 days ago from the current system date.
to	A date in the format of YYYY-MM-DD. Defaults to the current system date.
country	a character string with the country name to filter dividends data for (only for Investing.com)

Value

A data frame containing IPO calendar data for the specified date range.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getEconomic(from = Sys.Date(),to = Sys.Date()+35,country='Belgium')
```

---

getIPO   *Get IPO calendar data from Investing.com*

Description

This function retrieves IPO calendar data from Investing.com for a specified date range

Usage

```
getIPO(from = Sys.Date() - 10, to = Sys.Date())
```

Arguments

from   The start date of the date range in YYYY-MM-DD format. Default is the current
                  date minus 10 days.

to   The end date of the date range in YYYY-MM-DD format. Default is the current
                  date.

Value

A data frame containing IPO calendar data for the specified date range.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like
it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getIPO(from=Sys.Date(),to=Sys.Date()+3)
```

---

getOrderbook   *Get the order book for a given symbol from a supported exchange*

Description

This function retrieves order book information

Usage

```
getOrderbook(
  Symbols,
  depth = 10,
  src = "poloniex",
  adjust = FALSE,
  verbose = FALSE,
  auto.assign = TRUE,
  market = "shares",
  board = "tqbr",
  api.key = "",
  env = globalenv()
)
```

Arguments

Symbols	A character vector specifying the symbol to retrieve order book data for.
depth	An integer value specifying the number of levels of the order book to retrieve. Defaults to 10.
src	A character string specifying the exchange to retrieve data from. Possible values are kraken,poloniex,tinkoff,alor, binance.
adjust	A logical value indicating whether to adjust timestamps to match the system timezone. Defaults to FALSE.
verbose	A logical value indicating whether to print detailed messages during the function's execution. Defaults to FALSE.
auto.assign	A logical value indicating whether to automatically assign the resulting object to the current environment. Defaults to TRUE.
market	A character string specifying type of market
board	A character string specifying type of board
api.key	A character string specifying the API key to use when retrieving data from Alor or Tinkoff. Defaults to "".
env	environment where the data will be assigned

Value

A data.table object containing the order book data for the specified symbol.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getOrderbook('USDTGBP', src = 'kraken')
getOrderbook('BTC_USDT', src = 'poloniex')
```

---

getOrders   *Retrieve Orders Information from Brokers*

Description

This function retrieves information about orders from different brokers/exchanges

Usage

```
getOrders(
  src = "",
  board = "MOEX",
  api.key = "",
  orderId = "",
  clientId = "",
  stopOrders = FALSE,
  verbose = TRUE
)
```

Arguments

src	Character string specifying the source broker/exchange (e.g., "tinkoff", "finam", "alor")
board	Character string specifying the trading board (default is "MOEX")
api.key	Character string specifying the API key for the broker/exchange
orderId	Character string specifying the order ID to retrieve (default is "")
clientId	Character string specifying the client ID for the broker/exchange
stopOrders	Logical specifying whether to retrieve stop orders (default is FALSE)
verbose	Logical specifying whether to display additional information (default is TRUE)

Value

A list containing order information from the broker/exchange

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

See Also

getTrades cancelOrder placeOrder

Examples

```
# Retrieve all orders from Tinkoff
getOrders(src = "tinkoff", api.key = "your_api_key", clientId = "your_client_id")

# Retrieve all orders from Finam
getOrders(src = "finam", api.key = "your_api_key", clientId = "your_client_id")

# Retrieve all orders from Alor
getOrders(src = "alor", api.key = "your_api_key", clientId = "your_client_id")
```

---

getPortfolio *Retrieve portfolio data from different brokers/exchanges*

Description

This function retrieves portfolio data from different brokers/exchanges such as Tinkoff, Finam and Alor.

Usage

```
getPortfolio(
  src = "",
  board = "MOEX",
  api.key = "",
  clientId = "",
  verbose = TRUE
)
```

Arguments

src	character indicating the name of the broker/exchange (options are 'tinkoff', 'finam', or 'alor')
board	character indicating the name of the board (only required for Alor, default is 'MOEX')
api.key	character representing the authorization key required for accessing broker/exchange API
clientId	character representing the ID of the client whose portfolio data is being retrieved
verbose	logical value indicating whether to print verbose output (default is TRUE)

Value

A list of portfolio data containing the positions and other relevant information

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
# Retrieve portfolio data from Tinkoff
getPortfolio(src = 'tinkoff', api.key = 'my_api_key', clientId = 'my_client_id')

# Retrieve portfolio data from Finam
getPortfolio(src = 'finam', api.key = 'my_api_key', clientId = 'my_client_id')

# Retrieve portfolio data from Alor
getPortfolio(src = 'alor', api.key = 'my_api_key', clientId = 'my_client_id', board = 'MOEX')
```

---

getSplits	*Get stock split calendar data from investing.com*

Description

This function retrieves the stock split calendar data from investing.com between two given dates.

Usage

```
getSplits(from = Sys.Date() - 10, to = Sys.Date())
```

Arguments

from	A date in the format 'YYYY-MM-DD' representing the start of the date range to retrieve data for (default is Sys.Date()-10).
to	A date in the format 'YYYY-MM-DD' representing the end of the date range to retrieve data for (default is Sys.Date()).

Value

A data.table object containing the stock split calendar data from investing.com.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSplits(from=Sys.Date(),to=Sys.Date()+3)
```

---

getSymbolList	*Get a list of symbols for a given stock exchange*

Description

This function retrieves a list of symbols for a specified stock exchange from a variety of sources. The available sources are poloniex, rusquant, tinkoff, mfd, finam, alor, and kraken. The function returns a data.table object containing the symbol information for the requested exchange.

Usage

```
getSymbolList(
  src = "poloniex",
  verbose = FALSE,
  auto.assign = FALSE,
  country = "",
  api.key = "",
  type = "Shares",
  env = globalenv(),
  user_agent = NULL
)
```

Arguments

src	character indicating the source of the symbol list. Possible values are "poloniex", "rusquant", "tinkoff", "mfd", "finam", "alor", and "kraken".
verbose	logical indicating whether or not to print additional information. The default is FALSE.
auto.assign	logical indicating whether or not to automatically assign the data.table object to the global environment. The default is FALSE.
country	character indicating the country of the exchange. The default is an empty string.
api.key	character indicating the API key to be used for accessing the source. The default is an empty string.
type	character indicating the type of financial instruments to retrieve. Applicable for the "tinkoff", "gigapack","moex". Possible values are "Bonds", "Currencies", "Etfs", "Futures", "Options", and "Shares".
env	The environment where the data should be assigned. Defaults to the global environment.
user_agent	The special headers for parsing

Value

Returns a data.table object containing the symbol information for the specified exchange.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbolList()
#getSymbolList(src='moex')
#getSymbolList(src='moex',type='Forts')
```

---

getSymbols.Algopack   *Download AlgoPack data from MOEX*

Description

Download historical market data from AlgoPack for a given symbol and time range.

Usage

```
getSymbols.Algopack(
  Symbols = "",
  env = globalenv(),
  from = Sys.Date() - 30,
  to = Sys.Date(),
  date = Sys.Date(),
  verbose = FALSE,
  type = "tradestats",
  market = "eq",
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	a character vector of AlgoPack symbols to download data for.
env	environment where to create the downloaded data object.
from	a character string indicating the start date of the data to download, in YYYY-MM-DD format.
to	a character string indicating the end date of the data to download, in YYYY-MM-DD format.
date	a character string indicating the date of the data to download, in YYYY-MM-DD format.

verbose	a logical indicating whether to print the response details or not.
type	a character string indicating the AlgoPack type possible values are c('tradestats','orderstats','obstats','hi2' default 'tradestats'.
market	a character string indicating the market type possible values are c('eq','fo','fx'), default 'eq'.
auto.assign	a logical indicating whether to automatically assign the downloaded data to the global environment.
...	additional arguments passed to getSymbols.AlgoPack

Value

returns an data.table object containing financial data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Algopack('SBER',from = '2023-10-24',to='2023-11-04')
# open interest for available futures
getSymbols.Algopack(date = '2024-05-10',type = 'oi')
# open interest for Si futures
getSymbols.Algopack(Symbols = 'Si',type = 'oi')
# market concentration for available stocks
#getSymbols.Algopack(date = '2024-05-10',type = 'hi2')
# market concentration for current stock
#devgetSymbols.Algopack('SBER',from = '2023-10-24',to='2023-11-04',type = 'hi2')
# market concentration for available fx
#getSymbols.Algopack(date = '2024-05-10',type = 'hi2',market='fx')
# market concentration for available futures
#getSymbols.Algopack(date = '2024-05-10',type = 'hi2',market='fo')
# market concentration for CNYRUB_TOM
#getSymbols.Algopack(Symbols = 'CNYRUB_TOM',type = 'hi2',market='fx')
```

---

getSymbols.Alor *Download Alor data*

Description

Download historical market data from Alor for a given symbol and time range.

Usage

```
getSymbols.Alor(
  Symbols,
  env = globalenv(),
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  api.key = NULL,
  period = "day",
  verbose = TRUE,
  board = "MOEX",
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	a character vector of Alor symbols to download data for.
env	environment where to create the downloaded data object.
from	a character string indicating the start date of the data to download, in YYYY-MM-DD format.
to	a character string indicating the end date of the data to download, in YYYY-MM-DD format.
adjust	a logical indicating whether to adjust the data for stock splits or not.
api.key	an Alor API key.
period	a character string indicating the frequency of the data to download. Possible values are '1min', '5min', 'hour', 'day', 'week', and 'month'.
verbose	a logical indicating whether to print the response details or not.
board	a character string indicating the Alor exchange board to use. Possible values are 'MOEX' and 'SPB'.
auto.assign	a logical indicating whether to automatically assign the downloaded data to the global environment.
...	additional arguments passed to getSymbols.Alor

Value

returns an data.table object containing financial data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Alor('SBER',from = '2023-04-01',to='2023-05-04',period = '1min')
getSymbols('SBER',src='Alor')
```

---

getSymbols.Comon   *Download data from Comon copytrading platform*

Description

Download historical market data from Comon for a given trader id

Usage

```
getSymbols.Comon(
  Symbols,
  env = globalenv(),
  period = "day",
  verbose = TRUE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	a character id of strategy on Comon
env	environment where to create the downloaded data object.
period	a character string indicating the frequency of the data to download. Possible values are 'day'.
verbose	a logical indicating whether to print the response details or not.
auto.assign	a logical indicating whether to automatically assign the downloaded data to the global environment.
...	additional arguments passed to getSymbols.Comon

Value

returns an data.table object containing financial data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Comon('115038')
getSymbols('115039',src='Comon')
```

---

getSymbols.Finam   *Download historical data from Finam.ru*

Description

Download historical data from Finam.ru for one or more stock symbols. The data can be returned as
an xts object or assigned to a specified environment. This function uses the Finam.ru export service
to retrieve data.

Usage

```
getSymbols.Finam(
  Symbols,
  env = globalenv(),
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  period = "day",
  market = NULL,
  verbose = FALSE,
  auto.assign = FALSE,
  api.key = "",
  user_agent = NULL,
  ...
)
```

Arguments

Symbols	A character vector of one or more stock symbols.
env	The environment where the data should be assigned. Defaults to the global environment.
from	The start date for the data. Defaults to "2007-01-01".
to	The end date for the data. Defaults to the current date.
adjust	A logical indicating whether to adjust for dividends and splits. Defaults to FALSE.
period	The interval to use for the data. Must be one of "tick", "1min", "5min", "10min", "15min", "30min", "hour", "day", "week", or "month". Defaults to "day".
market	A character vector indicating the market for each symbol. If NULL, the function will attempt to determine the market automatically. Defaults to NULL.
verbose	A logical indicating whether to print progress messages. Defaults to FALSE.

`auto.assign`	A logical indicating whether to assign the data to an object with the same name as the symbol. Defaults to FALSE.
`api.key`	character representing the authorization key required for accessing broker/exchange API
`user_agent`	Header for user agent for Finam
`...`	additional arguments passed to getSymbols.Finam

Value

returns an data.table object containing the requested data with orders of current account.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
api_key = 'set_if_use_API'
getSymbols('SBER',src='Finam',from='2023-01-01',api.key = api_key)
```

---

getSymbols.Gigapack *Download Giga Candles data from GigaPack*

Description

Download historical market data from GigaPack for a given symbol and time range.

Usage

```
getSymbols.Gigapack(
  Symbols,
  env = globalenv(),
  from = Sys.Date() - 30,
  to = Sys.Date(),
  date = "",
  verbose = TRUE,
  field = "",
  type = "candles",
  fake = FALSE,
  reps = 1,
  trim = 0.1,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	a character vector of AlgoPack symbols to download data for.
env	environment where to create the downloaded data object.
from	a character string indicating the start date of the data to download, in YYYY-MM-DD format.
to	a character string indicating the end date of the data to download, in YYYY-MM-DD format.
date	a character string indicating the specific date of the data to download, in YYYY-MM-DD format.
verbose	a logical indicating whether to print the response details or not.
field	a character string indicating the GigaPack field
type	a character string indicating the GigaPack field candles or tech
fake	a bool string indicating the Giga Candles or Alter Giga
reps	number of alternative history geneartion
trim	number of entropy in data
auto.assign	a logical indicating whether to automatically assign the downloaded data to the global environment.
...	additional arguments passed to getSymbols.AlgoPack

Value

returns an data.table object containing financial data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Gigapack('SBER', field = 'disb.q20')
```

```
getSymbols.MarketWatch
```
MarketWatch data

Description

This function retrieves historical financial data for a given symbol from the MarketWatch website and returns it as a data frame or assigns it to an R object.

Usage

```
getSymbols.MarketWatch(
  Symbols,
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  period = "day",
  market = NULL,
  countrycode = NULL,
  verbose = FALSE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	A character vector specifying the name of the financial instrument(s) to retrieve.
from	A character string specifying the starting date for the historical data in the format "YYYY-MM-DD".
to	A character string specifying the ending date for the historical data in the format "YYYY-MM-DD".
adjust	A logical value indicating whether to adjust the prices for splits and dividends. The default value is FALSE.
period	A character string specifying the period for which to retrieve the data. The default value is "day".
market	A character string specifying the market where the financial instrument is listed.
countrycode	A character string specifying the country code of the market where the financial instrument is listed.
verbose	A logical value indicating whether to print additional information while running the function. The default value is FALSE.
auto.assign	A logical value indicating whether to assign the data to an R object with the same name as the symbol. The default value is FALSE.
...	Additional arguments.

Value

A data frame or an object of class "xts" containing the historical financial data for the given symbol.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.MarketWatch(Symbols = 'liborusd3m',market = 'interestrate',countrycode = 'mx')
#getSymbols.MarketWatch(Symbols = 'tmubmusd03m',market = 'bond',countrycode = 'bx')
```

---

getSymbols.Mfd   *Get financial data from Mfd.ru*

Description

This function retrieves financial data from Mfd.ru

Usage

```
getSymbols.Mfd(
  Symbols,
  env = globalenv(),
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  period = "day",
  verbose = TRUE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	character vector of symbols to retrieve
env	environment where the data will be assigned
from	a character string representing the starting date in 'YYYY-MM-DD' format. Default is '2007-01-01'.
to	a character string representing the ending date in 'YYYY-MM-DD' format. Default is the current system date.

adjust	a logical specifying whether to adjust for dividends and stock splits. Default is FALSE.
period	a character string specifying the period of the data to retrieve. Must be one of 'tick', '1min', '5min', '10min', '15min', '30min', 'hour', 'day', 'week', or 'month'. Default is 'day'.
verbose	a logical specifying whether to print progress messages. Default is TRUE.
auto.assign	a logical specifying whether to automatically assign the retrieved data to the symbols specified in the Symbols argument. Default is FALSE.
...	additional arguments passed to getSymbols.Mfd

Value

If auto.assign is TRUE, the function returns the Symbols argument with the retrieved data assigned to each symbol. If auto.assign is FALSE, the function returns an xts object.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

---

getSymbols.Moex   *Get financial data from MOEX Exchange*

Description

Retrieves historical data of a stock from Moscow Exchange (MOEX) using its API.

Usage

```
getSymbols.Moex(
  Symbols,
  env = globalenv(),
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  period = "day",
  market = NULL,
  verbose = FALSE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	character vector of ticker symbols to retrieve data for
env	environment where data is stored
from	character string specifying the start date (default: '2007-01-01')
to	character string specifying the end date (default: Sys.Date())
adjust	logical flag indicating whether to adjust prices for dividends and splits (default: FALSE)
period	character string specifying the interval of the data ('day', 'hour', '10min', or '1min'; default: 'day')
market	character string specifying the market of the data (default: NULL)
verbose	logical flag indicating whether to print progress messages (default: FALSE)
auto.assign	logical flag indicating whether to automatically assign the resulting data to objects with the same names as Symbols (default: FALSE)
...	Additional arguments.

Value

xts object with historical data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
data <- getSymbols.Moex('GAZP', from='2022-01-01', to='2022-01-10', period='day', auto.assign=FALSE)
```

---

getSymbols.Poloniex   *Get financial data from Poloniex*

Description

This function retrieves financial data from Poloniex.

Usage

```
getSymbols.Poloniex(
  Symbols,
  env,
  return.class = "xts",
  index.class = "Date",
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  period = "day",
  verbose = TRUE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	A character vector with the ticker symbols to retrieve data for.
env	The environment where to create the variables. Default is the current environment.
return.class	The class to return. Default is xts.
index.class	The class for the index column. Default is Date.
from	Start date of the data. Default is '2007-01-01'.
to	End date of the data. Default is Sys.Date().
adjust	Logical. Adjust the prices. Default is FALSE.
period	The period for the candle data. Default is 'day'.
verbose	Logical. Print progress messages. Default is TRUE.
auto.assign	Logical. Assign data to the environment. Default is FALSE.
...	Additional arguments to be passed to functions.

Value

A list of xts objects if `auto.assign` is TRUE, otherwise a single xts object.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Poloniex('BTC_USDT')
#getSymbols('BTC_USDT',src='Poloniex')
```

---

getSymbols.Rusquant   *Download alpha strategy data from Rusquant*

Description

This function retrieves alpha data for the specified alpha,symbols from Rusquant API. The data can
be returned in either xts or data.frame format.

Usage

```
getSymbols.Rusquant(
  Symbols,
  env = globalenv(),
  field = NULL,
  from = "2007-01-01",
  to = Sys.Date(),
  period = "day",
  market = NULL,
  api.key = NULL,
  verbose = FALSE,
  auto.assign = FALSE,
  ...
)
```

Arguments

Symbols	a character vector of symbols to be downloaded.
env	the environment where the data should be stored. Default is the global environment.
field	the name(s) of alpha strategy to retrieve from the API. Default is NULL.
from	the start date of the data to be retrieved. Default is '2007-01-01'.
to	the end date of the data to be retrieved. Default is Sys.Date().
period	a character value indicating the periodicity of the data. Default is 'day'.
market	a character value indicating the market where the symbols are traded. Default is NULL.
api.key	a character value indicating the API key to be used for the request. Default is NULL.
verbose	a logical value indicating whether to print informative messages. Default is FALSE.

| auto.assign | a logical value indicating whether to automatically assign the downloaded data to an object with the symbol name. Default is FALSE. |
| ... | Additional arguments to be passed to functions. |

Value

returns an data.table object containing the requested data with alpha strategy data

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
rusquant_key = 'get free key from rusquant.ru'
getSymbols.Rusquant('SBER',field = 'A1_L_P1',api.key = rusquant_key)
getSymbols('SBER',src='Rusquant',field = 'A1_L_P1',api.key = rusquant_key)
```

---

getSymbols.Tinkoff   *Get earnings data from investing.com*

Description

Retrieve financial data from Tinkoff API.

Usage

```
getSymbols.Tinkoff(
  Symbols,
  from = "2007-01-01",
  to = Sys.Date(),
  adjust = FALSE,
  api.key = NULL,
  period = "day",
  market = NULL,
  verbose = FALSE,
  auto.assign = FALSE,
  ...
)
```

Arguments

`Symbols`	A character vector with the names of the symbols to be retrieved.
`from`	A character string representing the start date for the financial data, in the format "YYYY-MM-DD". By default, it is "2007-01-01".
`to`	A character string representing the end date for the financial data, in the format "YYYY-MM-DD". By default, it is the current system date.
`adjust`	A logical value indicating whether to adjust prices for dividends and stock splits. By default, it is FALSE.
`api.key`	A character string containing the API key for accessing Tinkoff API.
`period`	A character string representing the interval of time between two candles. By default it is "day".
`market`	A character string representing the market to which the symbol belongs. By default, it is NULL.
`verbose`	A logical value indicating whether to print verbose output. By default, it is FALSE.
`auto.assign`	A logical value indicating whether to automatically assign the downloaded data to an object with the same name as the symbol. By default, it is FALSE.
`...`	Additional arguments.

Value

A data table with the financial data for the specified symbol(s).

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getSymbols.Tinkoff("BBG004730N88",from=Sys.Date()-5, ap.key = "your_api_key",verbose=TRUE)
```

---

getTradelog   *Get tradelog information from a brokerage or exchange*

Description

This function retrieves account information from a brokerage or exchange.

Usage

```
getTradelog(
  Symbols,
  depth = 500,
  src = "poloniex",
  api.key = "",
  adjust = FALSE,
  return.class = "data.table",
  index.class = "Date",
  market = "shares",
  board = "tqbr",
  verbose = FALSE,
  auto.assign = TRUE,
  env = globalenv()
)
```

Arguments

Symbols	Character vector specifying the trading pair, e.g. "BTC_ETH".
depth	Numeric scalar, specifying the number of trades to retrieve (default = 500).
src	Character scalar, specifying the exchange to retrieve trade logs from. The supported exchanges are: "tinkoff", "alor", "poloniex", "kraken", "binance", "bttrex", "cex", "gate", "gatecoin", "gdax", "gemini", "hitbtc", "liqui", and "lykke".
api.key	Character scalar, specifying the API key (if required by the exchange).
adjust	Logical scalar, specifying whether to adjust timestamps for time zones (default = FALSE).
return.class	Character scalar, specifying the class of the returned object. The supported classes are: "data.table", "data.frame", and "xts" (default = "data.table").
index.class	Character scalar, specifying the class of the index column. The supported classes are: "Date" and "POSIXct" (default = "Date").
market	A character string specifying type of market. 'shares' or 'forts'
board	A character string specifying type of board
verbose	Logical scalar, specifying whether to print verbose output (default = FALSE).
auto.assign	Logical scalar, specifying whether to automatically assign the resulting object to the global environment (default = TRUE).
env	environment where data is stored

Value

A data table or data frame with the retrieved trade logs, depending on the value of the `return.class` argument.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getTradelog('BTC_USDT', src = 'poloniex')
getTradelog('SBER', src = 'moex')
getTradelog('SiH5', src = 'moex',market='forts')
```

---

getTrades   *Download trades of account*

Description

Get trades for a given broker from a specified date to the current date.

Usage

```
getTrades(
  src = "",
  api.key = "",
  clientId = "",
  figi = "",
  from = Sys.Date() - 5,
  to = Sys.Date(),
  symbol_info = FALSE,
  time_transform = TRUE,
  verbose = FALSE
)
```

Arguments

src	Character string indicating the broker to use. Currently, only "tinkoff" and "alor" are supported.
api.key	Character string containing the API key for the broker's API.
clientId	Character string containing the client ID for the broker.

figi	Character string containing the Financial Instrument Global Identifier for the broker.
from	Date specifying the start date for the trade data. Defaults to 5 days ago.
to	Date specifying the end date for the trade data. Defaults to the current date.
symbol_info	Logical indicating whether to include symbol information in the returned data. Defaults to FALSE.
time_transform	Logical indicating whether to transform the timestamps to the local timezone. Defaults to TRUE.
verbose	Logical indicating whether to print verbose output. Defaults to FALSE.

Value

A data frame containing the trade data.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Author(s)

Vyacheslav Arbuzov

Examples

```
getTrades(src = "tinkoff", api.key = "tks token", clientId = "clientID", figi = "figi",verbose=TRUE)
```

---

placeOrder	*Place an order on a broker/exchange platform*

Description

This function place an existing order on a specified broker or exchange platform

Usage

```
placeOrder(
  src = "tinkoff",
  symbol = "SBER",
  board = "MOEX",
  action = "BUY",
  orderType = "LMT",
  totalQuantity = "10",
  lmtPrice = "100",
  auxPrice = "",
  api.key = "",
```

```
    live = TRUE,
    tif = "",
    orderId = "",
    clientId = "",
    verbose = TRUE
)
```

Arguments

src	A character string indicating the broker/exchange. Currently supported sources are 'tinkoff', 'finam', and 'alor'.
symbol	A character string indicating the security symbol.
board	A character string indicating the exchange board.
action	A character string indicating the action to perform. Possible values are 'BUY' and 'SELL'.
orderType	A character string indicating the type of order. Possible values are 'LMT' (limit) and 'MKT' (market).
totalQuantity	A character string or numeric value indicating the quantity of securities to order.
lmtPrice	A character string or numeric value indicating the limit price for the order.
auxPrice	A character string or numeric value indicating the auxiliary price for the order. This parameter is only used when orderType is 'STP' (stop).
api.key	A character string indicating the API key to use for authentication with the broker/exchange.
live	A logical value indicating whether to place a live (real) order or a test (paper) order.
tif	A character string indicating the time-in-force of the order. Possible values are 'DAY', 'GTC' (good till cancel), and 'IOC' (immediate or cancel).
orderId	A character string indicating the order ID to use. If empty, a random ID is generated.
clientId	A character string indicating the client ID to use.
verbose	A logical value indicating whether to print verbose output.

Value

A list with the result of the order placement.

Note

Not for the faint of heart. All profits and losses related are yours and yours alone. If you don't like it, write it yourself.

Examples

```
myorder = placeOrder(src = 'alor',
                     symbol = 'MTLR-6.23',
                     board = 'MOEX',
                     action = 'BUY',
                     orderType = 'LMT',
                     totalQuantity = 1,
                     lmtPrice = 20000,
                     api.key = '',
                     clientId = 'cliendID')
```

Index