Rusquant: another way trading with R

Vyacheslav Arbuzov [1,2,3]

[1] *Department of Economics, Perm State University, Perm, Russia*

[2] *Cryptoeconomics & Blockchain Systems Lab, Perm State University, Perm, Russia*

[3] *Olympia Capital, Moscow, Russia*

Abstract

We present our package which first time was published in 2011 as an extension of quantmod package for Russian market. Since that time, we have added different data sources and markets. With current popularity of cryptomarket and online exchanges with webAPI, we have created common functions for trading in crypto exchanges and connections for trading terminals.

Most of the R packages allow to download and manipulate data for creation quantitative models and visualization market dependencies and stylized facts, but implementation of trading strategy goes via special platforms like Marketcetera, Deltix and etc. There are small amount of packages which allow to execute signals from quantitative models, i.e. IBrokers (2014), Rbitcoin (2014), etc. At such packages there are standardized commands which can be used in different markets. We try to delete this gap and present package which can execute signals in different markets and platforms.

Trading universe

Universe of trading instruments can be divided into 3 main markets:

- Cryptomarket. For trading available next exchanges: poloniex, kraken, binance, bttrex, cex ,gemini and etc. More than 10 available exchanges.
- Forex market. We have created connection to platform Metatrader, which use many forex-traders.
- Equity market. We use 2 platforms: TWS of Interactive Brokers (using some functions from IBrokers) and Quik (80% of all trading in Moscow Exchange execute throw this platform).

In each financial market there are many trading instruments. List of actual instruments can be loaded by using command below:

loadSymbolList()

For each market list of characteristics can be different, but there are essential fields: Symbol, Name and Market.

Historical price data

For a more convenient work with historical data, we use familiar ***getSymbols*** function which allows you to download not only historical data but also live data for trading. In order to be able to work with different timeframes, a ***period*** argument was added.

getSymbols(Symbols = NULL, period = "1min",con=connection)

Some data sources (in particular, connection to terminals) require a connection object to work with data streams. Using the function ***Init***, you can specify from which connection will be taken data.

High frequency data

There are two main types of high-frequency data. The first is connected with an order book, which shows liquidity in the market. The second is related to transactions that occur on the market. In most cases, this information is useful for analyzing the execution of trading signals. Although, if we consider the crypto-currency market, then such kind of information can be used for arbitrage between few markets.

Function ***getTradelog*** allows to load trades of chosen instrument from specific market (***src*** parameter) and with maximum length (***last*** parameter) for excluding lagging processes.

getTradelog(Symbols = NULL, from=Sys.Date()-1, to = Sys.Date(), last=200, src="Poloniex")

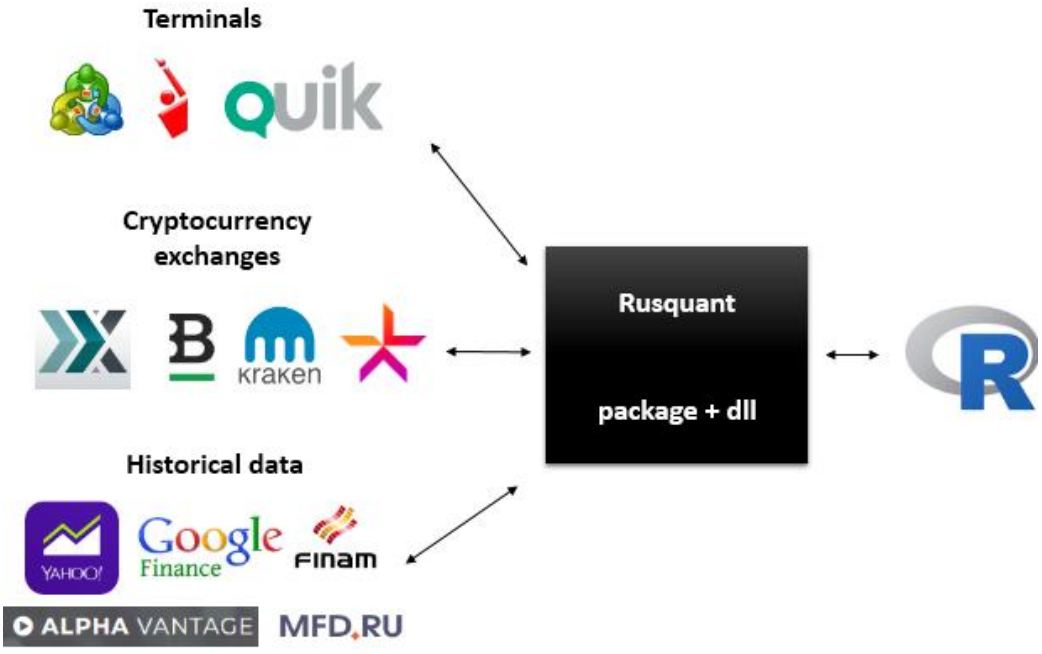
Function ***getOrderbook*** allows to load order book snapshot for current time of chosen instrument from specific market (***src*** parameter) and with maximum depth (depth parameter). When parameter depth=2, function gives you information about best bid and best ask prices in the market.

getOrderbook(Symbols = NULL, depth=2, src="Poloniex")

All data is returned as data table object.

Live trading and terminal connections

This package allows using R to interact with various exchanges and terminals, which provides any user a convenient work in different markets using one code for different API.



To initialize the connection for live trading, we use the ***Connect*** function which is primary when R sets with the trading server. For connection to the trading terminal, a special dll is used which creates a named pipe. The function uses several arguments which depend on the type of connection. All details are described in help of package.

Connect(host = 'localhost', port = '', Key='', Sign='', timeout = 5)

The main objects for live trading are the orders and methods for their sending and cancellation. For sending an order, the object should be created using command ***Order***, which specifies the type of order, direction, volume and price.

Order(Symbol = "", action = "BUY", Quantity = 10, orderType = "LMT", Price = 0)

The function **openOrder** denotes the result of order execution. In case of rejection of order, a corresponding error will be issued, which most often coincides with the code of exchange/terminal where this order was sent. The function of closing the order also returns the result of the order execution.

openOrder(conn, Orderid)

cancelOrder(conn, Orderid)

To control the execution of orders, it is possible, with the use of the ***getTrade*** function, to access the committed trades of a trader for a given instrument and for a given period

References

Jeffrey A. Ryan and Joshua M. Ulrich (2017). quantmod: Quantitative Financial Modelling Framework. R package version 0.4-11. https://CRAN.R-project.org/package=quantmod

Jeffrey A. Ryan (2014). IBrokers: R API to Interactive Brokers Trader Workstation. R package version 0.9-12. https://CRAN.R-project.org/package=IBrokers

Jan Gorecki (2014). Rbitcoin: R & bitcoin integration. R package version 0.9.2. https://CRAN.R-project.org/package=Rbitcoin