**Rusquant: another way trading with R**

Vyacheslav Arbuzov ¹˒²˒³

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

*² Criptoeconomics & Blockchain Systems Lab, Perm State University, Perm, Russia*

*³ Olympia Capital, Moscow, Russia*

**Abstract**

We present our package which first time was publish in 2011 as extension of quantmod package for Russian market. Since this time, we added different data source and markets. With current popularity of cryptomarket and online exchanges with webAPI, we create common functions for trading in cryptoexchanges and conections for trading terminals.

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

**Trading universe**

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

* Cryptomarket. For trading available next exchanges: poloniex, bitstamp, kraken, WEX, Lykke and etc. More than 15 available exchanges.
* Forex market. We create 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 executed throw this platform).

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

***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 used the familiar ***getSymbols*** function. This function 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***](http://127.0.0.1:23635/help/library/rusquant/html/getSymbols.html)***(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 the data will be taken.

**High frequency data**

There are 2 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***](http://127.0.0.1:23635/help/library/rusquant/html/getSymbols.html)***(Symbols = NULL, from=Sys.Date()-1, to = Sys.Date(), last=200, src="Poloniex")***

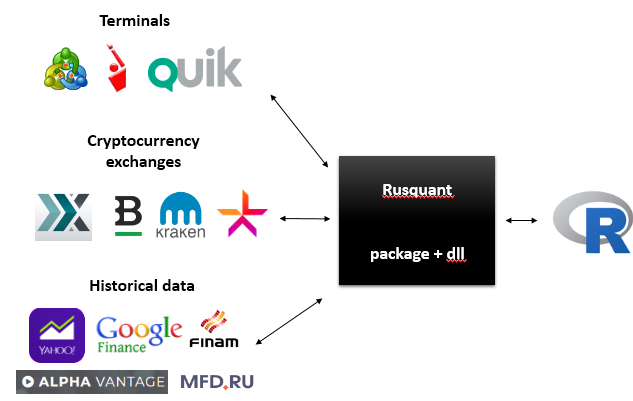
Function ***getOrderBook*** allows to load order book snapshot for current time of chosen instrument from specific market (***src*** parameter) and with maximum depth (depth parameter). When parameter depth=2, function give to you information about best bid and best ask prices in the market.

[***getTradelog***](http://127.0.0.1:23635/help/library/rusquant/html/getSymbols.html)***(Symbols = NULL, depth=2, src="Poloniex")***

All data is returned as data table object.

**Live trading and terminal connections**

The package allows using P to interact with various exchanges and terminals, which is quite convenient for the user to work in different markets using one code for different API.



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

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

The main object for live trading are the orders and methods for their sending and cancellation. To send 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 functions of the **openOrder** denote the result of the execution of the application. In case of rejected of the order, a corresponding error will be issued, which most often coincides with the code of the exchange/terminal in which the 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 the 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