NYU TANDON SCHOOL OF ENGINEERING

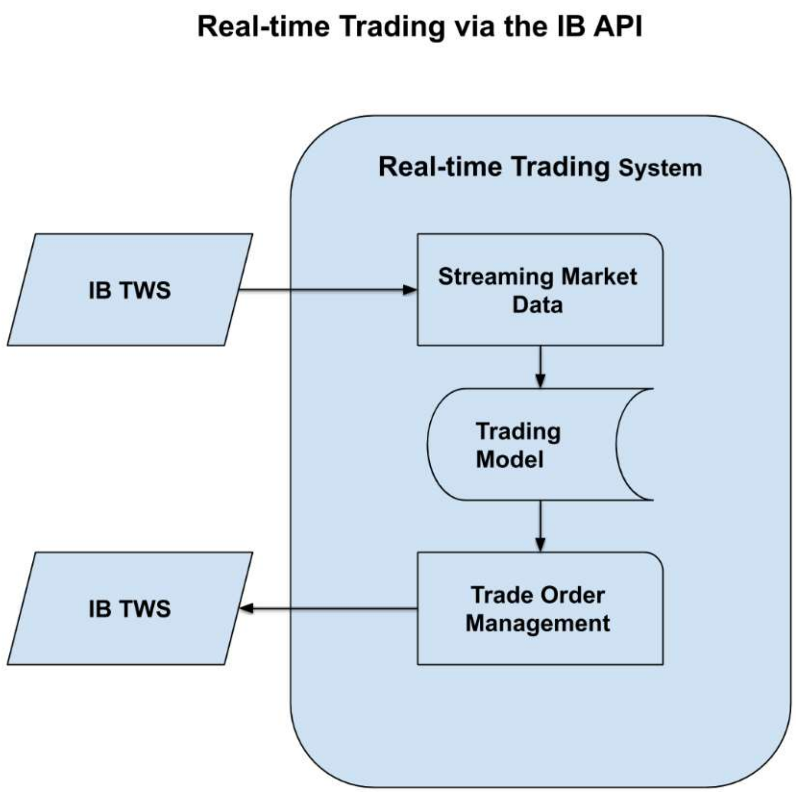
Package iBrokers2 for Real-time Trading With Interactive Brokers

Jerzy Pawlowski, NYU Tandon School of Engineering

May 17, 2019

Executing Real-time Trading Strategies

- Real-time trading requires running a programmatic loop.
- Continuous streaming market data is used to update a trading model.
- The model outputs are used by an order management system to place trade orders via an API.
- The package *IBrokers* contains *R* functions for downloading live market data via the API of Interactive Brokers (IB API), and for placing trade orders to Interactive Brokers.
- The function `iBrokers::reqRealTimeBars()` downloads live (real-time) *OHLC* bars of market data from Interactive Brokers.
- The function `IBrokers::twsOrder()` places trade orders to Interactive Brokers.



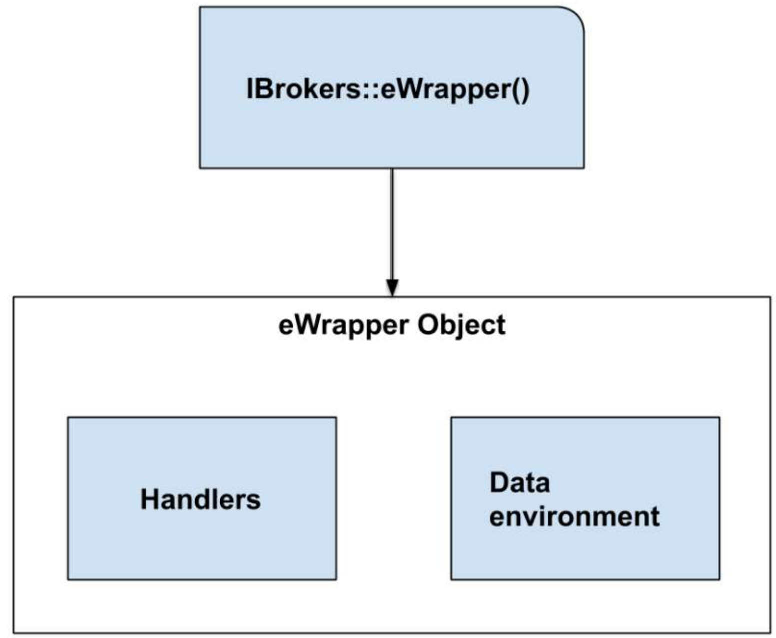
The Package *IBrokers* for Interactive Brokers

- The package *IBrokers* allows downloading live market data via the API of Interactive Brokers (IB API)
- The function `twsConnect()` opens a connection to the *IB API* via the IB Trader Workstation (TWS).
- The function `reqRealTimeBars()` downloads live (real-time) *OHLC* bars of market data.
- `reqRealTimeBars()` relies on the functions `eWrapper.RealTimeBars.CSV()` and `twsCALLBACK` to process real-time market events (trades and quotes).
- The function `eWrapper.RealTimeBars.CSV()` creates an *eWrapper* object designed for processing *OHLC* price data.

```r
# Install and load package IBrokers
install.packages("IBrokers")
library(IBrokers)
# Connect to Interactive Brokers TWS via API
ib_connect <- IBrokers::twsConnect(port=7497)
# Define S&P Emini future June 2019 contract
con_tract <- IBrokers::twsFuture(symbol="ES",
  exch="GLOBEX", expiry="201906")
# Get list with instrument information
IBrokers::reqContractDetails(conn=ib_connect,
                             Contract=con_tract)
# Open file connection for data download
file_name <- "C:/Develop/data/ib_data/ES_ohlc_live.csv"
file_connect <- file(file_name, open="w")
# Download live data to file
IBrokers::reqRealTimeBars(conn=ib_connect,
           Contract=con_tract, barSize="1",
           eventWrapper=eWrapper.RealTimeBars.CSV(1),
           file=file_connect)
# Close the Interactive Brokers API connection
IBrokers::twsDisconnect(ib_connect)
# Close data file
close(file_connect)
```

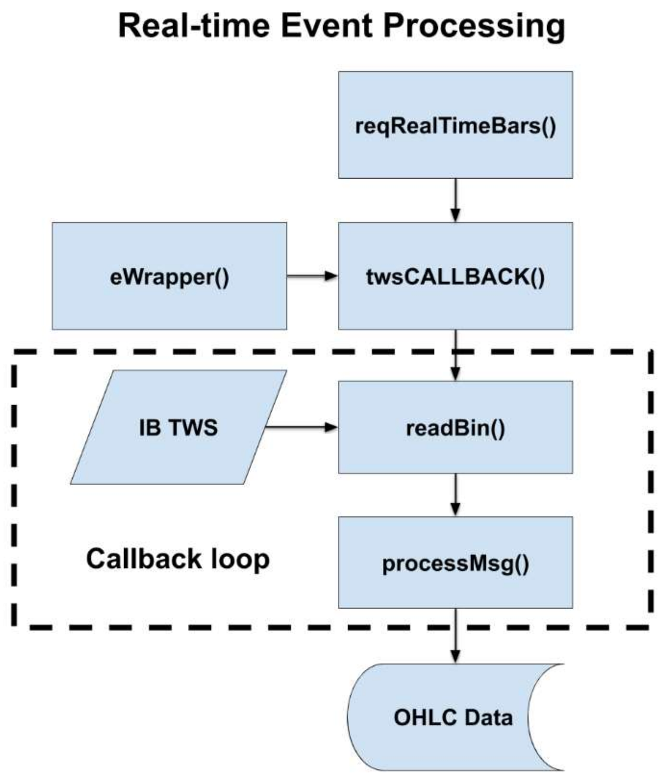
The *eWrapper* Object

- An *eWrapper* object consists of a *data environment* and *handlers* (methods) for formatting and adding new data to the *data environment*.
- The function `eWrapper()` creates a generic *eWrapper* object.
- The function `eWrapper.RealTimeBars.CSV()` creates an *eWrapper* object designed for processing *OHLC* price data.
- The functionality of package *IBrokers* can easily be extended by writing new *eWrapper* objects, designed for processing different types of data and performing different tasks.



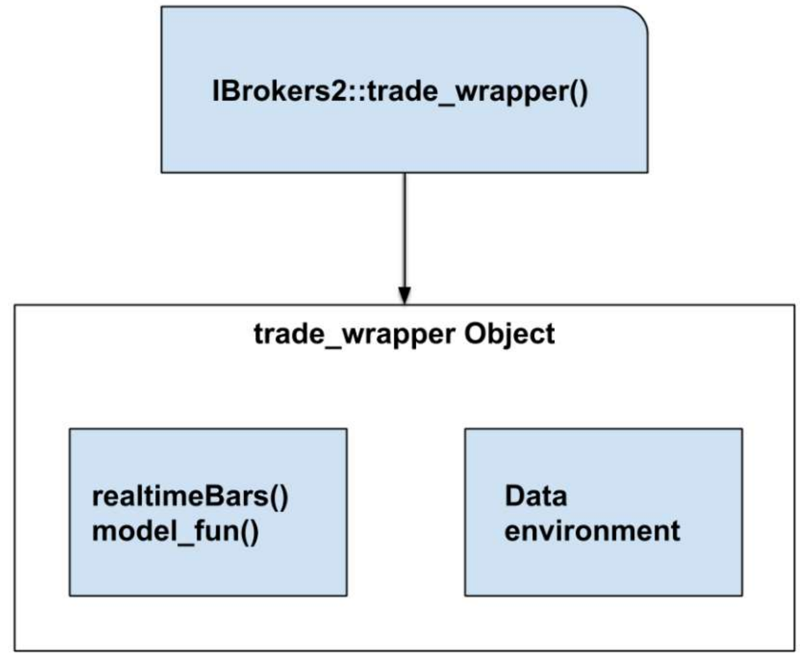
Market Event Processing

- Market events can be either *trade events* or *market quotes*.
- Streaming market events are processed in a *callback loop* which runs inside the function `twsCALLBACK()`.
- The function `twsCALLBACK()` first creates an *eWrapper* object by calling the function `eWrapper.RealTimeBars.CSV()`, and then passes it to the function `processMsg()`.
- The function `twsCALLBACK()` then calls `processMsg()` in a *callback loop*.
- The function `processMsg()` processes individual market events by calling the appropriate *eWrapper handlers* and saving the data into the *eWrapper* environment.



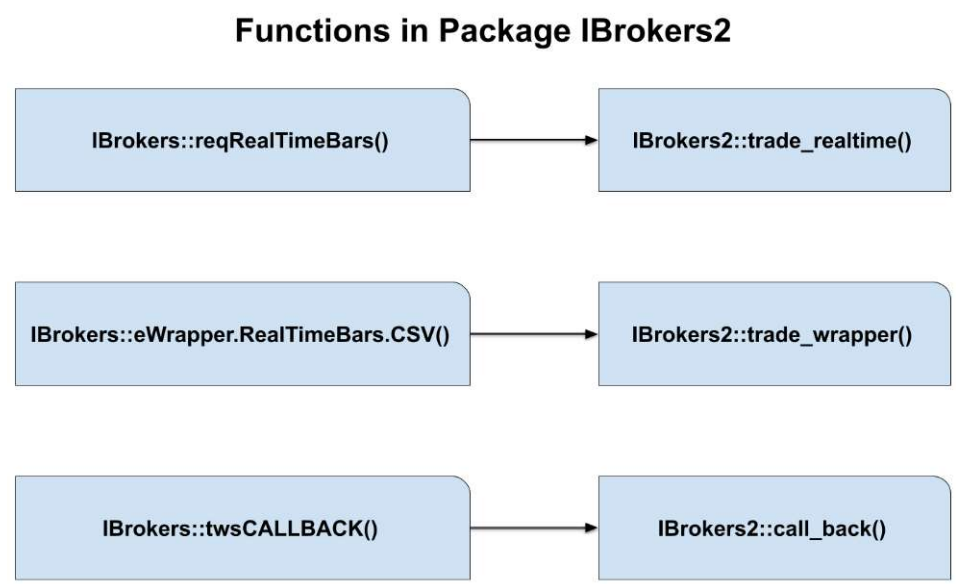
The *trade_wrapper* Object

- The functionality of package *IBrokers* can easily be extended to trading by writing a new *eWrapper* object.
- The function `IBrokers2::trade_wrapper()` creates a *trade_wrapper* object (a modified *eWrapper*) designed for real-time trading.
- The *trade_wrapper* data environment contains buffers for *OHLC* market data, trading model parameters, instrument positions, open trade orders, etc.
- The *trade_wrapper* contains the data handler `realtimeBars()` and the trading model `model_fun()`.
- The function `IBrokers2::trade_wrapper()` can be modified to support different market instruments and trading models.



Functions in Package *IBrokers2*

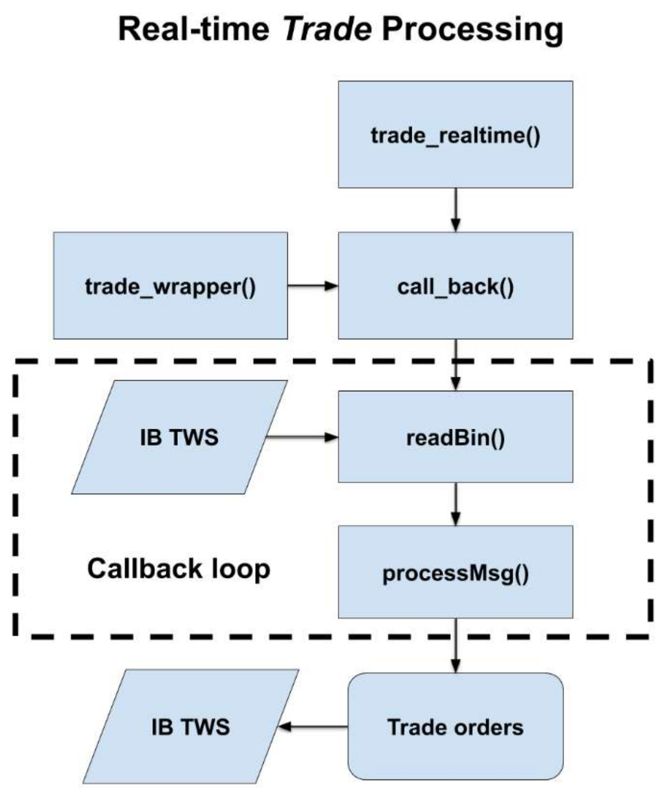
- Most of the functions in package *IBrokers2* were derived from those in *IBrokers*.
- `IBrokers2::trade_realtime()` initiates real-time trading.
- `IBrokers2::trade_wrapper()` creates a *trade_wrapper* object, containing the data handler `realtimeBars()` and the trading model `model_fun()`.
- The function `realtimeBars()` is a *trade_wrapper* handler which updates the data environment with new data and then runs the trading model `model_fun()`.
- `model_fun()` reruns the trading model using updated market data, and places trade orders using `IBrokers::twsOrder()`.
- The function `IBrokers2::call_back()` performs the *callback loop*.



Trade Processing Using Package *IBrokers2*

- Trade processing using package *IBrokers2* uses a similar framework to that for downloading real-time *OHLC* market data using `IBrokers::reqRealTimeBars()`.

- The main difference is that now trade orders are placed to the IB Trader Workstation (TWS).



Real-time Trading Using Package *IBrokers2*

- The function `trade_realtime()` accepts the `trade_wrapper()` and `call_back()` functions, and initiates real-time trading.
- The *OHLC* market data arrives from Interactive Brokers in 5-second intervals.
- Every time new data arrives, `model_fun()` reruns the trading model and places trade orders using `IBrokers::twsOrder()`.
- The instrument parameters specify the number of contracts, trade order types, etc.
- The trading instruments and their parameters are specified as lists to allow trading multiple instruments simultaneously.

```
# Install and load package IBrokers2
devtools::install_github(repo="algoquant/IBrokers2")
library(IBrokers2)
# Define named lists for trading one contract
con_tracts <- list(ES=IBrokers2::twsFuture(symbol="ES", exch="GL
trade_params <- list(ES=c(buy_spread=0.75, sell_spread=0.75, siz
# Open file connection for data download
file_names <- "C:/Develop/data/ib_data/ES_ohlc_live.csv"
file_connects <- lapply(file_names, function(file_name) file(fil
# Open the IB connection to TWS
ac_count <- "algoquant"
ib_connect <- IBrokers2::twsConnect(port=7497)
# Run the trading model (strategy):
IBrokers2::trade_realtime(ib_connect=ib_connect,
  Contract=con_tracts,
  eventWrapper=IBrokers2::trade_wrapper(ac_count=ac_count,
    con_tracts=con_tracts,
    trade_params=trade_params,
    file_connects=file_connects,
    warm_up=10),
  CALLBACK=IBrokers2::call_back,
  file=file_connects)
# Stop the trading loop by hitting the red STOP button in RStudi
# Close the Interactive Brokers API connection
IBrokers::twsDisconnect(ib_connect)
# Close data files
for (file_connect in file_connects) close(file_connect)
```

Future Development of Package *IBrokers2*

Current state

- The package *IBrokers2* is currently an initial proof of concept, rather than a working application.
- The package *IBrokers2* is derived from the package *IBrokers*, and is fully backward compatible with it.
- All the *IBrokers* functions and variables are preserved exactly in *IBrokers2*, while some additional functions have been added to provide functionality for real-time trading.

Future development

- Rewrite critical functions using *Rcpp* and the C++ API of Interactive Brokers.

Applications

- Education and trading competitions: Interactive Brokers has been kind to provide student *paper trading* accounts, together with market data.
- Crowd-sourced hedge fund: the package *IBrokers2* could become the foundation for a system similar to *Quantopian*.

Thank You

- Many thanks to Jeff Ryan for developing the package *IBrokers*.
- The package *IBrokers2* is available on GitHub: https://github.com/algoquant/iBrokers2

Contact information

NYU email: jp3900@nyu.edu

LinkedIn profile: https://www.linkedin.com/in/jerzypawlowski