COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

Introduction to Trading Systems

Guy Yollin

Applied Mathematics
University of Washington

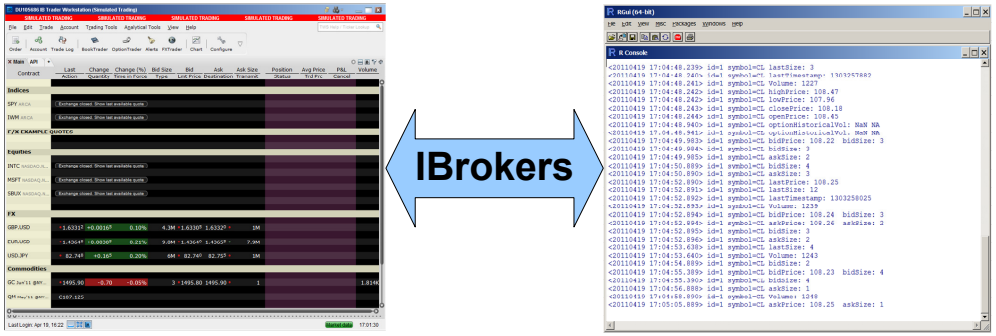
Outline

Lecture references

- IBrokers package vignette
  - http://cran.fhcrc.org/web/packages/IBrokers/vignettes/IBrokers.pdf
- IBrokers package reference manual
  - http://cran.fhcrc.org/web/packages/IBrokers/IBrokers.pdf
- Jeff Ryan's 2009 Meielisalp presentation
  - https://www.rmetrics.org/files/Meielisalp2009/Presentations/Ryan.pdf
- twsInstrument package reference
  - https://r-forge.r-project.org/R/?group_id=1113

Outline

The IBrokers package



IB TWS **R session**

Description

- Provides native R access to Interactive Brokers API

Key features

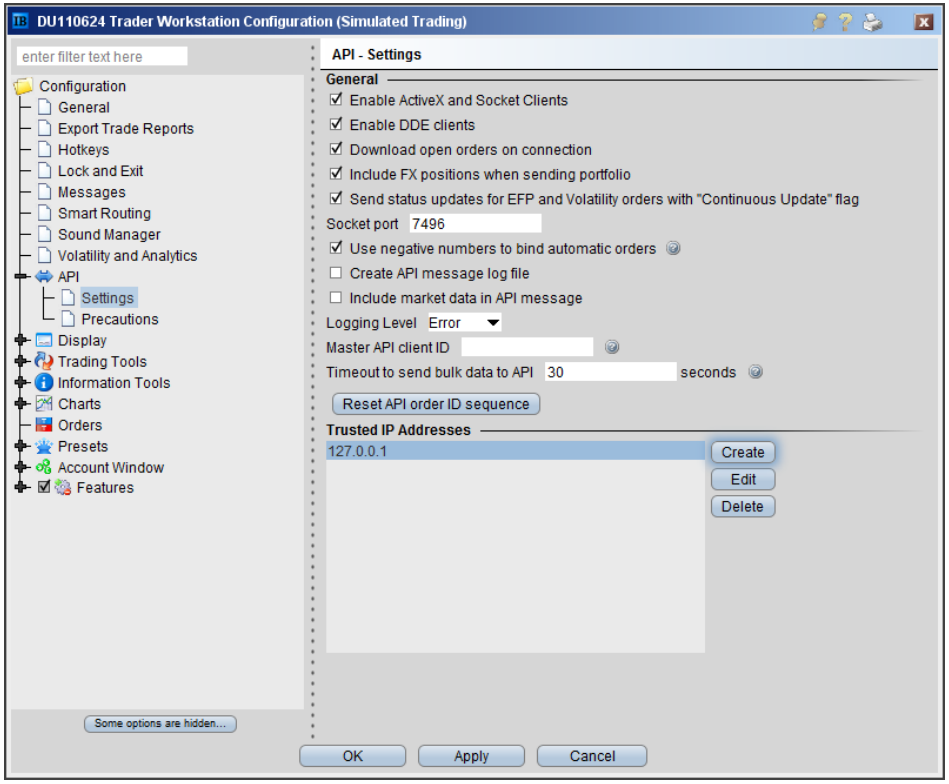
- Real-time market data feed
- Historic intra-day data
- Order placement
- Software comes with NO WARRANTY

Authors

- Jeffrey Ryan (also author of quantmod, and xts)

Enabling API access to TWS

API access must be enabled through TWS configuration:



- Enable ActiveX and Socket Clients - IBrokers
- Enable DDE Clients - Excel interface
- Trusted IP: 127.0.0.1 - no accept dialog

Testing the TWS connection

```
library(IBrokers)

##
## IBrokers comes with NO WARRANTY.  Not intended for production use!

tws <- twsConnect()
tws

## <twsConnection,1 @ 20140814 12:38:01 PST, nextId=1>

reqCurrentTime(tws)

## TWS Message: 2 -1 2104 Market data farm connection is OK:usfarm.us
## TWS Message: 2 -1 2104 Market data farm connection is OK:usfarm
## TWS Message: 2 -1 2106 HMDS data farm connection is OK:ushmds
## [1] "2014-08-14 12:38:01 PDT"

serverVersion(tws)

## [1] "71"

twsDisconnect(tws)
```

Connecting and disconnecting to TWS

twsConnect	connect to a running instance of Trader Workstation
twsDisconnect	disconnect from Trader Workstation

```
args(twsConnect)

## function (clientId = 1, host = "localhost", port = 7496, verbose = TRUE,
##     timeout = 5, filename = NULL, blocking = .Platform$OS.type ==
##         "windows")
## NULL

args(twsDisconnect)

## function (twsconn)
## NULL
```

- `twsConnect` returns a connection object which must be used for subsequent interactions with TWS

IB data retrieval

The following functions provide basic data retrieval:

reqMktData	retrieves real-time market data
reqMktDepth	retrieves real-time order book data
reqRealTimeBars	retrieves real-time OHLC data
reqHistoricalData	retrieves historical data
reqContractDetails	retrieves detailed product information

Contract object constructor functions

The IBrokers functions work with instrument objects and a number of constructor functions are supplied for common instruments:

twsEquity/twsSTK	create equity Contract objects
twsOption/twsOPT	create option Contract objects
twsFuture/twsFUT	create futures Contract objects
twsCurrency/twsCASH	create currency Contract objects

- The above functions are wrapper functions for the more generic function `twsContract`

Create a stock instrument

```
args(twsSTK)
```

```
## function (symbol, exch = "SMART", primary = "", strike = "0.0",
##     currency = "USD", right = "", local = "", multiplier = "",
##     include_expired = "0", conId = 0)
## NULL
```

```
sbux <- twsSTK("SBUX")
sbux
```

```
## List of 16
##  $ conId          : num 0
##  $ symbol         : chr "SBUX"
##  $ sectype        : chr "STK"
##  $ exch           : chr "SMART"
##  $ primary        : chr ""
##  $ expiry         : chr ""
##  $ strike         : chr "0.0"
##  $ currency       : chr "USD"
##  $ right          : chr ""
##  $ local          : chr ""
##  $ multiplier     : chr ""
##  $ combo_legs_desc: NULL
##  $ comboleg       : NULL
##  $ include_expired: chr "0"
##  $ secIdType      : chr ""
##  $ secId          : chr ""
```

Create a futures instrument

```
args(twsFUT)


## function (symbol, exch, expiry, primary = "", currency = "USD",
##     right = "", local = "", multiplier = "", include_expired = "0",
##     conId = 0)
## NULL


emini <- twsFUT("ES","GLOBEX","201409")
emini


## List of 16
##  $ conId          : num 0
##  $ symbol         : chr "ES"
##  $ sectype        : chr "FUT"
##  $ exch           : chr "GLOBEX"
##  $ primary        : chr ""
##  $ expiry         : chr "201409"
##  $ strike         : chr "0.0"
##  $ currency       : chr "USD"
##  $ right          : chr ""
##  $ local          : chr ""
##  $ multiplier     : chr ""
##  $ combo_legs_desc: NULL
##  $ comboleg       : NULL
##  $ include_expired: chr "0"
##  $ secIdType      : chr ""
##  $ secId          : chr ""
```

Create a forex instrument

```
args(twsCurrency)


## function (symbol, currency = "USD", exch = "IDEALPRO", primary = "",
##     strike = "0.0", right = "", local = "", multiplier = "",
##     include_expired = "0", conId = 0)
## NULL


gbpusd <- twsCurrency("GBP")
gbpusd


## List of 16
##  $ conId          : num 0
##  $ symbol         : chr "GBP"
##  $ sectype        : chr "CASH"
##  $ exch           : chr "IDEALPRO"
##  $ primary        : chr ""
##  $ expiry         : chr ""
##  $ strike         : chr "0.0"
##  $ currency       : chr "USD"
##  $ right          : chr ""
##  $ local          : chr ""
##  $ multiplier     : chr ""
##  $ combo_legs_desc: NULL
##  $ comboleg       : NULL
##  $ include_expired: chr "0"
##  $ secIdType      : chr ""
##  $ secId          : chr ""
```

The reqContractDetails function

The reqContractDetails function queries IB for the details of an IB tradeable product

```
args(reqContractDetails)

## function (conn, Contract, reqId = "1", verbose = FALSE, eventWrapper = eWrapper(
##     CALLBACK = twsCALLBACK, ...)
## NULL
```

Main arguments:

conn twd a twsConnection object

Contract a twsContract object

Return value:

a twsContractDetails object

Retrieve contract specifications

```
tws <- twsConnect()
reqContractDetails(tws, sbux)
```

```
## [[1]]
## List of 18
##  $ version       : chr "8"
##  $ contract      :List of 16
##   ..$ conId          : chr "274105"
##   ..$ symbol         : chr "SBUX"
##   ..$ sectype        : chr "STK"
##   ..$ exch           : chr "SMART"
##   ..$ primary        : chr "NASDAQ"
##   ..$ expiry         : chr ""
##   ..$ strike         : chr "0"
##   ..$ currency       : chr "USD"
##   ..$ right          : chr ""
##   ..$ local          : chr "SBUX"
##   ..$ multiplier     : chr ""
##   ..$ combo_legs_desc: chr ""
##   ..$ comboleg       : chr ""
##   ..$ include_expired: chr "0"
##   ..$ secIdType      : chr ""
##   ..$ secId          : chr ""
##   ..- attr(*, "class")= chr "twsContract"
##  $ marketName    : chr "NMS"
##  $ tradingClass  : chr "NMS"
##  $ conId         : chr "274105"
##  $ minTick       : chr "0.01"
##  $ orderTypes    : chr [1:59] "ACTIVETIM" "ADJUST" "ALERT" "ALGO" ...
##  $ validExchanges: chr [1:17] "SMART" "ISE" "CHX" "ARCA" ...
##  $ priceMagnifier: chr "1"
##  $ underConId    : chr "0"
##  $ longName      : chr "STARBUCKS CORP"
##  $ contractMonth : chr ""
##  $ industry      : chr "Consumer, Cyclical"
##  $ category      : chr "Retail"
##  $ subcategory   : chr "Retail-Restaurants"
##  $ timeZoneId    : chr "EST5EDT"
##  $ tradingHours  : chr "20140814:0400-2000;20140815:0400-2000"
##  $ liquidHours   : chr "20140814:0930-1600;20140815:0930-1600"
```

Retrieve contract specifications

```
reqContractDetails(tws, emini)
```

```
## [[1]]
## List of 18
##  $ version       : chr "8"
##  $ contract      :List of 16
##   ..$ conId          : chr "129513311"
##   ..$ symbol         : chr "ES"
##   ..$ sectype        : chr "FUT"
##   ..$ exch           : chr "GLOBEX"
##   ..$ primary        : chr ""
##   ..$ expiry         : chr "20140919"
##   ..$ strike         : chr "0"
##   ..$ currency       : chr "USD"
##   ..$ right          : chr ""
##   ..$ local          : chr "ESU4"
##   ..$ multiplier     : chr "50"
##   ..$ combo_legs_desc: chr ""
##   ..$ comboleg       : chr ""
##   ..$ include_expired: chr "0"
##   ..$ secIdType      : chr ""
##   ..$ secId          : chr ""
##   ..- attr(*, "class")= chr "twsContract"
##  $ marketName    : chr "ES"
##  $ tradingClass  : chr "ES"
##  $ conId         : chr "129513311"
##  $ minTick       : chr "0.25"
##  $ orderTypes    : chr [1:42] "ACTIVETIM" "ADJUST" "ALERT" "ALGO" ...
##  $ validExchanges: chr [1:2] "GLOBEX" "MIBSX"
##  $ priceMagnifier: chr "1"
##  $ underConId    : chr "11004968"
##  $ longName      : chr "E-mini S&P 500"
##  $ contractMonth : chr "201409"
##  $ industry      : chr ""
##  $ category      : chr ""
##  $ subcategory   : chr ""
##  $ timeZoneId    : chr "CST"
##  $ tradingHours  : chr "20140814:1700-1515,1530-1615;20140815:1700-1515,1530-1615"
##  $ liquidHours   : chr "20140814:0830-1515;20140815:0830-1515"
```

Retrieve contract specifications

```
reqContractDetails(tws, gbpusd)
```

```
## [[1]]
## List of 18
##  $ version       : chr "8"
##  $ contract      :List of 16
##   ..$ conId          : chr "12087797"
##   ..$ symbol         : chr "GBP"
##   ..$ sectype        : chr "CASH"
##   ..$ exch           : chr "IDEALPRO"
##   ..$ primary        : chr ""
##   ..$ expiry         : chr ""
##   ..$ strike         : chr "0"
##   ..$ currency       : chr "USD"
##   ..$ right          : chr ""
##   ..$ local          : chr "GBP.USD"
##   ..$ multiplier     : chr ""
##   ..$ combo_legs_desc: chr ""
##   ..$ comboleg       : chr ""
##   ..$ include_expired: chr "0"
##   ..$ secIdType      : chr ""
##   ..$ secId          : chr ""
##   ..- attr(*, "class")= chr "twsContract"
##  $ marketName    : chr "GBP.USD"
##  $ tradingClass  : chr "GBP.USD"
##  $ conId         : chr "12087797"
##  $ minTick       : chr "5.0E-5"
##  $ orderTypes    : chr [1:35] "ACTIVETIM" "ADJUST" "ALERT" "ALGO" ...
##  $ validExchanges: chr "IDEALPRO"
##  $ priceMagnifier: chr "1"
##  $ underConId    : chr "0"
##  $ longName      : chr "British pound"
##  $ contractMonth : chr ""
##  $ industry      : chr ""
##  $ category      : chr ""
##  $ subcategory   : chr ""
##  $ timeZoneId    : chr "EST5EDT"
##  $ tradingHours  : chr "20140814:1715-1700;20140815:1715-1700"
##  $ liquidHours   : chr "20140814:1715-1700;20140815:1715-1700"
```

```
twsDisconnect(tws)
```

Outline

The reqHistoricalData function

The reqHistoricalData function request historic market data from TWS

```
args(reqHistoricalData)

## function (conn, Contract, endDateTime, barSize = "1 day", duration = "1 M",
##     useRTH = "1", whatToShow = "TRADES", timeFormat = "1", tzone = "",
##     verbose = TRUE, tickerId = "1", eventHistoricalData, file)
## NULL
```

Main arguments:

conn	a twsConnection object
Contract	a twsContract
endDateTime	end date/time for request
barSize	bar size to retrieve
duration	time span the request will cover

Return value:

an xts object containing the requested data

Daily data retrieval

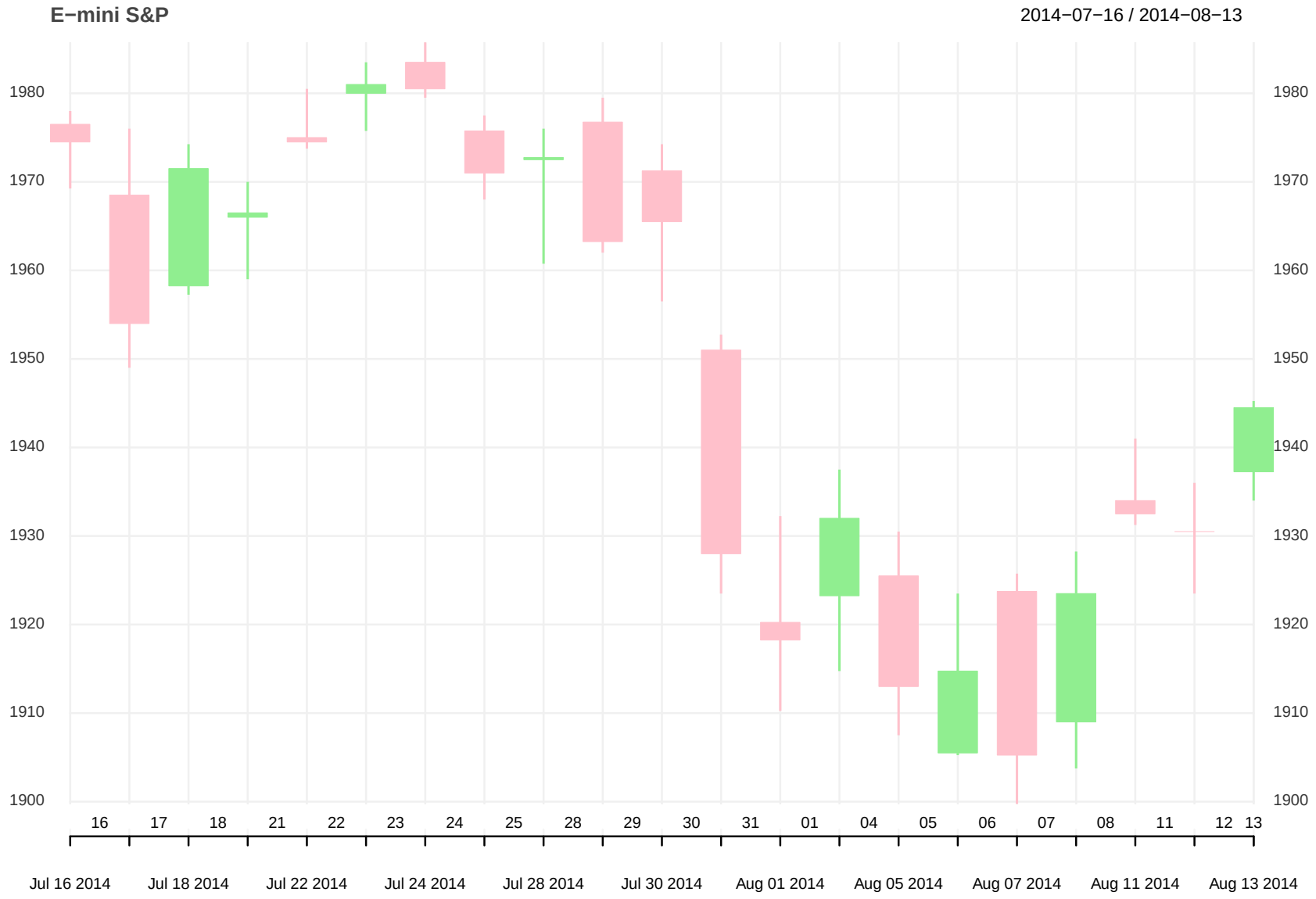
```
tws <- twsConnect()
ES.1D <- reqHistoricalData(tws, emini)
twsDisconnect(tws)
```

```
tail(ES.1D,3)

##             ESU4.Open ESU4.High ESU4.Low ESU4.Close ESU4.Volume ESU4.WAP
## 2014-08-11   1934.00   1941.00  1931.25     1932.5      923977 1935.550
## 2014-08-12   1930.50   1936.00  1923.50     1930.5     1016372 1929.475
## 2014-08-13   1937.25   1945.25  1934.00     1944.5      901580 1940.925
##             ESU4.hasGaps ESU4.Count
## 2014-08-11            0     245837
## 2014-08-12            0     284489
## 2014-08-13            0     235432

library(quantmod)
theme<-chart_theme()
theme$col$up.col<-'lightgreen'
theme$col$up.border<-'lightgreen'
theme$col$dn.col<-'pink'
theme$col$dn.border<-'pink'
plot(chart_Series(ES.1D,theme=theme,name="E-mini S&P"))
```

E-mini S&P daily data



The reqHistory function

The reqHistory function is a wrapper around the reqHistoricalData function which conveniently allows 1-year of 1-minute or 15-minute bars to be returned

```
args(reqHistory)

## function (conn, Contract, barSize = "1 min", ...)
## NULL
```

Main arguments:

conn	a twsConnection object
Contract	a twsContract
barSize	bar size to retrieve

Return value:

an xts object containing the requested data

Getting 1-year of 1-minute bars

Default settings of `reqHistory` return 1-year of 1-minute bars

```
tws <- twsConnect()
ES.1M <- reqHistory(tws, Contract=emini)
twsDisconnect(tws)
```

```
plot(chart_Series(ES.1M,subset="2014-08-11",theme=theme,name="E-mini S&P"))
```

- Function takes a few minutes to execute

E-mini S&P 1-minute bars

Getting 1-year of 15-minute bars

The `reqHistory` function also supports retrieving 1-year of 15-minute bars

```
etf1 <- twsEquity('IVV','SMART','ARCA')
tws <- twsConnect()
IVV.15M <- reqHistory(tws, Contract=etf1, barSize = "15 mins")
twsDisconnect(tws)
```

```
plot(chart_Series(IVV.15M,subset="2014-08",theme=theme,name="S&P ETF"))
```

- Function takes a few minutes to execute

S&P ETF 15-minute bars

Outline

reqMktData events

- reqMktData subscribes to market data updates for an instrument
- Whenever a change in the instrument's market data is seen, an update message is generated
- Market data changes that trigger messages include[†]:
  - bidPrice, bidSize
  - askPrice, askSize
  - lastPrice, lastSize
  - Volume, lastTimeStamp

[†] refer to IBrokers reference and IB API docs for more info

The reqMktData function

reqMktData allows for streaming market data to be handled in R

```
args(reqMktData)

## function (conn, Contract, tickGenerics = "100,101,104,106,165,221,225,236",
##     snapshot = FALSE, tickerId = "1", timeStamp = "%Y%m%d %H:%M:%OS",
##     playback = 1, file = "", verbose = TRUE, eventWrapper = eWrapper(),
##     CALLBACK = twsCALLBACK, ...)
## NULL
```

Main arguments:

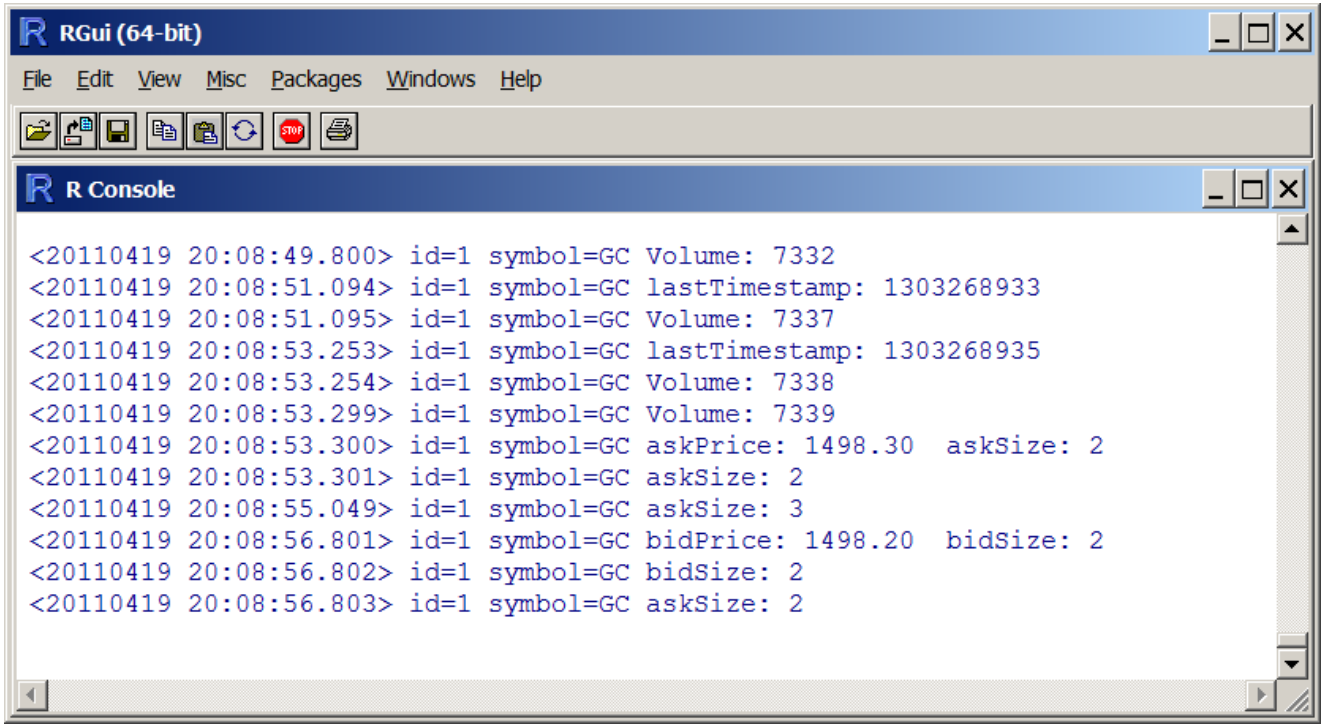
conn	twsConnection object (or twsPlayback connection)
Contract	twsContract object
playback	playback speed (for playback)
file	filename if data should be logged

Return value:

Real-time market data from TWS

Connecting to IB

```r
library(IBrokers)
tws <- twsConnect()
reqMktData(tws, twsFuture("GC","NYMEX","201106"))
twsDisconnect(tws)
```



Logging IB data

```r
reqMktData(tws, future1, file = "QM3.dat")
```

C:\Rprojects\HFcourse\QM3.dat - Notepad++

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

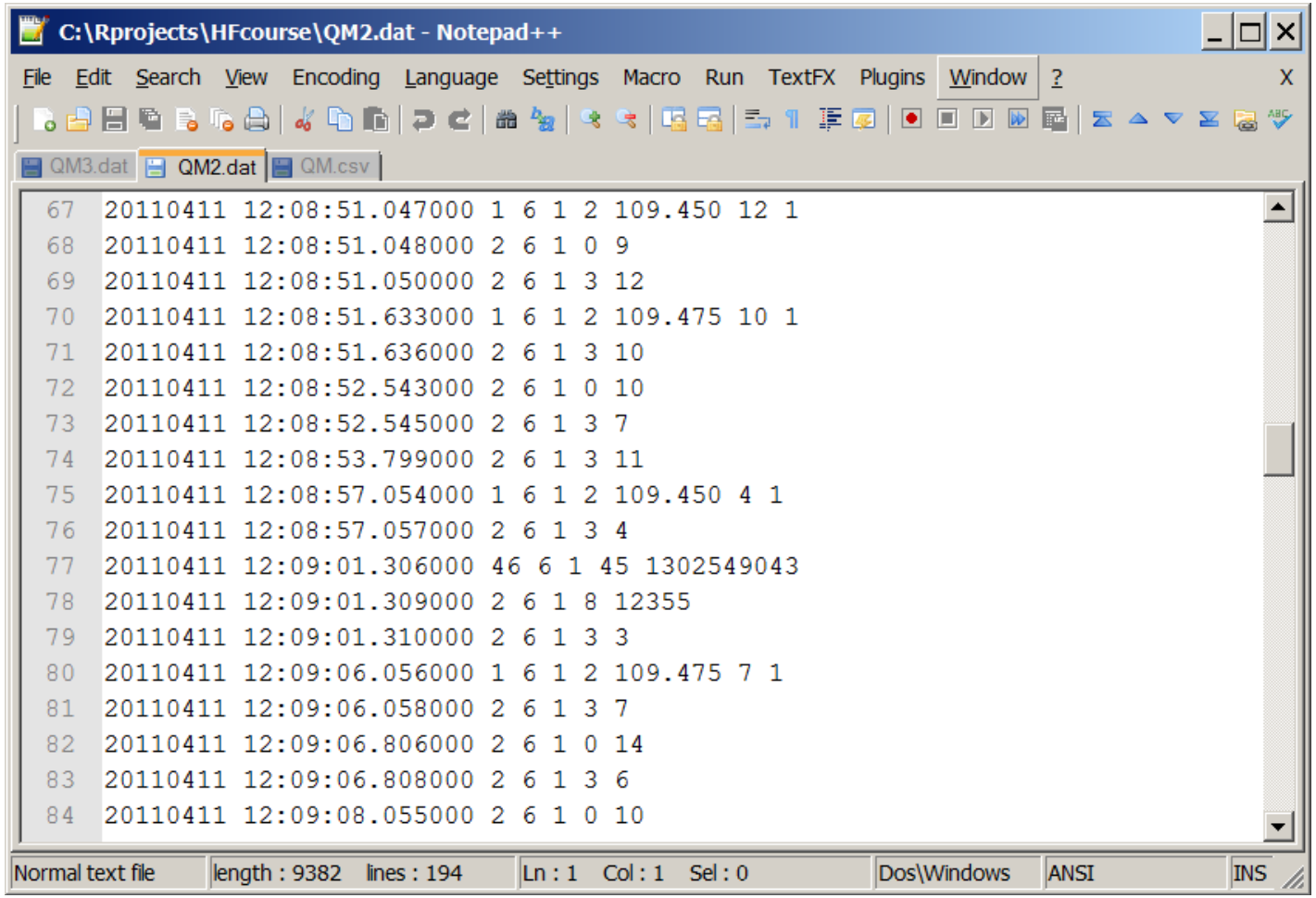
QM3.dat | QM2.dat | QM.csv

```
<20110411 12:17:39.141000> id=1 symbol=QM Volume: 12392
<20110411 12:17:39.145000> id=1 symbol=QM highPrice: 113.450
<20110411 12:17:39.148000> id=1 symbol=QM lowPrice: 109.250
<20110411 12:17:39.332000> id=1 symbol=QM bidPrice: 109.375  bidSize: 6
<20110411 12:17:39.336000> id=1 symbol=QM bidSize: 6
<20110411 12:17:39.338000> id=1 symbol=QM askPrice: 109.400  askSize: 11
<20110411 12:17:39.342000> id=1 symbol=QM askSize: 11
<20110411 12:17:39.428000> id=1 symbol=QM askPrice: 109.425  askSize: 11
<20110411 12:17:39.432000> id=1 symbol=QM lastPrice: 109.425
<20110411 12:17:39.435000> id=1 symbol=QM lastSize: 1
<20110411 12:17:39.438000> id=1 symbol=QM lastTimestamp: 1302549437
<20110411 12:17:39.441000> id=1 symbol=QM closePrice: 112.800
<20110411 12:17:39.444000> id=1 symbol=QM openPrice: 113.300
<20110411 12:17:40.126000> id=1 symbol=QM optionHistoricalVol: NaN NA
<20110411 12:17:40.129000> id=1 symbol=QM optionHistoricalVol: NaN NA
<20110411 12:17:40.419000> id=1 symbol=QM askPrice: 109.400  askSize: 4
<20110411 12:17:40.425000> id=1 symbol=QM askSize: 4
<20110411 12:17:41.082000> id=1 symbol=QM askPrice: 109.425  askSize: 11
```

Normal text file | length : 813438 lines : 13560 | Ln : 1 Col : 1 Sel : 0 | Dos\Windows | ANSI | INS

Logging IB data

```r
reqMktData(tws, future1, CALLBACK = NULL, file = "QM2.dat")
```




```
20110411 12:08:51.047000 1 6 1 2 109.450 12 1
20110411 12:08:51.048000 2 6 1 0 9
20110411 12:08:51.050000 2 6 1 3 12
20110411 12:08:51.633000 1 6 1 2 109.475 10 1
20110411 12:08:51.636000 2 6 1 3 10
20110411 12:08:52.543000 2 6 1 0 10
20110411 12:08:52.545000 2 6 1 3 7
20110411 12:08:53.799000 2 6 1 3 11
20110411 12:08:57.054000 1 6 1 2 109.450 4 1
20110411 12:08:57.057000 2 6 1 3 4
20110411 12:09:01.306000 46 6 1 45 1302549043
20110411 12:09:01.309000 2 6 1 8 12355
20110411 12:09:01.310000 2 6 1 3 3
20110411 12:09:06.056000 1 6 1 2 109.475 7 1
20110411 12:09:06.058000 2 6 1 3 7
20110411 12:09:06.806000 2 6 1 0 14
20110411 12:09:06.808000 2 6 1 3 6
20110411 12:09:08.055000 2 6 1 0 10
```

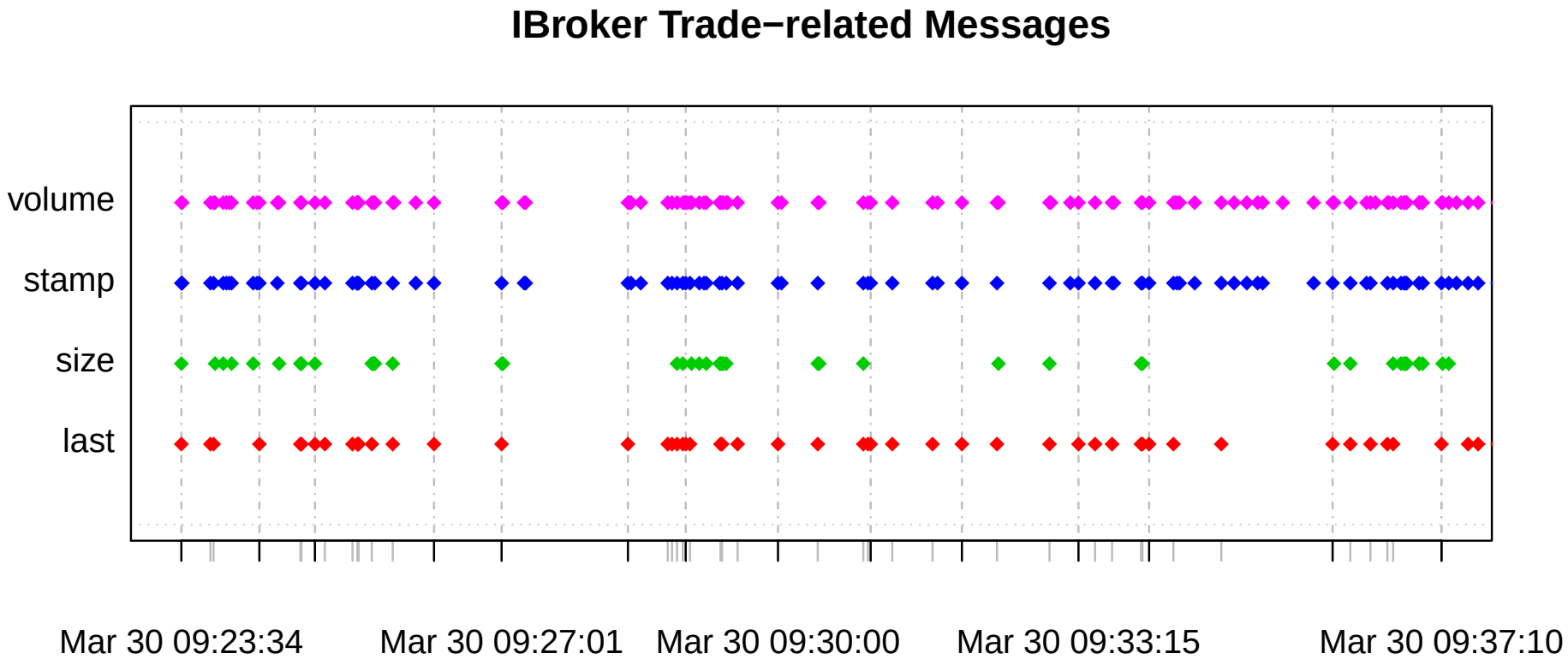
Market data messages

reqMktData messaging

- tick price (message = 1)
  - last price (tick type = 4)
- tick size (message = 2)
  - last size (tick type = 5)
  - volume (tick type = 8)
- tick string (message = 46)
  - timestamp (tick type = 45)

	A	B	C	D	E	F
1	Index	ID	MSG	tickType	parm1	parm2
37	09:24:37.720	1	2	8	5624	NA
38	09:24:51.500	1	1	4	104.85	1
39	09:24:51.503	1	2	5	1	NA
40	09:24:51.505	1	46	45	1301502374	NA
41	09:24:51.508	1	2	8	5625	NA
42	09:24:52.258	1	1	4	104.825	2
43	09:24:52.260	1	2	5	2	NA
44	09:24:52.263	1	46	45	1301502375	NA
45	09:24:52.265	1	2	8	5635	NA
46	09:25:00.960	1	1	4	104.85	1
47	09:25:00.964	1	2	5	1	NA
48	09:25:00.969	1	46	45	1301502383	NA
49	09:25:00.973	1	2	8	5636	NA
50	09:25:07.394	1	1	4	104.825	1
51	09:25:07.398	1	46	45	1301502390	NA
52	09:25:07.403	1	2	8	5637	NA
53	09:25:25.286	1	1	4	104.8	1
54	09:25:25.291	1	46	45	1301502408	NA
55	09:25:25.295	1	2	8	5638	NA
56	09:25:27.906	1	46	45	1301502410	NA
57	09:25:27.911	1	2	8	5639	NA

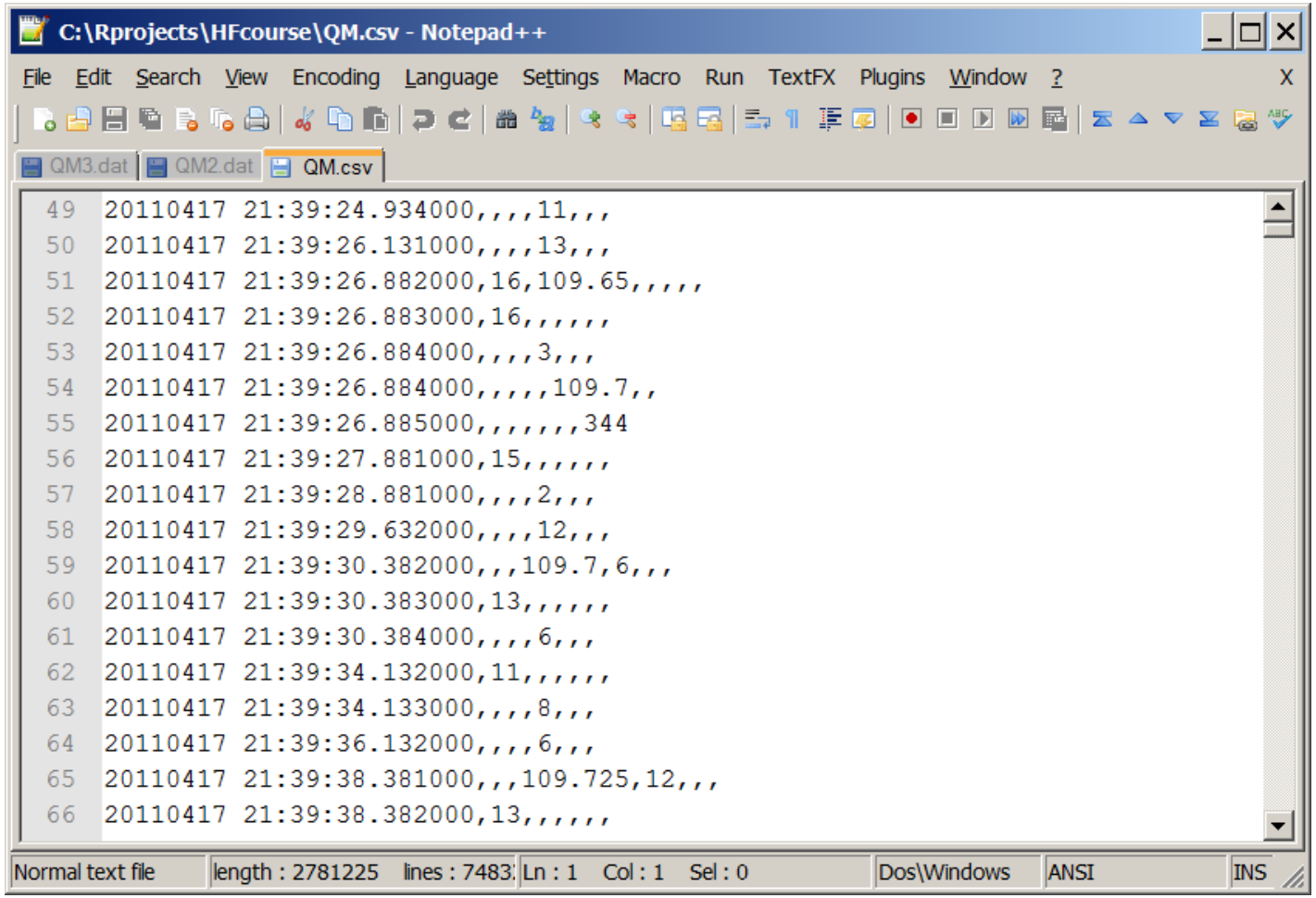
Trade-related IB messages through time



Outline

Logging IB data

```
reqMktData(tws, future1, eventWrapper=eWrapper.MktData.CSV(1), file="qm.csv")
```



Load IB log file

```r
library(IBrokers)
options(digits.secs = 3, digits=12)
dat <- read.table(file="QM.csv", sep=",", header=F, as.is=T)
colnames(dat) <- c("TimeStamp", "BidSize", "Bid", "Ask", "AskSize",
  "Last", "LastSize", "Volume")
head(dat,12)
```

```
##                    TimeStamp BidSize   Bid    Ask AskSize    Last LastSize Volume
## 1  20110417 21:37:53.437000      NA    NA     NA      NA      NA       NA    343
## 2  20110417 21:37:53.623000       4 109.7     NA      NA      NA       NA     NA
## 3  20110417 21:37:53.623000       4    NA     NA      NA      NA       NA     NA
## 4  20110417 21:37:53.624000      NA    NA 109.75      12      NA       NA     NA
## 5  20110417 21:37:53.624000      NA    NA     NA      12      NA       NA     NA
## 6  20110417 21:37:53.718000       9    NA     NA      NA      NA       NA     NA
## 7  20110417 21:37:53.719000      NA    NA     NA      15      NA       NA     NA
## 8  20110417 21:37:53.719000      NA    NA     NA      NA 109.575       NA     NA
## 9  20110417 21:37:53.720000      NA    NA     NA      NA      NA        1     NA
## 10 20110417 21:37:54.622000      11    NA     NA      NA      NA       NA     NA
## 11 20110417 21:37:55.622000      13    NA     NA      NA      NA       NA     NA
## 12 20110417 21:37:56.622000      11    NA     NA      NA      NA       NA     NA
```

Create an xts object

```
timeStamp.raw <- strptime(dat[,1], "%Y%m%d %H:%M:%OS")
class(timeStamp.raw)
```

```
## [1] "POSIXlt" "POSIXt"
```

```
head(timeStamp.raw,3)
```

```
## [1] "2011-04-17 21:37:53.437 PDT" "2011-04-17 21:37:53.623 PDT"
## [3] "2011-04-17 21:37:53.623 PDT"
```

```
x.raw <- xts(dat[,-1],timeStamp.raw)
class(x.raw)
```

```
## [1] "xts" "zoo"
```

```
head(x.raw)
```

```
##                          BidSize   Bid    Ask AskSize Last LastSize Volume
## 2011-04-17 21:37:53.437       NA    NA     NA      NA   NA       NA    343
## 2011-04-17 21:37:53.622        4 109.7     NA      NA   NA       NA     NA
## 2011-04-17 21:37:53.622        4    NA     NA      NA   NA       NA     NA
## 2011-04-17 21:37:53.624       NA    NA 109.75      12   NA       NA     NA
## 2011-04-17 21:37:53.624       NA    NA     NA      12   NA       NA     NA
## 2011-04-17 21:37:53.717        9    NA     NA      NA   NA       NA     NA
```

Fill in NAs

```
x <- na.locf(x.raw)
x['2011-04-17 21:45',]

##                          BidSize   Bid     Ask AskSize   Last LastSize Volume
## 2011-04-17 21:45:05.667       1 109.7 109.725       3 109.75        1    354
## 2011-04-17 21:45:06.667       1 109.7 109.725       5 109.75        1    354
## 2011-04-17 21:45:07.167       1 109.7 109.725       1 109.75        1    354
## 2011-04-17 21:45:09.417       1 109.7 109.750      18 109.75        1    354
## 2011-04-17 21:45:09.418       1 109.7 109.750      18 109.75        1    354
## 2011-04-17 21:45:10.417       1 109.7 109.725       6 109.75        1    354
## 2011-04-17 21:45:10.418       1 109.7 109.725       6 109.75        1    354
## 2011-04-17 21:45:24.763       1 109.7 109.725       4 109.75        1    354
## 2011-04-17 21:45:40.668       1 109.7 109.725       6 109.75        1    354
## 2011-04-17 21:45:45.673       1 109.7 109.725       8 109.75        1    354
## 2011-04-17 21:45:46.673       1 109.7 109.725      10 109.75        1    354
```

- `na.locf` - zoo function for last observation carried forward
- note `xts` indexing by date/hour/minute

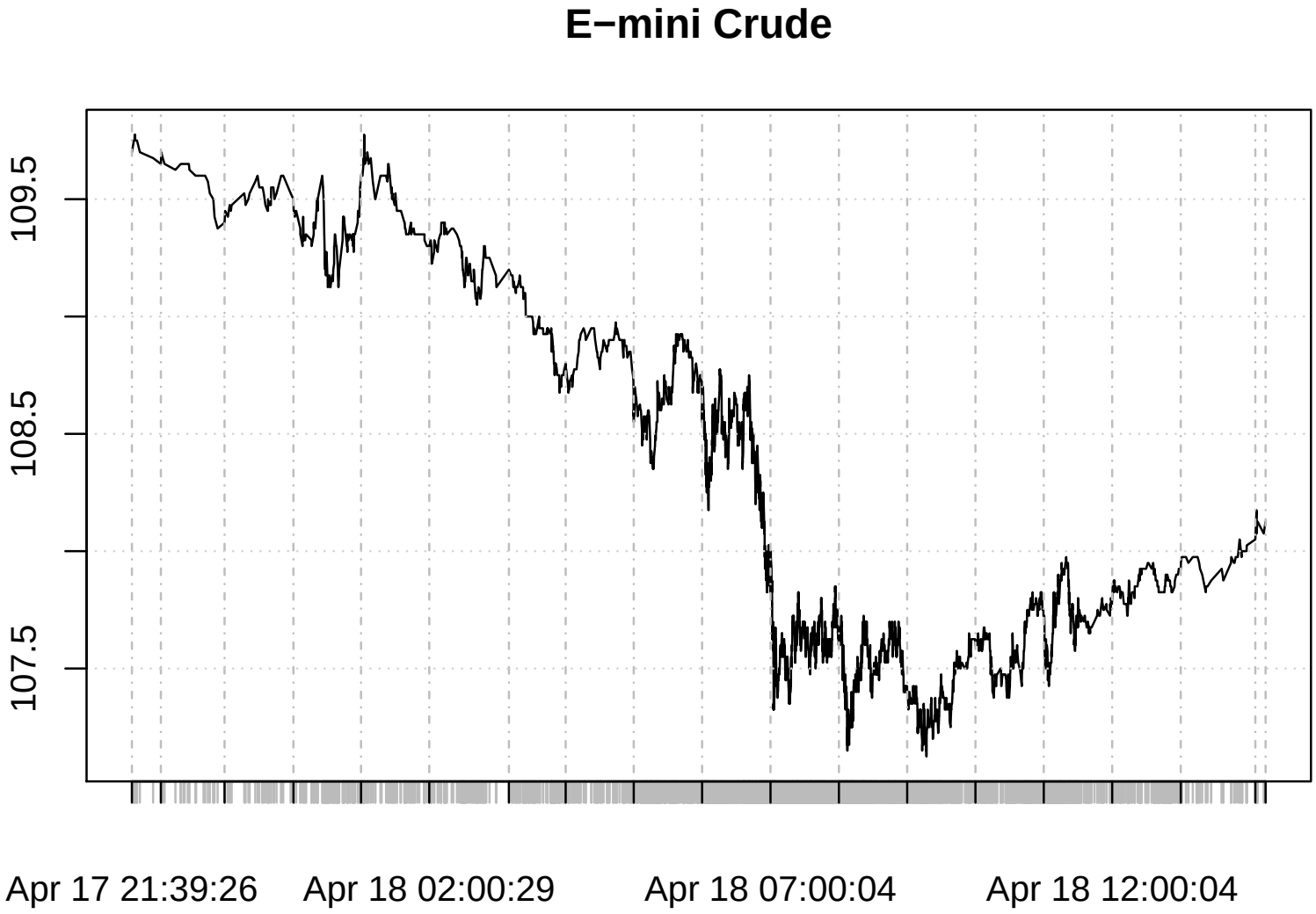
Extract trades from log of quotes and trades

```
dx <- diff(x)
trade.idx <- dx[,"Volume"]>0
trades <- x[trade.idx,]
trades[,"LastSize"] <- coredata(dx[trade.idx,"Volume"])
head(trades,10)
```

```
##                          BidSize     Bid     Ask AskSize    Last LastSize Volume
## 2011-04-17 21:39:26.884       16 109.650 109.725       3 109.700        1    344
## 2011-04-17 21:41:09.145        2 109.700 109.750      12 109.750        1    345
## 2011-04-17 21:41:38.648        6 109.700 109.750      14 109.750        1    346
## 2011-04-17 21:41:44.490        8 109.700 109.750       6 109.750        1    347
## 2011-04-17 21:41:49.401        1 109.750 109.775       8 109.775        2    349
## 2011-04-17 21:42:12.403        1 109.725 109.775      20 109.775        1    350
## 2011-04-17 21:42:21.654        1 109.725 109.750       2 109.750        1    351
## 2011-04-17 21:43:48.911        1 109.725 109.750      12 109.750        1    352
## 2011-04-17 21:43:50.911        1 109.725 109.750      10 109.750        1    353
## 2011-04-17 21:43:53.661        1 109.725 109.750       8 109.750        1    354
```

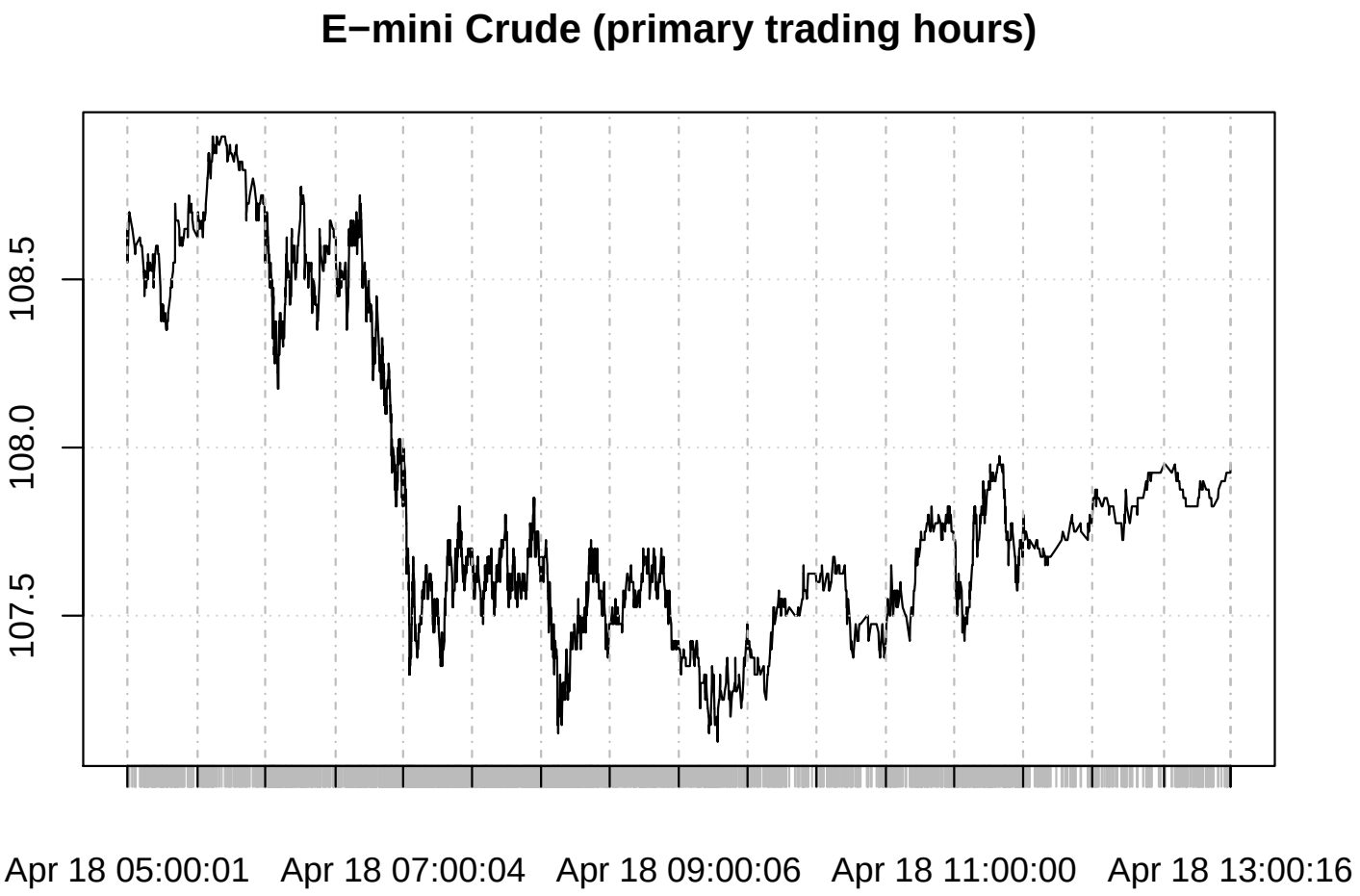
```
plot(trades[,"Last"],main="E-mini Crude")
```

Plot of trades from `plot.xts`



Plot of trades - primary session

```
plot(trades['2011-04-18 05:00::2011-04-18 13:00',"Last"],
  main="E-mini Crude (primary trading hours)")
```



Create 5-minute bars

```
trades.5 <- align.time(trades,60*5)
head(trades.5)

##                      BidSize     Bid     Ask AskSize    Last LastSize Volume
## 2011-04-17 21:40:00      16 109.650 109.725       3 109.700        1    344
## 2011-04-17 21:45:00       2 109.700 109.750      12 109.750        1    345
## 2011-04-17 21:45:00       6 109.700 109.750      14 109.750        1    346
## 2011-04-17 21:45:00       8 109.700 109.750       6 109.750        1    347
## 2011-04-17 21:45:00       1 109.750 109.775       8 109.775        2    349
## 2011-04-17 21:45:00       1 109.725 109.775      20 109.775        1    350

trades.ohlc <- to.minutes5(trades.5[,c("Last","Volume")])
colnames(trades.ohlc) <- c("Open","High","Low","Close","Volume")
head(trades.ohlc)

##                         Open    High     Low   Close Volume
## 2011-04-17 21:40:00 109.700 109.700 109.700 109.700    344
## 2011-04-17 21:45:00 109.750 109.775 109.750 109.750   3147
## 2011-04-17 21:50:00 109.700 109.700 109.700 109.700    355
## 2011-04-17 22:00:00 109.675 109.675 109.675 109.675    356
## 2011-04-17 22:05:00 109.650 109.650 109.650 109.650    357
## 2011-04-17 22:10:00 109.700 109.700 109.650 109.650   1079
```

Fill-in gaps in 5-minute bars

```
trades.ohlc <- merge(trades.ohlc[endpoints(trades.ohlc,'minutes')],
  xts( ,seq(start(trades.ohlc),end(trades.ohlc),by="5 mins")))
head(trades.ohlc,10)

##                          Open    High     Low   Close Volume
## 2011-04-17 21:40:00 109.700 109.700 109.700 109.700    344
## 2011-04-17 21:45:00 109.750 109.775 109.750 109.750   3147
## 2011-04-17 21:50:00 109.700 109.700 109.700 109.700    355
## 2011-04-17 21:55:00      NA      NA      NA      NA     NA
## 2011-04-17 22:00:00 109.675 109.675 109.675 109.675    356
## 2011-04-17 22:05:00 109.650 109.650 109.650 109.650    357
## 2011-04-17 22:10:00 109.700 109.700 109.650 109.650   1079
## 2011-04-17 22:15:00      NA      NA      NA      NA     NA
## 2011-04-17 22:20:00 109.625 109.625 109.625 109.625    362
## 2011-04-17 22:25:00 109.650 109.650 109.650 109.650   1458
```

Fill-in NAs in gaps

```
trades.ohlc[is.na(trades.ohlc[,"Volume"]),"Volume"] <- 0
trades.ohlc[,"Close"] <- na.locf(trades.ohlc[,"Close"])
trades.ohlc[is.na(trades.ohlc[,"Open"]),"Open"] <-
  trades.ohlc[is.na(trades.ohlc[,"Open"]),"Close"]
trades.ohlc[is.na(trades.ohlc[,"Low"]),"Low"] <-
  trades.ohlc[is.na(trades.ohlc[,"Low"]),"Close"]
trades.ohlc[is.na(trades.ohlc[,"High"]),"High"] <-
  trades.ohlc[is.na(trades.ohlc[,"High"]),"Close"]
head(trades.ohlc)

##                         Open    High     Low   Close Volume
## 2011-04-17 21:40:00 109.700 109.700 109.700 109.700    344
## 2011-04-17 21:45:00 109.750 109.775 109.750 109.750   3147
## 2011-04-17 21:50:00 109.700 109.700 109.700 109.700    355
## 2011-04-17 21:55:00 109.700 109.700 109.700 109.700      0
## 2011-04-17 22:00:00 109.675 109.675 109.675 109.675    356
## 2011-04-17 22:05:00 109.650 109.650 109.650 109.650    357

plot(chart_Series(trades.ohlc['2011-04-18 05:00::2011-04-18 13:00',],
  theme=theme,name="E-mini Crude"))
```

Candlestick plot from chartSeries

Outline

Order management

IBrokers supplies the following functions for order management:

placeOrder	Place an order to the TWS
cancelOrder	Cancel an order to the TWS
reqIds	Get the next valid order ID
reqAccountUpdates	Subscribe to account updates
reqOpenOrders	Subscribe to order status updates

The placeOrder function

The placeOrder function sends an order to TWS

```
args(placeOrder)

## function (twsconn, Contract, Order)
## NULL
```

Main arguments:

twsconn	twsConnection object
Contract	twsContract object
Order	twsOrder object

Return value:

orderId

The twsOrder function

twsOrder creates a twsOrder object for the placeOrder function

```
args(twsOrder)


## function (orderId, action = "BUY", totalQuantity = "10", orderType = "LMT",
##     lmtPrice = "0.0", auxPrice = "0.0", tif = "", outsideRTH = "0",
##     openClose = "O", origin = .twsOrderID$CUSTOMER, ocaGroup = "",
##     account = "", orderRef = "", transmit = TRUE, parentId = "0",
##     blockOrder = "0", sweepToFill = "0", displaySize = "0", triggerMethod = "0",
##     hidden = "0", discretionaryAmt = "0.0", goodAfterTime = "",
##     goodTillDate = "", faGroup = "", faMethod = "", faPercentage = "",
##     faProfile = "", shortSaleSlot = "0", designatedLocation = .twsOrderID$EMPTY_STR,
##     ocaType = "0", rule80A = "", settlingFirm = "", clearingAccount = "",
##     clearingIntent = "", allOrNone = "0", minQty = "", percentOffset = "",
##     eTradeOnly = "0", firmQuoteOnly = "0", nbboPriceCap = "",
##     auctionStrategy = "0", startingPrice = "", stockRefPrice = "",
##     delta = "", stockRangeLower = "", stockRangeUpper = "", overridePercentageConstraints = "0",
##     volatility = "", volatilityType = "", deltaNeutralOrderType = "",
##     deltaNeutralAuxPrice = "", continuousUpdate = "0", referencePriceType = "",
##     trailStopPrice = "", basisPoints = "", basisPointsType = "",
##     scaleInitLevelSize = "", scaleSubsLevelSize = "", scalePriceIncrement = "",
##     notHeld = FALSE, algoStrategy = "", algoParams = NULL, whatIf = FALSE,
##     clientId = "", permId = "")
## NULL
```

The twsOrder function

Main arguments:

orderID	The id for the order (use reqIds)
action	BUY, SELL, SSHORT
totalQuantity	Order quantity
orderType	MKT, LMT, STP, STPLMT, TRAIL, etc.
account	Account (may not need this)

Return value:

a twsOrder object

Sell 1 ESU3 at the market

RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
> future1 <- twsFuture("ES", "GLOBEX", "201309")
> (id <- reqIds(tws))
[1] "1"
> (placeOrder(tws, future1, twsOrder(orderId=id,action="SELL",totalQuantity="1",orderType="MKT",account="DI147392")))
[1] 1
>
> reqOpenOrders(tws)
TWS Message: 2 -1 2104 Market data farm connection is OK:usfuture
TWS Message: 2 -1 2104 Market data farm connection is OK:cashfarm
TWS Message: 2 -1 2104 Market data farm connection is OK:usfarm
TWS Message: 2 -1 2106 HMDS data farm connection is OK:ilhmds
TWS Message: 2 -1 2106 HMDS data farm connection is OK:cashhmds
TWS Message: 2 -1 2106 HMDS data farm connection is OK:ushmds
TWS Execution: orderId=1 time=2013-08-18 16:08:48 side=SLD shares=1 symbol=ES conId=108816234 price=1650.75
TWS OrderStatus: orderId=1 status=Filled filled=1 remaining=0 averageFillPrice=1650.75
TWS OrderStatus: orderId=1 status=Filled filled=1 remaining=0 averageFillPrice=1650.75
TWS OrderStatus: orderId=1 status=Filled filled=1 remaining=0 averageFillPrice=1650.75
```

Trades (Simulated Trading)

File Edit Trades View Settings Help

SIMULATED TRADING SIMULATED TRADING SIMULATED TRADING

Account Filter

Account: All

Show trades ☑ Sun ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ All

Trades Summary

Enter symbol filter More options

Action	Quantity	Underlying	Description	Price	Currency	Exch.	Time	Order Ref.	Account	Submitter	Commission
SLD	1	ES	SEP13 Futures	1650.75	USD	GLOBEX	16:08:48		DI147392		2.01

Buy 1 ESU3 at limit=1650.00

RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
> future1 <- twsFuture("ES", "GLOBEX", "201309")
> (id <- reqIds(tws))
[1] "3"
> (placeOrder(tws, future1, twsOrder(orderId=id,action="BUY",totalQuantity="1",
   orderType="LMT",lmtPrice="1650.00",account="DI147392")))
[1] 3
> reqOpenOrders(tws)
TWS OrderStatus: orderId=3 status=ApiPending filled=0 remaining=1 averageFillPrice=
TWS OrderStatus: orderId=3 status=Submitted filled=0 remaining=1 averageFillPrice=0
```

All IB Trader Workstation (Simulated Trading)

File Edit Trade Account Trading Tools Analytical Tools View Help

SIMULATED TRADING SIMULATED TRADING SIMULATED TRADING

Order Account Trade Log Information System LAUNCH Calendars News Fundamentals BookTrader OptionTrader Mkt Scanner Alerts FXTrader Chart News News & Research

API Portfolio Quotes Pending (All) +

Select Account All

Contract	Last / Account	Change / Action	Change (%) / Quantity	Bid Size / Time in Force	Bid / Type	Ask / Lmt Price	Ask Size / Destination	Position / Status	Trd Px	Cancel
ES Sep'13 @GL...	1652.00	+1.00	0.06%	42	1651.75	1652.00	59	20		
	DI147392	BUY	1	DAY	LMT	1650.00	GLOBEX			Cancel
	DI147392	SELL	0	DAY	MKT	MARKET	GLOBEX	1	1650.75	

Cancel limit order

RGui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
> future1 <- twsFuture("ES", "GLOBEX", "201309")
> (id <- reqIds(tws))
[1] "3"
> (placeOrder(tws, future1, twsOrder(orderId=id,action="BUY",totalQuantity="1",
   orderType="LMT",lmtPrice="1650.00",account="DI147392")))
[1] 3
> reqOpenOrders(tws)
TWS OrderStatus: orderId=3 status=ApiPending filled=0 remaining=1 averageFillPrice=
TWS OrderStatus: orderId=3 status=Submitted filled=0 remaining=1 averageFillPrice=0

>
> cancelOrder(tws,id)
> reqOpenOrders(tws)
TWS OrderStatus: orderId=3 status=PendingCancel filled=0 remaining=1 averageFillPrice=0
TWS Message: 2 3 202 Order Canceled - reason:
TWS OrderStatus: orderId=3 status=Cancelled filled=0 remaining=1 averageFillPrice=0
```

All IB Trader Workstation (Simulated Trading)

File Edit Trade Account Trading Tools Analytical Tools View Help

SIMULATED TRADING SIMULATED TRADING SIMULATED TRADING

Order Account Trade Log IB Information System LAUNCH Calendars News Fundamentals BookTrader OptionTrader Mkt Scanner Alerts FXTrader Chart News News & Research

API Portfolio Quotes Pending (All) +

Select Account All

Contract	Last / Account	Change / Action	Change (%) / Quantity	Bid Size / Time in Force	Bid / Type	Ask / Lmt Price	Ask Size / Destination	Position / Status	Trd Px	Cancel
ES Sep'13 @GL...	1652.00	+1.00	0.06%	52	1651.75	1652.00	78	20		
	DI147392	SELL	0	DAY	MKT	MARKET	GLOBEX	1	1650.75	

Outline

The twsInstrument package

Description

- Improves ease of use of IBrokers and FinancialInstrument

Key features

- Easily create twsContract and instrument objects
- Includes functions to make requesting historical data from IB easier (esp. TBBO data)
- Defines stocks from vector of ticker symbols, a csv file, or a data.frame
- Alpha software - still under development

Authors

- Garrett See

The getBAT function

The function getBAT downloads and merges data for Bid, Ask, and Trade

```
library(twsInstrument)
args(getBAT)

## function (Symbols, endDateTime, tws = NULL, barSize = "1 min",
##     duration = "5 D", useRTH = "1", auto.assign = TRUE, env = .GlobalEnv)
## NULL
```

```
getBAT("ES_Z4")
```

Bid, ask, trade data

```r
head(ES_Z4)
```

```
##                     ES.Bid.Price ES.Ask.Price ES.Trade.Price ES.Mid.Price ES.Volume
## 2014-08-07 06:30:00      1915.75      1916.25        1915.75     1916.000        10
## 2014-08-07 06:31:00      1916.00      1916.25        1916.25     1916.125         8
## 2014-08-07 06:32:00      1915.50      1915.75        1916.00     1915.625        32
## 2014-08-07 06:33:00      1914.75      1915.00        1915.50     1914.875        10
## 2014-08-07 06:34:00      1915.75      1916.00        1916.00     1915.875        21
## 2014-08-07 06:35:00      1914.00      1914.50        1914.00     1914.250        26
```

```r
tail(ES_Z4)
```

```
##                     ES.Bid.Price ES.Ask.Price ES.Trade.Price ES.Mid.Price ES.Volume
## 2014-08-13 13:09:00      1936.25      1936.75        1936.75      1936.50         0
## 2014-08-13 13:10:00      1936.25      1936.75        1936.75      1936.50         0
## 2014-08-13 13:11:00      1936.25      1936.75        1936.75      1936.50         0
## 2014-08-13 13:12:00      1937.00      1937.50        1936.50      1937.25         1
## 2014-08-13 13:13:00      1936.75      1937.25        1937.00      1937.00         1
## 2014-08-13 13:14:00      1936.50      1937.00        1937.00      1936.75         0
```

The get_quote function

The get_quote function downloads current quotes from IB

```
args(get_quote)

## function (Symbols, src = "IB", ...)
## NULL
```

```
define_FX(c("EUR.USD","GBP.USD"))
```

```
get_quote(ls_exchange_rates())

##          BidSize BidPrice AskPrice  AskSize
## EUR.USD 23370000  1.33685  1.33695 25019000
## GBP.USD 12409000  1.66885  1.66895  9580000
```

```
define_futures("ES","GLOBEX","201409")
```

```
get_quote(ls_futures())

##      BidSize BidPrice AskPrice AskSize    Last LastSize Volume
## ESU4     995  1950.25   1950.5    1371 1950.25        1 820788
```

The ATMQuote function

ATMQuote returns quotes for at-the-money options on a given stock

```
args(ATMQuote)

## function (symbol, Exp)
## NULL

ATMQuote('SPY')

##                     Strike Last   Chg  Bid  Ask   Vol    OI
## SPY140816C00195500  195.5 0.45  0.06 0.45 0.46 33480 34743
## SPY140822C00195500  195.5 1.26  0.17 1.25 1.27  3114  8846
## SPY140829C00195500  195.5 1.70  0.32 1.69 1.70  2296  2117
## SPY140816P00195500  195.5 0.37 -0.49 0.36 0.37 37303 26946
## SPY140822P00195500  195.5 1.17 -0.52 1.17 1.18  4224  2078
## SPY140829P00195500  195.5 1.62 -0.54 1.61 1.62  2658  2906
```

Outline

Final exam scope

- Reading
  - Lewis chapters 1-8
  - Tomasini chapters 1-6
  - quantmod, blotter, quantstrat, PerformanceAnalytics, IBrokers documentation (relevant to lectures and assignments)
- Lectures
  - 7 to 16
- Assignments
  - 4 to 6

Final exam logistics

- Test format
  - 90 minutes
  - Approximately 30 - 35 questions
  - Multiple choice and short answer
  - Closed book, no notes, no calculator, no scrape paper required
- Time and place
  - Wednesday 8/20 at 2:30-4:20 PM in Loew 206
  - All in-state students take the exam in 206 @ 2:30 PM on Wednesday
  - Out-of-state students will take a proctored exam on 8/20

Wrap up

- Reading
  - Tomasini/Jaekle Chapter 6
- Homework
  - Assignment #6 due Sunday by mid-night PDT
- Next lecture
  - Walk-forward analysis
- Final exam
  - Wednesday 2014-08-20 at 2:30 PM PDT in Loew 206
- Questions, comments, concerns
  - Post to the general discussion forum on Canvas
  - Guy, gyollin@uw.edu

COMPUTATIONAL FINANCE & RISK MANAGEMENT

UNIVERSITY *of* WASHINGTON

Department of Applied Mathematics

http://depts.washington.edu/compfin