High-Frequency Financial Data Analysis

Guy Yollin
Principal Consultant, r-programming.org

Scott Payseur
Quantitative Analyst, UBS Global Asset Management

Guy Yollin

- Professional Experience
  - Software Engineering
    - r-programming.org
    - Insightful Corporation
    - Electro Scientific Industries, Vision Products Division
  - Hedge Fund
    - Rotella Capital Management
    - J.E. Moody, LLC
  - Academic
    - University of Washington
    - Oregon Health & Science University
- Education
  - Oregon Health & Science University, MS Computational Finance
  - Drexel University, BS Electrical Engineering

Scott Payseur

- Professional Experience
  - Finance Industry
    - UBS Global Asset Management
    - Insightful Corporation
  - Software Engineering
    - ESPN.COM
    - ShareBuilder.com
  - Academic
    - PhD, Financial Ecomometrics, University of Washington
    - Author of 'Realized' R Package (available on CRAN)
- Education
  - University of Washington, PhD Economics
  - University of Colorado, BS Computer Science

Outline

Legal Disclaimer

- This presentation is for informational purposes
- This presentation should not be construed as a solicitation or offering of investment services
- The presentation does not intend to provide investment advice
- The information in this presentation should not be construed as an offer to buy or sell, or the solicitation of an offer to buy or sell any security, or as a recommendation or advice about the purchase or sale of any security
- The presenter(s) shall not shall be liable for any errors or inaccuracies in the information presented
- There are no warranties, expressed or implied, as to accuracy, completeness, or results obtained from any information presented

INVESTING ALWAYS INVOLVES RISK

Outline

Time series data

Time series

A *time series* is a sequence of observations in chronological order

Time series object

A time series object in R is a *compound data object* that includes a data matrix as well as a vector of associated time stamps

class	**package**	**overview**
`ts`	base	regularly spaced time series
`mts`	base	multiple regularly spaced time series
`timeSeries`	rmetrics	default for Rmetrics packages
`zoo`	zoo	reg/irreg and arbitrary time stamp classes
`xts`	xts	an extension of the zoo class

What is high-frequency?

- High-frequency begins when indexing by date alone is not enough
  - beyond the capabilities of `zoo` objects with `Date` indexes
    - intra-day bars
    - tick data
- High-frequency time series require formal time-based classes for indexing
- Recommended classes for high-frequency time series:
  - xts (e**x**tensible **t**ime **s**eries) class
  - POSIXct/POSIXlt date-time class for indexing

	Low frequency	**High Frequency**
time series class	`zoo`	`xts`
time index class	`Date`	`POSIXlt`

The xts package

Description

- xts provides for uniform handling of R's different time-based data classes by extending zoo, maximizing native format information preservation and allowing for user level customization and extension

Key features

- supports fine-grained time indexes
- extends the zoo class
- interoperability with other time series classes
- user-defined attributes

Authors

- Jeffrey Ryan (R/Finance 2011 committee member)
- Josh Ulrich (R/Finance 2011 committee member)

Date-time classes

POSIXct represents time and date as the number of seconds since 1970-01-01

POSIXlt represents time and date as 9 calendar and time components

R Code:

```
> d <- Sys.time()
> class(d)

[1] "POSIXct" "POSIXt"

> unclass(d)

[1] 1304350062

> sapply(unclass(as.POSIXlt(d)), function(x) x)

      sec       min      hour      mday       mon      year      wday      yday
 41.71847  27.00000  15.00000   2.00000   4.00000 111.00000   1.00000 121.00000
    isdst
  0.00000
```

The strptime function

The strptime function converts character strings to POSIXlt date-time objects

R Code: The strptime function

```
> args(strptime)

function (x, format, tz = "")
NULL
```

Arguments:

x	vector of character strings to be converted to POSIXlt objects
format	date-time format specification
tz	timezone to use for conversion

Return value:

POSIXlt object(s)

The xts function

The function xts is the constructor function for extensible time-series objects

R Code: The xts function

```
> library(xts)
> args(xts)

function (x = NULL, order.by = index(x), frequency = NULL, unique = TRUE,
    tzone = Sys.getenv("TZ"), ...)
NULL
```

Arguments:

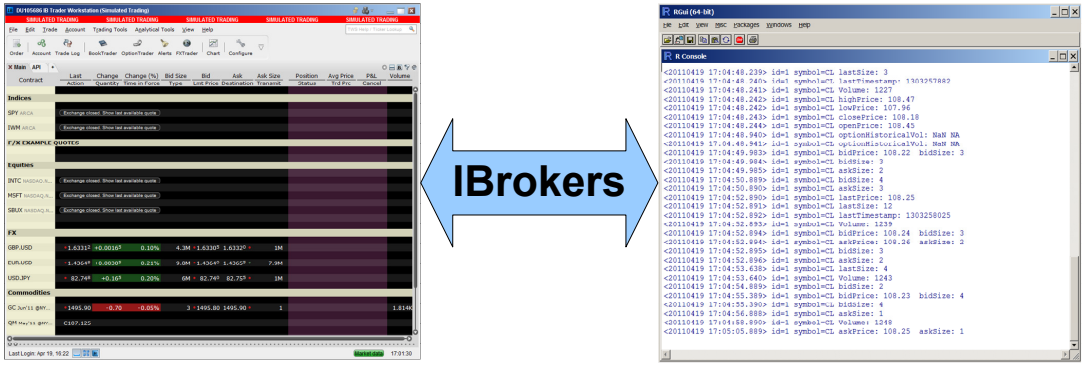
x	data matrix (or vector) with time series data
order.by	vector of unique times/dates (POSIXct is recommended)

Return value:

xts object

Outline

IBrokers - interface to the working man's datafeed



Description

- Provides native R access to Interactive Brokers Trader Workstation

Key features

- real-time market data feed: reqMktData, reqRealTimeBars
- live order execution: placeOrder, cancelOrder, reqAccountUpdates
- historic data: reqHistoricalData
- software comes with NO WARRANTY

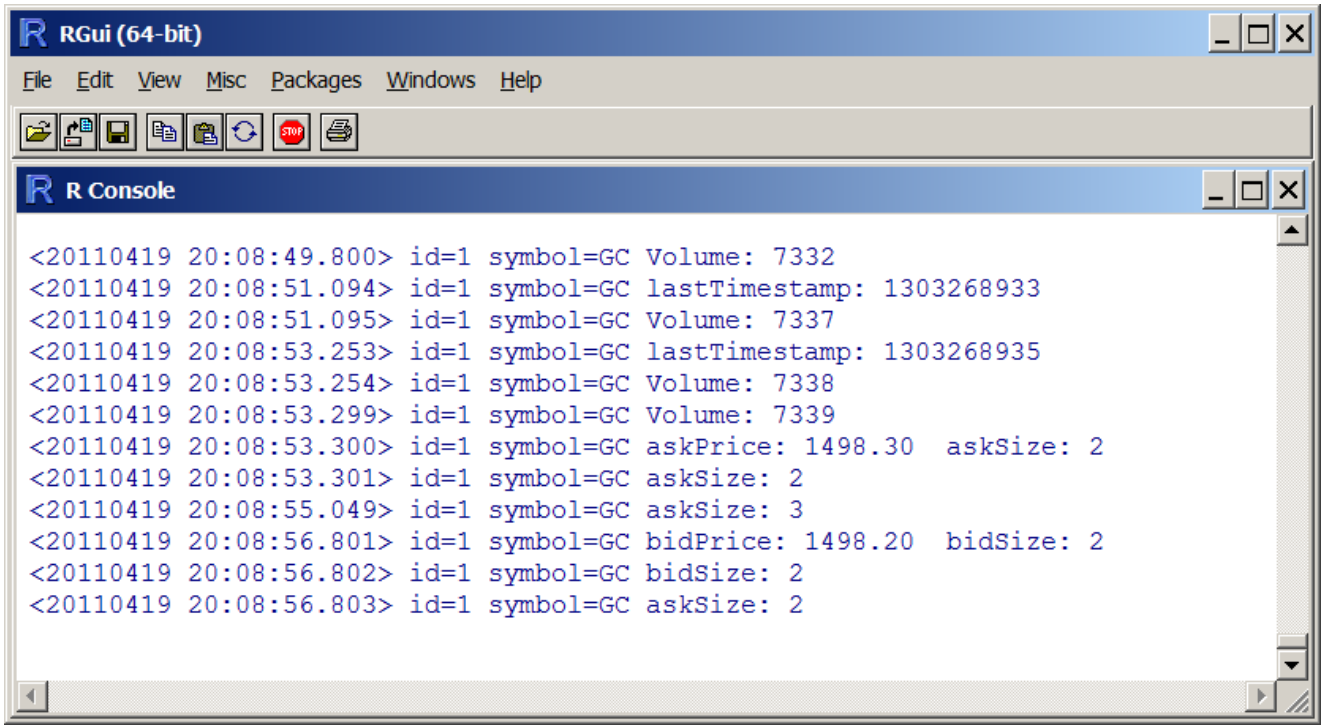
Authors

- Jeffrey Ryan

Connecting to IB

R Code: Get real-time market data from IB

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



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

Logging IB data

R Code: Log IB data to a file (default settings)

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

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

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

QM3.dat | QM2.dat | QM.csv

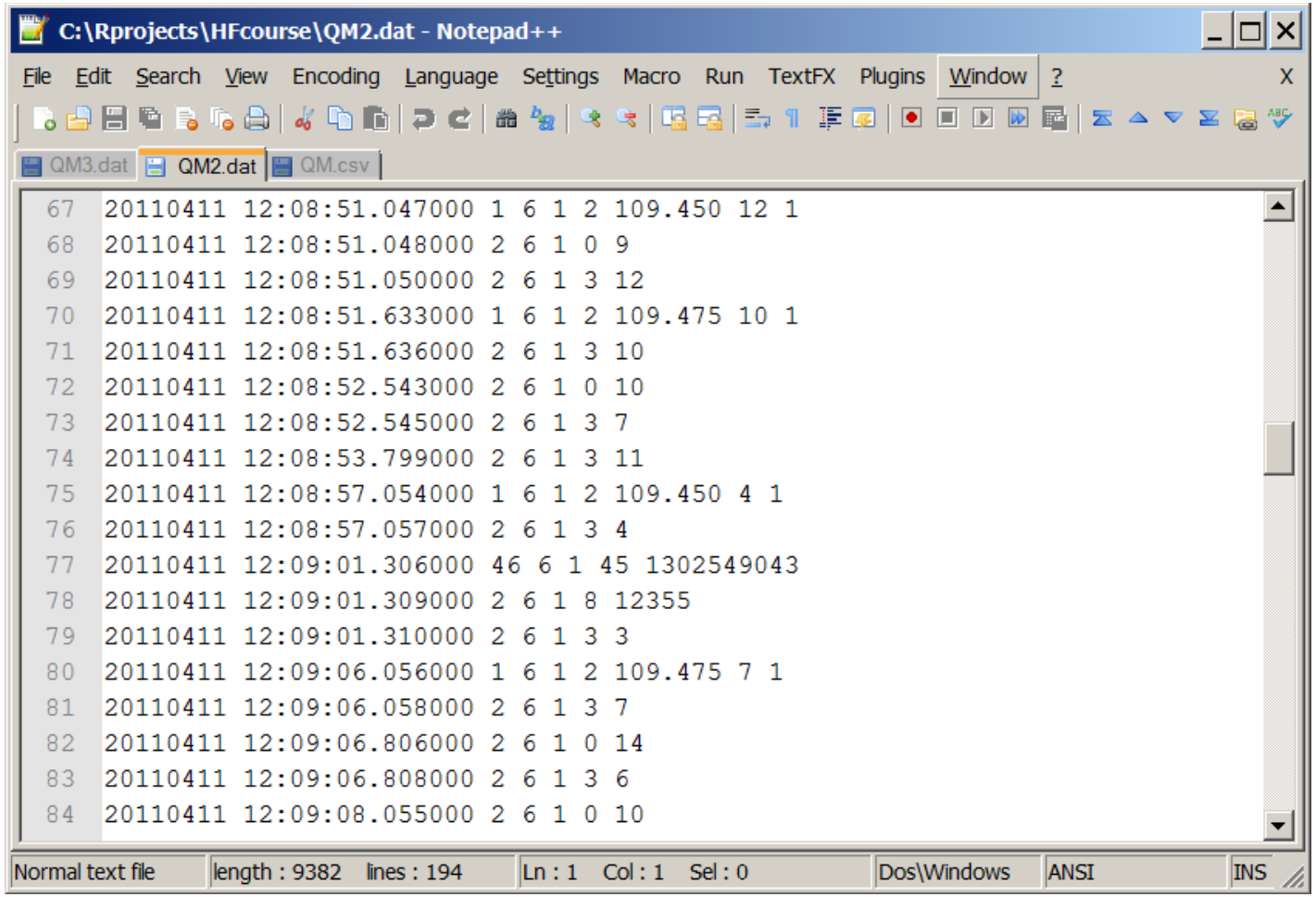
```
1  <20110411 12:17:39.141000> id=1 symbol=QM Volume: 12392
2  <20110411 12:17:39.145000> id=1 symbol=QM highPrice: 113.450
3  <20110411 12:17:39.148000> id=1 symbol=QM lowPrice: 109.250
4  <20110411 12:17:39.332000> id=1 symbol=QM bidPrice: 109.375  bidSize: 6
5  <20110411 12:17:39.336000> id=1 symbol=QM bidSize: 6
6  <20110411 12:17:39.338000> id=1 symbol=QM askPrice: 109.400  askSize: 11
7  <20110411 12:17:39.342000> id=1 symbol=QM askSize: 11
8  <20110411 12:17:39.428000> id=1 symbol=QM askPrice: 109.425  askSize: 11
9  <20110411 12:17:39.432000> id=1 symbol=QM lastPrice: 109.425
10 <20110411 12:17:39.435000> id=1 symbol=QM lastSize: 1
11 <20110411 12:17:39.438000> id=1 symbol=QM lastTimestamp: 1302549437
12 <20110411 12:17:39.441000> id=1 symbol=QM closePrice: 112.800
13 <20110411 12:17:39.444000> id=1 symbol=QM openPrice: 113.300
14 <20110411 12:17:40.126000> id=1 symbol=QM optionHistoricalVol: NaN NA
15 <20110411 12:17:40.129000> id=1 symbol=QM optionHistoricalVol: NaN NA
16 <20110411 12:17:40.419000> id=1 symbol=QM askPrice: 109.400  askSize: 4
17 <20110411 12:17:40.425000> id=1 symbol=QM askSize: 4
18 <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 Code: Log individual IB messages

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



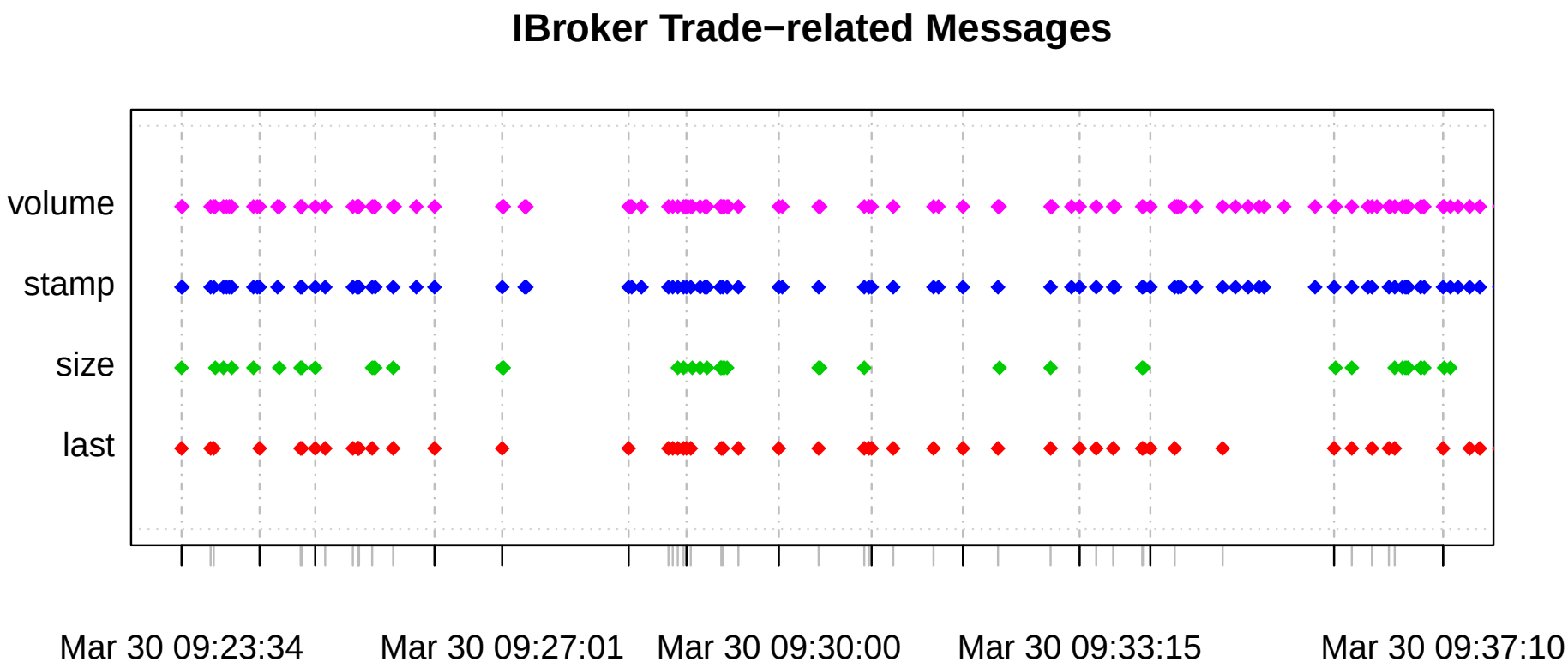
Market data messages

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

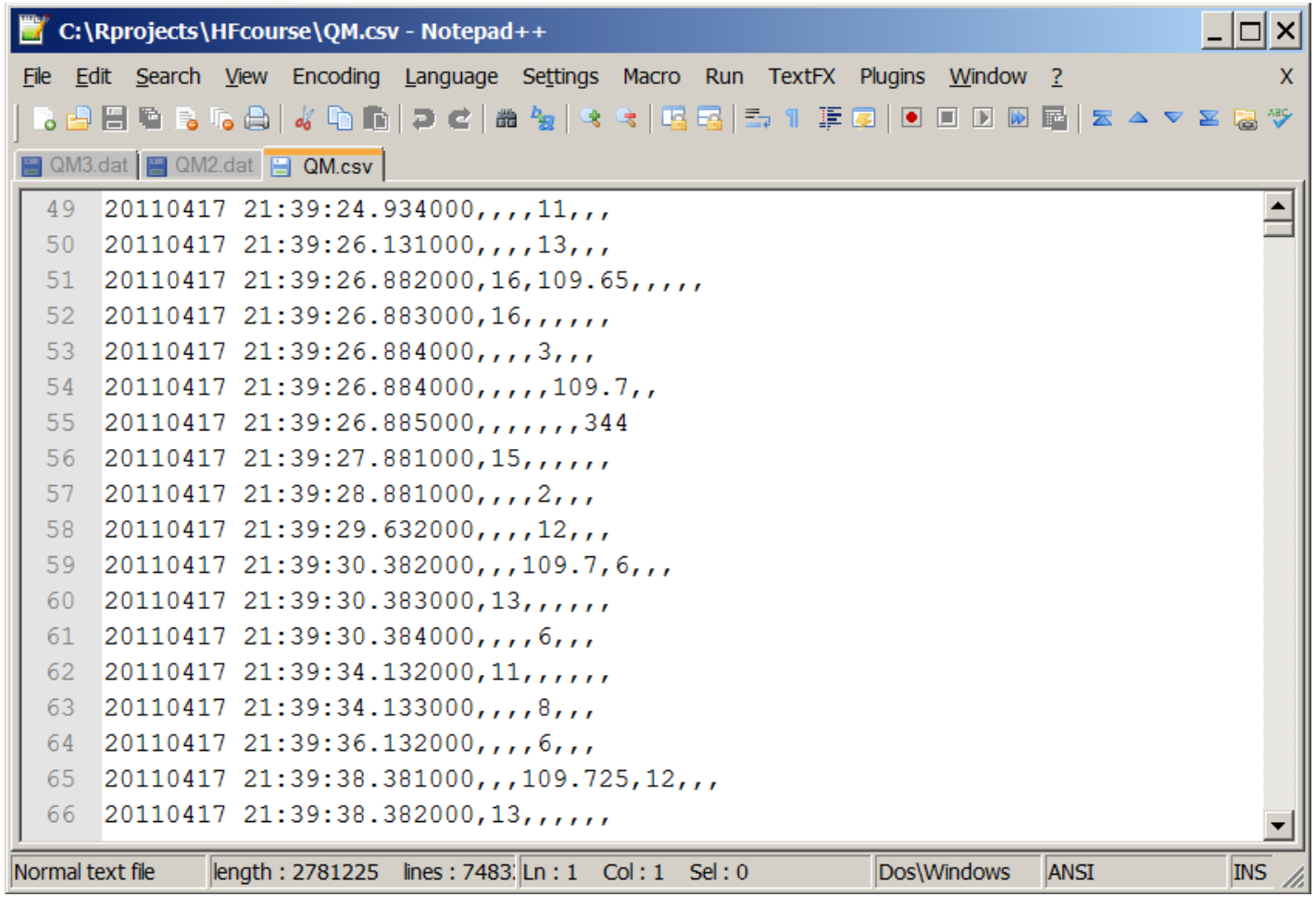
Trade-related IB messages through time



Logging IB data

R Code: Log IB updates in csv format

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



Load IB log file

R Code: Load IB log file

```
> library(IBrokers)

IBrokers version 0.9-1:
Implementing API Version 9.64

This software comes with NO WARRANTY.  Not intended for production use!
See ?IBrokers for details

> options(digits.secs = 3)
> options(digits=12)
> library(xts)
> dat <- read.table(file=paste(data.path,"QM.csv",sep=""),sep=",",
   header=F,as.is=T)
> colnames(dat) <- c("TimeStamp","BidSize","Bid","Ask","AskSize","Last",
   "LastSize","Volume")
> head(dat,4)

                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
```

Create an xts object

R Code: Call strptime and xts

```
> 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" "2011-04-17 21:37:53.623" "2011-04-17 21:37:53.623"

> 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 21:37:53.623       4 109.7     NA      NA   NA       NA     NA
2011-04-17 21:37:53.623       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 21:37:53.718       9    NA     NA      NA   NA       NA     NA
```

Fill in NAs

R Code: Fill in NAs with `na.locf`

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

	BidSize	Bid	Ask	AskSize	Last	LastSize	Volume
2011-04-17 21:45:05.668	1	109.7	109.725	3	109.75	1	354
2011-04-17 21:45:06.668	1	109.7	109.725	5	109.75	1	354
2011-04-17 21:45:07.168	1	109.7	109.725	1	109.75	1	354
2011-04-17 21:45:09.418	1	109.7	109.750	18	109.75	1	354
2011-04-17 21:45:09.419	1	109.7	109.750	18	109.75	1	354
2011-04-17 21:45:10.418	1	109.7	109.725	6	109.75	1	354
2011-04-17 21:45:10.419	1	109.7	109.725	6	109.75	1	354
2011-04-17 21:45:24.764	1	109.7	109.725	4	109.75	1	354
2011-04-17 21:45:40.669	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

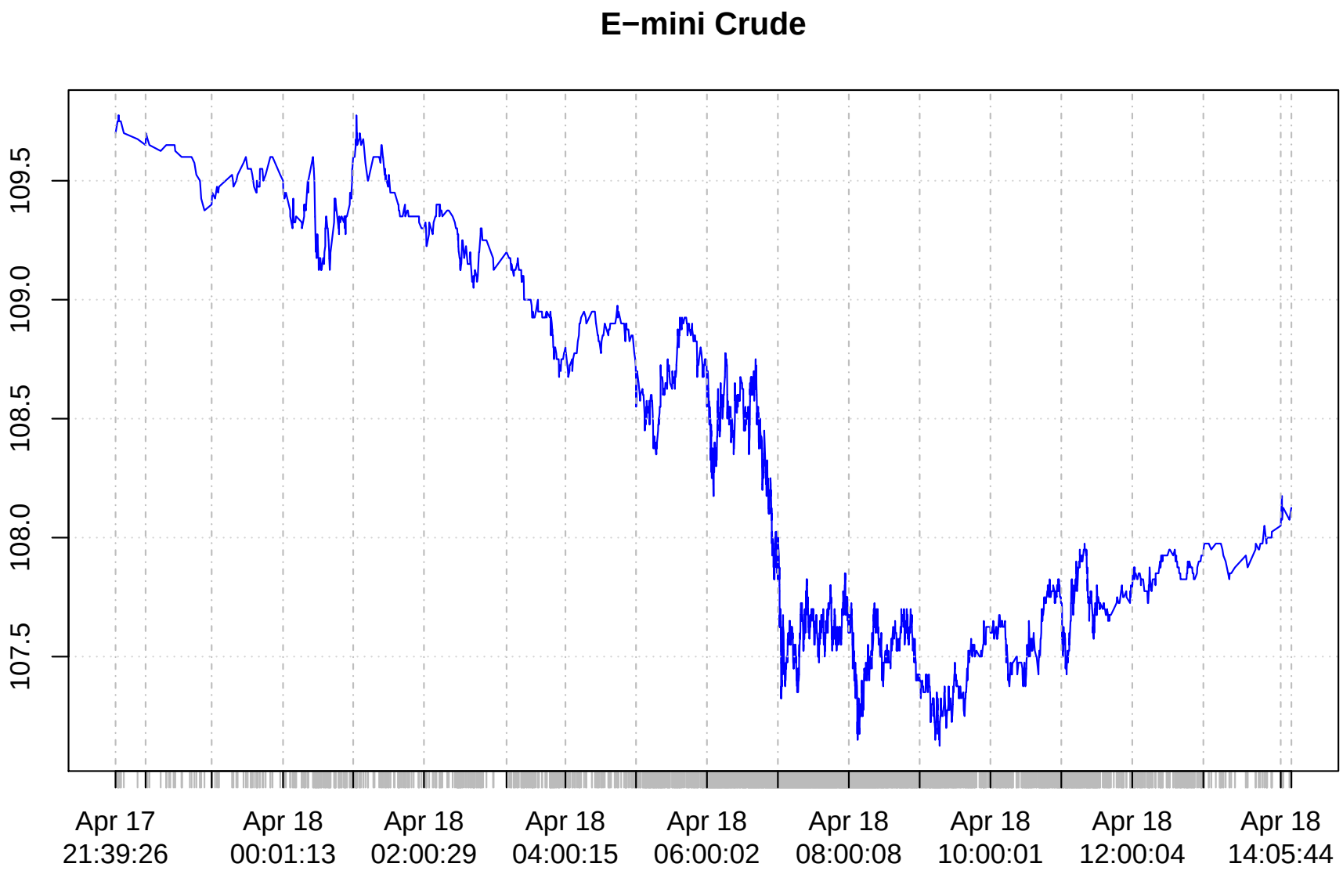
R Code: Create `xts` of trades from quotes and trades

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

```
                     BidSize     Bid     Ask AskSize    Last LastSize Volume
2011-04-17 21:39:26.885      16 109.650 109.725       3 109.700        1    344
2011-04-17 21:41:09.146       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 21:41:44.491       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
```

```
> plot(trades[,"Last"],col=4,main="E-mini Crude")
```

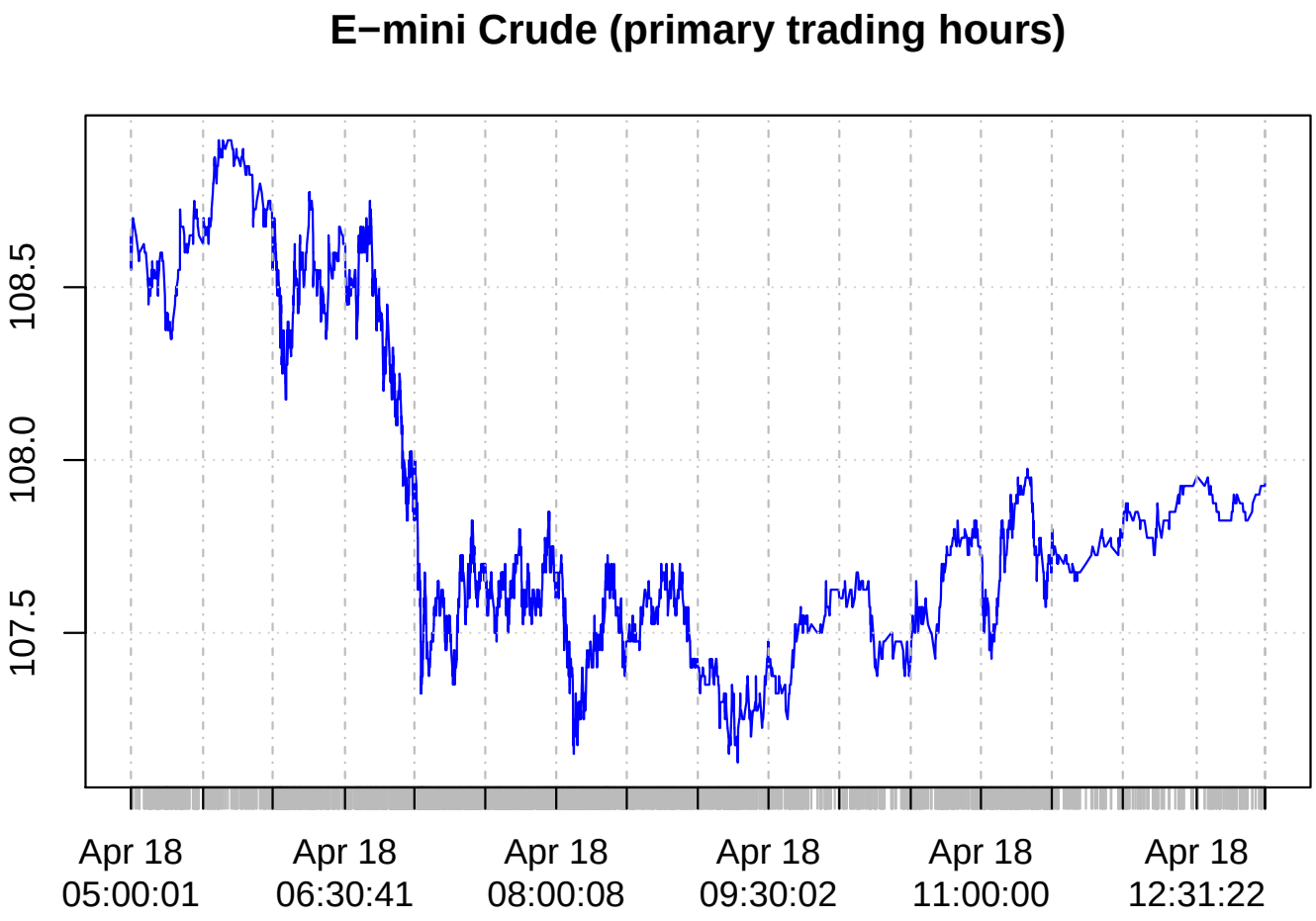
Plot of trades from `plot.xts`



Plot of trades - primary session

R Code: Plot during primary session via `xts` subsetting

```
> plot(trades['2011-04-18 05:00::2011-04-18 13:00',"Last"],col=4,
   main="E-mini Crude (primary trading hours)")
```



Create 5-minute bars

R Code: Create 5-minute bars with `align.time` and `to.period`

```
> 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

R Code: Fill-in gaps with `merge` and `endpoints`

```
> 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

R Code: 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
```

The quantmod package

Description

- Quantitative financial modelling framework

Key features

- data download (yahoo, FRED, google, oanda)
- fancy plotting
- technical analysis indicators

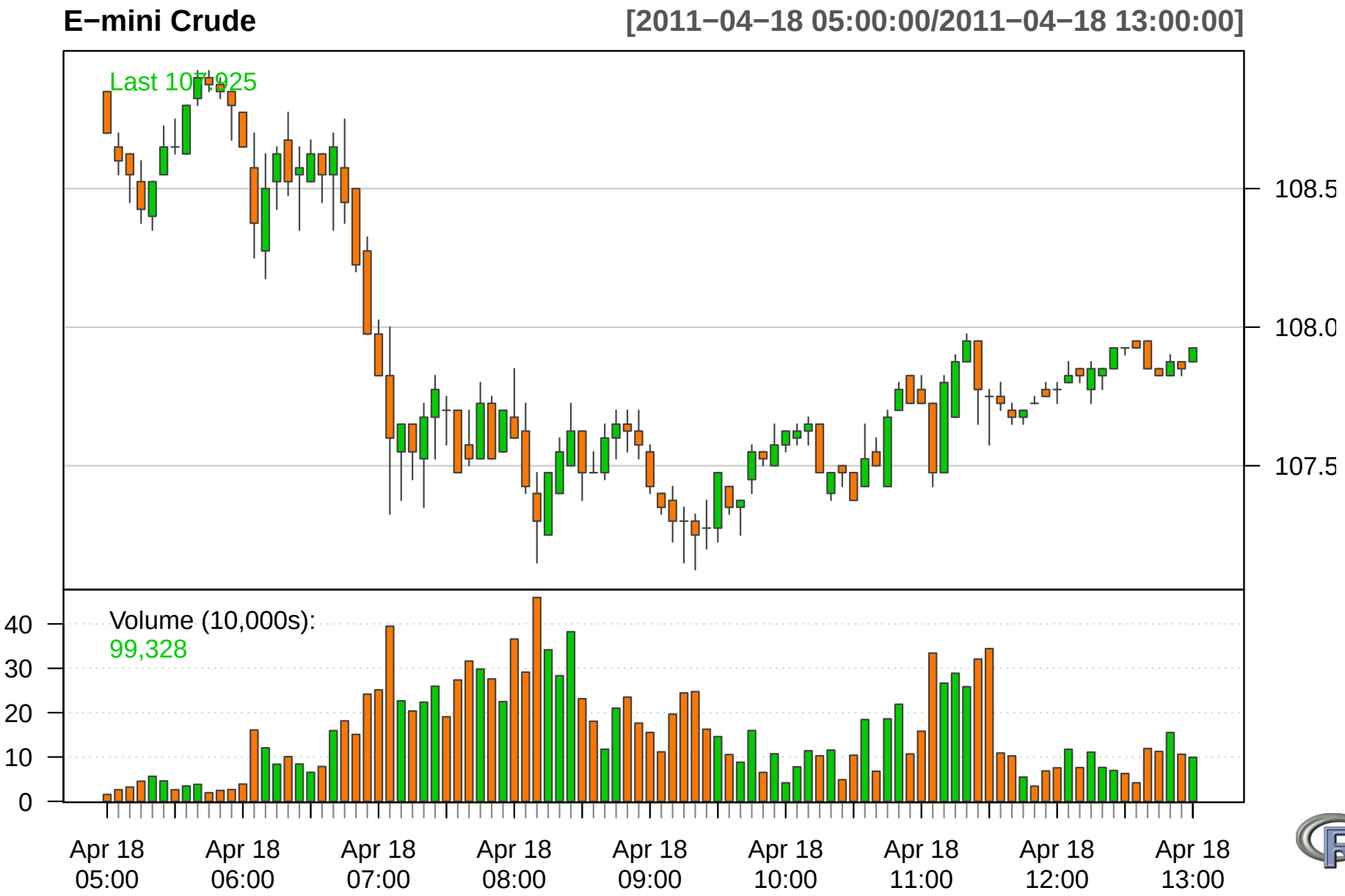
Authors

- Jeffrey Ryan

R Code: Plot candlesticks with `chartSeries` from `quantmod`

```
> library(quantmod)
> chartSeries(trades.ohlc['2011-04-18 05:00::2011-04-18 13:00',],
   theme=chartTheme("white",bg.col=0),name="E-mini Crude")
```

Candlestick plot from chartSeries



Outline

Trade and Quote (TAQ) database

The Trade and Quote (TAQ) database is a collection of intraday trades and quotes for all listed securities

- New York Stock Exchange
- American Stock Exchange
- Nasdaq National Market System
- SmallCap issues

Research facilitated by historical tick by tick data dating back to 1993

- intraday trading strategies
- liquidity and volatility measures
- analysis of market microstructure

The RTAQ package

Description

- RTAQ provides a suite of tools for the analysis of trades and quotes in the R environment

Key features

- R interface to TAQ data files
- data cleaning
- matching trades and quotes
- liquidity measures
- volatility measures

Authors

- Kris Boudt
- Jonathan Cornelissen

TAQ trade file format

Table 1:
Elements of raw trade data

SYMBOL	The stock's ticker
DATE	Date on which the trade was registered the function convert (section 2.1.1) assumes dd/mm/yyyy format
EX	Exchange on which the trade occurred see section 3.1.2
TIME	Time on which trade was registered function convert (section 2.1.1) assumes hh:mm:ss format
PRICE	Transaction price
SIZE	Number of shares traded
COND	Sales condition code
CORR	Correction indicator
G127	Combined "G", Rule 127, and stopped stock trade indicator

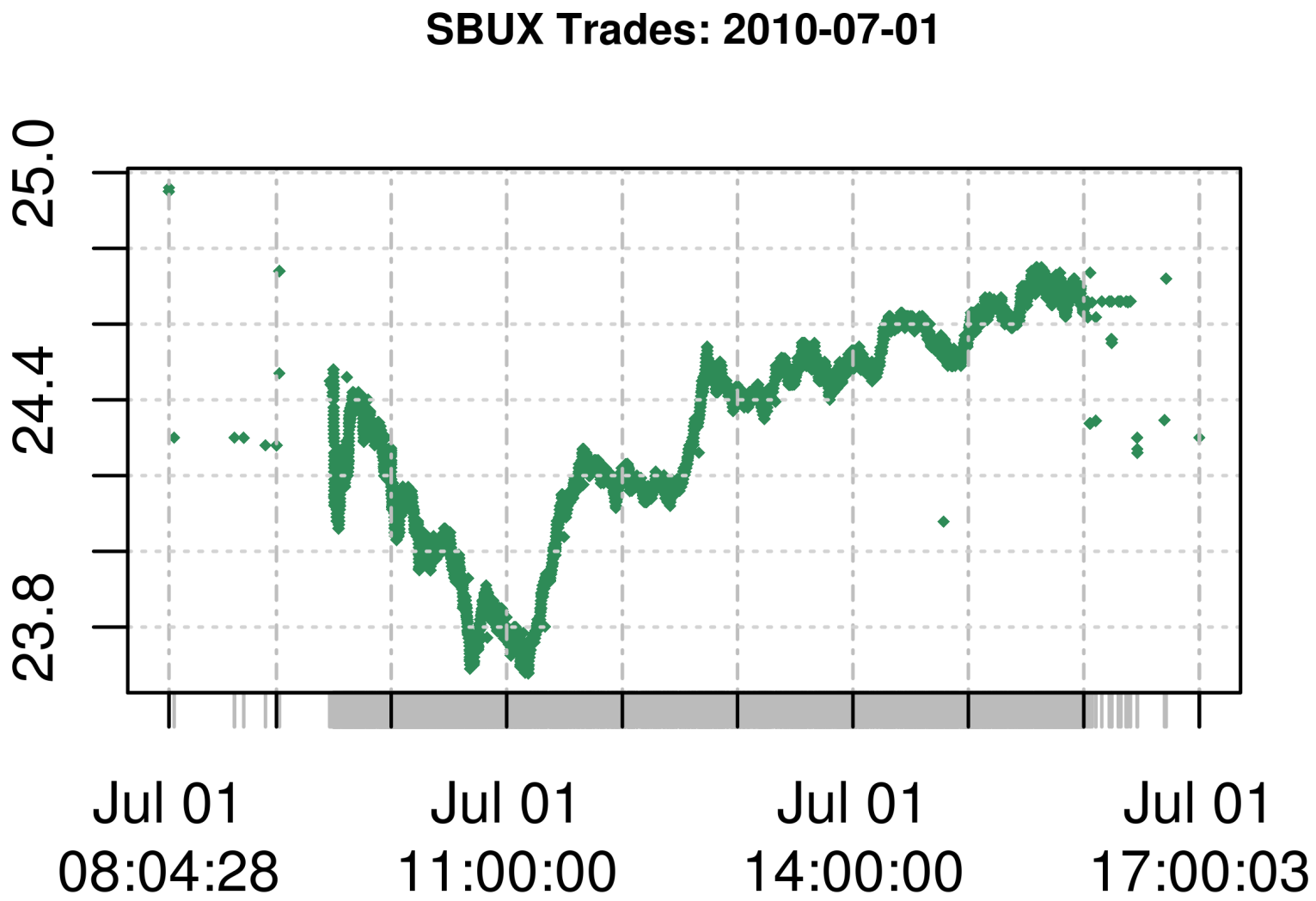
see RTAQ package vignette, J. Cornelissen and K. Boudt

Load TAQ trade data

R Code: Load TAQ trade data

```
> library(RTAQ)
> ticker <- "SBUX"
> TAQ.path <- "TAQRdata"
> sbux.t <- TAQLoad(tickers=ticker,from="2010-07-01",to="2010-07-01",
   trades=T,quotes=F,datasource=TAQ.path)
> class(sbux.t)

[1] "xts" "zoo"

> dim(sbux.t)

[1] 77956     7

> head(sbux.t,3)

                     SYMBOL EX  PRICE     SIZE     COND CORR G127
2010-07-01 08:04:28 "SBUX" "Q" "24.9500" "   100" "T"  "0"  "0"
2010-07-01 08:04:28 "SBUX" "Q" "24.9500" "   100" "T"  "0"  "0"
2010-07-01 08:04:28 "SBUX" "Q" "24.9600" "   300" "T"  "0"  "0"

> plot(sbux.t[,3],main=paste(ticker,"Trades:",as.Date(start(sbux.t))),type="p",
   cex.axis=0.25,cex.main=0.75,cex.lab=0.25,major.tick="hours",pch=18,cex=0.5,
   col="seagreen")
```

Raw TAQ trade data



TAQ quote file format

Table 2:
Elements of raw quote data

SYMBOL	The stock's ticker
DATE	Date on which the trade was registered the function convert (section 2.1.1) assumes dd/mm/yyyy format
EX	Exchange on which the quote occurred. see section 3.1.2
TIME	Time on which trade was registered function convert (section 2.1.1) assumes hh:mm:ss format
BID	Bid price
BIDSIZ	Bid size in number of round lots (100 share units)
OFR	Offer price
OFRSIZ	Offer size in number of round lots (100 share units)
MODE	Quote condition

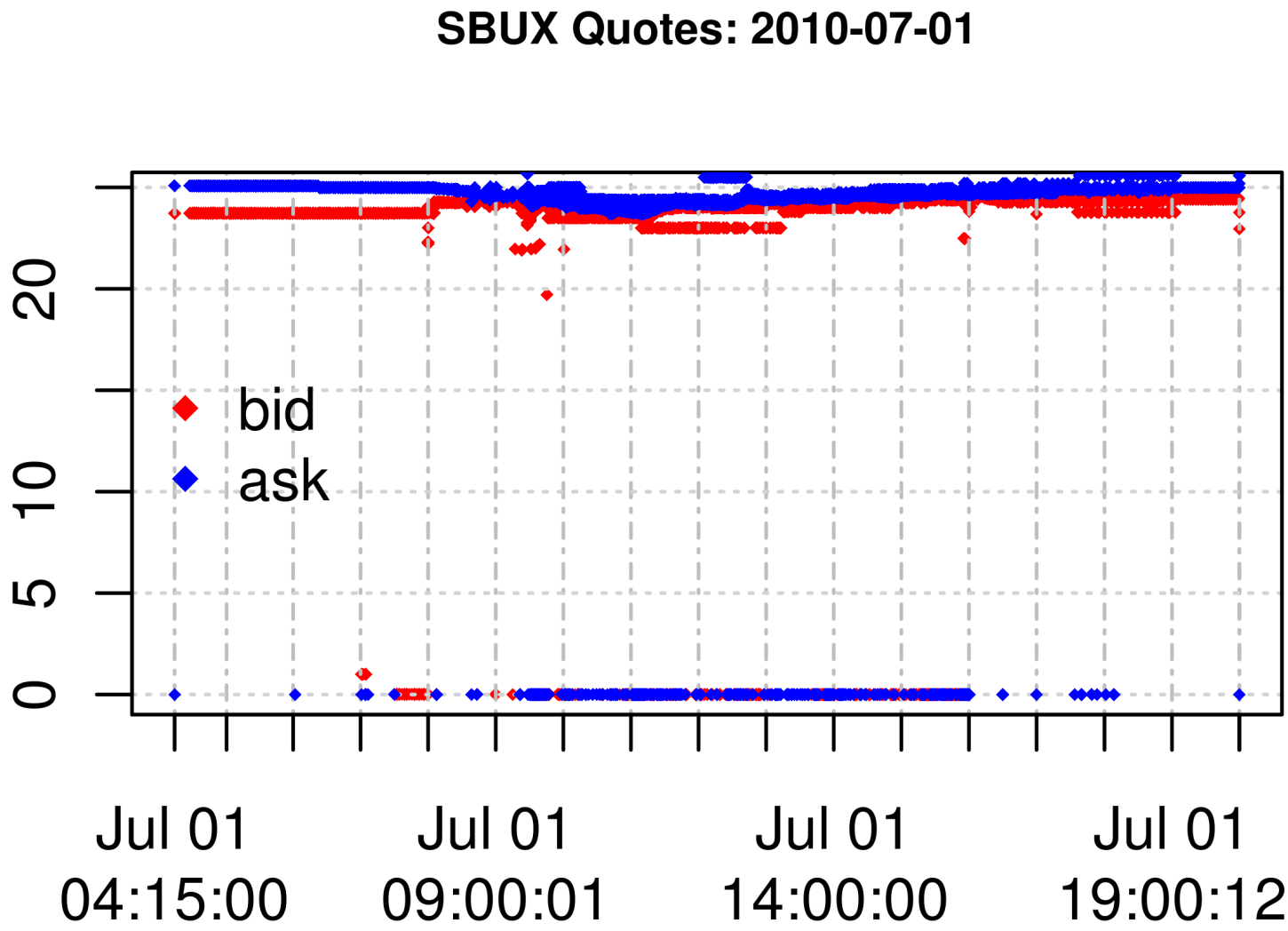
see RTAQ package vignette, J. Cornelissen and K. Boudt

Load TAQ quote data

R Code: Load TAQ quote data

```
> sbux.q <- TAQLoad(tickers="SBUX",from="2010-07-01",to="2010-07-01",
   trades=F,quotes=T,datasource=TAQ.path)
> class(sbux.q)

[1] "xts" "zoo"

> dim(sbux.q)

[1] 1093706       7

> head(sbux.q,4)

                    SYMBOL EX  BID       BIDSIZ OFR         OFRSIZ MODE
2010-07-01 04:15:00 "SBUX" "P" "23.7200" "   3" "     0.00" "  0"  "12"
2010-07-01 04:15:00 "SBUX" "P" "23.7200" "   3" "    27.99" "  1"  "12"
2010-07-01 04:15:00 "SBUX" "P" "23.7200" "   3" "    25.08" "  2"  "12"
2010-07-01 04:28:27 "SBUX" "P" "23.7300" "   5" "    25.08" "  2"  "12"

> plot(sbux.q[,3],main=paste(ticker,"Quotes:",as.Date(start(sbux.t))),
   major.tick="hours",minor.ticks=F,type="p",pch=18,cex=0.5,col=2,
   cex.main=0.75)
> points(x=index(sbux.q),y=sbux.q[,5],pch=18,cex=0.5,col=4)
> legend(x="left",legend=c("bid","ask"),pch=18,col=c(2,4),bty="n")
```

Raw TAQ quote data



RTAQ data cleaning functions

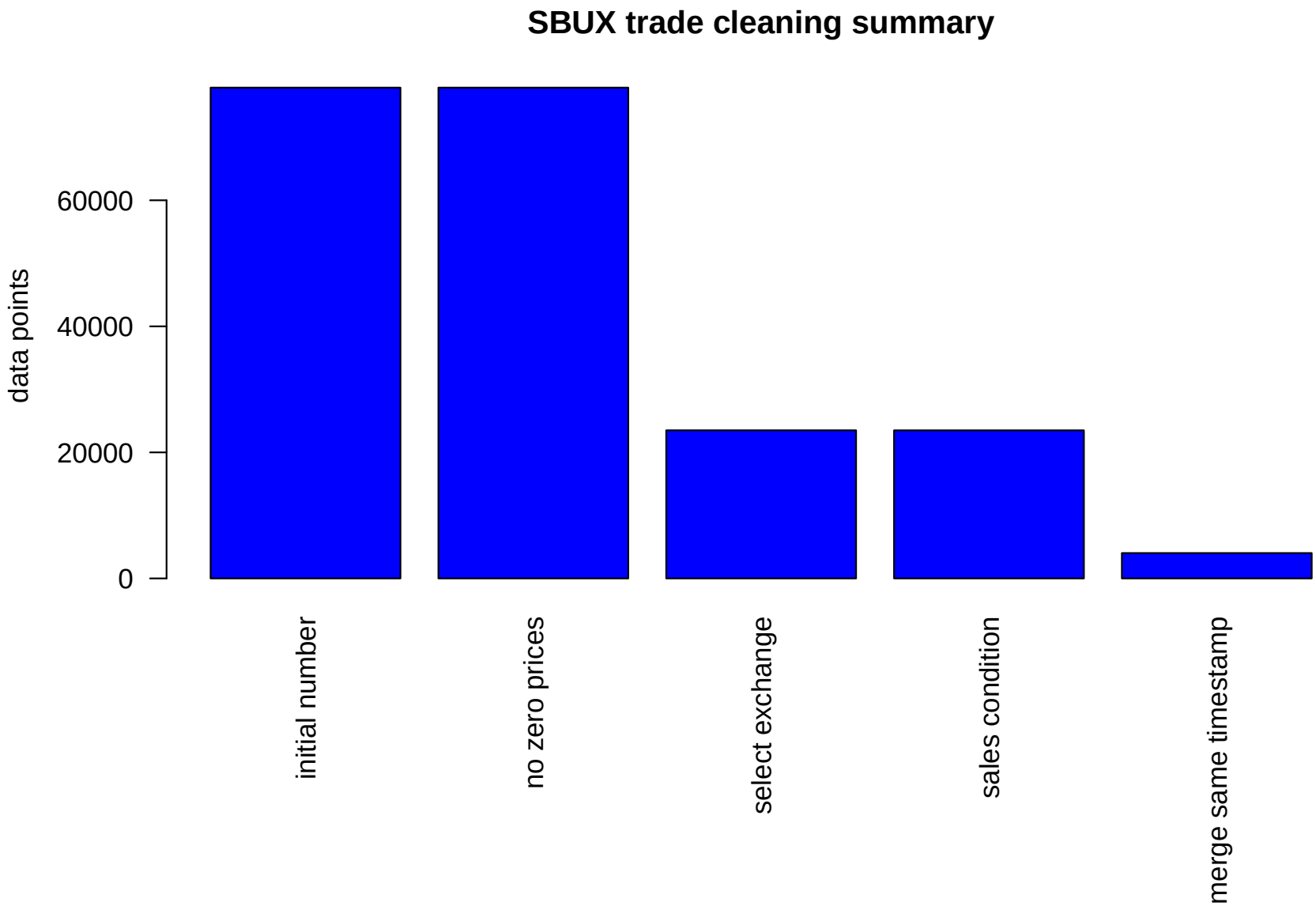
function	Description
all data	
exchangeHoursOnly	retain data for exchange hours only
selectExchange	retain only data from a single stock exchange
trade data	
tradesCleanup	cleans trade data
noZeroPrices	delete the observations where the price is zero
salesCondition	delete entries with abnormal sale condition
mergeTradesSameTimestamp	merge multiple transactions with the same time stamp
rmTradeOutliers	delete transactions with unlikely transaction prices
tradesCleanupFinal	perform a final cleaning procedure on trade data
quote data	
noZeroQuotes	delete the observations where the bid or ask is zero
rmLargeSpread	delete entries for which the spread is more than threshold
mergeQuotesSameTimestamp	merge multiple quote entries with the same time stamp
rmOutliers	delete entries for which the mid-quote is outlying

Clean trade data

R Code: Clean trade data and plot report

```
> nrow(sbux.t)

[1] 77956

> tdata <- exchangeHoursOnly(sbux.t)
> tdata.c <- tradesCleanup(tdataraw=tdata,exchanges="Q")
> tdata <- tdata.c$tdata
> nrow(tdata)

[1] 4023

> op <- par(mar=c(11,7,4,2))
> barplot(tdata.c$report,las=2,col=4,
   main=paste(ticker,"trade cleaning summary"))
> mtext("data points",side=2,line=4)
> par(op)
```

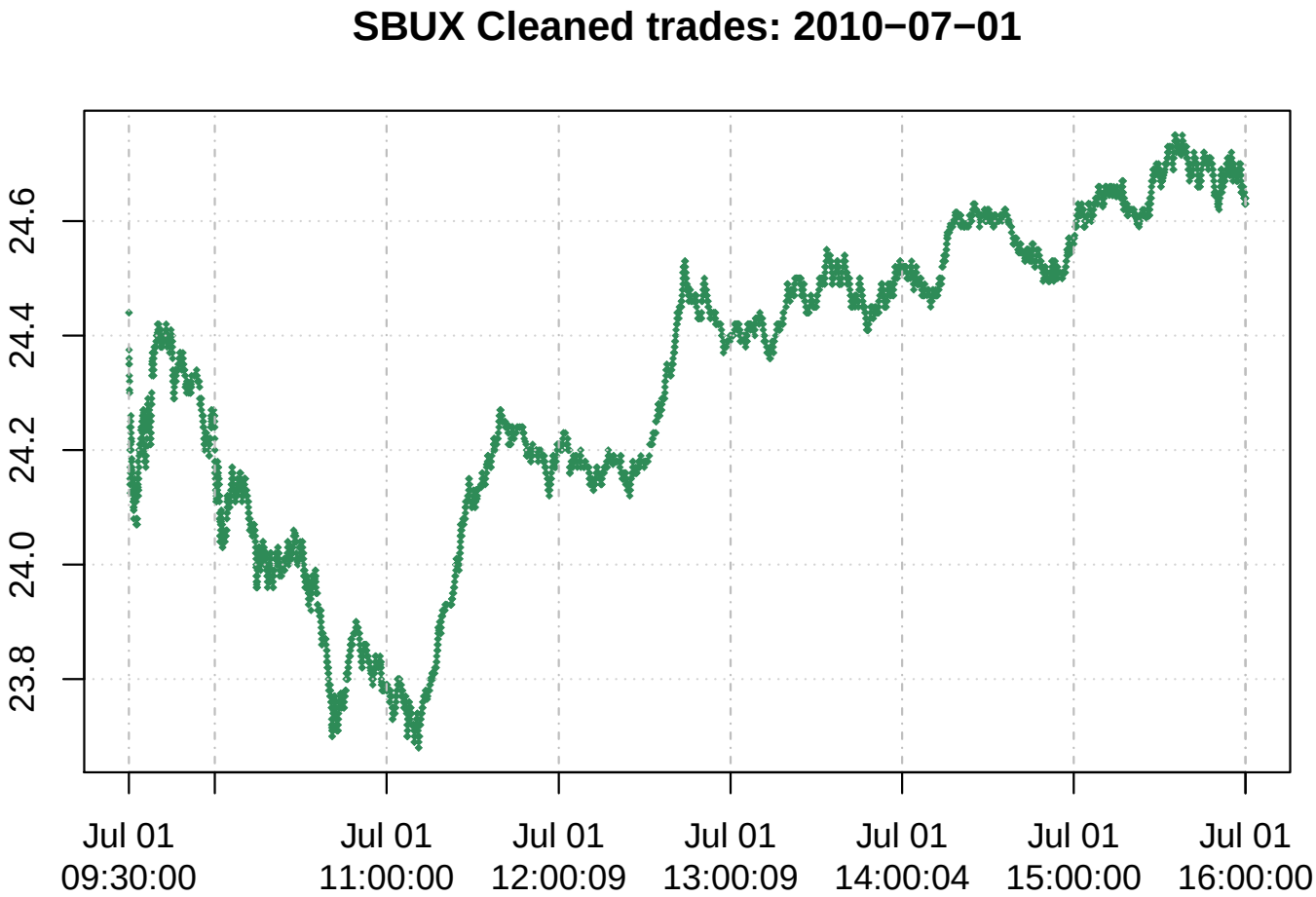
Trade cleaning report



Clean trades

R Code: Plot time series of clean trades

```
> plot(tdata[,3],main=paste(ticker,"Cleaned trades:",as.Date(start(sbux.t))),
   major.tick="hours", minor.ticks=F,type="p",pch=18,cex=0.5,col="seagreen")
```

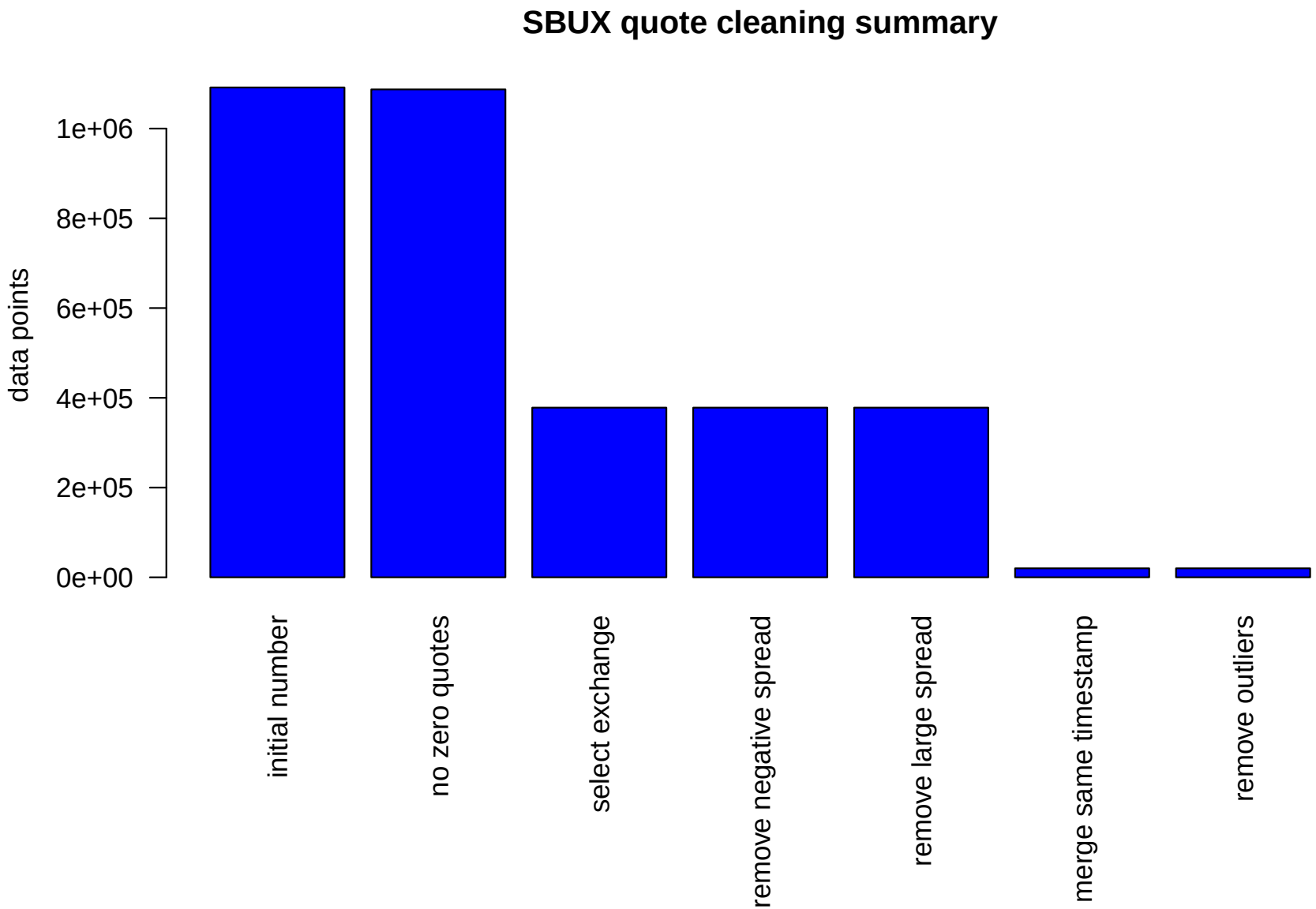


Clean quote data

R Code: Clean trade data and plot report

```
> nrow(sbux.q)

[1] 1093706

> qdata <- exchangeHoursOnly(sbux.q)
> qdata.c <- quotesCleanup(qdataraw=qdata,exchanges="T")
> qdata <- qdata.c$qdata
> nrow(qdata)

[1] 19994

> op <- par(mar=c(11,7,4,2))
> barplot(qdata.c$report,las=2,col=4,
   main=paste(ticker,"quote cleaning summary"))
> mtext("data points",side=2,line=4)
> par(op)
```

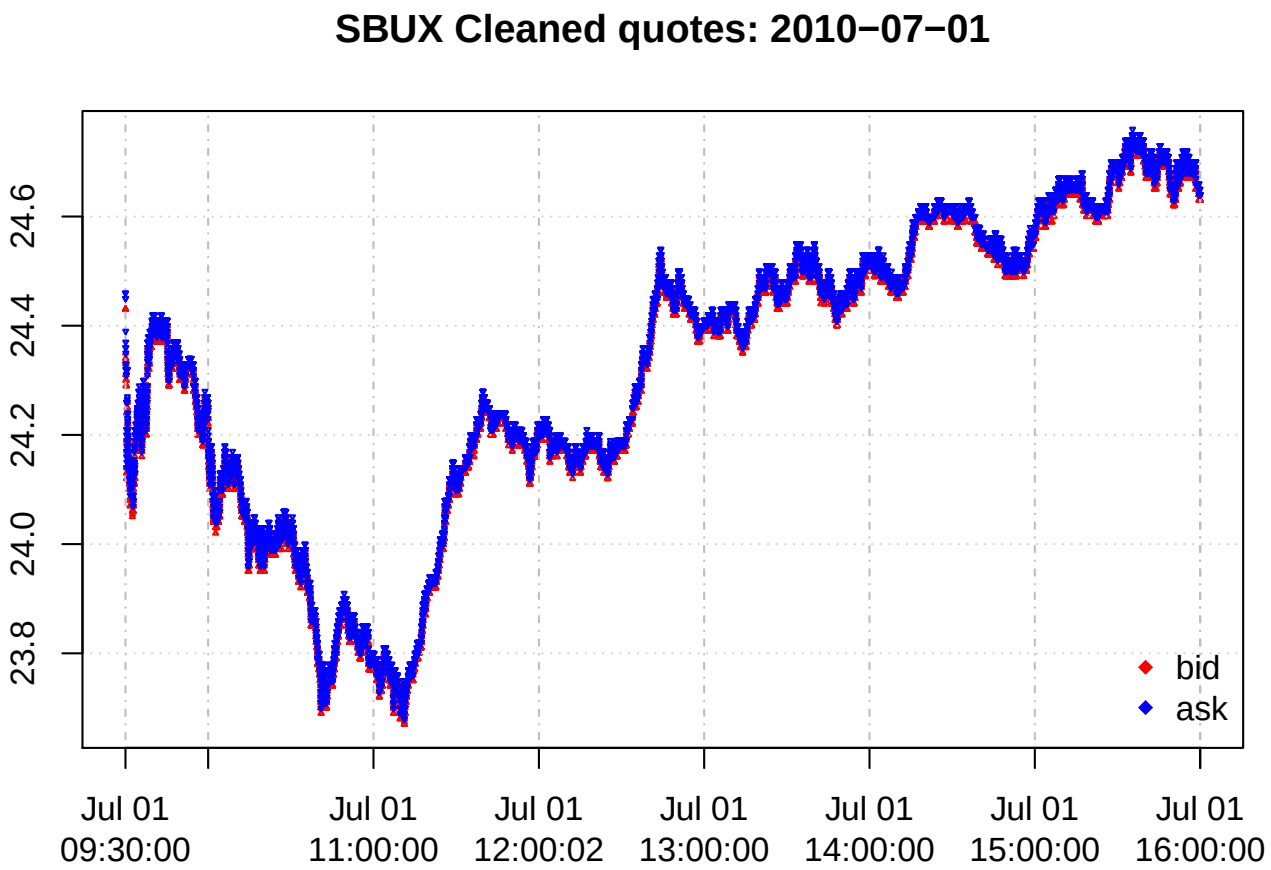
Quote cleaning report



Clean trades

R Code: Plot time series of clean quotes

```
> plot(qdata[,3],main=paste(ticker,"Cleaned quotes:",as.Date(start(sbux.t))),
   major.tick="hours", minor.ticks=F,type="p",pch=24,cex=0.25,col=2)
> points(x=index(qdata),y=qdata[,5],pch=25,cex=0.25,col=4)
> legend(x="bottomright",legend=c("bid","ask"),pch=18,col=c(2,4),bty="n")
```

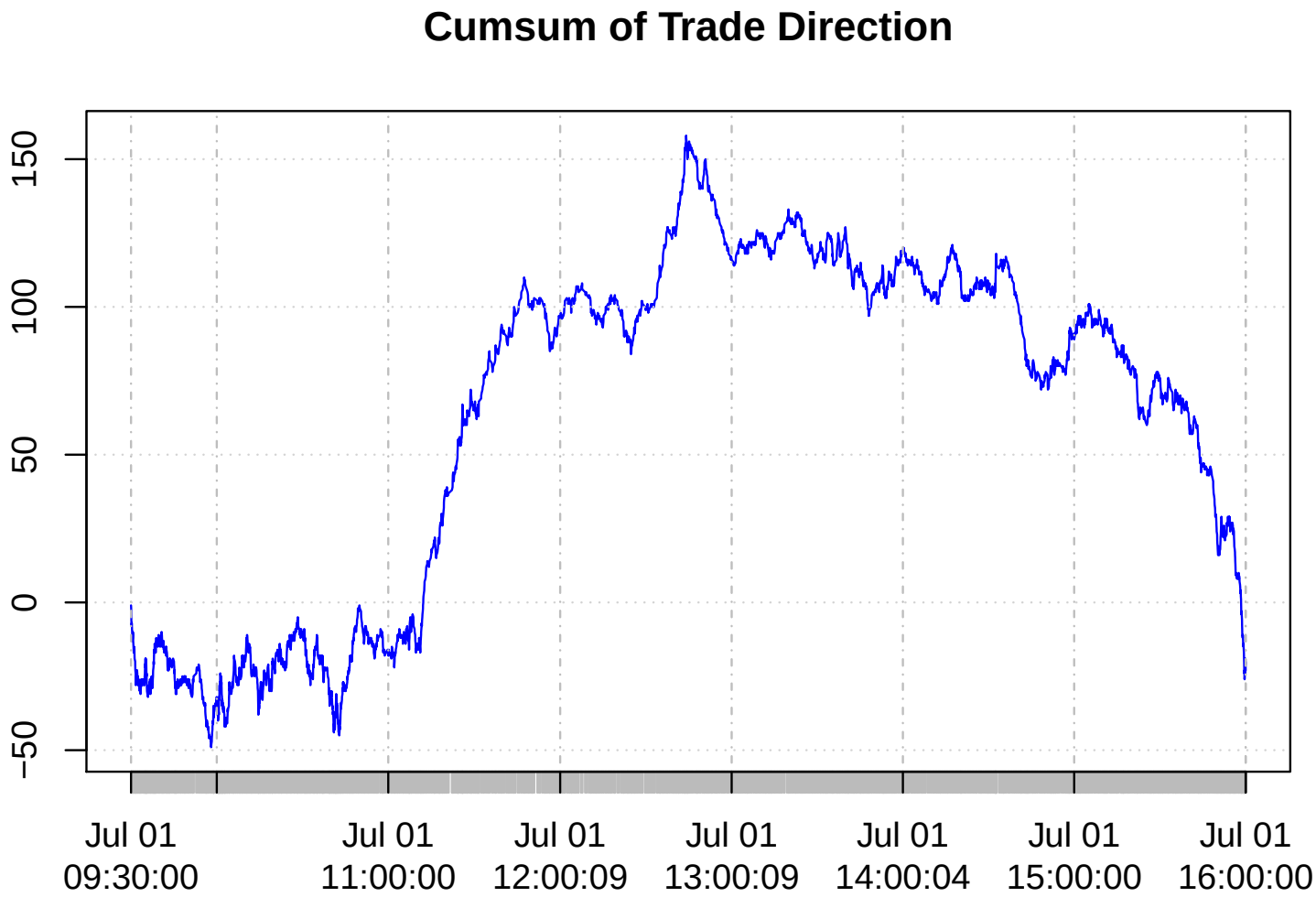


Match trades/quotes and calculate trade direction

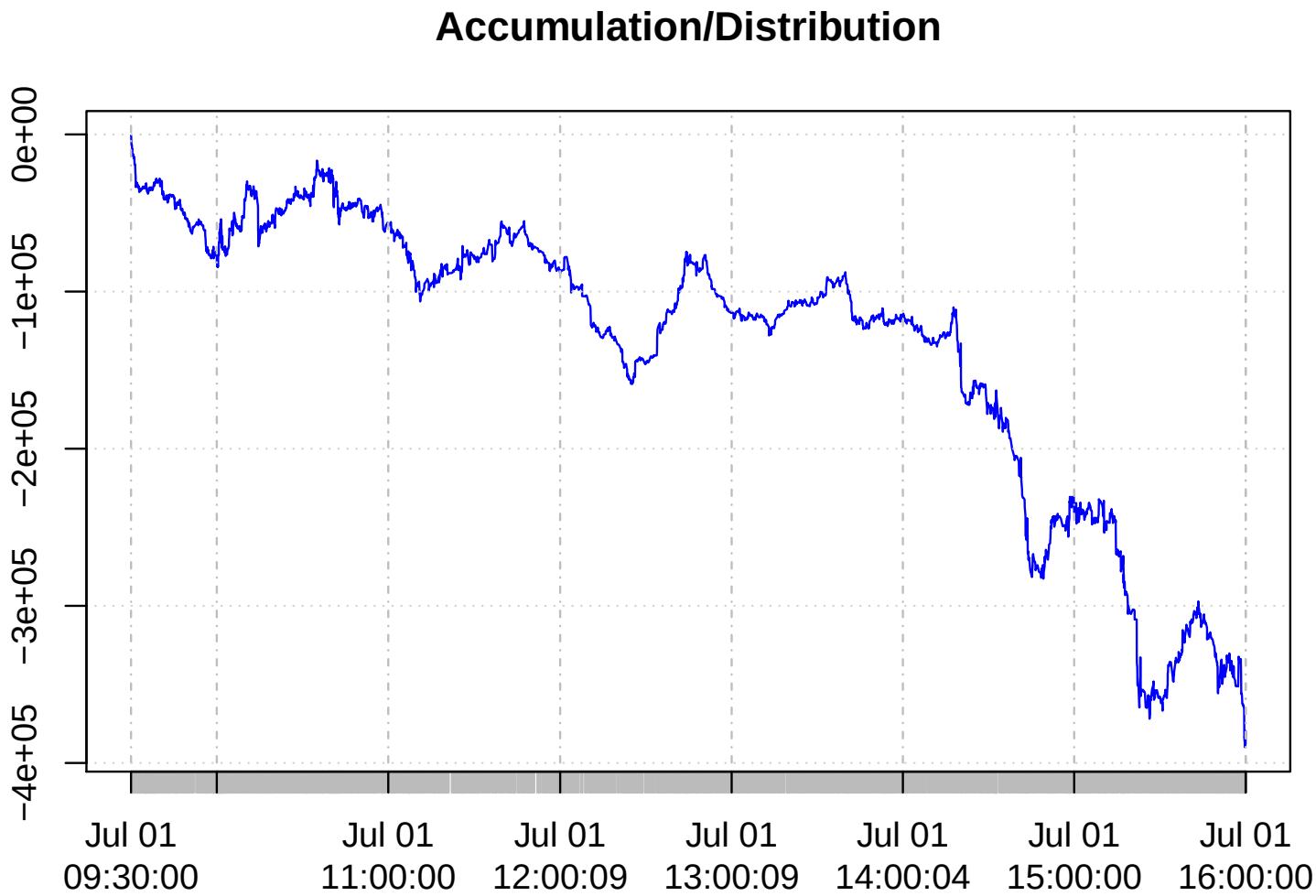
R Code: Match trades/quotes and calculate trade direction

```
> tqdata <- matchTradesQuotes(tdata,qdata)
> head(tqdata,2)

                     SYMBOL EX  PRICE   SIZE  COND CORR G127 BID     BIDSIZ
2010-07-01 09:30:00 "SBUX" "Q" "24.44" "700" "@F" "0"  "0"  "24.43" "13"
2010-07-01 09:30:03 "SBUX" "Q" "24.44" "462" ""   "0"  "0"  "24.43" "13"
                     OFR     OFRSIZ MODE
2010-07-01 09:30:00 "24.46" "41"   "12"
2010-07-01 09:30:03 "24.46" "41"   "12"

> tdir <- getTradeDirection(tqdata)
> head(tdir,25)

 [1] -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1  1 -1 -1  1  1  1 -1 -1 -1 -1 -1

> tdx <- xts(tdir,index(tdata))
> tdvx <- tdata[,"SIZE"]*tdx
> accDist <- cumsum(tdvx)
> plot(cumsum(tdx),main="Cumsum of Trade Direction",col=4,major.ticks="hours")

> plot(accDist,main="Accumulation/Distribution",col=4,major.ticks="hours")
```

Trade direction pattern



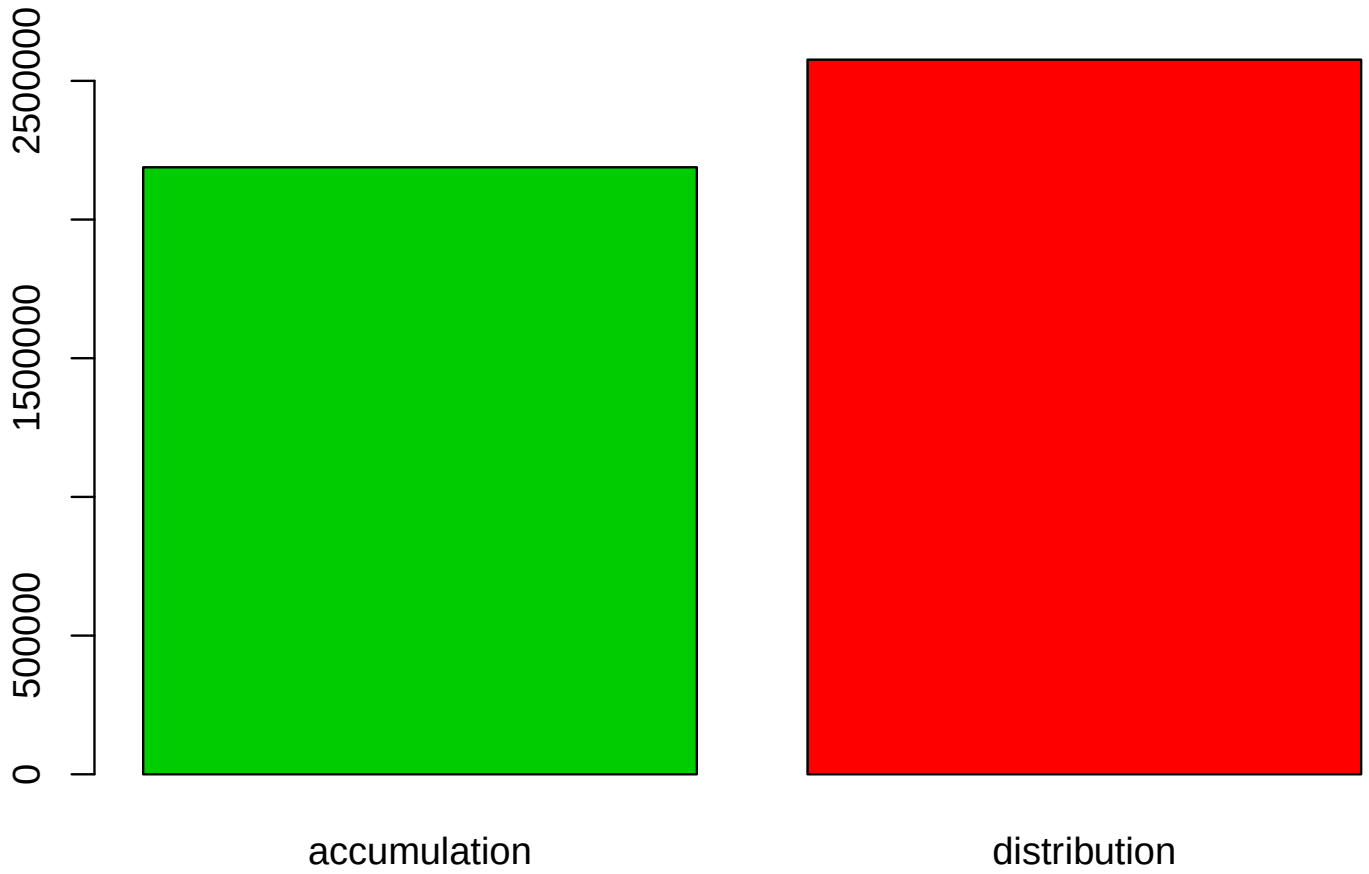
Time series of Accumlation/Distribution



Comparison of Accumulation and Distribution

R Code: Accumulation and distribution bar chart

```
> barplot(c(sum(tdvx[tdvx>0]),-sum(tdvx[tdvx<0])),col=c(3,2),
   names.arg=c("accumulation","distribution"))
```



Outline

Realized Variance

- Introduction to Realized Variance
- Estimation Problems
- Estimation Solutions and Tuning
- Example: SBUX Realized Vol vs Implied Vol

NOTE: I have left out citations, however, let me know if you want to be pointed to papers.

Goal: Estimate Realized Variance

Why Estimate Variance?

- Manage Risk
- Trading Strategies
- Portfolio Optimization
- Option Pricing
- Market Sentiment

Realized Variance is a one day calculation of variance that uses tick data. In addition to a different calculation method, it also has a slightly different the interpretation than traditional estimates.

Traditional Variance Estimation

Different Volatility Estimates:

- Unconditional
- Unconditional Rolling Window
- Conditional (GARCH)
- Factor Models (BARRA)

Possible Issues:

- Lack of parsimony
- Today's volatility estimate depends on more than today's data
- A large spike in the past creates a sustained upwards estimate, this bias disappears when the spike falls out of the estimation window.
- Uses only one price per day (usually close price)

Realized Variance Motivation

Advantages:

- Realized Variance for today is based on today's market only
- Theory tells us that RV $\rightarrow$ Integrated Variance as sampling frequency increases.
- Quotes or trades can be used

Issues:

- 20-50% of volatility is overnight
- Expensive to compute
- Not smooth $\rightarrow$ forecast not obvious
- Practice vs Theory

Sampling Frequency

Define

M is the total number of m length returns in the day.

Example (using a NYSE trading day):

- One Minute Returns: $m = mins$ and $M = 450$
- 30 Second Returns: $m = 30sec$ and $M = 900$
- 1 Second Returns: $m = 1sec$ and $M = 27000$

Frequency?

Higher frequency $\rightarrow$ higher M over a fixed period.

Realized Variance Theory

If true prices follow the following process:

$$dp^* = \sigma(t)dw(t)$$

Realized Variance:

$$RV_*^{(m)} = \sum_{i=1}^{M} r_{i,m}^{*2}$$

converges to Integrated Variance, $\sigma(t)^2$, as $M \to \infty$.

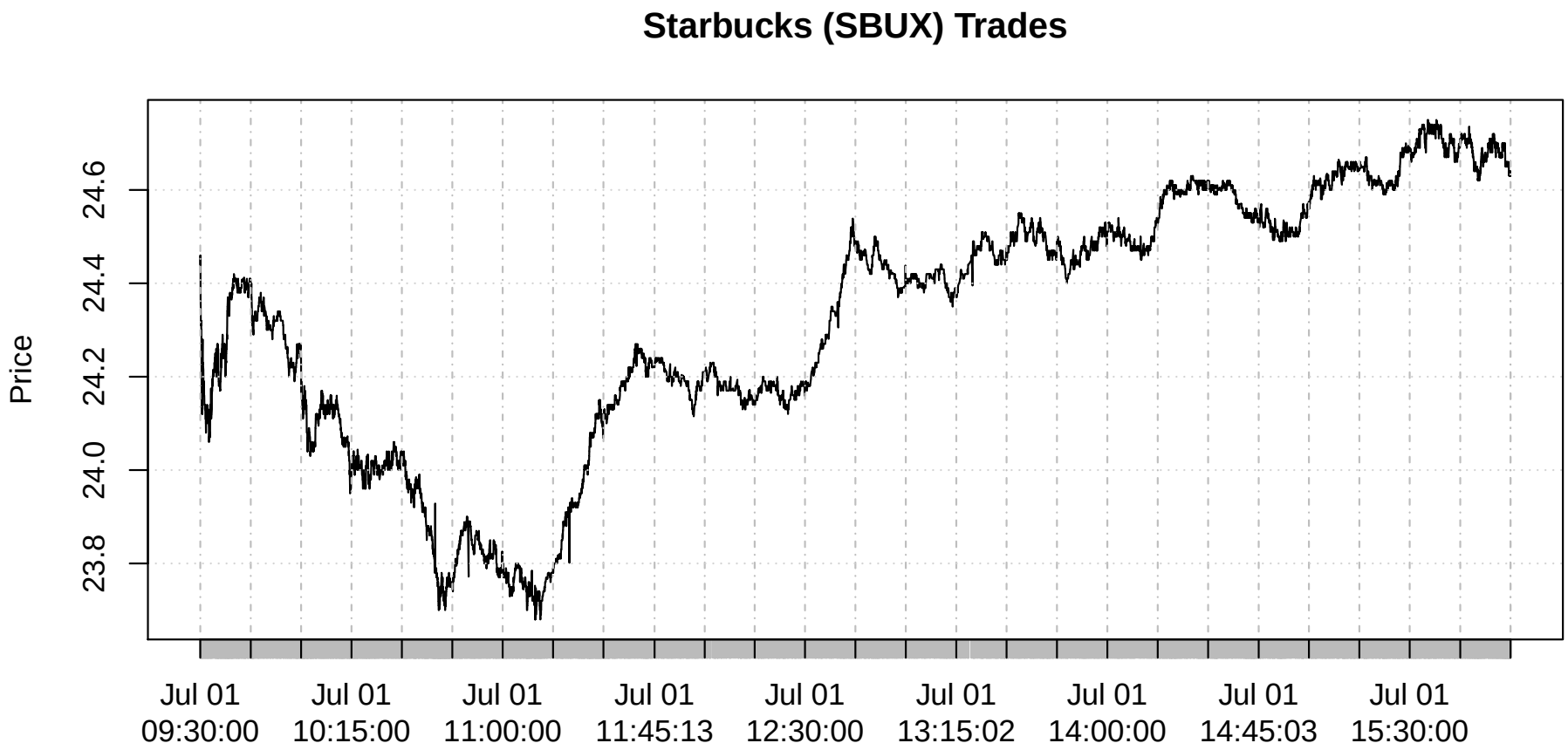
For example:

$$RV_*^{(mins)} = \sum_{i=1}^{450} r_{i,mins}^{*2}, \quad RV_*^{(30secs)} = \sum_{i=1}^{900} r_{i,30secs}^{*2}$$

Starbucks Trades, July 01 2010

R Code: Starbucks Trade Prices

```
> shared.dir <- "/shared/scott.payseur@gmail.com/RFinance_Presentation"
> load(paste(shared.dir,"data/sbux.20110701.rda", sep="/"))
> plot(sbux.20110701.xts[,"PRICE"], main="Starbucks (SBUX) Trades", ylab="Price")
```



Realized Package

Realized Variance Package (Available on CRAN)

- Author: Scott Payseur
- Calculation of realized variance, covariance and correlation
- Naive, sub-sampling, kernel, Hayashi-Yoshida
- Graphical tools

R Code: Load Realized Library

```
> library(realized)

Realized Library:  Realized Variance, Covariance, Correlation Estimation and Tools.
 R:  http://cran.r-project.org
Copyright (C) 2010  Scott W. Payseur <scott.payseur@gmail.com>

Please report any bugs or feature requests to the author.  User's manual
in the 'realized' library directory.

> # Sourcing realized 1.0
> source(paste(shared.dir,"realized/realized.R", sep="/"))
```

Computing Naive Realized Variance

R Code: rRealizedVariance

```
> args(rRealizedVariance)

function (x, y = NULL, type = "naive", period = 1, align.by = "seconds",
    align.period = 1, cor = FALSE, rvargs = list(), cts = TRUE,
    makeReturns = FALSE, lags = NULL)
NULL
```

- x: an xts return object
- y: an xts return object (used for covariance)
- type: "naive", "kernel", ...
- period: period to use in calculation (one second)
- align.by: align using seconds, minutes, hours
- align.period: aligned to this many seconds, minutes, hours

Computing Naive Realized Variance

R Code: rRealizedVariance with one second returns

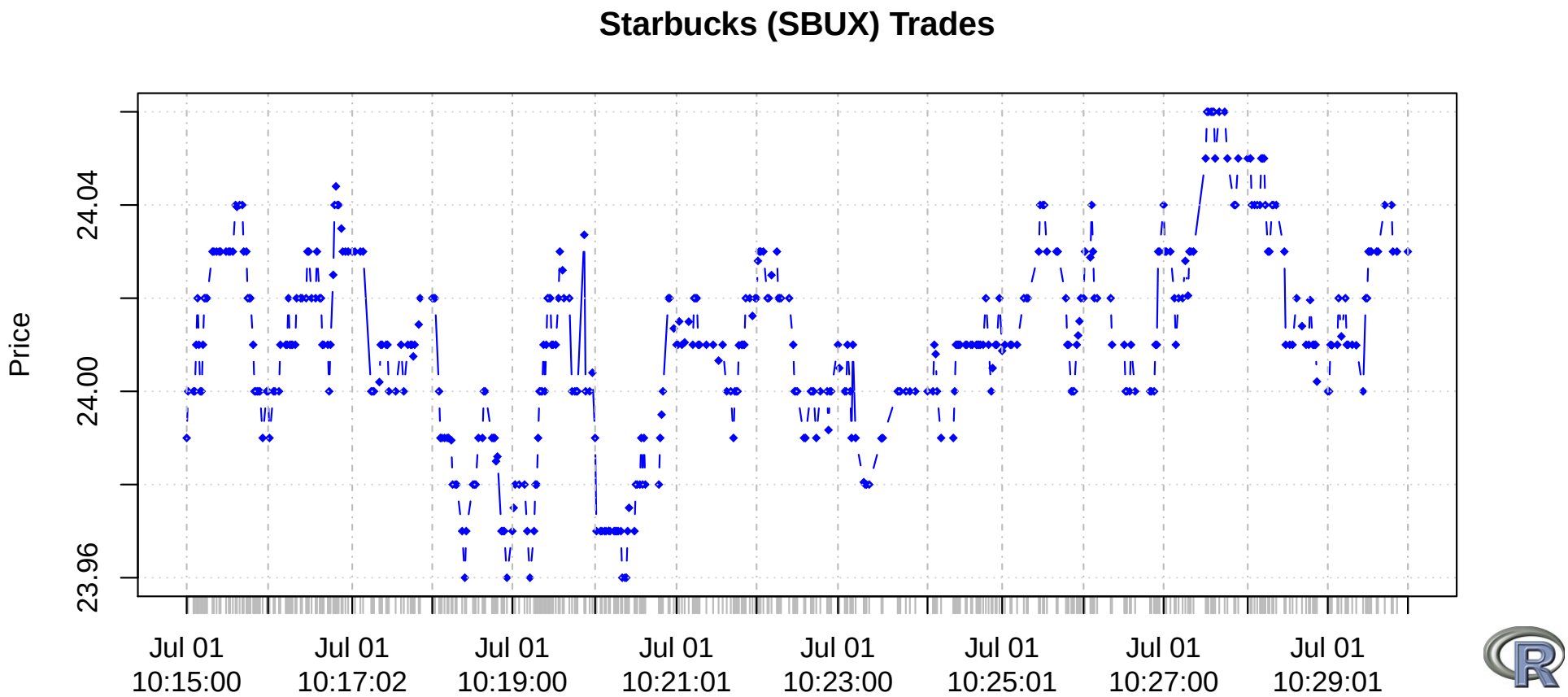
```
> sbux.rets.xts <- makeReturns(sbux.20110701.xts[,"PRICE"])
> rv <- rRealizedVariance(sbux.rets.xts,
                          period=1,
                          align.by="seconds",
                          align.period=1,
                          type="naive")
> rv

[1] 0.00103305303084

> # Annualized Vol
> sqrt(rv)*sqrt(250)*100

[1] 50.819608195
```

- Estimated at highest frequency (calendar time sampling)
- Theory says this $\rightarrow$ Integrated Variance
- We call it Naive for a reason!

Bid Ask Bounce

R Code: 15 Minutes of Starbucks Prices

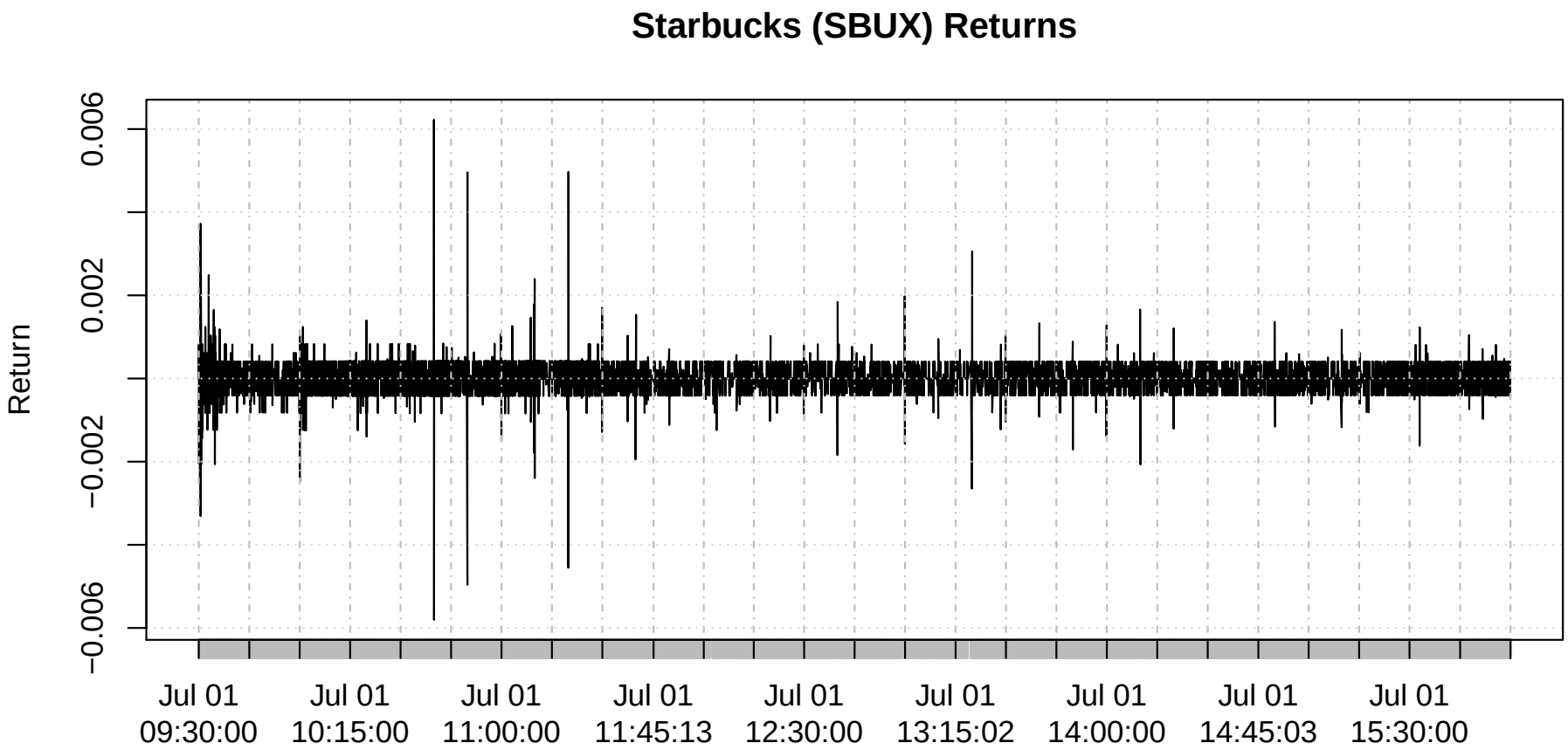
```
> Sys.setenv(TZ="GMT")
> plot(sbux.20110701.xts["2010-07-01 10:15:00::2010-07-01 10:30:00",3],pch=18,col=4
      main="Starbucks (SBUX) Trades", ylab="Price", type="b", cex=.7)
```



Bid Ask Bounce: Returns

R Code: Starbucks Returns

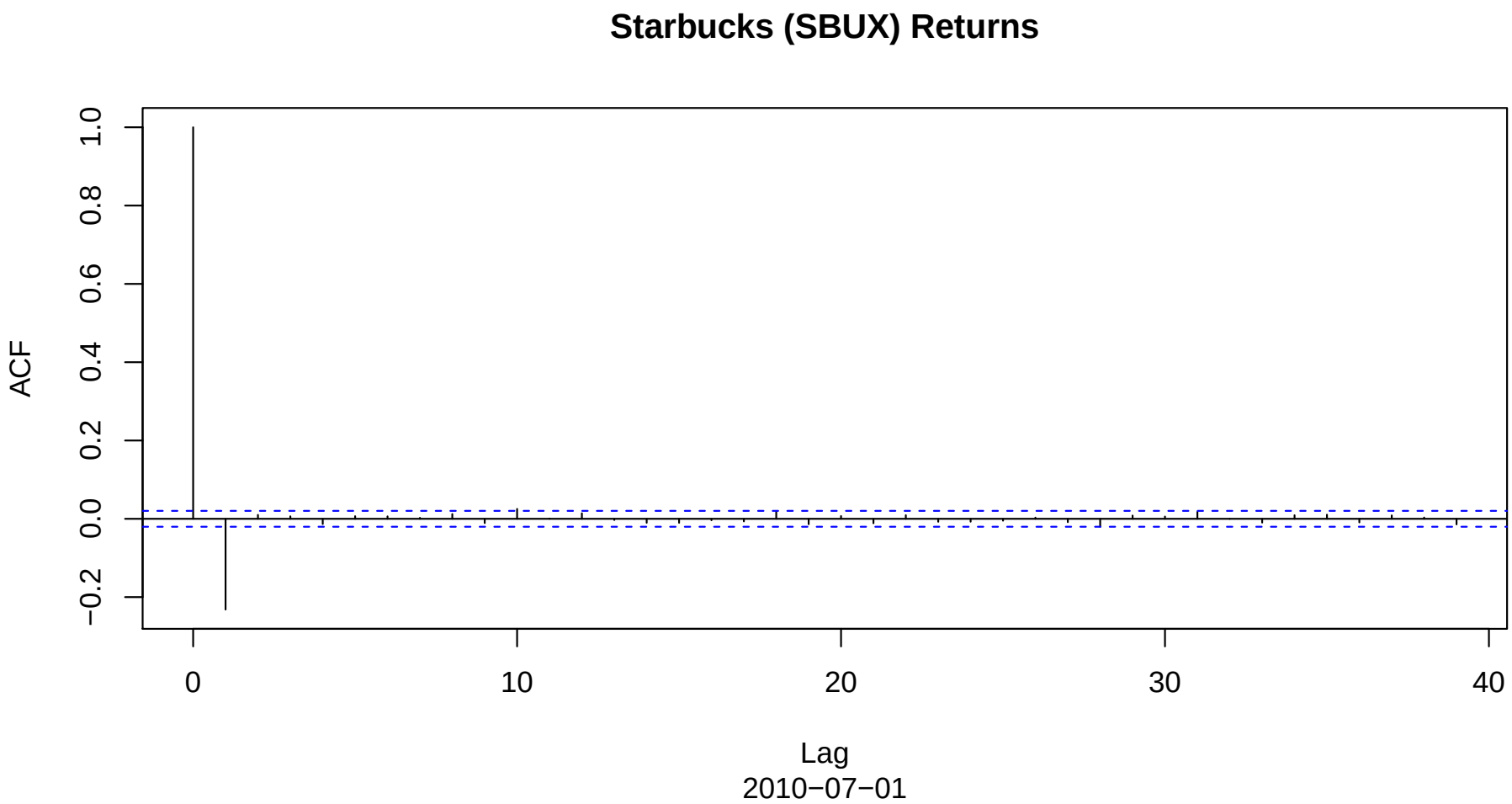
```
> plot(sbux.rets.xts, main="Starbucks (SBUX) Returns", ylab="Return")
```



Bid Ask Bounce: ACF

R Code: Return ACF

```
> acf(as.numeric(sbux.rets.xts), main="Starbucks (SBUX) Returns",
      sub="2010-07-01")
```



Bid Ask Bounce: Bias

Observed Price Process:

$$\underbrace{p_{t_i,m} - p_{t_{i-1},m}}_{r_{i,m}} \equiv \underbrace{p^*_{t_i,m} - p^*_{t_{i-1},m}}_{r^*_{i,m}} + \underbrace{u_{t_i,m} - u_{t_{i-1},m}}_{e_{i,m}} \qquad i = 1, 2, ..., m$$

The bias can be shown with a simple expansion:

$$RV^{(m)} = \sum_{i=1}^{M} r^2_{i,m} = \underbrace{\sum_{i=1}^{M} r^{*2}_{i,m}}_{RV^{(m)}_*} + \underbrace{\sum_{i=1}^{M} e^2_{i,m}}_{RV^{(m)}_u} + \underbrace{2 \cdot \sum_{i=1}^{M} r^*_{i,m} e_{i,m}}_{2RC^{(m)}_{*,u}}$$

Bid Ask Bounce: Bias

$$RV^{(m)} = \sum_{i=1}^{M} r_{i,m}^2 = \underbrace{\sum_{i=1}^{M} r_{i,m}^{*2}}_{RV_*^{(m)}} + \underbrace{\sum_{i=1}^{M} e_{i,m}^2}_{RV_u^{(m)}} + \underbrace{2 \cdot \sum_{i=1}^{M} r_{i,m}^* e_{i,m}}_{2RC_{*,u}^{(m)}}$$

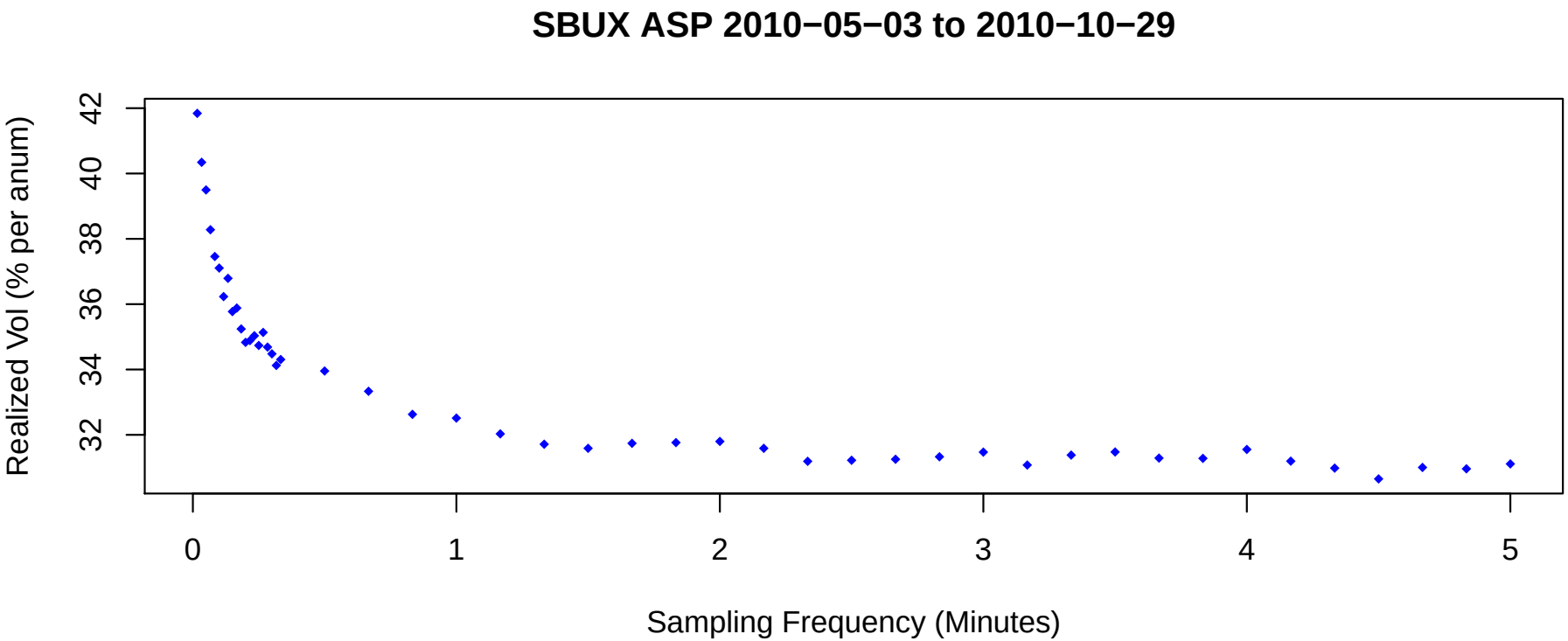
- We assume that $2RC_{*,u}^{(m)} = 0$
- As $m \to \infty$ the term $RV_u^{(m)}$ is greater than $RV_*^{(m)}$ and $RV^{(m)}$ is biased upward.
- As $m \to \infty$ we should converge to the true estimate of IV but instead we are overwhelmed with noise.

Computing Naive Realized Variance

R Code: rRealizedVariance at different *m*'s

```
> # 1 second estimate
> rRealizedVariance(sbux.rets.xts, period=1, align.by="seconds",
                    align.period=1, type="naive")

[1] 0.00103305303084

> # 30 Second estimate
> rRealizedVariance(sbux.rets.xts, period=30, align.by="seconds",
                    align.period=1, type="naive")

[1] 0.000655917834693

> # One Minute estimate
> rRealizedVariance(sbux.rets.xts, period=60, align.by="seconds",
                    align.period=1, type="naive")

[1] 0.000611898721492

> # One Minute estimate
> rRealizedVariance(sbux.rets.xts, period=1, align.by="Minutes",
                    align.period=1, type="naive")

[1] 0.000611898721492
```

Average Signature plot



- At highest frequency there is an upward bias
- As frequency is lowered the bias disappears

Starbucks: Multiple Days

R Code: Multiple Days of SBUX

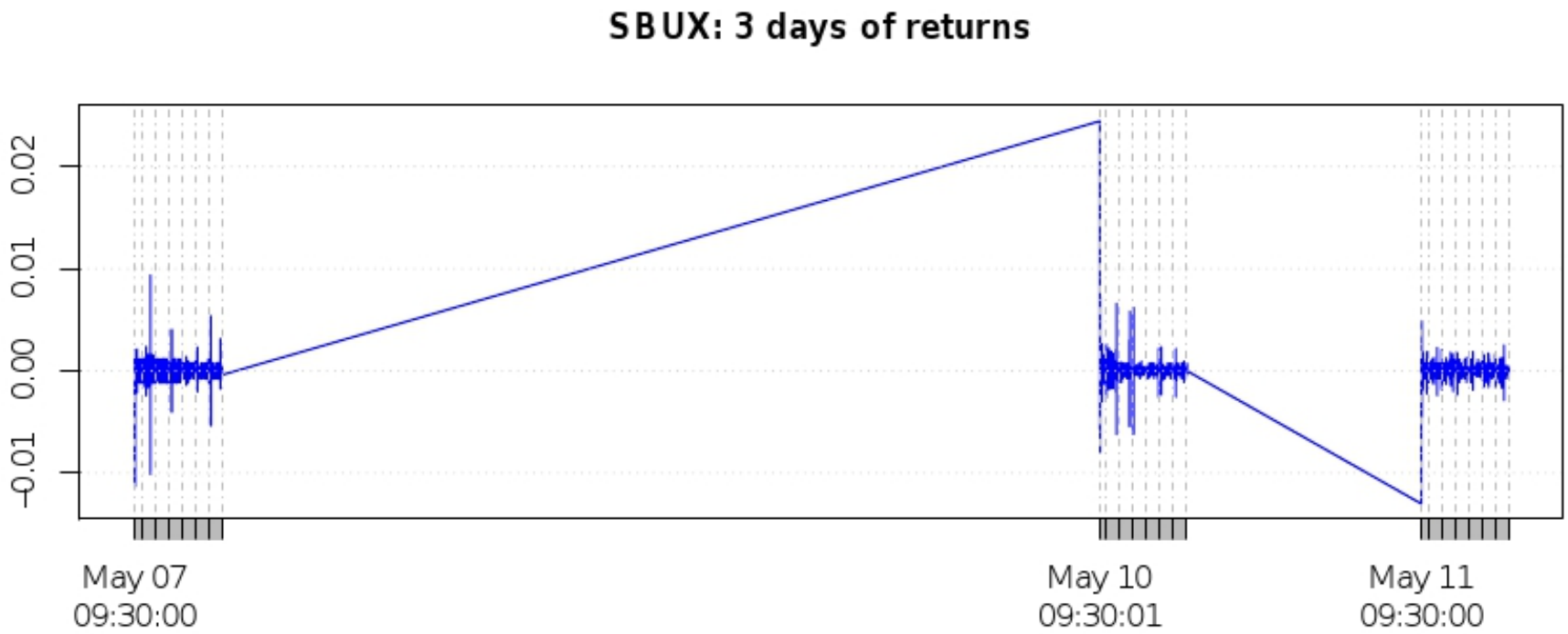
```
> load(paste(shared.dir,"data/sbux.rets.big.xts.rda", sep="/"))
> # Get Dates
> dates <- union(substr(index(sbux.rets.big.xts),1,10),
                 substr(index(sbux.rets.big.xts),1,10))

> length(dates)

[1] 127

> dates[1:3]

[1] "2010-05-03" "2010-05-04" "2010-05-05"
```

- 127 days of SBUX trade data
- Cleaned using RTAQ

Starbucks: Multiple Days

R Code: Plot Multiple Days

```
> plot(sbux.rets.big.xts[dates[5:7],], main="SBUX: 3 days of returns")
```



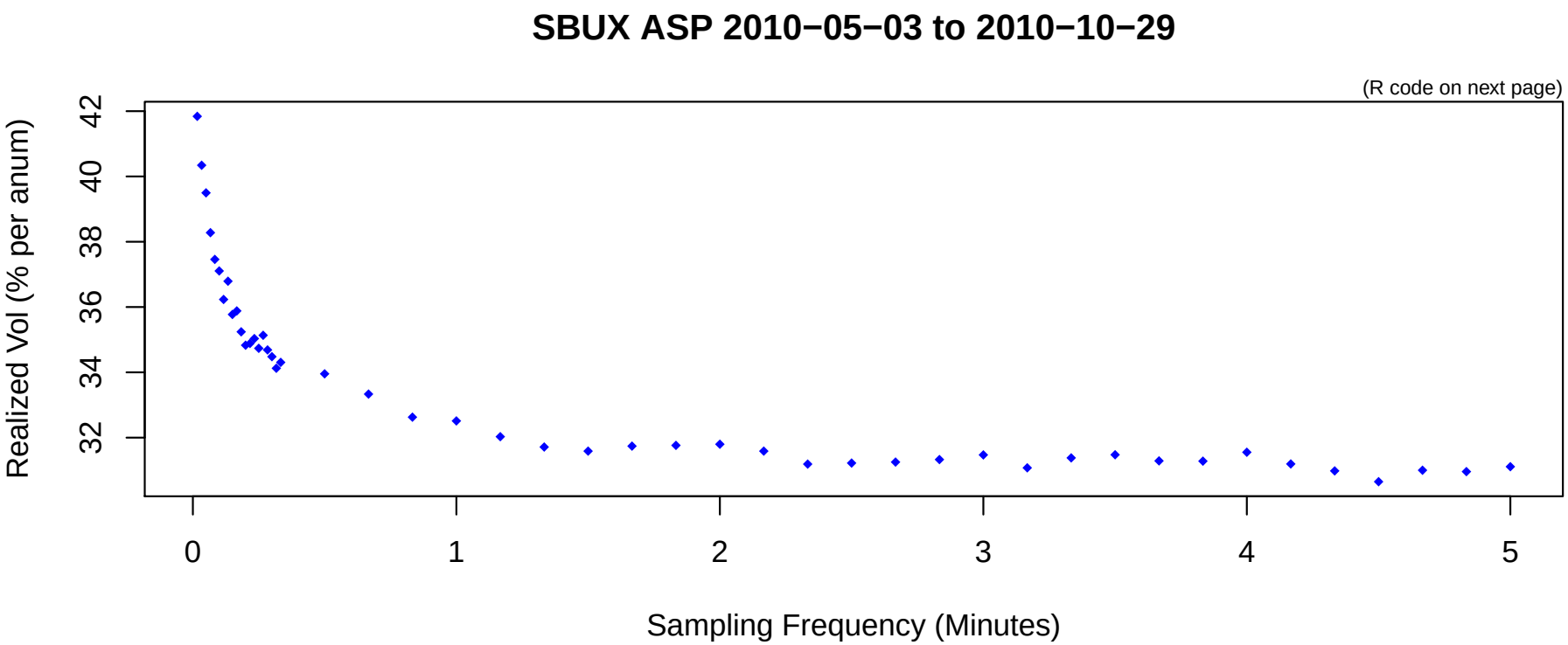
Signature Plot

R Code: rSignature

```
> args(rSignature)

function (range, x, y = NULL, type = "naive", cor = FALSE, rvargs = list(),
    align.by = "seconds", align.period = 1, xscale = 1, plotit = FALSE,
    cts = TRUE, makeReturns = FALSE, iteration.funct = NULL,
    iterations = NULL, lags = NULL)
NULL
```

- range: x axis of plot (sampling period for naive)
- x: an xts return object
- y: an xts return object (used for covariance)
- type: "naive", "kernel", ...
- align.by: align using seconds, minutes, hours
- align.period: aligned to this many seconds, minutes, hours

Average Signature plot



- Let's choose and "optimal" frequency of 2 minute returns

Average Signature plot

R Code: Average Signature Plot

```
> datestr <- paste(dates, " 09:30:00::",dates, " 16:00:00", sep="")
> li <- list()
> for(i in 1:length(dates))
 {
     li[[i]] <- rSignature(c(1:20,10*(3:30)),
                  sbux.rets.big.xts[datestr[[i]],])
 }
> rvs <- li[[1]]$y
> for(i in 2:length(dates))
     rvs <- rvs+li[[i]]$y
> plot(c(1:20,10*(3:30))/60, sqrt(rvs/length(dates))*sqrt(250)*100,
       xlab="Sampling Frequency (Minutes)",
       ylab="Realized Vol (% per anum)", cex=.7, pch=18, col="blue",
       main="SBUX ASP 2010-05-03 to 2010-10-29")
```

Realized Variance Estimation Solutions

Timing:

- Choose an optimal sampling frequency, using an ASP (we chose 2 mins for our data)

Kernel:

- Similar to long run variance
- Eliminates auto-correlation

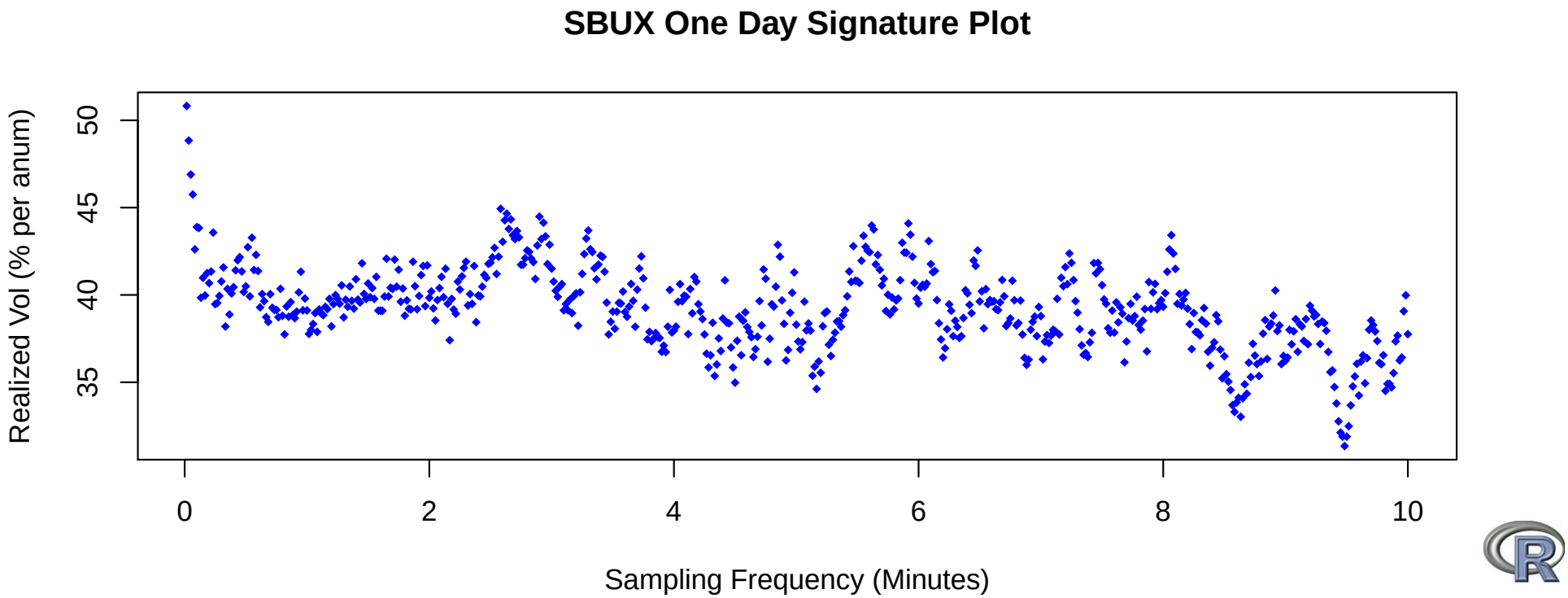
Other:

- Two-Timescale
- Average RV

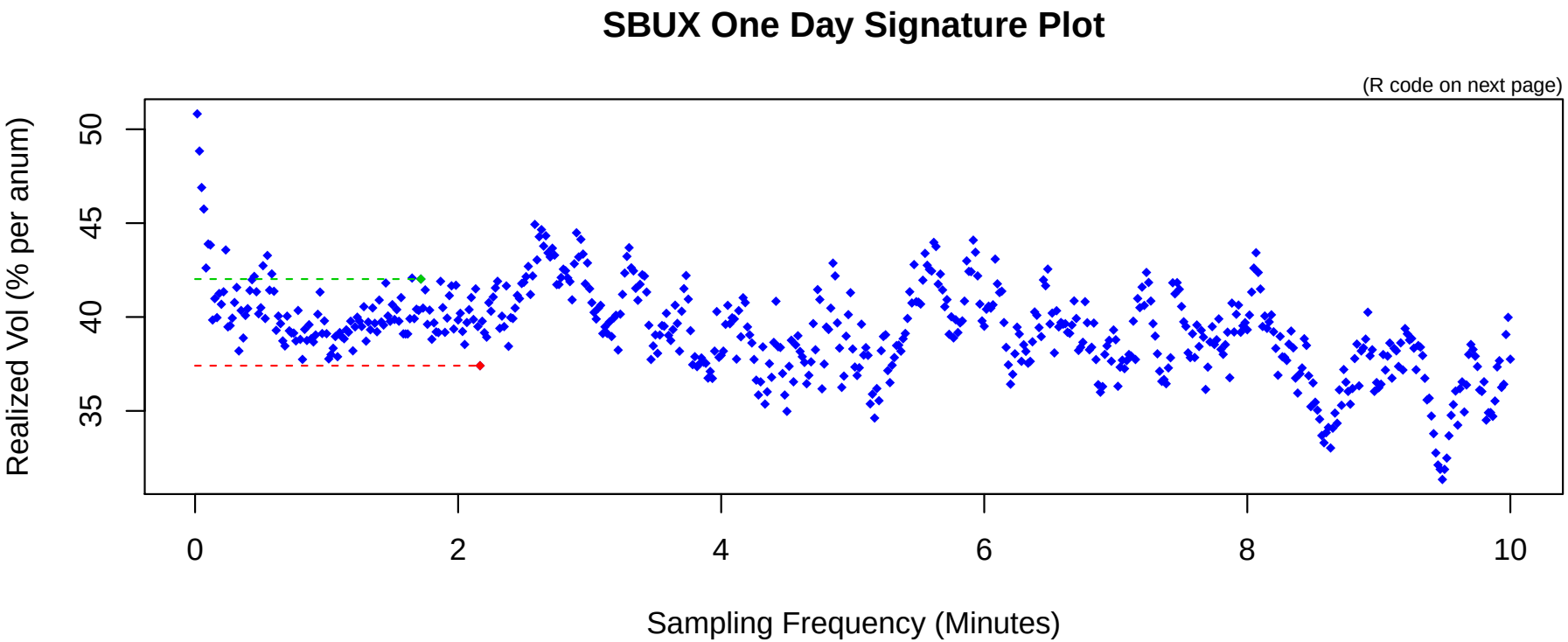
One Day Signature Plot

R Code: One Day Signature Plot

```
> sig.naive <- rSignature(1:600,sbux.rets.xts, align.period=1, xscale=1)
> sig.naive$x <- sig.naive$x/60
> sig.naive$y <- sqrt(sig.naive$y)*sqrt(250)*100

> plot(sig.naive$x, sig.naive$y, xlab="Sampling Frequency (Minutes)",
       ylab="Realized Vol (% per anum)", cex=.7, pch=18, col="blue",
       main="SBUX One Day Signature Plot")
```



One Day Signature Plot



- Sampling difference: 27 seconds
- Max Vol: 42%
- Min Vol: 37%

One Day Signature Plot

R Code: Investigate 2 Minute Region

```
> plot(sig.naive$x, sig.naive$y, xlab="Sampling Frequency (Minutes)",
 ylab="Realized Vol (% per anum)", cex=.7, pch=18, col='blue',
 main="SBUX One Day Signature Plot")
> # Find the minimum and maximum values of RV for a
> # time period near two minutes (our optimal)
> themin <- which(sig.naive$y[100:140]==min(sig.naive$y[100:140]))
> themax <- which(sig.naive$y[100:140]==max(sig.naive$y[100:140]))
> # Plot these points
> points(sig.naive$x[99+themin], sig.naive$y[99+themin],
        pch=18, cex=.7, col=2)
> lines(c(0,sig.naive$x[99+themin]),
       c(sig.naive$y[99+themin],sig.naive$y[99+themin]),
       lty=2, col=2)
> points(sig.naive$x[99+themax], sig.naive$y[99+themax],
        pch=18, cex=.7, col=3)
> lines(c(0,sig.naive$x[99+themax]),
       c(sig.naive$y[99+themax],sig.naive$y[99+themax]),
       lty=2,  col=3)
```

Kernel Estimators

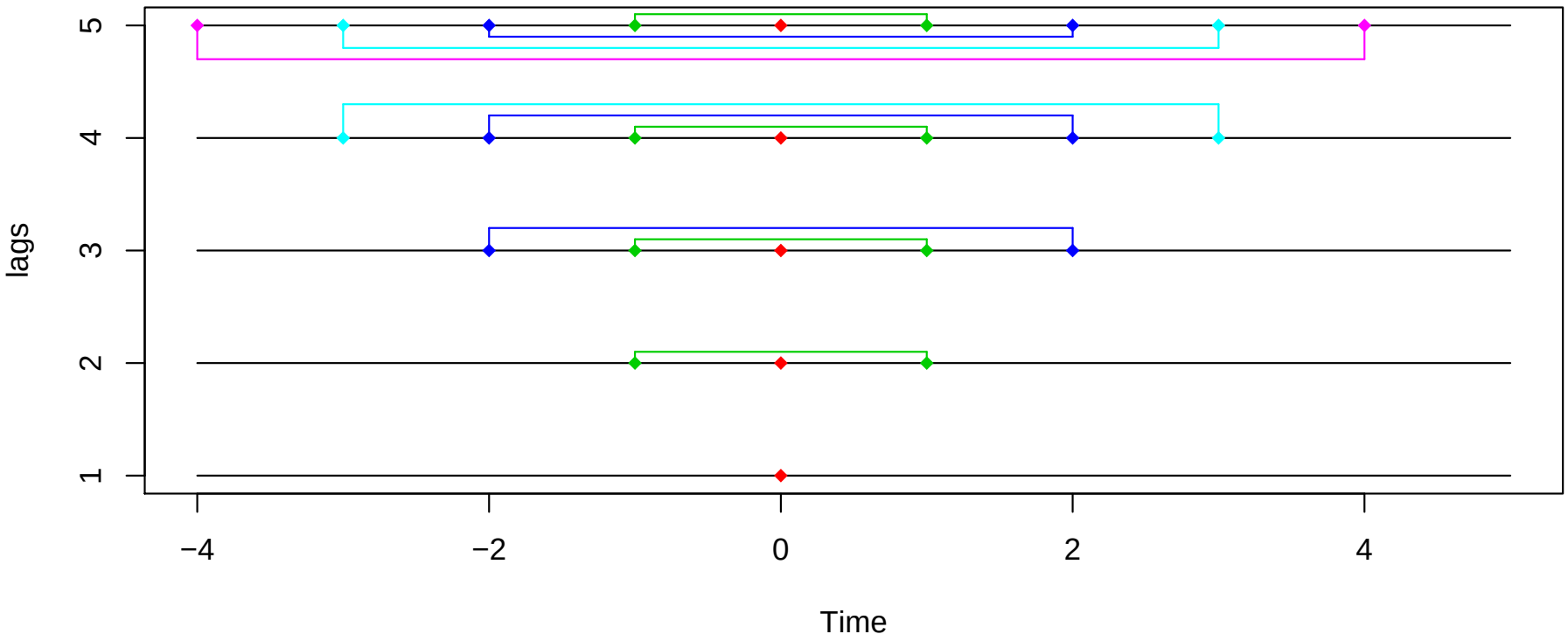
$$RV_{Kernel} = \hat{\gamma}_0 + \sum_{h=1}^{H} k\left(\frac{h-1}{H}\right)\left\{\hat{\gamma}_h + \hat{\gamma_{-}}_h\right\}$$

$$\hat{\gamma}_h \equiv \sum_{i=1}^{m} r_{i,m} r_{i+h,m}$$

- Similar to Long Run Variance estimation (Newey-West)
- Attempts to solve the bid-ask-bounce auto-correlation problem
- Provides stable realized variance estimates

Kernel Estimators

$$RV_{Kernel} = \hat{\gamma}_0 + \sum_{h=1}^{H} k\left(\frac{h-1}{H}\right)\left\{\hat{\gamma}_h + \hat{\gamma_{-h}}\right\}, \hat{\gamma}_h \equiv \sum_{i=1}^{m} r_{i,m} r_{i+h,m}$$



Kernel Estimators

$$RV_{Kernel} = \hat{\gamma}_0 + \sum_{h=1}^{H} k\left(\frac{h-1}{H}\right)\left\{\hat{\gamma}_h + \hat{\gamma_{-h}}\right\}, \hat{\gamma}_h \equiv \sum_{i=1}^{m} r_{i,m} r_{i+h,m}$$

	(h-1)/H	Rectangular Kernel	Bartlett Kernel
One Lag	0	1	1
Two Lags	0, 1/2	1, 1	1, .5
Three Lags	0, 1/3, 2/3	1, 1, 1	1, .66, .33
Four Lags	0, 1/4, 2/4, 3/4	1, 1, 1, 1	1, .75, .5, .25
Five Lags	0, 1/5, 2/5, 3/5, 4/5	1, 1, 1, 1, 1	1, .8, .6, .4, .2

rKernel, rKernel.available

R Code: rKernel

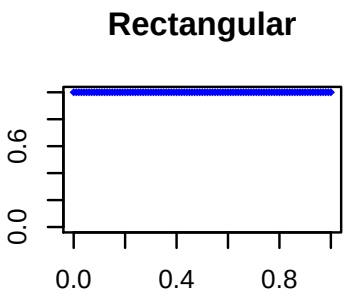
```
> args(rKernel)

function (x, type = 0)
NULL
```

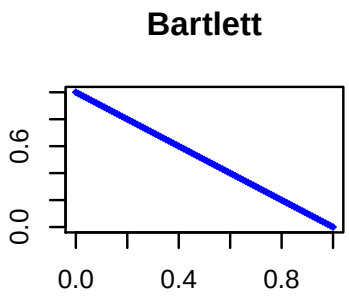
- x: value between 0 and 1 to calculate kernel weight
- type: type of kernel 0-11 represents the 12 kernels in rKernel.available. Can also be specified as text (type="Bartlett", etc)

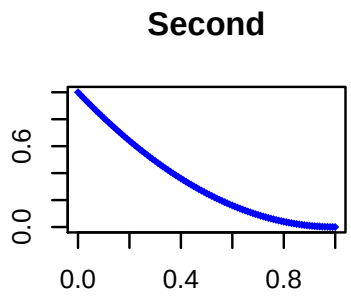
R Code: rKernel.available

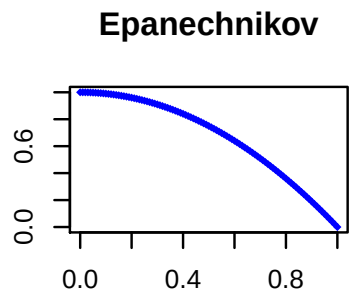
```
> rKernel.available()

 [1] "Rectangular"          "Bartlett"              "Second"
 [4] "Epanechnikov"         "Cubic"                 "Fifth"
 [7] "Sixth"                "Seventh"               "Eighth"
[10] "Parzen"               "TukeyHanning"          "ModifiedTukeyHanning"
```

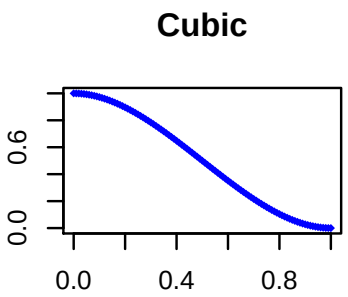
realized Kernels

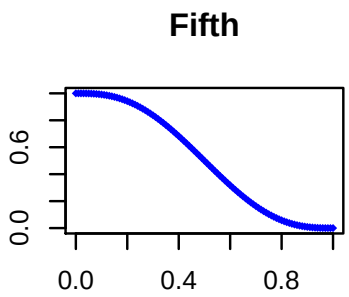


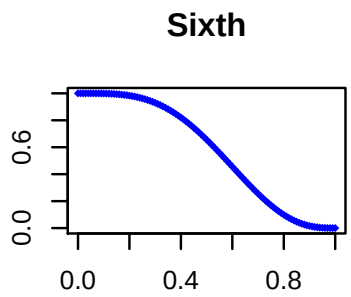


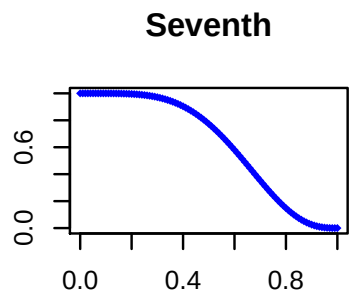


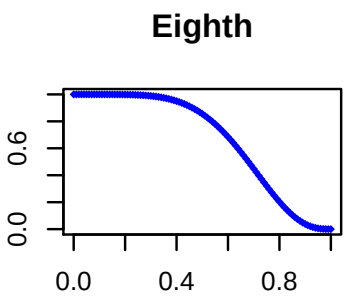


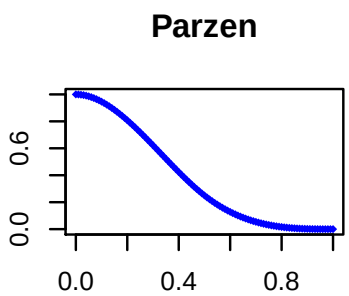


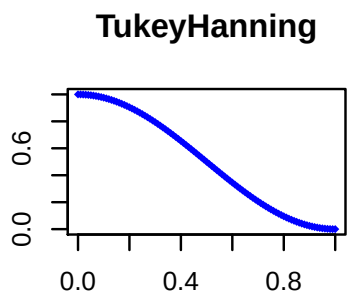


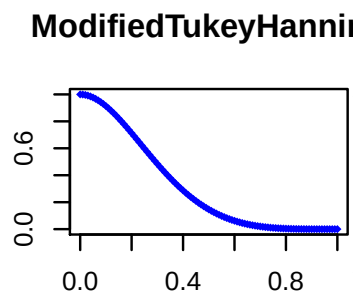












(R code on next page)

realized Kernels

R Code: Plot Kernels

```
> par(mfrow=c(3,4))
> x <- (0:100)*.01
> for(i in 1:length(rKernel.available()))
    plot(x=x,
         y=sapply(x,FUN="rKernel",type=rKernel.available()[i]),
         xlab="",
         ylab="",
         main=rKernel.available()[i],
         ylim=c(0,1),
         pch=18,
         col="blue",
         cex=.7)
```

Computing Kernel Based Realized Variance

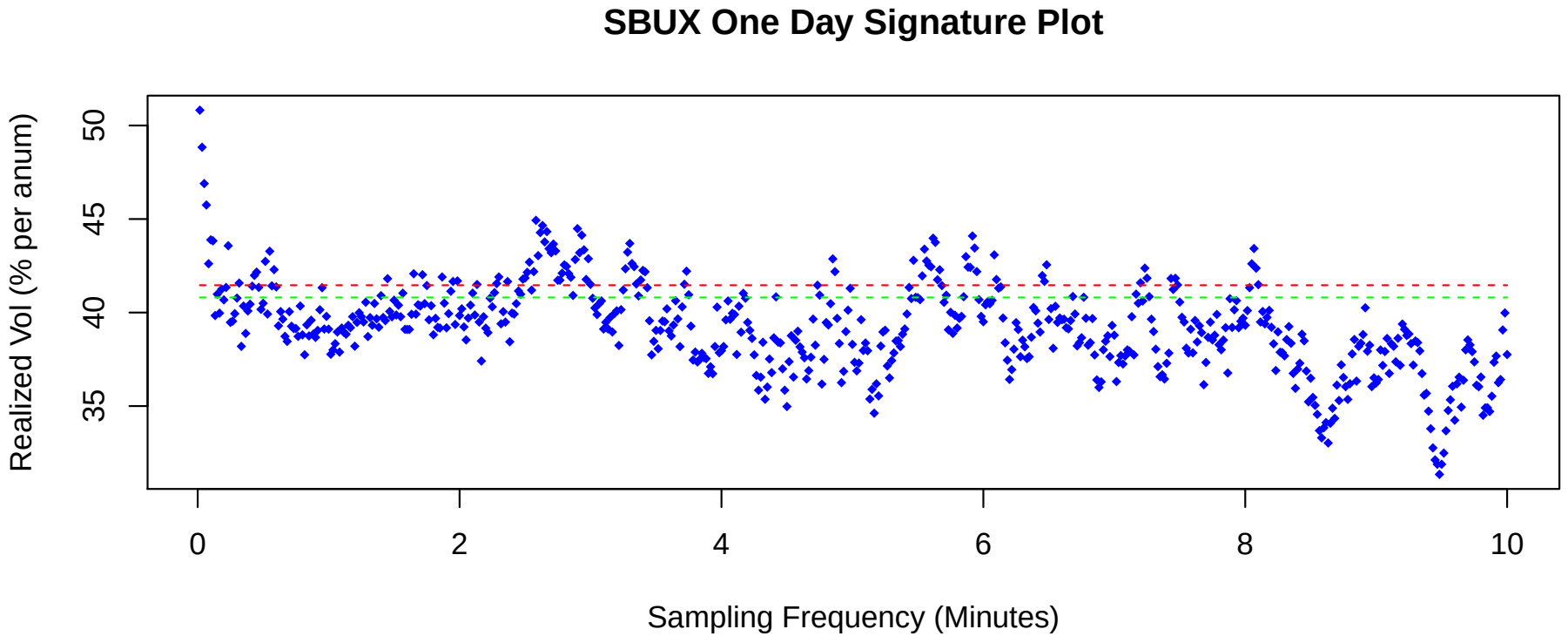
R Code: rRealizedVariance Kernel Estimates

```
> # Bartlett Kernel on one second returns, lag = 10
> bart.10 <- rRealizedVariance(sbux.rets.xts, align.by="secs", align.period=1,
                               type="kernel", rvargs=list(kernel.type="bartlett",
                                                          kernel.param=10))
> bart.10

[1] 0.00066605818814

> # Bartlett Kernel on one MINUTE returns, lag = 3
> rRealizedVariance(sbux.rets.xts, align.by="mins", align.period=1,
                    type="kernel", rvargs=list(kernel.type="bartlett",
                                               kernel.param=3))

[1] 0.000683761298184

> # Tukey-Hanning Kernel on one second returns, lag = 10
> th.10 <- rRealizedVariance(sbux.rets.xts, align.by="secs", align.period=1,
                             type="kernel", rvargs=list(kernel.type=11,
                                                        kernel.param=10))
> th.10

[1] 0.000687564800786
```

Kernel Points

R Code: One Day Signature Plot + Kernel Esimates

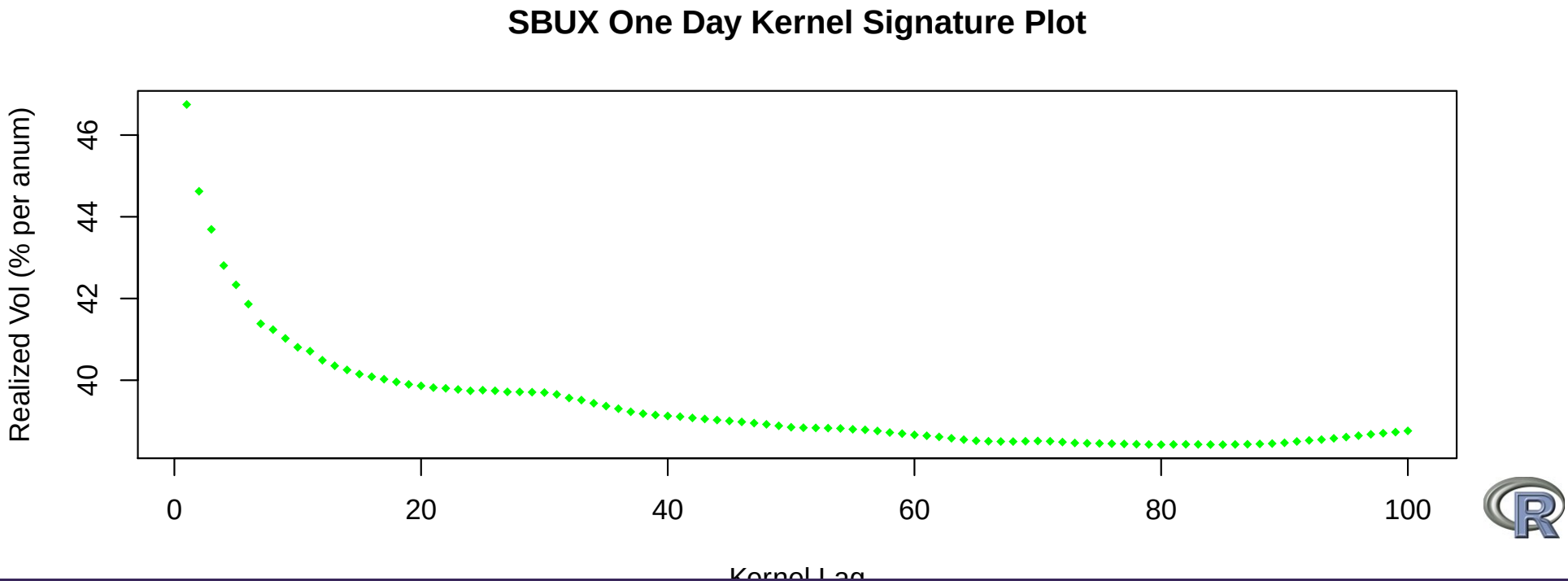
```
> plot(sig.naive$x, sig.naive$y, xlab="Sampling Frequency (Minutes)",
      ylab="Realized Vol (% per anum)", cex=.7, pch=18, col="blue",
      main="SBUX One Day Signature Plot")
> kern.bart.10.av <- sqrt(bart.10)*sqrt(250)*100
> kern.th.10.av <- sqrt(th.10)*sqrt(250)*100
> lines(sig.naive$x, rep(kern.bart.10.av, 600), col='green', lty=2)
> lines(sig.naive$x, rep(kern.th.10.av, 600), col='red', lty=2)
```



One Day Kernel Signature Plot

R Code: One Day Kernel Signature Plot

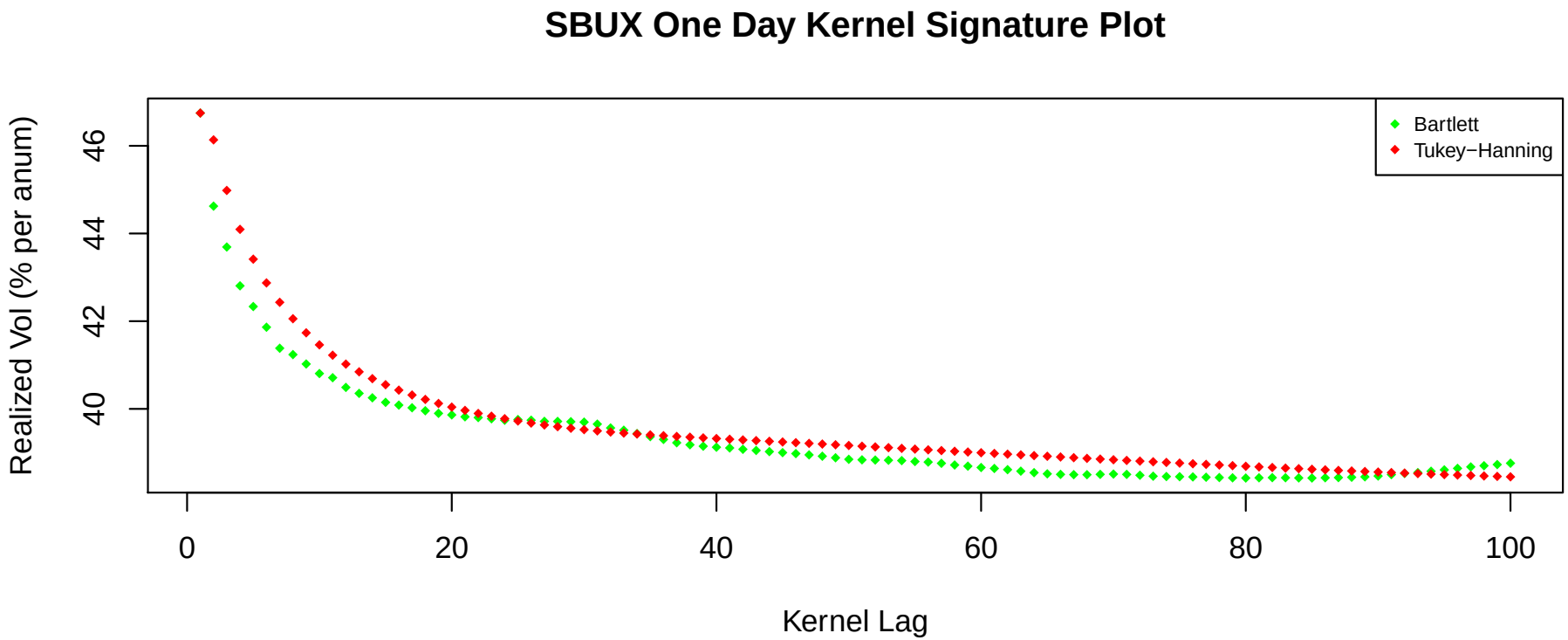
```
> sig.bart <- rSignature(1:100, sbux.rets.xts, type="kernel",
                         rvargs=list(kernel.type="bartlett"),
                         align.period=1, align.by="seconds")
> sig.bart$y <- sqrt(sig.bart$y)*sqrt(250)*100

> plot(sig.bart$x, sig.bart$y, xlab="Kernel Lag",
       ylab="Realized Vol (% per anum)", cex=.7, pch=18, col="green",
       main="SBUX One Day Kernel Signature Plot")
```



One Day Kernel Signature Plot

R Code: One Day Kernel Signature Plot, Add Tukey-Hanning

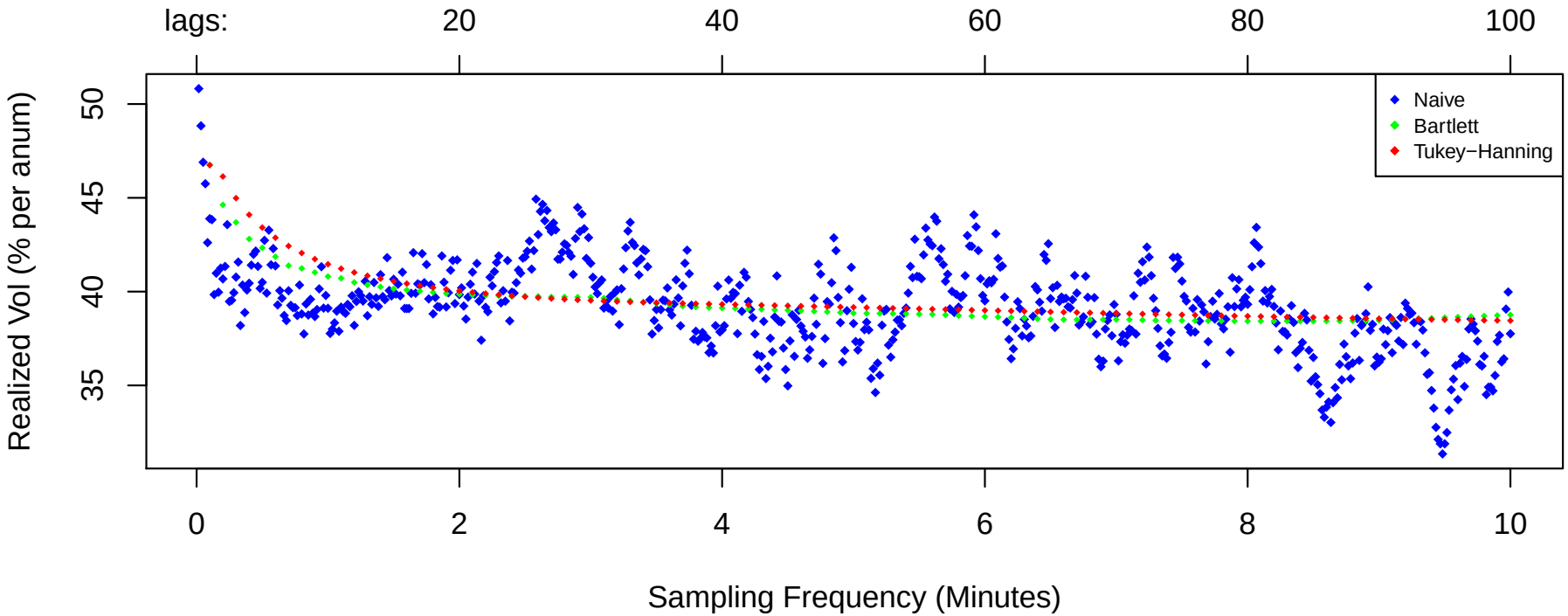
```
> sig.th <- rSignature(1:100, sbux.rets.xts, type="kernel",
                       rvargs=list(kernel.type=11),
                       align.period=1, align.by="seconds")
> sig.th$y <- sqrt(sig.th$y)*sqrt(250)*100
> points(sig.th$x, sig.th$y, cex=.7, pch=18, col='red')
> legend("topright", c('Bartlett', 'Tukey-Hanning'), col=c('green', 'red'),
         cex=.7,pch=18)
```



One Day Kernel Signature Plot

R Code: One Day Kernel Signature Plot Overlay

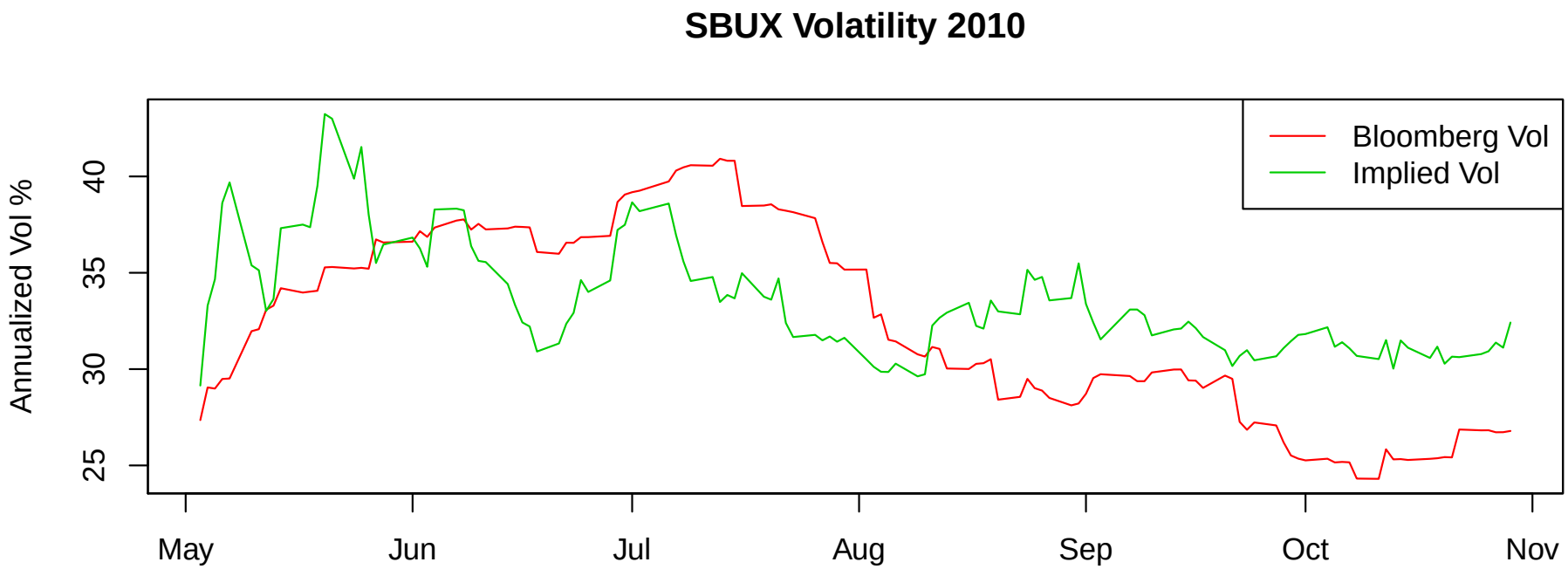
```
> plot(sig.naive$x, sig.naive$y, xlab="Sampling Frequency (Minutes)",
      ylab="Realized Vol (% per anum)", cex=.7, pch=18, col="blue")
> points(sig.bart$x*.1, sig.bart$y, cex=.5, pch=18, col='green')
> points(sig.th$x*.1, sig.th$y, cex=.5, pch=18, col='red')
> legend("topright", c('Naive', 'Bartlett', 'Tukey-Hanning'),
        col=c('blue', 'green', 'red'), cex=.7, pch=18)
> axis(3, c(0, 2, 4, 6, 8, 10), c('lags:', 20, 40, 60, 80, 100))
```



Implied Volatility

R Code: SBUX Volatility Graph

```
> load(paste(shared.dir,"data/bloomberg.ivol.xts.rda", sep="/"))
> plot(as.zoo(bloomberg.ivol.xts), plot.type="single", col=c(2,3),
      main="SBUX Volatility 2010", xlab="", ylab="Annualized Vol %")
> legend("topright", c("Bloomberg Vol", "Implied Vol"), col=c(2,3), lwd=c(1,1))
```



Multiple Days of Realized Variance

R Code: Calcuate Multiple Days of Realized Variance

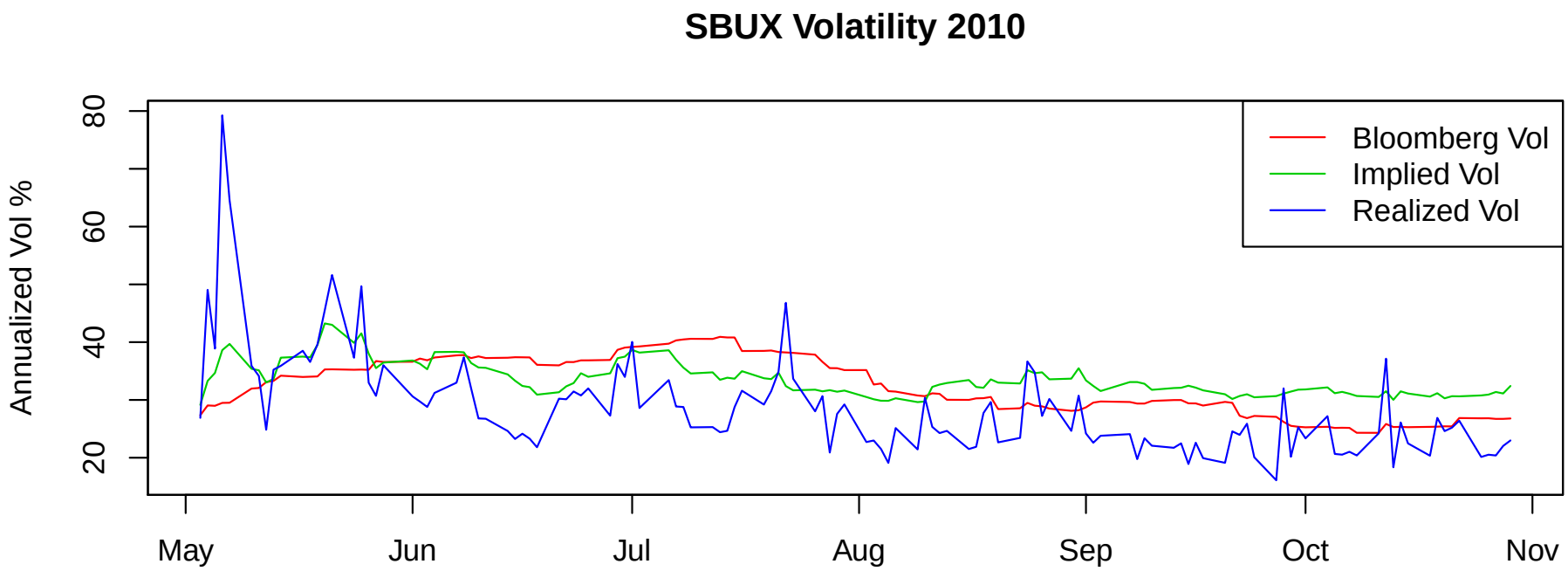
```
> datestr <- paste(dates, " 09:30:00::",dates, " 16:00:00", sep="")
> rvs <- matrix(NA, nrow = length(dates), ncol=1)
> for(i in 1:length(dates))
 {
     rvs[i,1] <- rRealizedVariance(sbux.rets.big.xts[datestr[[i]],],
                          type = "kernel",
                          rvargs = list(kernel.type = "bartlett",
                                        kernel.param = 70),
                          align.by = "seconds",
                          align.period = 1)
 }
> rvs <- sqrt(rvs)*sqrt(250)*100
> dimnames(rvs)[[1]] <- dates
> rvs.xts <- as.xts(rvs)
```

- NOTE: 'kernel.param = 70' represents 70 lags as we picked from the one day signature plot

Multiple Days of Realized Variance

R Code: Plot Multiple Days of Realized Variance

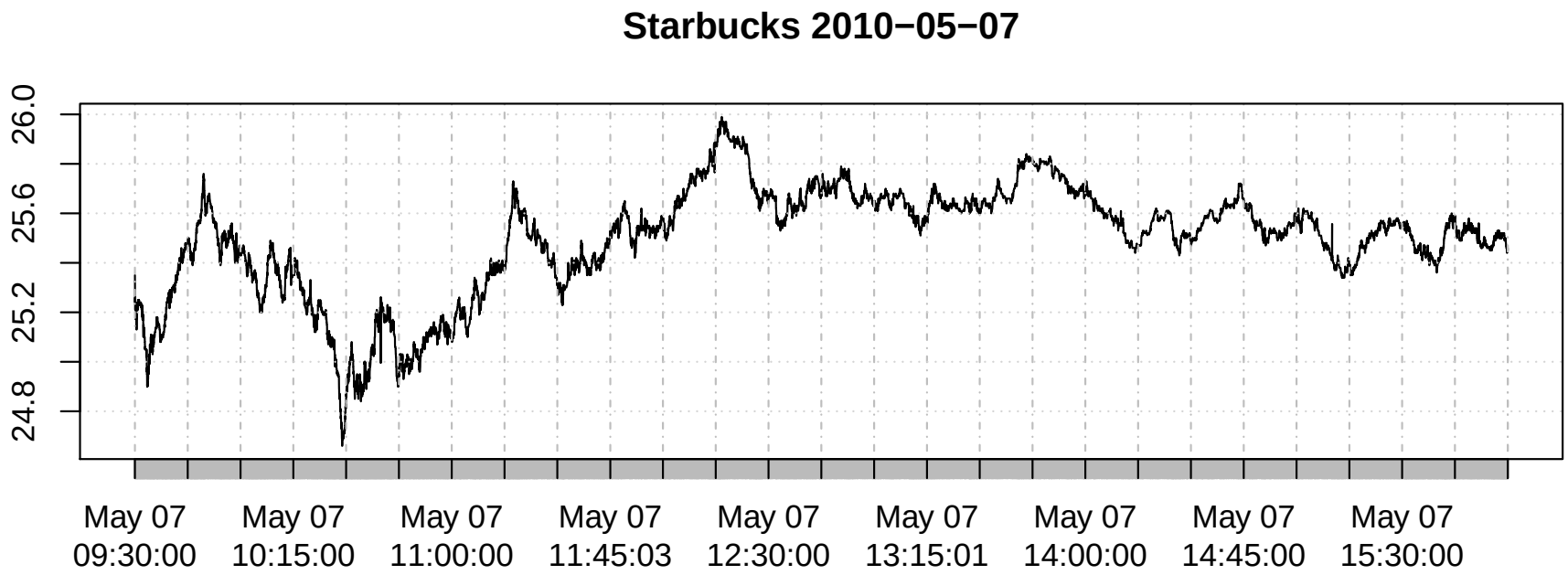
```
> plot(as.zoo(merge(bloomberg.ivol.xts,rvs.xts)), plot.type="single",
       col=c(2,3,4,1), main="SBUX Volatility 2010",
       xlab="",ylab="Annualized Vol %")
> legend("topright", c("Bloomberg Vol", "Implied Vol", "Realized Vol"),
         col=c(2,3,4), lwd=c(1,1,1))
```



Starbucks 2010-05-07

R Code: Plotting Starbucks 2010-05-07

```
> load(paste(shared.dir,"data/sbux.20100507.prices.xts.rda", sep="/"))
> plot(sbux.20100507.prices.xts[,"PRICE"], main="Starbucks 2010-05-07",
      xlab="",ylab="")
```



- Open to close prices: 25.35, 25.45 (return = 39 bps)
- Min and Max prices: 24.66, 25.99 (% change = 540 bps)

Realized Variance and the Realized package

Realized Variance:

- As $M \to \infty, RV_*^{(m)} \to \sigma(t)^2$ (where $dp^* = \sigma(t)dw(t)$).
- Bid-ask-bounce causes an upward bias

Solutions and Tuning :

- One day signature plots can help us diagnose and tune
- Kernel estimators

Realized Package:

- Realized Variance Estimation and Tuning tools
- ** Also contains Realized Covariance and Correlation tools

Outline

Summary

- R has a wealth of tools for high-frequency financial data analysis
  - accessing
  - manipulating
  - analyzing
  - visualizing
- Packages that support these activities include
  - xts/zoo, POSIXlt/POSIXct
  - IBrokers
  - RTAQ
  - quantmod
  - realized

Going Further

Course	Introduction to Electronic Trading Systems
Dates	June 20, 2011 through August 22, 2011
Time	Monday & Wednesday 6:00 PM - 7:30 PM PDT
Instructor	Guy Yollin, Visiting Lecturer, Applied Mathematics
Where	Online or on-campus at the University of Washington, Seattle
Info	`http://www.amath.washington.edu/studies/computational-finance/`

Topics

- financial markets, exchanges, and the electronic trading process
- trading system development, optimization, and backtesting
- paper trading with Interactive Brokers Student Trading Lab

Running R in the clouds

Special thanks to RStudio, Inc.

- For the free use of RStudio™ server

This Sweave presentation was created using RStudio™

Wrap up

- Questions and comments
- Contacting the Presenters
  - Guy Yollin
    - `http://www.r-programming.org`
    - gyollin@r-programming.org
  - Scott Payseur
    - `http://uk.linkedin.com/in/spayseur`
    - scott.payseur@gmail.com