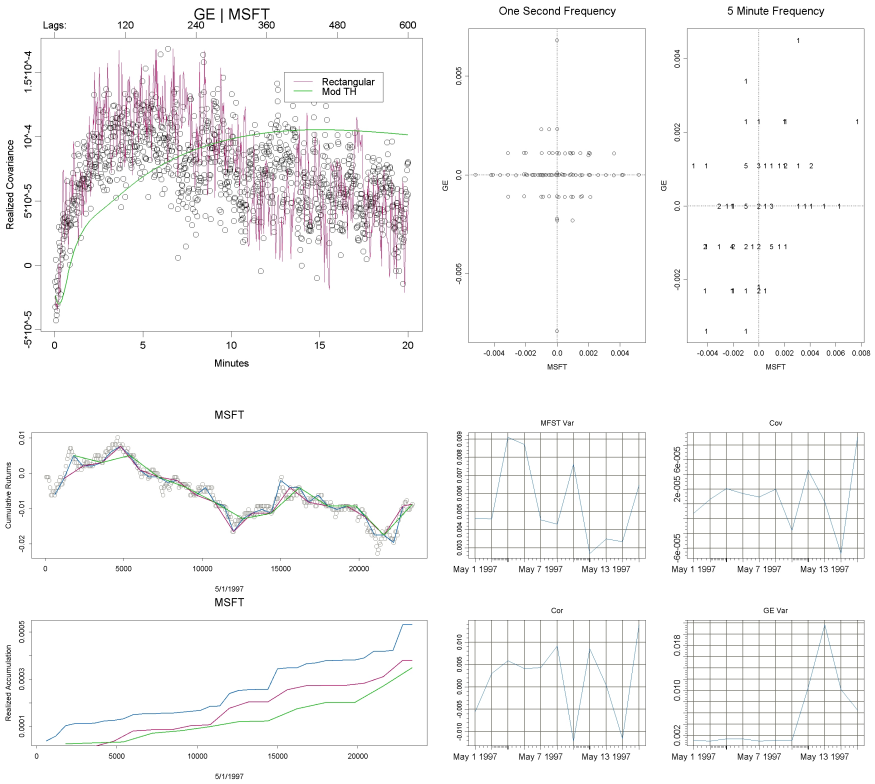
"Realized" Software Package 0.8
SPlus Supplement

Scott Payseur (spayseur@u.washington.edu)

October 23, 2007

http://students.washington.edu/spayseur/realized



Abstract

S+Finmetrics 3.0 and Yan and Zivot (2003) contain high frequency functions. Data importing, cleaning, and alignment functions are particularly useful to create inputs to this Realized library. This is a beta release and is meant to solidify the accuracy and useability of the software. Please contact me with any bugs, suggestions or comments at spayseur@u.washington.edu. Please site if used for academic purposes.

1 Interfacing with Finmetrics 3.0

S+Finmetrics 3.0 and Yan and Zivot (2003) contain high frequency manipulation functions, including data cleaning, data alignment, and data importing. Each library creates S+TimeSeries objects. In this supplement I discuss how to interact with the Realized library functions using S+TimeSeries objects with examples. This is meant to supplement to the User's Manual, which contains a more robust discussion of the realized variance and covariance literature. All of the Realized library[1] functions have the ability to take timeSeries objects as the x and y parameters. However, the conversion from a timeSeries object to an object of type realizedObject is compuationally intensive. I recommend that you convert the timeSeries to a realizedObject if you will be doing multiple calculations. Below is a script created to interact with S+Finmetics 3.0 and the first part of it is described in Zivot (available in the Finmetrics 3.0 directory). The data is available in Eric Zivot's website: http://faculty.washington.edu/ezivot/.

Below the trade_msft.txt and trade_ge.txt files from the NYSE Trades and Quotes database are loaded into S+TimeSeries objects, reordered, cleaned and aligned to the trading day.

```
> module(finmetrics)
S+FinMetrics Version 3.0.1 for Microsoft Windows :  2007
> dataPath = "d:/dev/zivot.highFreq/" # Change to your data directory!
> taqData = paste(dataPath, "trade_msft.txt", sep="")

> msftt.df = importData(file=taqData, keep=c("price", "siz", "tdate", "ttim"),
 + delim="|", stringsAsFactors=F, time.in.format="%d%m%y", big=F)
> pos = msftt.df$tdate + msftt.df$ttim/(24*60*60)
> pos@format = "%02m/%02d/%04Y %02H:%02M:%02S"
> msftt.ts = timeSeries(pos=pos, data=msftt.df[, c("price", "siz")])
> msftt.ts = seriesReorder(msftt.ts)
> msftt.cleanPrice.ts = seriesClean(msftt.ts[,"price"], n=51, k = 25, maxDev=3)
> msftt.cleanPrice.ts = excludeTimes(msftt.cleanPrice.ts, exclude="closed", openTime=timeCalendar(h=9,min=30),
 + closeTime=timeCalendar(h=16), type="d")

> taqData = paste(dataPath, "trade_ge.txt", sep="")
> get.df = importData(file=taqData, keep=c("price", "siz", "tdate", "ttim"), delim="|", stringsAsFactors=F,
 + time.in.format="%d%m%y", big=F)
> pos = get.df$tdate + get.df$ttim/(24*60*60)
> pos@format = "%02m/%02d/%04Y %02H:%02M:%02S"
> get.ts = timeSeries(pos=pos, data=get.df[, c("price", "siz")])
> get.ts = seriesReorder(get.ts)
> get.cleanPrice.ts = seriesClean(get.ts[,"price"], n=51, k = 25, maxDev=3)
> get.cleanPrice.ts = excludeTimes(get.cleanPrice.ts, exclude="closed", openTime=timeCalendar(h=9,min=30),
 + closeTime=timeCalendar(h=16), type="d")
```

The dayPlot function is used to plot the high frequency data for MSFT. Figures 12 and 13 show clean and unclean data respectively.

```
> # Figure 12
> dayPlot(msftt.ts[,1])
> # Figure 13
> dayPlot(msftt.cleanPrice.ts)
```

[1] From Beta 0.8 on.

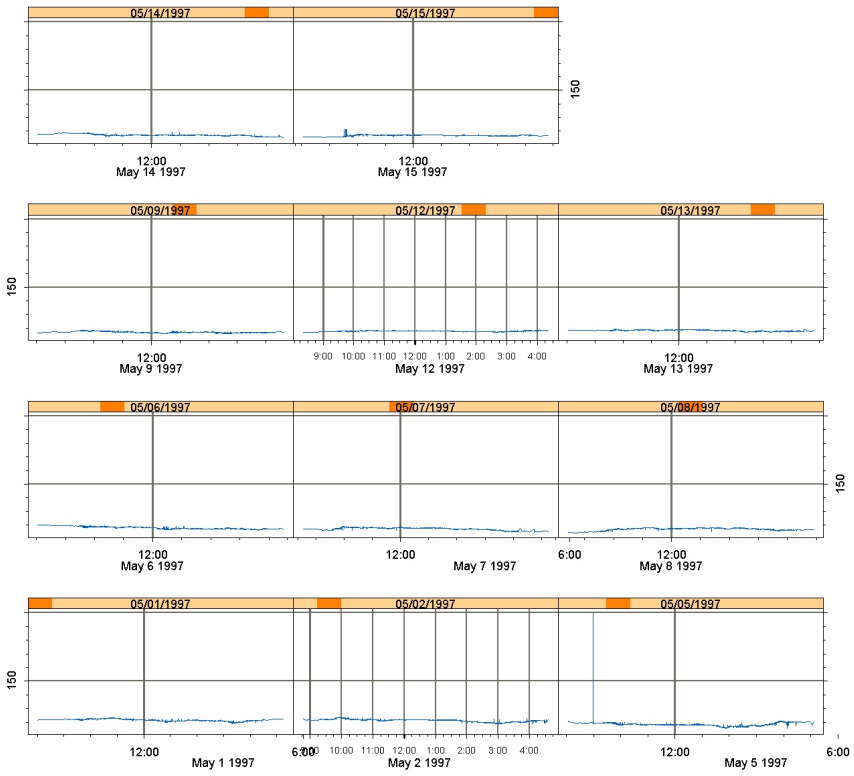


Figure 12: MSFT Day Plot Pre-Clean

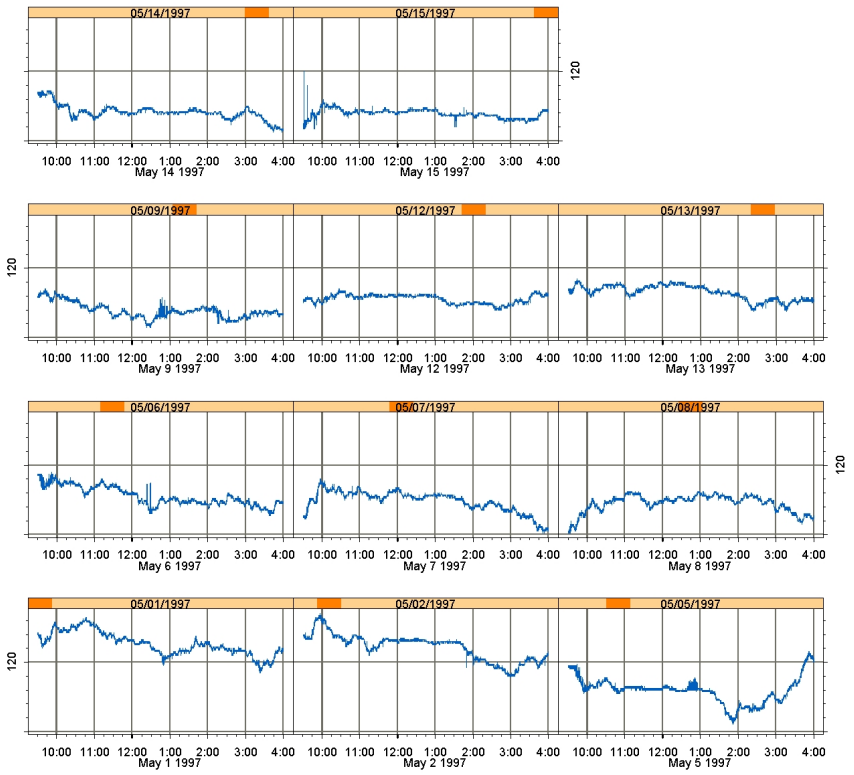


Figure 13: MSFT Day Plot Post-Clean

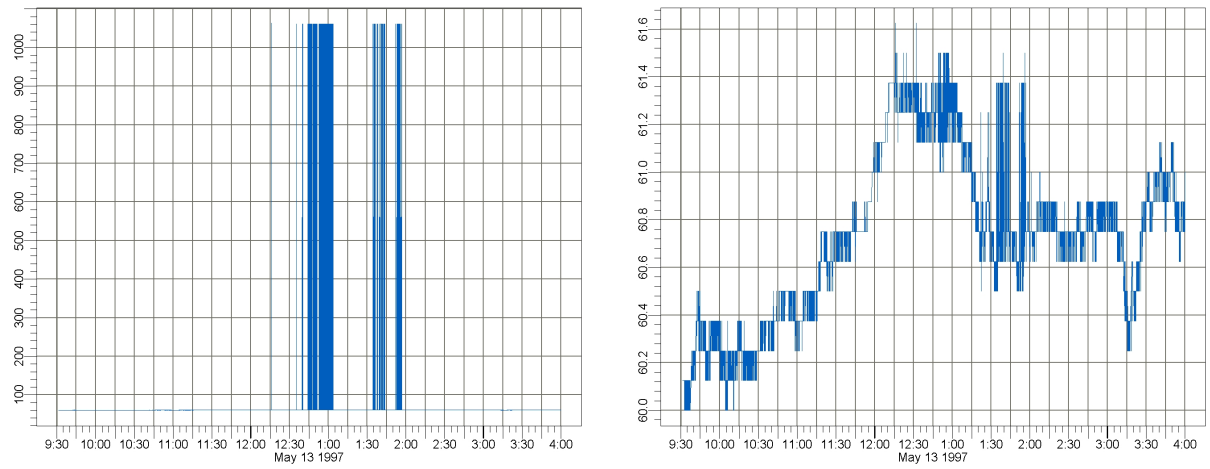


Figure 14A and 14B : Futher cleaning

If you do not have pre-cleaned data then it is important to realized that the seriesClean function does not work perfect for all datasets and requires tuning of the parameters to properly catch outliers, as well as, prevent over filtering of the data. In the TAQ data from the files in this supplement there are some days that contain a high amount of outliers. One of these days is May 5th, 1997 for General Electric. Figure 14A is a plot of the data after it has been cleaned.

```
> # Figure 14A
> plot(get.cleanPrice.ts[timeEvent("5/13/1997"), ])
```

Figure 14A shows that there are some prices that are over $1000. Studying the data further it appears that there were some trades that erronously had $1000 added to each. For a quick fix in this supplement I just subtract 1000 from each of these trades. Again, if you are using unclean data you should develope a strategy for cleaning that involves seriesClean, as well as, spot checks to be sure your parameters are tuned properly. The nature of realized calculations can make outliers a very big issue.

```
> ind <- get.ts[, "price"] > 1000
> get.ts@data[ind, "price"] <- (get.ts[ind, "price"]@data - 1000)
> get.cleanPrice.ts = seriesClean(get.ts[, "price"], n = 51, k = 25, maxDev = 3)
> get.cleanPrice.ts = excludeTimes(get.cleanPrice.ts, exclude = "closed", openTime = timeCalendar(h = 9, min = 30),
 + closeTime = timeCalendar( h = 16), type = "d")
> # Figure 14B
> plot(get.cleanPrice.ts[timeEvent("5/13/1997"), ])
```

Figure 14B shows May 5th, 1997 for General Electric after I altered the data by hand and called seriesData again. while this is still not perfect it will work for now.

Now that we have our TAQ data loaded we can use the realized library to perform various calculations. The theory and citations can be found in the Realized User's Manual. The realized library can take timeSeries as arugments to it'a main functions, however, it is more efficient to convert these to class 'realizedObject'

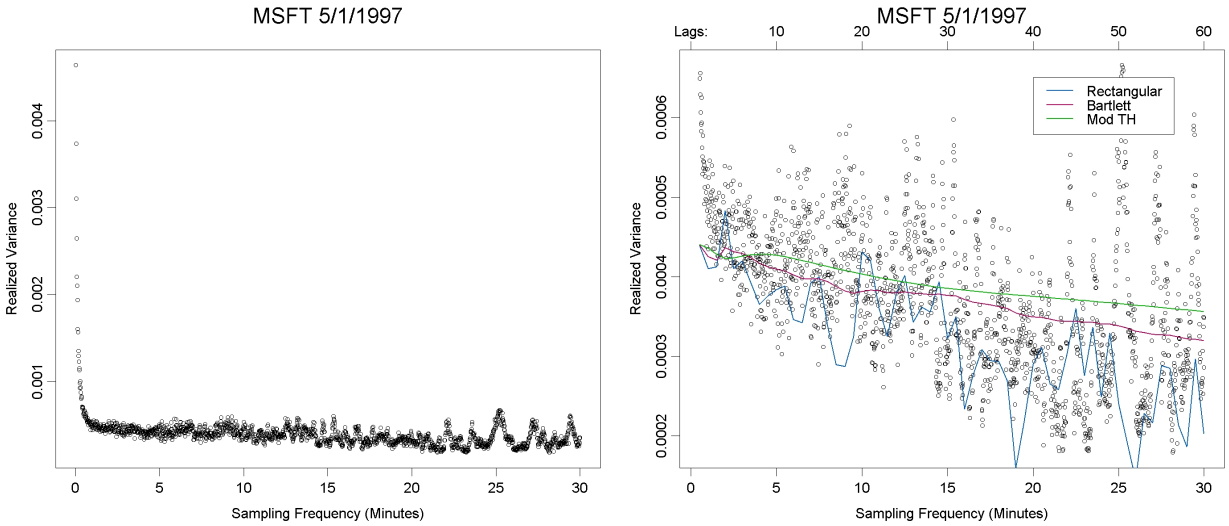


Figure 15 and 16

so that each call to the realized functions does not need to convert the timeSeries. To compute traditional realized variance sample at five minute sampling frequency for May 1, 1997 Microsoft:

```
> library(realized)
> msftt.real.19970105 <- realizedObject(msftt.cleanPrice.ts[timeEvent("5/1/1997"), ], cts = T, makeReturns = T)

> rRealizedVariance(msftt.cleanPrice.ts[timeEvent("5/1/1997"), ], period = 5, args = list(align.period = 60),
 + cts = T, makeReturns = T)
[1] 0.0004817088
```

or using the realizedObject:

```
> rRealizedVariance(msftt.real.19970105, period = 5, args = list(align.period = 60), cts = T)
[1] 0.0004817088
```

A one day realized variance signature plot for May 1, 1997 for MSFT for every second from one second to 20 minutes is displayed in Figure 15.

```
> # Figure 15
> real.naive <- rSignature(1:1800, x = msftt.real.19970105, type = "naive", xscale = 1/60)
> plot(real.naive, xlab = "Sampling Frequency (Minutes)", ylab = "Realized Variance", main = "MSFT 5/1/1997", cex
= 0.5)
```

Figure 16 shows the same signature plot omitting the first 30 seconds in order to show the variability of the estimation. It also displays three different kernel estimators as a function of lags (shown at the top of the plot). These three kernel estimates use the Rectangular, Bartlett and Modified Tukey-Hanning kernels with the data aligned to a one minute frequency.

```
> # Figure 16
> plot(x = real.naive$x[ - c(1:30)], y = real.naive$y[ - c(1:30)], xlab = "Sampling Frequency (Minutes)",
```

```
 + ylab = "Realized Variance", main = "MSFT 5/1/1997", cex = 0.5)
> # Bartlett kernel
> Rectangular kernel real.rect <- rSignature(1:60, x = msftt.real.19970105, type = "kernel", xscale = 1/2,
 + args = list(align.period = 60, type = "rectangular"))
> lines(real.rect, col = 2, lwd = 2, lty = 1)
> axis(3, c(0, (1:6) * 5), c("Lags:", as.character((1:6) * 10)))
> # Bartlett kernel
real.bart <- rSignature(1:60, x = msftt.real.19970105, type = "kernel", xscale = 1/2,
 + args = list(align.period = 60, type = "bartlett"))
> lines(real.bart, col = 3, lwd = 2, lty = 1)
> # Modified Tukey Hanning kernel
real.mth <- rSignature(1:60, x = msftt.real.19970105, type = "kernel", xscale = 1/2,
 + args = list(align.period = 60, type = "mth"))
> lines(real.mth, col = 4, lwd = 2, lty = 1)
```

Using a one minute sampling frequency with a kernel estimator is what Barndorff-Nielsen et al. (2004) suggest. However, with the realized library it is easy to try different sampling frequencies. I have found that if the data is clean the highest sampling frequency create estimates that are less variable with respect to their inputs. Figure 17 displays the same one day signature plot as above, however, the kernel estimates use the highest sampling frequency possible.

```
> # Figure 17
> plot(x = real.naive$x[ - c(1:30)], y = real.naive$y[ - c(1:30)], xlab = "Sampling Frequency (Minutes)",
 + ylab = "Realized Variance", main = "MSFT 5/1/1997", cex = 0.5)
> real.rect <- rSignature(1:300, x = msftt.real.19970105, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "rectangular"))
> lines(real.rect, col = 2, lwd = 1, lty = 2)
> axis(3, c(0, (1:6) * 5), c("Lags:", as.character((1:6) * 50)))
> # Bartlett kernel
> real.bart <- rSignature(1:300, x = msftt.real.19970105, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "bartlett"))
> lines(real.bart, col = 3, lwd = 2, lty = 1)
> # Modified Tukey Hanning kernel
> real.mth <- rSignature(1:300, x = msftt.real.19970105, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "mth"))
> lines(real.mth, col = 4, lwd = 2, lty = 1)
> legend(20, 0.00065, c("Rectangular", "Bartlett", "MTH"), lwd = c(1, 2, 2), col = c(2, 3, 4), lty = c(2, 1, 1))
```

The plots above were created all in calendar time sampling (CTS). To see how these estimates change when tick time sampling (TTS) is used we set cts = F. These are displayed in Figure 18. The difference here is that instead of sampling every second in the signature plot, we are sampling every tick. Notice that TTS cannot be used for most covariance or correlation calculations.

```
> # Figure 18
> msftt.real.19970105.tts <- realizedObject(msftt.cleanPrice.ts[timeEvent( "5/1/1997"), ], cts = F,
 + makeReturns = T)
> real.naive.tts <- rSignature(1:1800, x = msftt.real.19970105.tts, type = "naive", xscale = 1/60)
> plot(x = real.naive.tts$x[ - c(1:30)], y = real.naive.tts$y[ - c(1:30)],
 + xlab = "Sampling Frequency (Ticks)", ylab = "Realized Variance", main = "MSFT 5/1/1997", cex = 0.5)
> real.rect <- rSignature(1:300, x = msftt.real.19970105.tts, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "rectangular"))
> lines(real.rect, col = 2, lwd = 1, lty = 2)
> axis(3, c(0, (1:6) * 5), c("Lags:", as.character((1:6) * 50)))
```

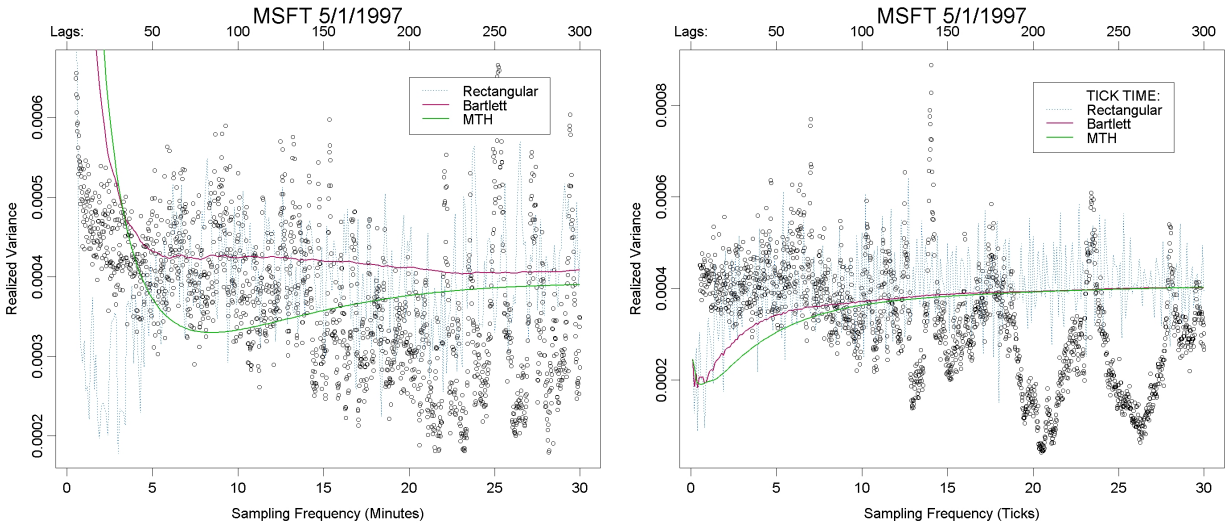


Figure 17 and 18

```
> # Bartlett kernel
> real.bart <- rSignature(1:300, x = msftt.real.19970105.tts, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "bartlett"))
> lines(real.bart, col = 3, lwd = 2, lty = 1)
> # Modified Tukey Hanning kernel
> real.mth <- rSignature(1:300, x = msftt.real.19970105.tts, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "mth"))
> lines(real.mth, col = 4, lwd = 2, lty = 1)
> legend(20, 0.00085, c("TICK TIME:", "Rectangular", "Bartlett", "MTH"), lwd = c( 0, 1, 2, 2),
 + col = c(0, 2, 3, 4), lty = c(0, 2, 1, 1))
```

There can be a big difference between realized variance estimates when the sampling frequency differs by a small amount. For an example of this we turn back to the CTS one day signature plot for MSFT May 1, 1997. In the top panel of Figure 19 we pick two estimates of realized variance that are 14 seconds apart. An accumulation plot appears in the bottom panel of this figure along with how the timings line up in the center panel.

```
> # Figure 19
> par(mfrow = c(3, 1))
> plot(x = real.naive$x[ - c(1:30)], y = real.naive$y[ - c(1:30)], xlab = "Sampling Frequency",
 + ylab = "Realized Variance", main = "MSFT 5/1/1997", cex = 0.5)
> tmp <- real.naive$y[880:920]
> ind.max <- which(tmp == max(tmp)) + 880
> ind.min <- which(tmp == min(tmp)) + 880
> points(real.naive$x[c(ind.max, ind.min)], real.naive$y[c(ind.max, ind.min)],
 + col = c(2, 3), cex = 1.5)
> plot(rCumSum(msftt.real.19970105, 1), pch = ".", xlab = "Seconds", ylab = "Cummulative Return")
> lines(rCumSum(msftt.real.19970105, ind.max), col = 2)
> lines(rCumSum(msftt.real.19970105, ind.min), col = 3)
> plot(rAccumulation(msftt.real.19970105, ind.max), col = 2, type = "l",
 + ylab = "RV Accumulation", xlab = "Seconds")
> lines(rAccumulation(msftt.real.19970105, ind.min), col = 3)
```

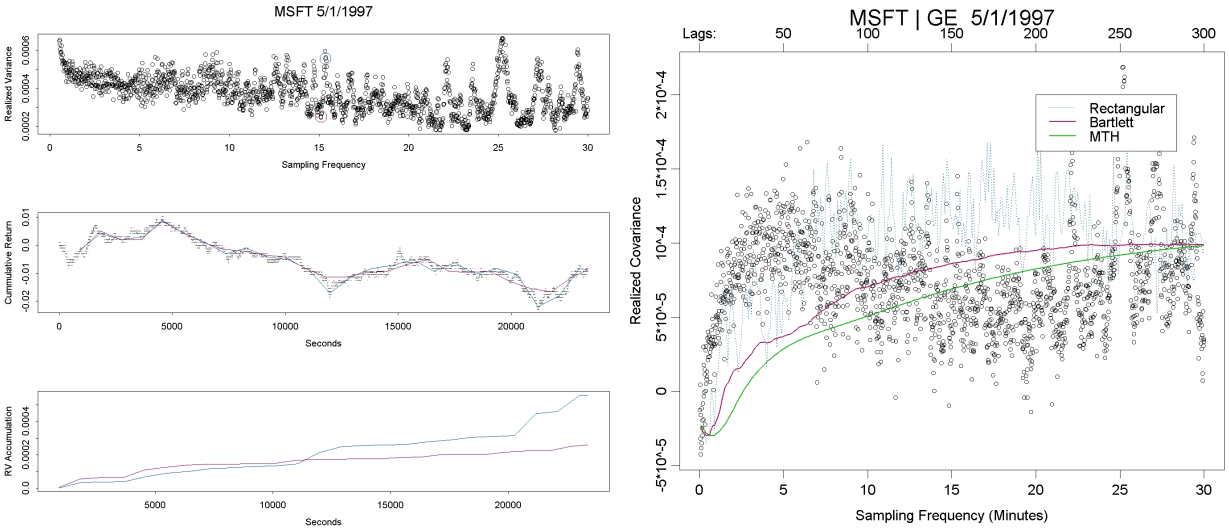


Figure 19 and 20

Figure 20 displays multiple covariance estimates for MSFT and GE May 1, 1997. These are the kernel type estimates using Rectangular, Bartlett, and Modified Tukey-Hanning kernels at the highest sampling frequency.

```
> # Figure 20
> get.real.19970105 <- realizedObject(get.cleanPrice.ts[timeEvent("5/1/1997"), ],
 + cts = T, makeReturns = T)
> real.naive <- rSignature(1:1800, x = msftt.real.19970105, y = get.real.19970105,
 + type = "naive", xscale = 1/60)
> par(mfrow = c(1, 1))
> plot(real.naive, xlab = "Sampling Frequency (Minutes)", ylab = "Realized Covariance",
 + main = "MSFT | GE 5/1/1997", cex = 0.5)
> rect <- rSignature(1:300, x = msftt.real.19970105, y = get.real.19970105, type = "kernel", xscale = 1/10,
 + args = list(align.period = 1, type = "rectangular"))
> lines(real.rect, col = 2, lwd = 1, lty = 2)
> axis(3, c(0, (1:6) * 5), c("Lags:", as.character((1:6) * 50)))
> # Bartlett kernel
> real.bart <- rSignature(1:300, x = msftt.real.19970105, y = get.real.19970105, type = "kernel",
 + xscale = 1/10, args = list(align.period = 1, type = "bartlett"))
> lines(real.bart, col = 3, lwd = 2, lty = 1)
> # Modified Tukey Hanning kernel
> real.mth <- rSignature(1:300, x = msftt.real.19970105, y = get.real.19970105, type = "kernel",
 + xscale = 1/10,args = list(align.period = 1, type = "mth"))
> lines(real.mth, col = 4, lwd = 2, lty = 1)
> legend(20, 0.0002, c("Rectangular", "Bartlett", "MTH"), lwd = c(1, 2, 2), col = c(2, 3, 4), lty = c(2, 1, 1))
```

Again, we study the difference between two realized covariance estimates. Figure 21 shows two different estimates that are 75 seconds apart, along with the alignment of each asset and a realized covariance accumulation plot.

```
> #Figure 21
> par(mfrow = c(4, 1))
```

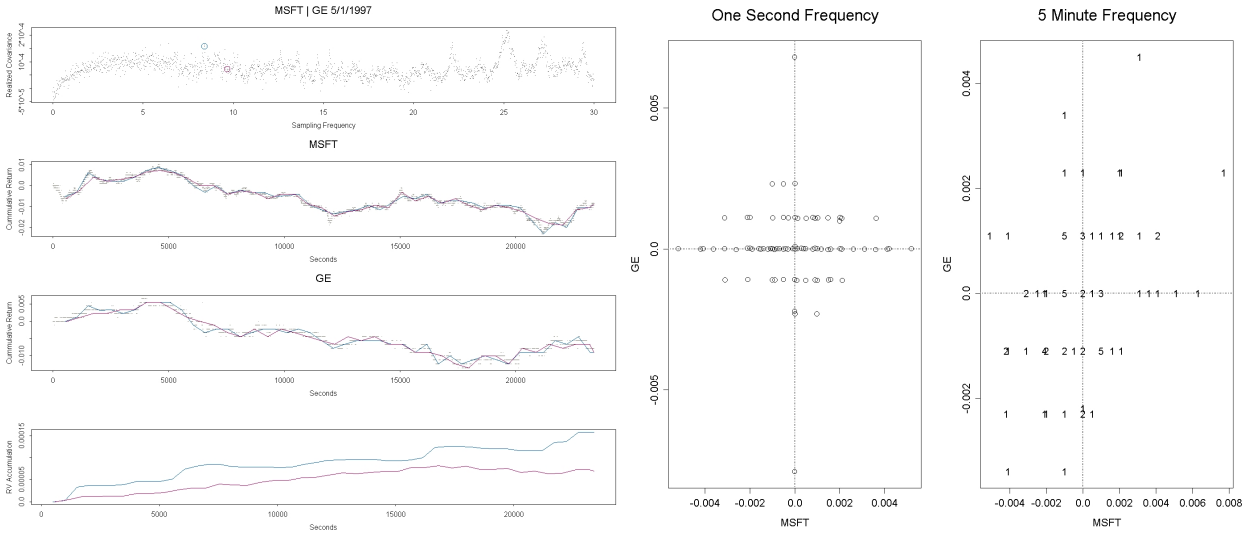


Figure 21 and 22

```
> plot(real.naive, xlab = "Sampling Frequency", ylab = "Realized Covariance", main = "MSFT | GE 5/1/1997"
 + , pch = ".")
> tmp <- real.naive$y[500:600]
> ind.max <- which(tmp == max(tmp)) + 500
> ind.min <- which(tmp == min(tmp)) + 500
> points(real.naive$x[c(ind.max, ind.min)], real.naive$y[c(ind.max, ind.min)], col = c(2, 3), cex = 0.8)
> plot(rCumSum(msftt.real.19970105, 1), pch = ".", xlab = "Seconds", ylab = "Cummulative Return",
 +main = "MSFT", col = 16)
> lines(rCumSum(msftt.real.19970105, ind.max), col = 2)
> lines(rCumSum(msftt.real.19970105, ind.min), col = 3)
> plot(rCumSum(get.real.19970105, 1), pch = ".", xlab = "Seconds", ylab = "Cummulative Return",
 + main = "GE", col = 16)
> lines(rCumSum(get.real.19970105, ind.max), col = 2)
> lines(rCumSum(get.real.19970105, ind.min), col = 3)
> plot(rAccumulation(msftt.real.19970105, y = get.real.19970105, ind.max), col = 2, type = "l",
 + ylab = "RV Accumulation", xlab = "Seconds")
> lines(rAccumulation(msftt.real.19970105, y = get.real.19970105, ind.min), col = 3)
```

Nonsynchronous trading is a large problem in the calculation of realized covariance. To see how co-returns occur for different timings the rScatterReturns function is used as in Figure 22.

```
> # Figure 22
> par(mfrow = c(1, 2))
> rScatterReturns(msftt.real.19970105, y = get.real.19970105, period = 1, xlab = "MSFT", ylab = "GE",
 + main = "One Second Frequency")
> rScatterReturns(msftt.real.19970105, y = get.real.19970105, period = 300, xlab = "MSFT", ylab = "GE",
 + numbers = T, main = "5 Minute Frequency")
```

Further analysis to see the co-zero returns is the rc.zero function which is similar to any rc.* function. It can also be put in the rSignature function as in Figure 23.

```
> # Figure 23
> par(mfrow = c(1, 1))
> plot(rSignature(1:1800, msftt.real.19970105, y = get.real.19970105, type = "zero"))
```

In order to calculate a realized covariance or correlation matrix you should pass the rRealizedVariance function a 'seriesMerged' timeSeries or you can use the merge function objects of type "realizedObject"

```
> both.real.19970105 <- merge(msftt.real.19970105, get.real.19970105)
> rRealizedVariance(x = both.real.19970105, cts = T, args = list(align.period = 5 ))
[,1] [,2]
[1,] 0.0022036758 -0.0000162177
[2,] -0.0000162177 0.0008898262

> rRealizedVariance(x = seriesMerge(msftt.cleanPrice.ts[timeEvent("5/1/1997"), ], get.cleanPrice.ts[timeEvent("5/1/1997"),
], pos = "union"), cts = T, args = list(align.period = 5), makeReturns=T)
[,1] [,2]
[1,] 0.0022036758 -0.0000162177
[2,] -0.0000162177 0.0008898262
```

The last example in this supplement is an example of calculating realized variance, covariance and correlation over all days in the sample. The results appear in Figure 22.

```
> days <- c("5/1/1997", "5/2/1997", "5/5/1997", "5/6/1997", "5/7/1997", "5/8/1997", "5/9/1997", "5/12/1997",
 + "5/13/1997", "5/14/1997", "5/15/1997")
> dates <- timeEvent(days)
> covs <- array(dim = c(length(dates), 2, 2))
> cors <- array(dim = c(length(dates), 2, 2))
> for(i in 1:length(dates)) {
 + msft.real <- realizedObject(msftt.cleanPrice.ts[dates[[i]], ], cts = T, makeReturns = T)
 + ge.real <- realizedObject(get.cleanPrice.ts[dates[[i]], ], cts = T, makeReturns = T)
 + both.real <- merge(msft.real, ge.real)
 + covs[i, , ] <- rRealizedVariance(x = both.real, cts = T, type = "avg", lags = 600, args = list(align.period =
1), cor = F)
 + cors[i, , ] <- cov2cor(covs[i, , ])
 + }

> # Figure 22
>par(mfrow = c(2, 2))
> plot(timeSeries(covs[1:length(dates), 1, 1], positions = days), main = "MFST Var")
> plot(timeSeries(covs[1:length(dates), 1, 2], positions = days), main = "Cov")
> plot(timeSeries(cors[1:length(dates), 1, 2], positions = days), main = "Cor")
> plot(timeSeries(covs[1:length(dates), 2, 2], positions = days), main = "GE Var" )
```

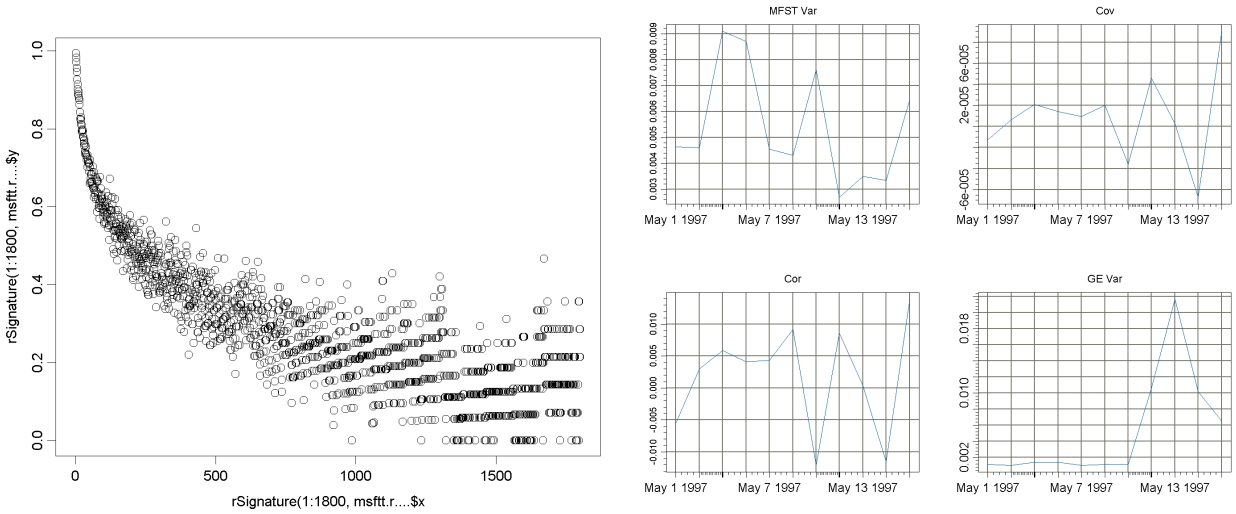


Figure 21 and 22

References

Ole E. Barndorff-Nielsen, Peter Reinhard Hansen, Asger Lunde, and Neil Shephard. Regular and modified kernel-based estimators of integrated variance: The case with independent noise. (2004fe20), 2004. available at http://ideas.repec.org/p/sbs/wpsefe/2004fe20.html.

B. Yan and E. Zivot. Analysis of high-frequency financial data with s-plus. *Software Library: http://faculty.washington.edu/ezivot/research/HFLibrary.SSC*, 2003.

E. Zivot. Finmetrics 3.0 high frequency supplement.