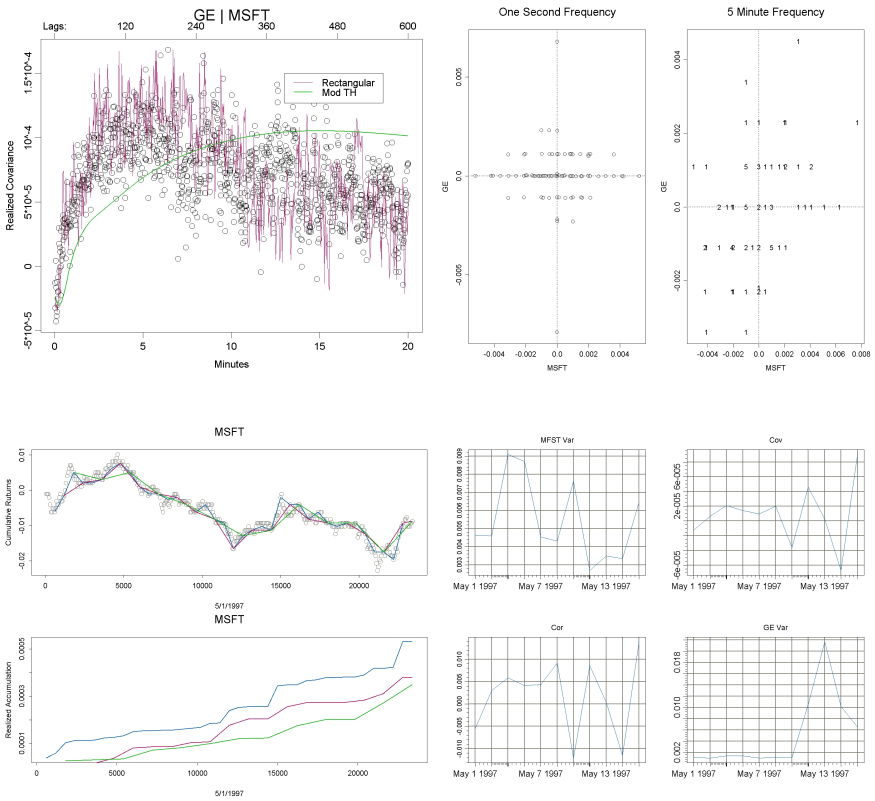
"Realized" Software Package 0.8

Scott Payseur (spayseur@u.washington.edu)

September 26, 2007

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



Abstract

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. This version contains the following improvements as compared to version 0.7: an interface to SPlus timeSeries objects (see section SPlus supplement), a new realizedObject, new example data, and multiple bug fixes.

Contents

1 Introduction

This library contains current realized variance, covariance and correlation estimators. Shephard and Barndorff-Nielsen (2005) provide a thorough survey of this literature. The library also contains multiple plotting functions to help study the problem of realized calculations, as well as, the differences between estimators. While there has been many advances in solving the problem of realized estimation it is apparent that the best estimate depends on each different data set, duration, and asset class. This library will not only calculate estimates of realized quantities but also help the practitioner verify that the estimation method they are using gives a reliable estimate. The library does not include any optimal sampling frequency, optimal subgrid, or optimal lag length calculations, however, this is planned for version 1.0.

1.1 Data

The focus of this library is the calculation and exploration of realized variance, covariance and correlation estimates. I have attempted to make the package data agnostic and leave it up to the user to clean and align the data. S+Finmetrics 3.0 and Yan and Zivot (2003) have S+ utility functions that will help in this effort. After discussions with many academics and practitioners it appears that everyone has their own data management practices and this library was written to handle very generic data types that can be created from any data management strategy.

For those starting from scratch, I will briefly discuss the data management strategy that I use for my dissertation. High frequency data poses a memory issue that is not common in most time-series analysis. For a ballpark figure, say I have 10 years worth of data for 10 assets that trade at a similar frequency that MSFT in 2004 does (roughly 15,000 trades per day). The amount of transactions for the whole period is 10*10*250*15,000 = 375,000,000. At some point loading this into memory becomes infeasible. However, all of the realized variance, covariance and correlation calculations are computed over a particular time interval. This time interval corresponds directly to the integration bounds for integrated variance[1] (This time interval is usually one trading day). This allows us to keep only one day of data in memory at a time.

I imported NYSE Trades and Quotes (TAQ) files into a MySQL database with indices on trade date and ticker. From there I used the RODBC library for R (Lapsley (2007)) or the importData command for S+ to query for one day and one stock at a time. After aligning the log-returns to the highest possible frequency, using the `realizedObject` function, the Realized library is used. For more information on setting up your

[1] Assuming that the log-price process follows an Ito process

own data see Section 6 or the S+Supplement for S+ users with S+Finmetrics 3.0 or Yan and Zivot (2003)'s script[2].

There are 11 days of Microsoft (MSFT) and General Electric (GE) high frequency transactions included in this package. This data comes from the TAQ. I have cleaned this data using S+Finmetrics 3.0 and used the median transaction for transactions with the exact same time stamp. The dataset `msft.real.tts` is aligned to tick-time sampling (TTS) and `msft.real.cts` is aligned to calendar-time sampling (CTS) using the previous tick method (for further discussion on alignment of trades see Hansen and Lunde, 2006). It is important to note that calculation of realized covariance and correlation must be performed on CTS data, except for the Hayashi and Yoshida (2005) estimator.

The sample data is loaded with the following commands:

```
> library(realized)
> data(msft.real.cts)
> data(msft.real.tts)
> data(ge.real.cts)
```

Each of `msft.real.cts`, `msft.real.cts`, and `msft.real.cts` objects is a list of objects representing a particular market day of data from 5/1/1997 to 5/15/1997. These objects are of type 'realizedObject', Section 6 contains examples of creating your own realizedObjects. The first 10 MSFT returns for 5/1/1997 CTS aligned data are:

```
> msft.real.cts[[1]][1:10]
Realized Object:  (length = 10 , cts= TRUE )
data milliseconds
0.000000000 34201000
0.000000000 34202000
0.001024066 34203000
0.000000000 34204000
0.000000000 34205000
0.000000000 34206000
-0.001024066 34207000
0.000000000 34208000
0.000000000 34209000
0.000000000 34210000
```

The `rCumSum` function plots the cumulative returns for a particular alignment of the data. The first six days of MSFT TTS data at the highest frequency of every tick is plotted in Figure 1.

```
# Figure 1:  Multiple days of MSFT cumulative returns.
> par(mfrow=c(2,3))
> tmp <- sapply(1:6, function(x, rets){plot(rCumSum(rets[[x]]), ylab="Cumulative Return", xlab="")}, rets=msft.real.tts)
```

The dates for the example data are included for ease of use and are:

```
> data(dates.example)
> dates.example
```

[2] Available at http://faculty.washington.edu/ezivot/

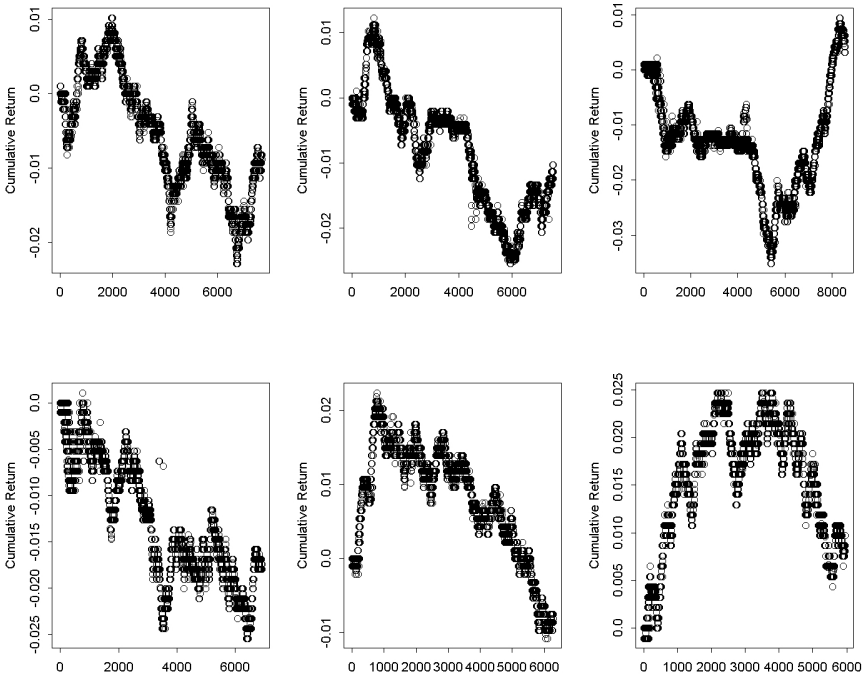


Figure 1: MSFT cumulative returns aligned at highest frequency in tick time.

```
[1] "5/1/1997" "5/2/1997" "5/5/1997" "5/6/1997" "5/7/1997" "5/8/1997"
[7] "5/9/1997" "5/12/1997" "5/13/1997" "5/14/1997" "5/15/1997"
```

2 Realized Variance

This section discusses the theory behind realized variance, problems with realized variance estimators and introduces the realized variance functionality of the this package. This functionality includes the traditional realized variance estimator (Andersen et al., 2001; Barndorff-Nielson and Sheppard, 2002), kernel based estimators (Zhou, 1996; Hansen and Lunde, 2006, 2004; Barndorff-Nielsen et al., 2004), sub-sample based estimators (Zhang et al., 2005), and signature plots (Fang, 1996; Andersen et al., 2000; Payseur, 2007).

Under certain assumptions, realized variance is an unbiased estimate of conditional variance. An overview of the theory follows. For each day consider the observed price process to be:

$$P_{t_i,m} = P^*_{t_i,m} U_{t_i,m} \quad i = 1, 2, ..., (m+1) \tag{1}$$

where t_i represents the ith index the $m+1$ observation price series, which can be TTS or CTS with a $\frac{1}{m}$ sampling frequency. $P^*_{t_i,m}$ is the efficient price and $U_{t_i,m}$ is microstructure noise. A log-return transformation

yields[3]:

$$\underbrace{p_{t_i,m} - p_{t_{i-1},m}}_{y_{i,m}} \equiv \underbrace{p^*_{t_i,m} - p^*_{t_{i-1},m}}_{y^*_{i,m}} + \underbrace{u_{t_i,m} - u_{t_{i-1},m}}_{e_{i,m}} \quad i = 1, 2, ..., m \tag{2}$$

Where $p^* \equiv ln(P^*)$, $p \equiv ln(P)$, and $u \equiv ln(U)$. Assume that the efficient log price process, p^*, is a continuous local martingale:

$$dp^* = \sigma(t)dw(t) \tag{3}$$

where $dw(t)$ is a standard Brownian motion and $\sigma(t)$ is the independent cadlag spot volatility process. The quadratic variation over the sub-interval $T_{i,m}$ to $T_{i-1,m}$ ($T_{i,m} - T_{i-1,m}$ is usually one day) is defined by:

$$QV_{i,m} = E(y^{*2}_{i,m}) \;\; 1, 2, ..., m \tag{4}$$

The realized variance of the efficient price process

$$RV^{(m)}_* = \sum_{i=1}^{m} y^{*2}_{i,m} \tag{5}$$

converges to quadratic variation as $m \to \infty$. (This quantity converges to integrated variance if the log-price process is Ito.)

However, $y^*_{i,m}$ is latent and the observable realized variance based on $y_{i,m}$

$$RV^{(m)} = \sum_{i=1}^{m} y^2_{i,m} \tag{6}$$

is biased as $m \to \infty$ due to the micro structure noise, $e_{i,m}$. The bias can be shown with a simple expansion:

$$RV^{(m)} = \sum_{i=1}^{m} y^2_{i,m} = \underbrace{\sum_{i=1}^{m} y^{*2}_{i,m}}_{RV^{(m)}_*} + \underbrace{\sum_{i=1}^{m} e^2_{i,m}}_{RV^{(m)}_u} + \underbrace{2 \cdot \sum_{i=1}^{m} y^*_{i,m} e_{i,m}}_{2RC^{(m)}_{*,u}} \tag{7}$$

[3] All of the functions from the Realized package assume that your log return vector is in the form $y_{i,m}$ with m at as high a frequency as possible.

The bias is composed of two components. The realized variance of the noise process, $RV_u^{(m)}$, and the realized covariance between the efficient price and noise processes, $RC_{*,u}^{(m)}$. This bias is typically positive as $m \to \infty$ (always positive assuming that the noise and efficient price processes are uncorrelated) due to a negative auto correlation in the noise process known as bid-ask-bounce.

2.1 Signature Plots

The upward bias in the traditional realized variance estimator at high sampling frequencies can be seen graphically in a volatility signature plot (Andersen et al., 2000 and Fang, 1996). This tool displays the sample average realized variance across n days (often for one year or month), $\overline{RV}^{(m)} = \frac{1}{n}\sum_{t=1}^{n} RV_t^{(m)}$, as a function of different sampling frequencies, $\frac{1}{m}$.

The function `rSignature` is used to create an average signature plot. As mentioned above this package takes a hands off approach to data management. For this reason the when the `rSignature` is used for multiple days the "iteration function" and iterations arguments must be specified[4]. The iteration function must return an object of type `realizedObject` the interval $T_{i,m} - T_{i-1,m}$ (usually each trading day). The other parameters in the `rSignature` function will be discussed in the next section. Since the example data is a simple list of `realizedObjects` the iteration function simply returns the next element of the list:

```
> par(mfrow=c(1,1))
> simpleIteration <- function(x, i,args){x[[i]]}
```

A six day variance signature plot for MSFT 5/1/1997 to 5/8/1997 is displayed in Figure 2.

```
# Figure 2
> plot(rSignature((1:120)*10+1, msft.real.cts, xscale=1/60, iteration.funct="simpleIteration",
                  iterations=1:6), ylab="Realized Variance", xlab="Sampling Frequency (Minutes)", main="MSFT",
                sub=paste(dates.example[[1]], dates.example[[11]], sep=" - "))
```

2.1.1 One Day Signature Plots

Averaging across many days can obfuscate the volatile behavior realized variance estimates at low frequencies. This range is where the estimator is believed to be unbiased. In the realized variance and covariance literature there is often a discussion of whether to sample at 5, 10, 15 or 20 minutes. To make this determination practitioners often use the frequency where the bias ends on the signature plot (e.g. somewhere around a 3 minute sampling frequency in Figure 2) . However, there can be extremely large variability between estimates even after the sampling frequency is low enough to correct the micro structure bias. Payseur (2007) uses

[4]In my own research the dataset is an open ODBC connection to my database, and the iteration function allows me to query the next day.

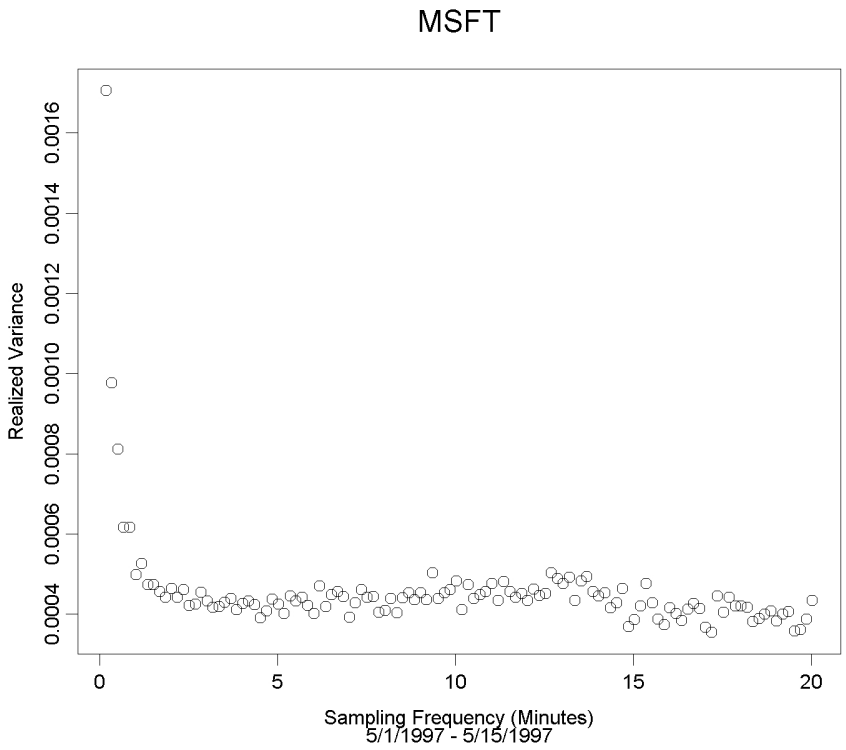


Figure 2: Six Day Average Signature Plot for MSFT 5/1/1997 to 5/8/1997

the one day signature plot to demonstrate how volatile estimates can be to small differences in the sampling frequency.

The `rSignature` function takes the following inputs:

```
> args(rSignature)
function (range, x, y = NULL, type = "naive", cor = FALSE, args = list(),
 xscale = 1, iteration.funct = "", iterations = NULL, plotit = FALSE, cts
 = TRUE, makeReturns = FALSE)
```

- `range` is a numeric vector that specifies a range of values as inputs to realized estimators.
- `x`is a `realizedObject` for the first asset
- `y`is a `realizedObject` for the second asset in the case of covariance or correlation
- `type`is the type of estimator to use. If `y=NULL`then then type must be the * string from the `rv.*`functions and if `y is not NULL` it must be the * string from the `rc.*`functions.
- `cor`if TRUE and `y is not NULL` a realized correlation signature is returned.
- `xscale` controls the scale of the x axis (for instance for returns sampled at every second, if we want the xaxis to be in minutes then xscale=1/60)

A one day signature plot of MSFT for May 1, 1997, at a sampling frequency of each second from one second to 20 minutes the command is:

```
> test.sig <- rSignature(1:1200, msft.real.cts[[1]], xscale=1/60)
```

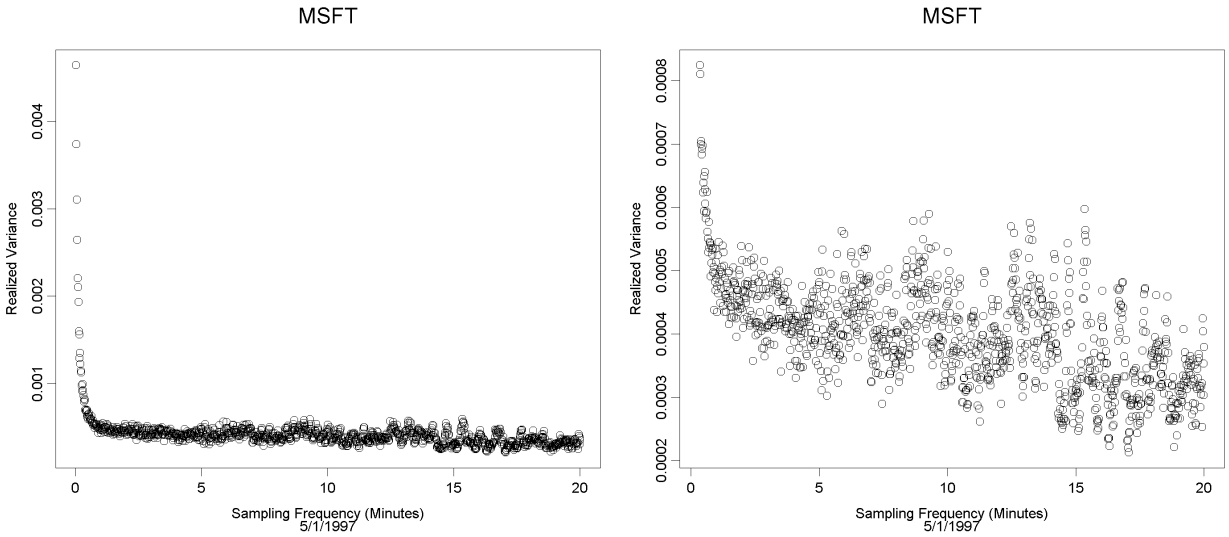


Figure 3A and 3B: One Day Signature Plots for MSFT, 5/1/1997

the object that is returned by the `rSignature` function has an `x` and a `y` component so that a simple call to plot will plot the sampling frequency and realized variance on the horizontal and verical axis respectively.

```
> names(test.sig)
[1] "x" "y" "xgrid" "type" "cor" "cov" "cts"
```

Figure 3A shows this one day realized variance signature plot.

```
# Figure 3A
> plot(test.sig, ylab="Realized Variance", xlab="Sampling Frequency (Minutes)"
             , main="MSFT", sub=dates.example[[1]])
```

These one day signature plots are used throughout this users manual to demonstrate the variability of each estimator with respect to their inputs. In order to demonstrate the variability of the realized variance estimator it is helpful to leave out the highest frequencies on the plot:

```
# Figure 3B
>plot(x=test.sig$x[-(1:20)], y=test.sig$y[-(1:20)], ylab="Realized Variance", xlab="Sampling Frequency (Minutes)",
main="MSFT", sub=dates.example[[1]])
```

The variability of the traditional realized variance estimator is easy to see on this one day plot. If a sampling frequency of around 5 minutes is used then the estimate can lie anywhere between 0.0003 and 0.0006. This is a significant difference and should discourage simply choosing a sampling frequency for the realized variance estimate. These graphs present only stylized facts about the variability of an estimate. An obvious tool that is not implemented in this package is the use of standard errors with estimates of realized quantities. The problem with these standard errors is that they are based on the fourth moments of the log-returns and suffer from similar bias problems as the realized estimators themselves.

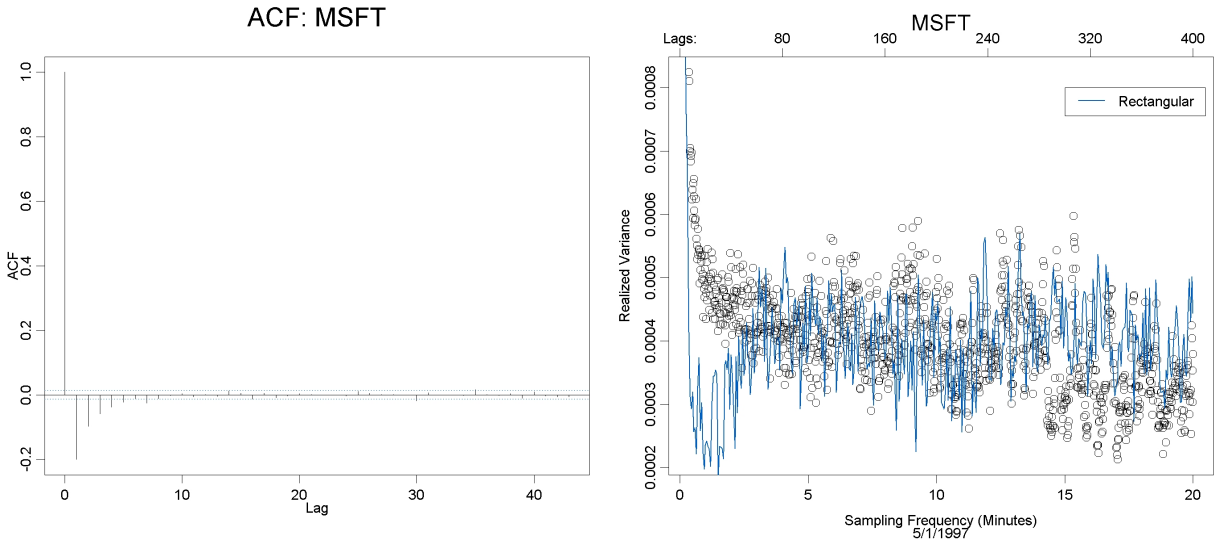


Figure 4A and 4B: ACF for MSFT May 1, 1997 and Rectangular Kernel Estimator of Realized Variance as a Function of Lags.

2.2 Kernel Estimators

The realized variance signature plots in Figures 2, 3A and 3B demonstrate the micro structure noise bias in realized variance estimates at high frequencies. These biases are created by a bid-ask-bounce induced negative auto correlation in the micro structure noise. This can be observed directly from an auto-correlation plot of the returns at the highest frequency:

```
> acf(msft.real.cts[[1]]$data, main="ACF: MSFT")
```

Hansen and Lunde (2006, 2004); Zhou (1996); Barndorff-Nielsen et al. (2004) (add BNSHL) propose the use of kernel estimators to eliminate the bias that is caused by the auto-correlation microstructure noise auto-correlation. These estimators are in the same spirit as long run variance estimators that are common in time series analysis.

$$RV_{Kernel} = \hat{\gamma_0} + \sum_{h=1}^{H} k\Big(\frac{h-1}{H}\Big)\Big\{\hat{\gamma_h} + \hat{\gamma_{-h}}\Big\} \tag{8}$$

$$\hat{\gamma}_h \equiv \sum_{i=1}^{m} y_{i,m} y_{i+h,m} \tag{9}$$

and can be estimated by:

$$\tilde{\gamma}_h \equiv \frac{m}{m-h} \sum_{i=1}^{m} y_{i,m} y_{i+h,m} \tag{10}$$

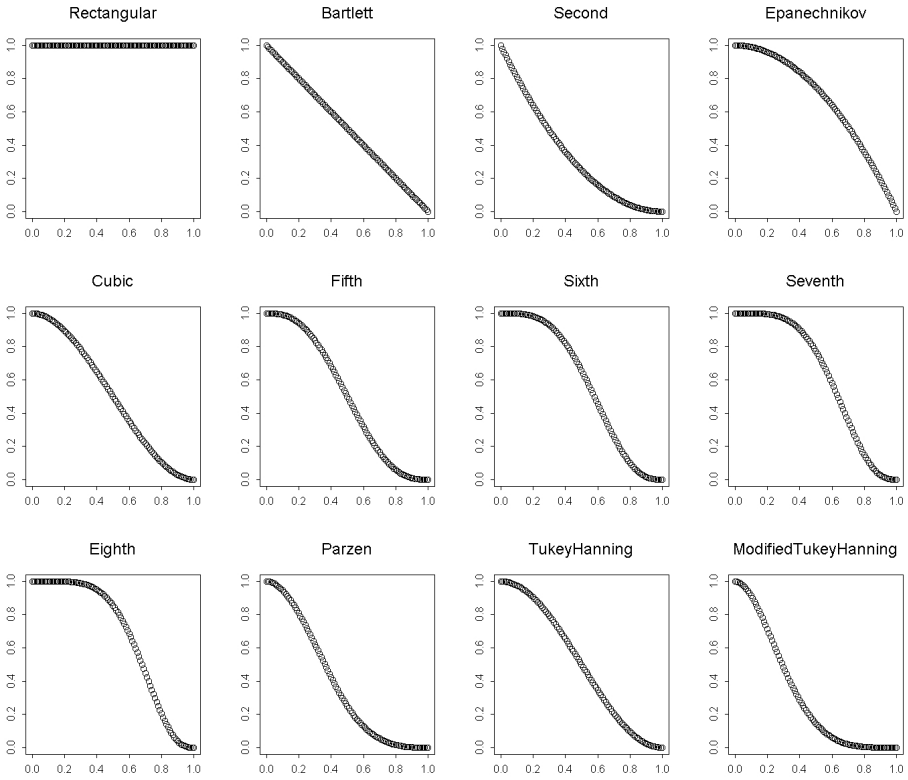


Figure 5: Available Kernels

Figure 4B demonstrates kernel estimates for the rectangular kernel and sampled at the highest frequency as a function of lags overlayed on the original signature plot from Figure 3B.

```
# Figure 4B
> plot(x=test.sig$x[-(1:20)],y=test.sig$y[-(1:20)],ylab="Realized Variance", xlab="Sampling Frequency (Minutes)",
main="MSFT",sub=dates.example[[1]])
 > test.rect <- rSignature(1:400, msft.real.cts[[1]], xscale=1/20, type="kernel", args=list(type="rectangular"))
 > lines(test.rect, col=2, lwd=2)
 > axis(3, c(0,(1:5)*4), c("Lags:",as.character((1:5)*80)))
 > legend(15,.0008,c("Rectangular"), lwd=c(2), col=c(2))
```

The rectangular kernel realized variance estimator does have less variability with respect to its inputs (lags) than the traditional realized variance estimator with respect to sampling frequency. However, we will see that different kernels will give far better results.

The available kernels, $k\left(\frac{h-1}{H}\right)$, are the same as in Barndorff-Nielsen et al. (2004):

```
> rKernel.available()
[1] "Rectangular" "Bartlett" "Second" "Epanechnikov" "Cubic" "Fifth"
[7] "Sixth" "Seventh" "Eighth" "Parzen" "TukeyHanning" "ModifiedTukeyHanning"
```

And are plotted below:

```
# Figure 5
> 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))
```

In Figure 6A the Modified Tukey-Hanning and Bartlett kernels are displayed and show better performance

with respect to variability of the estimators with respect to their inputs. This is important to the practitioner because it means that their estimate will not vary much with their choice of lag length. `Payseur (2007)` recommends the use of this type of plot as a "spot check" to verify that the estimators are performing in a consistent manor as the data used changes.

```
# Figure 6A
> par(mfrow=c(1,1))
> plot(x=test.sig$x[-(1:20)],y=test.sig$y[-(1:20)],ylab="Realized Variance", xlab="Minutes", main="MSFT",sub=dates.example[[1]])

> test.mth <- rSignature(1:400, msft.real.cts[[1]], xscale=1/20, type="kernel", args=list(type="mth"))
> test.bart <- rSignature(1:400, msft.real.cts[[1]], xscale=1/20, type="kernel", args=list(type="bartlett"))
> lines(test.mth, col=3, lwd=2)
> lines(test.bart, col=4, lwd=2)
> axis(3, c(0,(1:5)*4), c("Lags:",as.character((1:5)*80)))
> legend(15,.0008,c("Mod T-H", "Bartlett"), lwd=c(2,2), col=c(3,4))
```

2.3 Subsample Estimators

Zhang et al. (2005) categorized $RV^{(m)}$ as the fifth-best realized variance estimator, choosing a lower sampling frequency as fourth-best, and sampling at an optimal lower sampling frequency as third-best. Their first and second best approaches are based on the idea of sub-sampling. When we choose a lower frequency, or sparse grid, we are only using one portion of the data set. For example, if we use minute prices for an asset and m =5 minutes then every fifth data point is used in our realized variance calculation, defined by observations: 1, 6, 11, 16, ... This ignores the other sub-samples that are available, such as: 2, 7, 12, 17, ... and 3, 8, 13, 18, ... The full grid, containing every observation, is defined as $\mathcal{G}$ and n $= |\mathcal{G}|$, or the size of $\mathcal{G}$. $\mathcal{G}$ is partitioned into k non-overlapping sub-grids $\mathcal{G}^{(k)}$ of size ,n_k . The scenario above yields $\mathcal{G} = \{p_{t_1}, p_{t_2}, p_{t_3}, p_{t_4}, p_{t_5}, p_{t_6}, p_{t_7}, p_{t_8}, p_{t_9}, p_{t_{10}}, p_{t_{11}}, ...\}$ and returns are calculated from the appropriate sub-grid, $\mathcal{G}_1^{(5)} = \{p_{t_1}, p_{t_6}, p_{t_{11}}, ...\}$ and $\mathcal{G}_2^{(5)} = \{p_{t_2}, p_{t_7}, p_{t_{12}}, ...\}$. Then realized variance of each sparse grid, $\mathcal{G}_i^{(k)}$, is calculated by:

$$RV_{sparse}^{(k,i)} \equiv \sum_{t_{jt}, t, +_j \in \mathcal{G}_i}^{n_k} (p_{t_{j,+}, } - p_{t_j})^2 \tag{11}$$

where $p_{t_{j,+}}$, is the next observation of grid i. The traditional realized variance estimator, $RV^{(m)}$, is equal to $RV_{sparse}^{(\frac{1}{m},1)}$.

Their second best realized variance estimate is the average of (11) over all of the possible grids, or sub-samples,

$$RV_{Avg}^{(k)} = \frac{1}{k} \sum_{i=1}^{k} RV_{sparse}^{(k,i)} \tag{12}$$

The estimator (12), however, is still biased at high frequencies but it greatly lowers the variability of the

realized variance estimate. The first best estimator, known as the two timescales estimator, couples $RV_{Avg}^{(k)}$ with realized variance calculated at the highest frequency possible, $RV_{sparse}^{(all)}$:

$$RV_{TS}^{(k)} = RV_{Avg}^{(k)} - \frac{n_k}{n} RV_{sparse}^{(all)} \tag{13}$$

The two-timescale estimator may be improved with a small sample adjustment

$$RV_{TS,Adj}^{(k)} = (1 - \frac{n_k}{n})^{-1} RV_{TS}^{(k)} \tag{14}$$

Finally, Ait-Sahalia et al. (2005) introduced a further improvement to compensate for a slight underestimation of $RV_{Avg}^{(k)}$ when k is large:

$$RV_{TS,AA}^{(k)} = \frac{n}{(k-1)n_k} RV_{TS}^{(k)} \tag{15}$$

which has the same asymptotics as $RV_{TS,Adj}^{(k)}$.

Figure 6B displays a signature plot for the same data, but this data is aligned first to a 30 second frequency. Then the three versions of the two-timescale estimator (13, 14,15) are plotted with respect to the number of subgrids.

```
# Figure 6B

test.sig.min <- rSignature(1:120, msft.real.cts[[1]], xscale=1/2, args=list(align.period=30))
> plot(test.sig.min, ylab="Realized Variance", xlab="Minutes", main="MSFT", sub=dates.example[[1]])
> test.tt <- rSignature(1:20, msft.real.cts[[1]], xscale=3, type="timescale", args=list(adj.type="classic", align.period=60))

> test.tt.adj <- rSignature(1:20, msft.real.cts[[1]], xscale=3, type="timescale", args=list(adj.type="adj", align.period=60))

> test.tt.aa <- rSignature(1:20, msft.real.cts[[1]], xscale=3, type="timescale", args=list(adj.type="aa", align.period=60))

> lines(test.tt, col=3, lwd=2) > lines(test.tt.adj, col=4, lwd=2)
> lines(test.tt.aa, col=5, lwd=2)
> axis(3, c(0,(1:5)*12), c("Subgrids:",as.character((1:5)*4)))
> legend(45,.0006,c("Classic", "Adj", "AA"), lwd=c(2,2,2), col=c(3,4,5))
```

2.4 Single Estimates

The previous two sections demonstrated various realized variance estimators on signature plots. The analysis shows that one must be careful when they choose what type of estimator to use as well as the inputs to pass to that estimator. To calculate an estimate with a particular estimator and inputs the `rRealizedVariance` function is used. The inputs to this function are similar to the `rSignature` function.

Below are a few examples of how to use the `rRealizedVariance` function. The actual input values to each type of estimator can be found on the help pages for each estimator. These are linked to on the `rRealizedVariance` help page which you can find using:

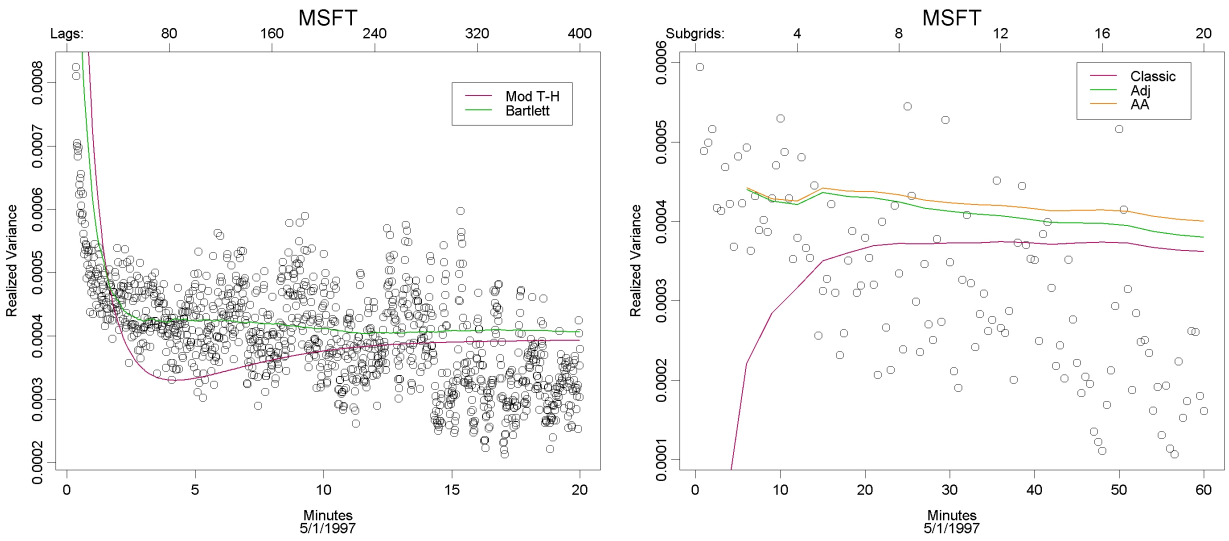


Figure 6A and 6B:Kernel and Two-Timescale Estimates of Realized Variance as a Function of Lags and Subgrids.

```
> ?rRealizedVariance
```

Below are various estimates of realized variance.

```
>> # Traditional Estimate at highest frequency
> rRealizedVariance(x=msft.real.cts[[1]], type="naive", period=1)
[1] 0.004642229

> # Traditional Estimate at one minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], type="naive", period=1, args=list(align.period=60))
[1] 0.0004884795

> # Traditional Estimate at 10 minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], type="naive", period=10, args=list(align.period=60))
[1] 0.0005299257

> # Bartlett Kernel Estimate with minute aligned data at 20 lags
> rRealizedVariance(x=msft.real.cts[[1]], type="kernel", lags=20, args=list(align.period=60, type="Bartlett"))
[1] 0.0003815077

> # Cubic Kernel Estimate with second aligned data at 400 lags
> rRealizedVariance(x=msft.real.cts[[1]], type="kernel", lags=400, args=list(type="Cubic"))
[1] 0.0003986213

> # Two-Timescale Estimate with minute aligned data at 10 subgrids
> rRealizedVariance(x=msft.real.cts[[1]], type="timescale", period=10, args=list(align.period=60))
[1] 0.0003724935

> # Subsample Average Estimate with second aligned data at 600 subgrids
> rRealizedVariance(x=msft.real.cts[[1]], type="avg", period=600) [1] 0.0004016684
```

3 Realized Covariance

Realized covariance suffers from a bias toward zero at high frequencies and, similar to realized variance, this bias disappears at lower frequencies at the cost of more volatile estimates. The high volatility of the traditional realized covariance estimator motivates the use of one day realized covariance signature plots. There have been many alternative estimators proposed (Bandi and Russell, 2007,de Pooter et al.,

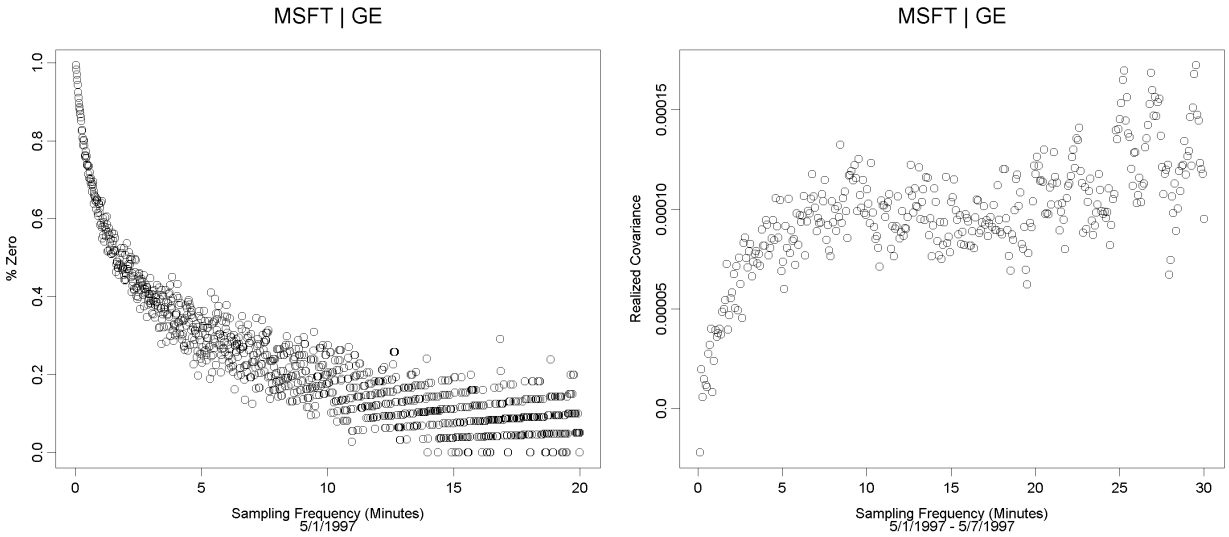


Figure 7A and 7B: Percentage of zero co-returns and Average Covariance Signature Plot for January 2, 2004 MSFT and GE

2005,Flemming et al., 2002,Hayashi and Yoshida, 2005,Griffin and Oomen, 2006,Voev and Lunde, 2006 and Sheppard, 2006). The traditional realize covariance estimator is a natural extension of $RV^{(m)}$:

$$RC^{(m)} = \sum_{i=1}^{m} y_{i,m} x_{i,m} \tag{16}$$

The high frequency bias results from non-synchronous trading and zero returns. Campbell et al. (1997) discuss this problem in detail. When one stock does not change price over an interval, or has a zero return, then even if the other stock moves, $y_{i,m}x_{i,m} = 0$. As the sampling frequency increases there are more and more zero returns and $RC^{(m)}$ becomes a sum of zeros. This produces a downward bias as $m \to \infty$. The function rc.zero calculates the percentage of zero co-returns at a given sampling frequency.

```
> rc.zero(x=msft.real.cts[[1]], y=ge.real.cts[[1]], period=1)
[1] 0.9944022
> rc.zero(x=msft.real.cts[[1]], y=ge.real.cts[[1]], period=60)
[1] 0.6419437
```

This function can be coupled with the `rSignature` function to view the percentage of zero co-returns at each sampling frequency:

```
# Figure 7A
> test.zero <- rSignature(1:1200, x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="zero", xscale=1/60)
> plot(test.zero, ylab="% Zero", xlab="Sampling Frequency (Minutes)", main="MSFT | GE", sub=dates.example[[1]])
```

The plot shows that for a sampling frequency of five minutes or higher there is at least 20% of the co-returns that are zero. Some of this could be due to the CTS alignment and further investigation using the `rAccumuation`, `rMarginal`, and `rScatterReturns` functions discussed in section five is encouraged.

Most of the functions in the realized library work for realized variance and covariance calculations. When the `x` and `y` variables are both be specified a covariance result is returned. Griffin and Oomen (2006) were the first to extend the variance signature plot to covariance. To plot a six day average covariance signature plot at every five seconds of sampling frequency:

```
# Figure 7B
> plot(rSignature((1:360)*5+1, x=msft.real.cts, y=ge.real.cts, xscale=1/60, iteration.funct="simpleIteration", iterations=1:6),
 ylab="Realized Covariance", xlab="Sampling Frequency (Minutes)", main="MSFT | GE", sub=paste(dates.example[[1]],
dates.example[[5]], sep=" - "))
```

This six day average signature shows the bias towards zero at the highest sampling frequencies like expected.

3.1 Lead-Lag / Kernel Estimators

Griffin and Oomen (2006); Voev and Lunde (2006)and De Pooter, Martens, and van Dijk (2005) implement a lead-lag estimator which is just a simple extension of the univariate rectangular kernel estimator. They set the number of lead and lags to one and adjust the sampling frequency. I have implemented all of the univariate kernel estimators in a multivariate context as defined below.

$$RC_{Kernel} = \hat{\gamma_0} + \sum_{h=1}^{H} k\Big(\frac{h-1}{H}\Big)\Big\{\hat{\xi_h} + \hat{\xi_{-h}}\Big\} \tag{17}$$

$$\hat{\xi}_h \equiv \sum_{i=1}^{m} y_{i,m} x_{i+h,m} \tag{18}$$

and if `adj=T` then a degree of freedom adjustment is made:

$$\tilde{\xi_h} \equiv \frac{m}{m-h} \sum_{i=1}^{m} y_{i,m} x_{i+h,m} \tag{19}$$

A one day covariance volatility plot shows that the kernels are less variable with respect to their inputs than the traditional estimator with respect to sampling period:

```
# Figure 8A
> test.cov <- rSignature(1:1200,x=msft.real.cts[[1]], y=ge.real.cts[[1]], xscale=1/60)
> test.rect <- rSignature(1:600,msft.real.cts[[1]], ge.real.cts[[1]],type="kernel",args=list(type="rectangular"),
xscale=1/30)
> test.mth <- rSignature(1:600,msft.real.cts[[1]], ge.real.cts[[1]],type="kernel",args=list(type="mth"), xscale=1/30)

> plot(test.cov, ylab="Realized Covariance", xlab="Minutes", main="GE | MSFT")
> lines(test.rect, col=3, lwd=1)
> lines(test.mth, col=4, lwd=2)
> axis(3, c(0,(1:5)*4), c("Lags:",as.character((1:5)*120)))
```

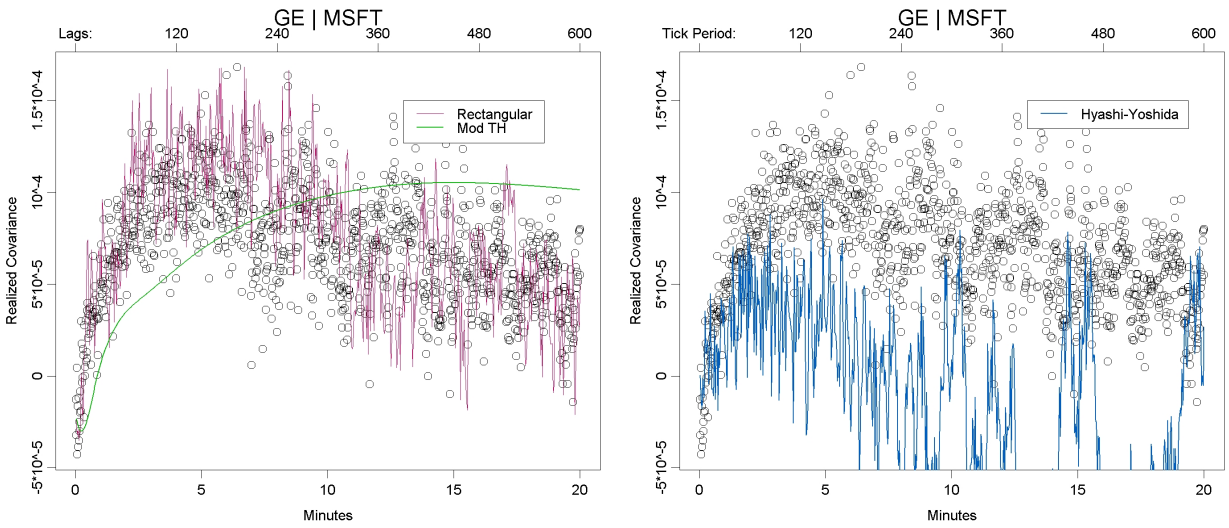


Figure 8A and 8B: Percentage of zero co-returns and Average Covariance Signature Plot for January 2, 2004 MSFT and GE

```
> legend(13,.00015,c("Rectangular", "Mod TH"), lwd=c(1,2), col=c(3,4))
```

3.2 Hayashi-Yoshida

An estimation procedure that solves the problem of non-synchronicity, assuming no micro structure noise, is proposed by Hayashi and Yoshida (2005). The defining feature of this TTS estimator is that it adds products of returns to the sum as long as the corresponding intervals overlap. Thus, a given tick return of asset y is multiplied by several tick returns of asset x. First we define the transaction times of the returns for each asset, y, and x.

$$\Pi_y = \{t_1^y, t_2^y, ..., t_{n_y}^y\}, \Pi_x = \{t_1^x, t_2^x, ..., t_{n_x}^x\}$$

where the size of Π_y may not be the same as the size of Π_x. We choose a base asset, say y, and we multiply the first return of y by all of the returns of x where $t_i^x \leq t_1^y$. The second return of y is multiplied by all of the returns of x where $t_1^y < t_i^x \leq t_2^y$ and so on. The results from each time period are summed to form the estimator:

$$RC_{HY}^{(m)} = \sum_{i=1}^{n_y} y_{t_i^y,m} x_{t_j^x \in (t_{i-1}^y, t_i^y],m} \tag{20}$$

The following code displays a realized covariance signature plot with the Hyashi-Yoshida at different TTS

sampling frequencies[5].

```
# Figure 8B
> test.hy <- rSignature(1:600,msft.real.tts[[1]], ge.real.tts[[1]],type="hy",args=list(align.period=1), xscale=1/30)

> plot(test.cov, ylab="Realized Covariance", xlab="Minutes", main="GE | MSFT")
> lines(test.hy, col=2, lwd=2)
> axis(3, c(0,(1:5)*4), c("Tick Period:",as.character((1:5)*120)))
> legend(13,.00015,c("Hyashi-Yoshida"), lwd=c(2), col=c(2))
```

3.3 Sub-Sample Estimators

There has been very little attention focused on multivariate extensions of the average sub-sample (12) and timescale estimators (13 - 15). de Pooter et al. (2005) use a multivariate extension of the adjusted timescale estimator, $RV^{(k)}_{TS,Adj}$, in their extension of Flemming et al. (2002) and found that there is only a marginal improvement in their evaluation criteria. We find, however, that the sub-sampling estimator drastically improves on the efficiency of the realized covariance estimates. Given two assets that are pre-aligned to the same calendar time the multivariate sub-sample estimator of realized covariance are

$$RC^{(k,i)}_{sparse} \equiv \sum_{t_{jt},t,+_j,\in\mathcal{G}_i}^{n_k} (p^y_{t_{j,+},} - p^y_{t_j})(p^x_{t_{j,+},} - p^x_{t_j}) \tag{21}$$

$$RC^{(k)}_{Avg} = \frac{1}{k}\sum_{i=1}^{k} RC^{(k,i)}_{sparse} \tag{22}$$

Due to the bias towards zero at high frequencies, any multivariate extension of the timescale estimators (13 - 15) are essentially equivalent to (22), except for the constant used to adjust for small samples or a large k.

$$RC^{(k)}_{TS} = RC^{(k)}_{Avg} - \frac{n_k}{n} RC^{(all)}_{sparse} \approx RC^{(k)}_{Avg} \tag{23}$$

This means that, at high frequencies, setting the type of realized covariance to "timescale" and "avg" should yield very similar results. However, this could it is still useful to specify two-timescale for correlation calculations.

3.4 Single Estimates and Covariance Matrices

Single estimates can be calculated with the rRealized.variance function by specifying an x and y argument. An error message is printed if the returns are not aligned using CTS for the estimators that need that.

[5]Note that this is currently the only function that needs a list as its data object. This is because the Hayashi-Yoshida estimator uses execution times.

```
>> # Traditional Estimate at highest frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=1)
[1] -1.279923e-05

> # Traditional Estimate at one minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=1, args=list(align.period=60))
[1] 5.116529e-05

> # Traditional Estimate at 10 minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=10, args=list(align.period=60))
[1] 7.856014e-05

> # Bartlett Kernel Estimate with minute aligned data at 20 lags
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=20, args=list(align.period=60,
type="Bartlett"))
[1] 6.553482e-05

> # Cubic Kernel Estimate with second aligned data at 400 lags
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=400, args=list(type="Cubic"))

[1] 0.0001019466

> # Lead-Lag with one lag at one minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=1, args=list(align.period=60))
 [1] 0.0001179901

> # Subsample Average Estimate with second aligned data at 600 subgrids > rRealizedVariance(x=msft.real.cts[[1]],
y=ge.real.cts[[1]], type="avg", period=600)
[1] 7.610897e-05
```

A covariance matrix can be calculated by specifying only the x input variable and merging multiple `realizedObjects`. Currently this is not a vectorized calculation so it will become slower and slower as the number of assets increases. The number of individual calculations is $\frac{k(k+1)}{2}$.

```
> # Traditional Estimate at highest frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="naive", period=1)
    [,1] [,2]
[1,] 4.642229e-03 -1.279923e-05
[2,] -1.279923e-05 1.140889e-03

> # Traditional Estimate at 10 minute frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="naive", period=10, args=list(align.period=60))
    [,1] [,2]
[1,] 5.299257e-04 7.856014e-05
[2,] 7.856014e-05 1.212815e-04

> # Lead-Lag with one lag at one minute frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="kernel", lags=1, args=list(align.period=60))

    [,1] [,2]
[1,] 0.0004402754 0.0001179901
[2,] 0.0001179901 0.0001567376

> # Subsample Average Estimate with second aligned data at 600 subgrids
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="avg", period=600)
    [,1] [,2]
[1,] 4.016684e-04 7.610897e-05
[2,] 7.610897e-05 1.329599e-04
```

4 Realized Correlation

Realized correlation is a natural extension in the realized variance and covariance literature. `Epps (1979)` was the first to document that increasing the frequency of observations in a correlation calculation will lead to a bias toward zero. The equation for realized correlation is:

$$R\rho^{(m)} = \frac{RC_{x,y}^{(m)}}{\sqrt{RV_x^{(m)}}\sqrt{RV_y^{(m)}}} \tag{24}$$

and it is easy to see why this bias towards zero exists since the numerator is biased towards zero and the denominator is biased upwards at high frequencies. The estimation of realized correlation with the realized library is simple. The rRealized.variance and rSignature function have an argument called cor that is set to true. Along with this flag both x and y must be specified, or a matrix for x for a correlation matrix.

The idea of an optimal realized correlation calculation is slightly more complicated than a realized variance or covariance due to the fact that it involves choosing the best estimator and inputs for $RC_{x,y}^{(m)}$, $RV_y^{(m)}$, and $RV_x^{(m)}$. Sheppard (2006) introduces the idea of "pseudo" realized correlation where the realized variances are sampled at a different frequency then the realized covariance. The realized library only calculates realized correlation with all the same estimator and inputs, however, using the `rv.*` and `rc.*` functions you can create any realized correlation estimate. This also means that there is no Hayashi-Yoshida correlation estimator.

4.1 Single Estimates and Correlation Matrices

The `rRealizedVariance` and `rSignature` functions will calculate realized correlation when there are multiple log-return vectors passed in and the cor flag is set to true. Below is an example of different types of realized correlation for MSFT and GE May 1, 1997.

```
> # Traditional Estimate at highest frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=1, cor=T)
 [1] -0.005561591

> # Traditional Estimate at one minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=1, args=list(align.period=60),
cor=T)
[1] 0.1381217

> # Traditional Estimate at 10 minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="naive", period=10, args=list(align.period=60),
cor=T)
[1] 0.3098827

> # Bartlett Kernel Estimate with minute aligned data at 20 lags
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=20, args=list(align.period=60,
type="Bartlett"), cor=T)
 [1] 0.31984

> # Cubic Kernel Estimate with second aligned data at 400 lags
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=400, args=list(type="Cubic"),
cor=T)
[1] 0.4530968

> # Lead-Lag with one lag at one minute frequency
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="kernel", lags=1, args=list(align.period=60),
cor=T)
 [1] 0.4491556
```

```
> # Subsample Average Estimate with second aligned data at 600 subgrids
> rRealizedVariance(x=msft.real.cts[[1]], y=ge.real.cts[[1]], type="avg", period=600, cor=T)
[1] 0.3293378
```

For correlation matrices:

```
>> # Correlation Matrices
> # Traditional Estimate at highest frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="naive", period=1, cor=T)
    [,1] [,2]
[1,] 1.000000000 -0.005561591
[2,] -0.005561591 1.000000000

> # Traditional Estimate at 10 minute frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="naive", period=10, args=list(align.period=60),
cor=T)
    [,1] [,2]
[1,] 1.0000000 0.3098827
[2,] 0.3098827 1.0000000

> # Lead-Lag with one lag at one minute frequency
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="kernel", lags=1, args=list(align.period=60),
cor=T)
    [,1] [,2]
[1,] 1.0000000 0.4491556
[2,] 0.4491556 1.0000000

> # Subsample Average Estimate with second aligned data at 600 subgrids
> rRealizedVariance(x=merge(msft.real.cts[[1]], ge.real.cts[[1]]), type="avg", period=600, cor=T)
    [,1] [,2]

[1,] 1.0000000 0.3293378
[2,] 0.3293378 1.0000000
```

5 Plotting Tools

One day signature plots allow us to see the variation of realized variance estimates with respect to their inputs. Realized accumulation and marginal contribution plots gives us the ability to attribute this variation to particular observations in the high frequency data set.

5.1 Realized Accumulation and Marginal Contribution Plots

Payseur (2007) uses marginal contribution and realized accumulation plots over the period of a single estimate for traditional realized estimators. This analysis shows that it is not only the sampling frequency that creates highly volatile estimates, but also the alignment of the returns. It is useful to couple the `rCumSum` function with the `rAccumulation` function as follows:

```
# Figure 9A
> cumm <- list()
> cumm[[1]] <- rCumSum(msft.real.cts[[1]], period=1, align.period=60)
> cumm[[2]] <- rCumSum(msft.real.cts[[1]], period=10, align.period=60)
> cumm[[3]] <- rCumSum(msft.real.cts[[1]], period=20, align.period=60)
> cumm[[4]] <- rCumSum(msft.real.cts[[1]], period=30, align.period=60)
> accum <- list()
> accum[[1]] <- rAccumulation(msft.real.cts[[1]], period=10, align.period=60)
```

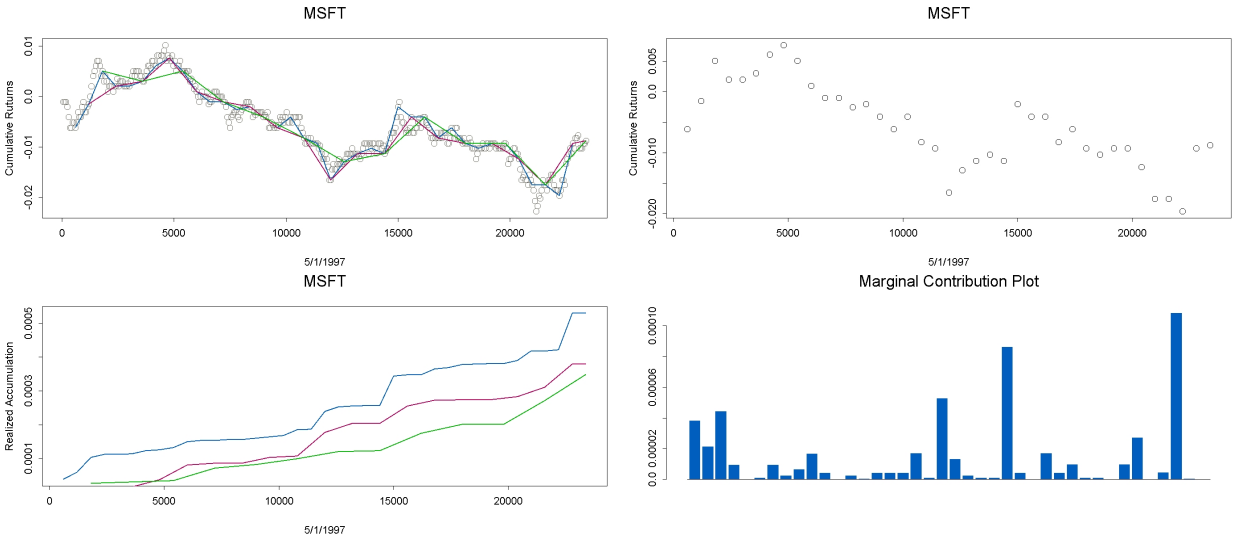


Figure 9A and 9B:Realized Accumulation and Marginal Contribution to Realized Variance Plots

```
> accum[[2]] <- rAccumulation(msft.real.cts[[1]], period=20, align.period=60)
> accum[[3]] <- rAccumulation(msft.real.cts[[1]], period=30, align.period=60)

> par(mfrow=c(2,1))
> plot(cumm[[1]], xlab="", ylab="Cumulative Ruturns", main="MSFT", sub=dates.example[[1]], type="p", col=16, lwd=2)

> lines(cumm[[2]], col=2, lwd=2)
> lines(cumm[[3]], col=3, lwd=2)
> lines(cumm[[4]], col=4, lwd=2)
> plot(accum[[1]], xlab="", ylab="Realized Accumulation", type="l",main="MSFT", sub=dates.example[[1]], col=2, lwd=2)

> lines(accum[[2]], col=3, lwd=2)
> lines(accum[[3]], col=4, lwd=2)
```

The three realized variance accumulation lines display the estimate builds throughout the trading day. Near the end of the day the 10 minute aligned realized variance estimate, in red, is aligned in a particular way to include a valley in the realized variance calculation that is not included in the other two. The accumulation plot allows us to see how big of an impact this alignment has on the estimate.

The marginal contribution to a realized estimate shows how much each high frequency observation effects the final estimate.

```
# Figure 9B
>par(mfrow=c(2,1))
> plot(cumm[[2]], xlab="", ylab="Cumulative Ruturns", main="MSFT", sub=dates.example[[1]], type="p")
> barplot(rMarginal(msft.real.cts[[1]], period=10, align.period=60)$y, main="Marginal Contribution Plot")
```

Similar to all of the functions in this library the `rMarginal` and `rAccumulation` plots can also be used for realized covariance and correlation. The only difference in the function call is to specify an `x` and a `y`. To plot an accumulation plot for realized covariance aligned at 10 minutes intervals:

```
# Figure 10A
> cumm.ge <- list()
> cumm.ge[[1]] <- rCumSum(ge.real.cts[[1]], period=1, align.period=60)
> cumm.ge[[2]] <- rCumSum(ge.real.cts[[1]], period=10, align.period=60)
```

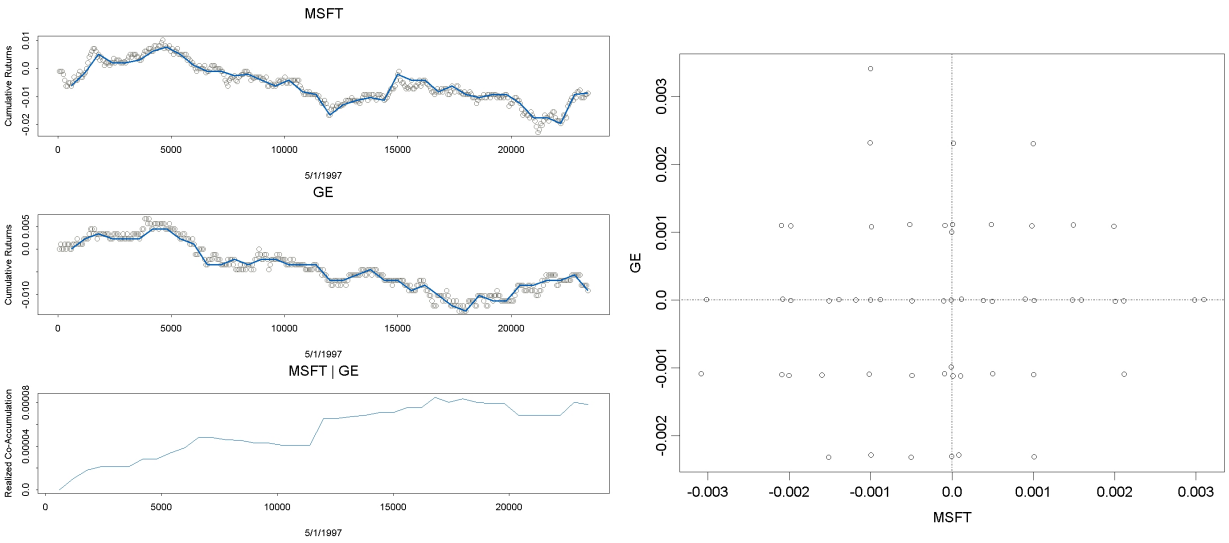


Figure 10A and 10B : Accumulation Plot for the Realized Covariance and Returns Scatter Plot for MSFT and GE January 2, 2002

```
> accum <- list()
> accum[[1]] <- rAccumulation(msft.real.cts[[1]], y=ge.real.cts[[1]], period=10, align.period=60)
> par(mfrow=c(3,1)) > plot(cumm[[1]], xlab="", ylab="Cumulative Ruturns", main="MSFT", sub=dates.example[[1]], type="p",
col=16)
> lines(cumm[[2]], col=2, lwd=3)
> plot(cumm.ge[[1]], xlab="", ylab="Cumulative Ruturns", main="GE", sub=dates.example[[1]], type="p", col=16)
> lines(cumm.ge[[2]], col=2, lwd=3)
> plot(accum[[1]], xlab="", ylab="Realized Co-Accumulation", type="l",main="MSFT | GE", sub=dates.example[[1]],
col=2)
```

In this example the realized covariance appears to build at a fairly constant pace except for two or three big jumps where the stocks move together. To see how the variability in the realized covariance estimate is created, run the same plot commands with sampling periods of six to 20 minutes and you will most likely see that for some of the timings the alignment will be such that the stocks will not move together during these same two or three times.

The additive nature of the realized estimators allows implementation of both the accumulation and marginal contribution plots for all of the estimators above by adjusting the interval $T_{i,m} - T_{i-1,m}$. First, the estimation is performed on a subset of the whole periods worth of data. Second, one more observation is added and the estimation is performed again. As observations are added an accumulation plot is created and the marginal plot can be constructed easily. This functionality will be included in the 1.0 release.

5.2 Scatter Plots

The `rMarginal` function plots the products or cross products of returns, however, it is also informative to look at the returns themselves using a scatter plot. The function `rScatterReturns` displays a scatter plot of returns for a given sampling period. For one minute returns of MSFT and GE on January 2, 2004:

```
# Figure 10B
>rScatterReturns(msft.real.cts[[1]],y=ge.real.cts[[1]], period=1, align.period=20,ylab="GE",xlab="MSFT",numbers=F)
```

There are many points that lie on the two zero lines, which shows the non synchronous trading problem. There is also one big positively correlated log-return that will add to the final estimate where GE and MSFT moved .002 and .003 during the same sample period.

6 Example: Importing Data

Each example in this users manual uses cleaned data that is preloaded into a `realizedObject`. However, everyone using this library will use different data from different sources. In this section I give an example of how to get a flat file of data into a `realizedObject`.

For this example I import FX data from http://ratedata.gaincapital.com/. I have also included the three files `eurusd.csv`, `usdjpy.csv`, and `eurjpy.csv` in the library zip file. There are three simple functions that you must write to import this data properly, all of these depend on the datasource, data format and timestamp format. Before we get to these functions let's load the file and look at it the first ten rows to see the format:

```
> path <- "d:/dev/" #change this to the directory that the .csv files live in
> eur.usd.05.2007 <- read.table(paste(path, "eurusd.csv", sep=""), stringsAsFactors=F, sep=",")
 > usd.jpy.05.2007 <- read.table(paste(path, "usdjpy.csv", sep=""), stringsAsFactors=F, sep=",")
> eur.jpy.05.2007 <- read.table(paste(path, "eurjpy.csv", sep=""), stringsAsFactors=F, sep=",")
> eur.usd.05.2007[1:10,]
```

```
         V1      V2                  V3     V4     V5 V6
1 328975061 EUR/USD 2007-05-27 17:00:05 1.3449 1.3452 D
2 328975103 EUR/USD 2007-05-27 17:05:24 1.3448 1.3451 D
3 328975235 EUR/USD 2007-05-27 17:36:02 1.3447 1.3450 D
4 328975395 EUR/USD 2007-05-27 17:44:06 1.3446 1.3449 D
5 328975412 EUR/USD 2007-05-27 17:44:21 1.3447 1.3450 D
6 328975424 EUR/USD 2007-05-27 17:44:36 1.3446 1.3449 D
7 328975466 EUR/USD 2007-05-27 17:45:33 1.3445 1.3448 D
8 328975514 EUR/USD 2007-05-27 17:45:50 1.3446 1.3449 D
9 328975520 EUR/USD 2007-05-27 17:45:52 1.3445 1.3448 D
10 328975522 EUR/USD 2007-05-27 17:45:54 1.3446 1.3449 D
```

For this simple example I will demonstrate how to get the first days worth of data and work with that, obviously your approach may vary. This data is 24 hour data so our integration bounds for integrated variance are from 0:00 to 23:59:59 for each day. This first function will subset the original data to return a particular day of data by matching the date string in column three.

```
> getT <- function(x, dateStr,...)
{
    y <- x[,3]
    x[substring(y,1,10)==dateStr,]
}
```

We will work with data for May 5, 2007:

```
> eur.usd.05.30.2007 <- getT(eur.usd.05.2007, "2007-05-30")
> usd.jpy.05.30.2007 <- getT(usd.jpy.05.2007, "2007-05-30")
> eur.jpy.05.30.2007 <- getT(eur.jpy.05.2007, "2007-05-30")
```

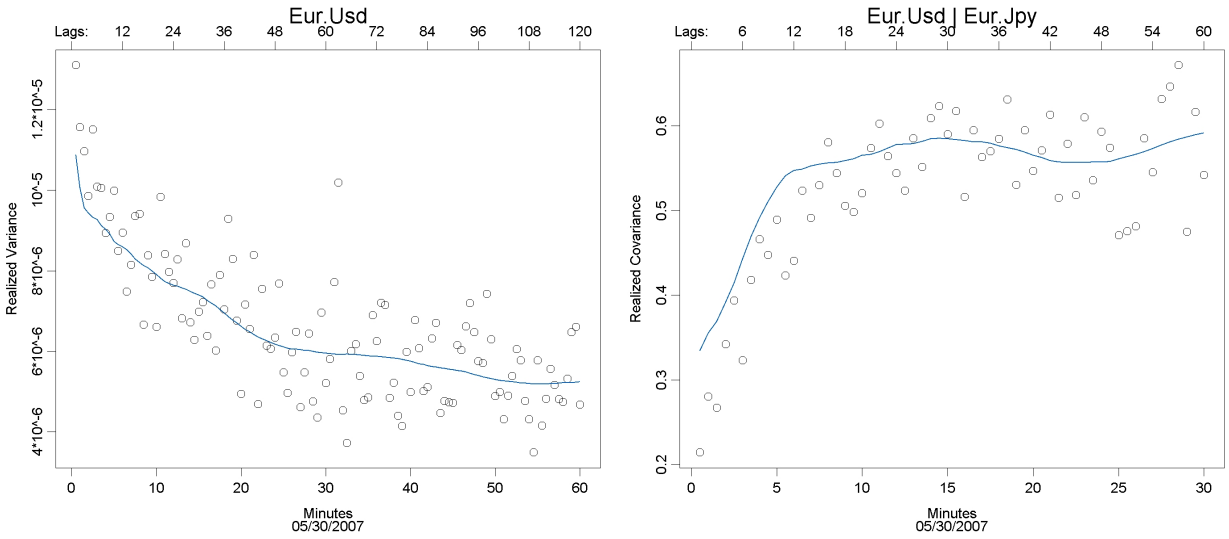


Figure 11A and 11B : Realized Variance Signature Plot and Realized Correlation Signature Plot

The second function that we will create is due to the fact that we are looking at quote data with a bid and ask. It is common practice in the realized variance literature to use the midquote, or average of the two quotes.

```
> midQuote <- function(x, bid.index = 4, ask.index = 5)
{
    (x[,bid.index] + x[,ask.index])/2
}
```

The `realizedObject` constructor function needs data (in the form of prices or returns), as well as, the timestamp in milliseconds. We will calculate milliseconds from the time portion of colum three.

```
> toMilliseconds <- function(x,...)
{
    ans <- 1000 * as.numeric(substring(x, 12,13)) * 60 * 60 +
    1000 * as.numeric(substring(x, 15,16)) * 60 +
    as.numeric(substring(x, 18,19)) * 1000
    ans
}
```

We can use these two functions with the `realizedObject` constructor function to create our `realizedObjects`. The parameters `millisstart` and `millisend` specify the integration bounds for the integrated variance and are defaulted to normal market hours. Since we are looking at 24 hour FX data we need to change these to 0:00 and 23:59:59 respectively.

```
eur.usd.real <- realizedObject(list(data=midQuote(eur.usd.05.30.2007),
 milliseconds=toMilliseconds(eur.usd.05.30.2007[,3])),
makeReturns=T, cts=T, millisstart=0000, millisend=1000*24*60*60)

> eur.jpy.real <- realizedObject(list(data=midQuote(eur.jpy.05.30.2007),
milliseconds=toMilliseconds(eur.jpy.05.30.2007[,3])),
makeReturns=T, cts=T, millisstart=0000, millisend=1000*24*60*60)

> usd.jpy.real <- realizedObject(list(data=midQuote(usd.jpy.05.30.2007),
milliseconds=toMilliseconds(usd.jpy.05.30.2007[,3])),
makeReturns=T, cts=T, millisstart=0000, millisend=1000*24*60*60)
```

Now these objects can be used with any function from in this manual. A variance signature plot of eur.usd.05.30.2007 with data aligned to 30 seconds with Bartlett kernel estimates overlayed as a function of lags:

```
# Figure 11A
> test.sig <- rSignature(1:120, eur.usd.real, xscale=1/2, args=list(align.period=30))
> test.bart <- rSignature(1:120, eur.usd.real, type="kernel",xscale=1/2, args=list(align.period=30, type="bartlett"))
```

```
> plot(test.sig, ylab="Realized Variance", xlab="Minutes", main="Eur.Usd", sub="05/30/2007")
> lines(test.bart, col=2, lwd=2)
> axis(3, c(0,(1:10)*6), c("Lags:",as.character((1:10)*12)))
```

A correlation signature plot:

```
# Figure 11B
> test.sig <- rSignature(1:60, x=eur.usd.real, y=eur.jpy.real, xscale=1/2, args=list(align.period=30), cor=T)
> test.bart <- rSignature(1:60, x=eur.usd.real, y=eur.jpy.real, type="kernel",xscale=1/2, args=list(align.period=30,
type="bartlett"), cor=T)
> plot(test.sig, ylab="Realized Covariance", xlab="Minutes", main="Eur.Usd | Eur.Jpy", sub="05/30/2007")
> lines(test.bart, col=2, lwd=2)
> axis(3, c(0,(1:10)*3), c("Lags:",as.character((1:10)*6)))
```

References

Yacine Ait-Sahalia, Per A. Mykland, and Lan Zhang. Ultra high frequency volatility estimation with dependent microstructure noise. (11380), May 2005. URL http://www.nber.org/papers/w11380.

T.G. Andersen, T. Bollerslev, F.X. Diebold, and P. Labys. Great realizations. *Risk*, 13:105–108, 2000.

T.G. Andersen, T. Bollerslev, F.X. Diebold, and P. Labys. The distribution of exchange rate volatility. *Journal of the American Statistical Association*, 96:42–55, 2001.

F.M. Bandi and Zhu Y. Russell, J.R. Using high-frequency data in dynamic portfolio choice. *Forthcoming, Econometric Reviews*, 2007.

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.

O. E. Barndorff-Nielson and N. Sheppard. Econometric analysis of realised volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society*, Series B 64:253–280, 2002.

J. Campbell, A. Lo, and C. MacKinlay. *The Econometrics of Financial Markets*. Princeton University Press, New Jersey, 1997.

Michiel de Pooter, Martin Martens, and Dick van Dijk. Predicting the daily covariance matrix for sp100 stocks using intraday data - but which frequency to use? (05-089/4), October 2005. available at http://ideas.repec.org/p/dgr/uvatin/20050089.html.

T. Epps. Comovements in stock prices in the very short run. *Journal of the American Statistical Association*, 74(366):291–298, 1979.

Y. Fang. Volatility modeling and estimation of high-frequency data with gaussian noise. *unpublished doctoral thesis, MIT, Sloan School of Management*, 1996.

J. Flemming, C. Kirby, and B. Ostdiek. The economic value of volatility timing using "realized" volatility. *Journal of Financial Economics*, 2002.

J. E. Griffin and R. C. A. Oomen. Covariance measurement in the presence of non-synchronous trading and market microstructure noise. *Working Paper, June 27th*, 2006.

P. Hansen and A. Lunde. Realized variance and market microstructure noise. *Journal of Business and Economic Statistics*, 24:127–218, 2006.

P.R. Hansen and A. Lunde. An unbiased measure of realized variance. *Working Paper*, 2004.

T. Hayashi and N. Yoshida. On covariance estimation of non-synchronously observed diffusion processes. *Bernoulli*, 11:359–379, 2005.

M. Lapsley. The rodbc package. *R-Project, CRAN: http://cran.r-project.org/doc/packages/RODBC.pdf*, 2007.

S. W. Payseur. One day comparison of realized variance and covariance estimators. *Working Paper, http://students.washington.edu/spayseur/realized/payseur.one.day.signature.plots.pdf*, 2007.

Neil Shephard and Ole E. Barndorff-Nielsen. Variation, jumps, market frictions and high frequency data in financial econometrics. (240), 2005. available at http://ideas.repec.org/p/oxf/wpaper/240.html.

Kevin Sheppard. Realized covariance and scrambling. *Working paper University of Oxford*, 2006.

V. Voev and A. Lunde. Integrated covariance estimation using high-frequency data in the presence of microstructure noise. *Journal of Fiancial Econometrics*, 5, 2006.

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.

L. Zhang, P.A Mykland, and Y. Ait-Sahalia. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association*, 2005.

B. Zhou. High-frequency data and volatility in foreign-exchange rates. *Journal of Buiness & Economic Statistics*, 14:45–52, 1996.