www.portfolioeffect.com
High Frequency Portfolio Analytics

User Manual

PortfolioEffectEstim R Package

High Frequency Price Estimators & Models

Andrey Kostin
andrey.kostin@portfolioeffect.com

August 20, 2015

Contents

1 Package Installation

PortfolioEffectEstim package for R relies on the rJava package, which assumes that Java runtime is installed and configured on your system. To install Java runtime and to configure your R engine to work with it, follow these steps:

1.1 Install Latest JDK/JRE Runtime

Download and install latest Java distribution (JDK or JRE) for your platform from Oracle's website

1.2 Configure Java Environment (Optional)

If you are using Windows, installation wizard from the previous step should have done everything for you. If you are on Linux or Mac and you used a tarball file, you will need to manually append the following lines to /etc/environment using your favorite text editor:

```
export JAVA_HOME=/path/to/java/folder
export PATH=$PATH:$JAVA_HOME/bin
```

Apply environment changes:

```
source /etc/environment
```

To complete with the set-up of Java environment inside R, run the following line:

```
sudo R CMD javareconf
```

1.3 Install Required Packages (Optional)

If you are manually installing PortfolioEffectEstim package (you don't want to use CRAN repositories for some reason), you would need to install all required package dependencies first. Start R from the command line or in your GUI editor and type

```
install.packages(c("PortfolioEffectHFT", "rJava"))
```

You are now ready to install the PortfolioEffectEstim package directly from www.portfolioeffect.com downloads section.

2 API Credentials

All computations are performed on PortfolioEffect cloud servers. To obtain a free non-professional account, you need to follow a quick sign-up process on our website: www.portfolioeffect.com/registration.

Please use a valid sign-up address - it will be used to email your account activation link.

2.1 Locate API Credentials

Log in to you account and locate your API credentials on the main page

2.2 Set API Credentials in R

Run the following commands to set your account API credentials for the PortfolioEffectEstim R Package installed. You will need to do it only once as your credentials are stored between sessions on your local machine to speed up future logons.

You would need to repeat this procedure if you change your account password or install PortfolioEffectEstim package on another computer.

```r
require(PortfolioEffectEstim)
util_setCredentials("API Username", "API Password", "API Key")
```

You are now ready to call PortfolioEffectEstim methods.

3 Estimator Construction

3.1 User Data

Users may supply their own historical datasets for asset entries. This external data could be one a OHLC bar column element (e.g. 1-second close prices) or a vector of actual transaction prices that contains non-equidistant data points.

3.1.1 Create Estimator

Method estimator_create(). takes a vector of asset prices in the format (UTC timestamp, price) with UTC timestamp expressed in milliseconds from 1970-01-01 00:00:00 EST.

```
      Time          Value
 [1,] 1412256601000 99.30
 [2,] 1412256602000 99.33
 [3,] 1412256603000 99.30
 [4,] 1412256604000 99.26
 [5,] 1412256605000 99.36
 [6,] 1412256606000 99.36
 [7,] 1412256607000 99.36
 [8,] 1412256608000 99.38
 [9,] 1412256609000 99.40
[10,] 1412256610000 99.37
```

If asset symbol is specified, it is silently ignored.

```
data(spy.data)

# Create estimator
estimator=estimator_create(priceData=goog.data)
```

3.2 Server Data

At PortfolioEffect we are capturing and storing 1-second intraday bar history for a all NASDAQ traded equites. This server-side dataset spans from January 2013 to the latest trading time minus five minutes. It could be used to construct asset estimator and compute intraday estimator metrics.

3.2.1 Create Estimator

Method estimator_create() creates new asset estimator or overwrites an existing estimator object with the same name.

When using server-side data, it only requires a time interval that would be treated.

Interval boundaries are passed in the following format:

- "yyyy-MM-dd HH:MM:SS" (e.g. "2014-10-01 09:30:00")
- "yyyy-MM-dd" (e.g. "2014-10-01")
- "t-N" (e.g. "t-5" is latest trading time minus 5 days)
- UTC timestamp in milliseconds (mills from "1970-01-01 00:00:00") in EST time zone

```
# Timestamp in "yyyy-MM-dd HH:MM:SS" format
estimator=estimator_create(asset='AAPL',fromTime="2014-09-01 09:00:00",
                           toTime="2014-09-14 16:00:00")

# Timestamp in "yyyy-MM-dd" format
estimator=estimator_create(asset='AAPL',fromTime="2014-09-01",
                           toTime="2014-09-14")

# Timestamp in "t-N" format
estimator=estimator_create(asset='AAPL', fromTime="t-5",
                           toTime="t")
```

3.2.2 Get Symbols List

Once estimator is created, estimator_availableSymbols() method could be called to receive the list of all available symbols for asset creation. Each symbol is accompanied by a full company/instrument description and listing exchange name.

```
estimator_availableSymbols(estimator)

      id      description                                            exchange
 [1,] "BBC"   "BioShares Biotechnology Clinical Trials Fund"         "NASDAQ"
 [2,] "SCS"   "Steelcase Inc. Common Stock"                          "NYSE"
 [3,] "BBD"   "Banco Bradesco Sa American Depositary Shares"         "NYSE"
 [4,] "BBG"   "Bill Barrett Corporation Common Stock"                "NYSE"
 [5,] "STPP"  "Barclays PLC - iPath US Treasury Steepener ETN"       "NASDAQ"
 [6,] "BBF"   "BlackRock Municipal Income Investment Trust"          "NYSE"
 [8,] "BBH"   "Market Vectors Biotech ETF"                           "NYSEARCA"
 [9,] "SCON"  "Superconductor Technologies Inc. - Common Stock"      "NASDAQ"
[10,] "SCX"   "L.S. Starrett Company (The) Common Stock"             "NYSE"
[11,] "BBK"   "Blackrock Municipal Bond Trust"                       "NYSE"
```

4 Estimator Settings

These settings regulate how estimator metrics are computed, returned and stored.

4.0.1 Results Sampling Interval

Interval to be used for sampling computed results before returning them to the caller. Available interval values are:

- "Xs" - seconds
- "Xm" - minutes
- "Xh" - hours
- "Xd" - trading days (6.5 hours in a trading day)
- "Xw" - weeks (5 trading days in 1 week)
- "Xmo" - month (21 trading day in 1 month)
- "Xy" - years (256 trading days in 1 year)
- "none" - no sampling.
- "last" - only the very last data point is returned

Large sampling interval would produce smaller vector of results and would require less time spent on data transfer. Default value of "1s" indicates that data is returned for every second during trading hours.

```
estimator=estimator_create(asset='AAPL',fromTime="2014-10-01 09:30:00",
                           toTime="2014-10-01 16:00:00")

# sample results every 30 seconds
estimator_settings(estimator, resultsSamplingInterval="30s")
variance_30s=variance_tsrv(estimator);

# sample results every 5 minutes
estimator_settings(estimator, resultsSamplingInterval="15m")
variance_15m=variance_tsrv(estimator);

util_plot2d(variance_30s,title="TSRV, resultsSamplingInterval",legend="30s")+ util_line2d(variance_15m,legend="15m
    ")
```

4.0.2 Input Sampling Interval

Interval to be used as a minimum step for sampling input prices. Available interval values are:

- "Xs" - seconds

- "Xm" - minutes
- "Xh" - hours
- "Xd" - trading days (6.5 hours in a trading day)
- "Xw" - weeks (5 trading days in 1 week)
- "Xmo" - month (21 trading day in 1 month)
- "Xy" - years (256 trading days in 1 year)
- "none" - no sampling

Default value is "none", which indicates that no sampling is applied.

```
estimator=estimator_create(asset='AAPL',fromTime="2014-10-01 09:30:00",
                           toTime="2014-10-01 16:00:00")

# sample input prices every 30 seconds
estimator_settings(estimator, inputSamplingInterval="30s")
variance_30s=variance_tsrv(estimator);

# sample input prices every 5 min
estimator_settings(estimator, inputSamplingInterval="5m")
variance_5m=variance_tsrv(estimator);

util_plot2d(variance_30s,title="TSRV,inputSamplingInterval",legend="30s")+ util_line2d(variance_5m,legend="5m")
```

4.0.3 Jumps/Outliers Model

Used to select jump filtering mode when computing return statistics. Available modes are:

- "none" - price jumps are not filtered anywhere
- "moments" - price jumps are filtered only when computing return moments (i.e. for expected return, variance, skewness, kurtosis and derived metrics)
- "all" - price jumps are filtered from computed returns, prices and all return metrics.

```
estimator=estimator_create(asset='AAPL',fromTime="2014-10-01 09:30:00",
                           toTime="2014-10-02 16:00:00")

# Price jumps detection is enabled for returns and moments
estimator_settings(estimator, jumpsModel="all")
variance_all=variance_tsrv(estimator);

# Price jumps detection is disabled
estimator_settings(estimator, jumpsModel="none")
variance_none=variance_tsrv(estimator);

util_plot2d(variance_all,title="Variance,jumpsModel",legend="all")+ util_line2d(variance_none,legend="none")
```

5 Price Variance

5.1 Integrated Variance

Assume that the logarithmic equilibrium price of a financial asset is given by the following diffusion process

$$X_t = \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW(s) \tag{5.1}$$

where

- W_t is a standard Brownian Motion,
- the mean process μ is continuous and of finite variation,
- $\sigma(t) > 0$ denotes the cadlag instantaneous volatility.

The object of interest is the integrated variance (IV), i.e. the amount of variation at time point t accumulated over a past time interval Δ according to [1, Pigorsch et al.]:

$$IV_t = \int_{t-\Delta}^t \sigma^2(s)ds \tag{5.2}$$

5.1.1 Returns

Suppose there exist m intraday equilibrium returns, the ith intraday return is then defined as:

$$r_i^{X(m)} = X_{i/m} - X_{(i-1)/m}, \quad i = 1, 2, \ldots, m. \tag{5.3}$$

5.2 Realized Variance

5.2.1 Assumptions

The equilibrium price process.

1. The logarithmic equilibrium price process p_t^X is a continuous stochastic volatility semimartingale. Specifically,
$$p_t^X = \alpha_t + m_t; \tag{5.4}$$
where α_t (with $\alpha_0 = 0$) is a predictable drift process of finite variation and m_t is a continuous local martingale defined as $\int_0^t \sigma_s dW_s$ with $W_t : t \geq 0$ denoting standard Brownian motion.
2. The spot volatility process σ_t is cadlag and bounded away from zero.
3. The process $\int_0^t \sigma_s^4 ds$ is bounded almost surely for all $t < \infty$.

5.2.2 Estimator

For discretely observed path X_{t_i}, $i = 0, \ldots, n$ of the X, the realized quadratic variation could be estimated consistently using the realized variance measure, defined as [2, Zu and Boswijk, 2014]:

$$RV_t = \sum_{i=1}^{n} (X_{t_i} - X_{t_{(i-1)}})^2 \xrightarrow{p} IV \tag{5.5}$$

However the realized variance estimator for integrated volatility is not consistent when data is contaminated by market microstructure noise. In particular, when we only observe data with microstructure noise, the realized variance measure will diverge. When sampling frequency increases, realized variance actually estimates the sum of infinite many variances of noises.

5.2.3 Properties

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Unbiased: no
- Consistent: no
- Jump Robust: no
- Allows for time dependent noise: no
- Allows for endogenous noise: no

5.2.4 Usage

```
variance_rv(estimator)
variance_rvRolling(estimator,wLength=23400)
```

5.3 Two Series Realized Variance

5.3.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

5.3.2 Estimator

Two Scale Realized Variance (TSRV) estimates integrated volatility consistently. The idea is to use realized variance type estimators over two time scales to correct the effect of market microstructure noise. Define as:

$$[Y,Y]_t^{avg} = \frac{1}{K}\sum_{i=K}^{n}(Y_{t_i} - Y_{t_{(i-K)}})^2 \tag{5.6}$$

$$[Y,Y]_t^{all} = \sum_{i=1}^{n}(Y_{t_i} - Y_{t_{(i-1)}})^2 \tag{5.7}$$

$$\bar{n} = \frac{n-K+1}{K}, \tag{5.8}$$

the TSRV estimator is defined as [2, Zu and Boswijk, 2008]:

$$TSRV_t = [Y,Y]_t^{(avg)} - \frac{\bar{n}}{n}[Y,Y]_t^{(all)} \xrightarrow{p} IV \tag{5.9}$$

A small sample refinement to the estimator give correction:

$$TSRV_t^{adjust} = \left(1 - \frac{\bar{n}}{n}\right)^{-1} TSRV_t \xrightarrow{p} IV. \tag{5.10}$$

If we have possibly dependent noise we should use an alternative estimator that is also based on the two time scales idea.

To pin down the optimal sampling frequency K [3, Zhang, Mykland, and Ait-Sahalia, 2005], one can minimize the expected asymptotic variance and to obtain

$$c^* = \left(\frac{16(E\epsilon^2)^2}{TE\eta^2}\right)^{1/3} \tag{5.11}$$

which can be consistently estimated from data in past time periods (before time $t_0 = 0$), using $\hat{E\epsilon^2}$ and an estimator of η^2. η^2 can be taken to be independent of K as long as one allocates sampling points to grids regularly. Hence one can choose c, and so also K, based on past data.

5.3.3 Properties

- Convergence speed: $m^{1/6}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependent noise: no
- Allows for endogenous noise: no

5.3.4 Usage

```
variance_tsrv(estimator,K=2)
variance_tsrvRolling(estimator,K=2,wLength=23400)
```

where

- K is number of subsamples in the slow time series (default: 2)

5.4 Multiple Series Realized Variance

5.4.1 Assumptions

Dependent Noise Structure [4, Podolskij and Vetter, 2009]:

1. The microstructure noise, $\epsilon_{t,i}$, has a zero mean, stationary, and strong mixing stochastic process, with the mixing coefficients decaying exponentially. In addition, $E[(\epsilon_{t,i})^{4+\kappa}]$, for some $\kappa > 0$.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

5.4.2 Estimator

Under most assumptions, this estimator violates the sufficiency principle, whence we define the average lag j realized volatility as

$$[Y,Y]_T^{(J)} = \frac{1}{J}\sum_{r=0}^{J-1}[Y,Y]_T^{(J,r)} = \frac{1}{J}\sum_{t=0}^{n-J}(Y_{t_{i+J}} - Y_{t_i})^2 \tag{5.12}$$

A generalization of TSRV can be defined for $1 \leq J < K \leq n$ as

$$MSRV_t = [Y,Y]_T^{(K)} - \frac{\bar{n}_K}{\bar{n}_J}[Y,Y]_T^{(J)} \xrightarrow{p} IV \tag{5.13}$$

thereby combining the two time scales J and K. Here $\bar{n}_K = (n - K + 1)/K$ and similarly for $\bar{n}_J$.

5.4.3 Properties

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependent noise: yes
- Allows for endogenous noise: no

5.4.4 Usage

```
variance_msrv(estimator,K=2,J=1)
variance_msrvRolling(estimator,K=2,J=1,wLength=23400)
```

where

- K is number of subsamples in the slow time series (default: 2)
- J is number of subsamples in the fast time series (default: 1)

5.5 Modulated Realized Variance

5.5.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

5.5.2 Estimator

Modulated Bipower Variation is written as [4, Podolskij and Vetter, 2009]:

$$MBV(Y,r,l)_n = n^{(r+l)/4-1/2} \sum_{m=1}^{M} \mid \bar{Y}_m^{(K)} \mid^r \mid \bar{Y}_{m+1}^{(K)} \mid, \quad r,l \geq 0; \tag{5.14}$$

$$Y_m^{(K)} = \frac{1}{n/M - K + 1} \sum_{i=(m-1)n/M}^{mn/M-K} (Y_{(i+K)/n} - Y_{i/n}) \tag{5.15}$$

with

$$K = c_1 n^{1/2}, \quad M = \frac{n}{c_2 K} = \frac{n^{1/2}}{c_1 c_2} \tag{5.16}$$

We can choose the constants c_1 and c_2 from specific process:

$$c_1 = 0.25, \quad c_2 = 2. \tag{5.17}$$

Modulated Realized Variance is written as:

$$MRV(Y)_n = \frac{c_1 c_2 MBV(Y,2,0)_n - \nu_2 \hat{\omega}^2}{\nu_1} \xrightarrow{p} IV \tag{5.18}$$

where

$$\nu_1 = \frac{c_1(3c_2 - 4 + (2 - c_2)^3 \bigvee 0)}{3(c_2 - 1)^2} \tag{5.19}$$

$$\nu_2 = \frac{2((c_2 - 1) \bigwedge 1)}{c_1(c_2 - 1)^2} \tag{5.20}$$

5.5.3 Properties

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependent noise: no
- Allows for endogenous noise: no

5.5.4 Usage

```
variance_mrv(estimator)
variance_mrvRolling(estimator,wLength=23400)
```

5.6 Jump Robust Modulated Realized Variance

5.6.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.
4. The price is $Z = Y + J$ where Y is a noisy diffusion process and J denotes a finite activity jump process, that is, J exhibits finitely many jumps on compact intervals.

5.6.2 Estimator

We can construct consistent estimates for the integrated volatility, which are robust to noise and finite activity jumps [4, Podolskij and Vetter, 2009].

$$MBV(Y,r,l)_n = \frac{(c_1c_2/\mu_1^2)(MBV(Z,1,1)_n - \nu_2\hat{\omega}^2)}{\nu_1} \xrightarrow{p} IV \tag{5.21}$$

where

$$\nu_1 = \frac{c_1(3c_2 - 4 + (2-c_2)^3 \bigvee 0)}{3(c_2-1)^2} \tag{5.22}$$

$$\nu_2 = \frac{2((c_2-1) \bigwedge 1)}{c_1(c_2-1)^2} \tag{5.23}$$

Mixed normal distribution with conditional variance [4, Podolskij and Vetter, 2009]:

$$\beta^2 = \frac{2c_1c_2}{\nu_1^2}\int_0^1 (\nu_1\sigma_u^2 + \nu_2\omega^2)^2 du \tag{5.24}$$

Consistent estimator of β is:

$$\beta_n^2 = \frac{2c_1^2c_2^2}{3\nu_1^2} MBV(Y,4,0)_n \tag{5.25}$$

We can choose the constants c_1 and c_2 that minimise the conditional variance. In order to compare our asymptotic variance with the corresponding results of other methods we assume that the volatility process σ is constant. In that case the conditional variance β^2 is minimised by

$$c_1 = \sqrt{\frac{18}{(c_2-1)(4-c_2)} \cdot \frac{\omega}{\sigma}}, \quad c_2 = \frac{8}{5} \tag{5.26}$$

We can choose the constants c_1 and c_2 from specific process:

$$c_1 = 0.25, \quad c_2 = 2. \tag{5.27}$$

5.6.3 Properties

- Converges to integrated variance
- Convergence speed: $m^{1/6}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: yes
- Allows for time dependent noise: no
- Allows for endogenous noise: no

5.6.4 Usage

```
variance_jrmrv(estimator)
variance_jrmrvRolling(estimator,wLength=23400)
```

5.7 Kernel-based Realized Variance

5.7.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is distributed random variable.
2. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

5.7.2 Kernel Types

Flat-Top Kernel	
Bartlett Kernel	$k(x) = 1 - x$
Epanichnikov Kernel	$k(x) = 1 - x^2$
Second order Kernel	$k(x) = 1 - 2x + x^2$
Non-Flat-Top Kernel	
Cubic Kernel	$k(x) = 1 - 3x^2 + 2x^3$
Parzen Kernel	$k(x) = \begin{cases} 2(1-x)^3 & x > 0.5 \\ 1 - 6x^2 + 6x^3 & x < 0.5, \end{cases}$
Tukey Hanning Kernel	$k(x) = \frac{(1 + \sin(\pi/2 - \pi x))}{2}$
Tukey Hanning Modified Kernel	$k(x) = \frac{(1 - \sin(\pi/2 - \pi(1-x)^2))}{2}$
Fifth Order Kernel	$k(x) = 1 - 10x^3 + 15x^4 - 6x^5$
Sixth Order Kernel	$k(x) = 1 - 15x^4 + 24x^5 - 10x^6$
Seventh Order Kernel	$k(x) = 1 - 21x^5 + 35x^6 - 15x^7$
Eighth Order Kernel	$k(x) = 1 - 28x^6 + 48x^7 - 21x^8$

5.7.3 Estimator

Kernel-based Realized Variance is [5, Barndorff-Nielsen et al., 2006]:

$$\tilde{K}(X_\delta) = \gamma_0(X_\delta) + \sum_{h=1}^{H} k\left(\frac{h-1}{H}\right)\tilde{\gamma}_h(X_\delta) \tag{5.28}$$

where $k()$ is kernel function and:

$$\tilde{\gamma}_h(X_\delta) = \gamma_h(X_\delta) + \gamma_{-h}(X_\delta) \tag{5.29}$$

$$\gamma_h(X_\delta) = \sum_{j=1}^{n}(X_{\delta j} - X_{\delta(j-1)})(X_{\delta(j-h)} - X_{\delta(j-h-1)}) \tag{5.30}$$

with $h = -H, \ldots, -1, 0, 1, \ldots, H$ and $n = [1/\delta]$.

Bandwidth for all kernel is:

$$H = cn^{2/3} \tag{5.31}$$

In this case we have the asymptotic distribution given:

$$n^{1/6}\{\tilde{K}(X_\delta) - \int_0^t \sigma_u^2 du\} \xrightarrow{L_s} MN\left[0, 4ck_\bullet^{0,0} t \int_0^t \sigma_u^4 du + 4\omega^4 c^{-2}\{k'(0)^2 + k'(1)^2\}\right] \tag{5.32}$$

where $\omega = (1, 1, k(\frac{1}{H}), \ldots, k\left(\frac{H-1}{H}\right))^T$

$$k_\bullet^{0,0} = \int_0^1 k(x)^2 dx, \quad k_\bullet^{0,2} = \int_0^1 k(x)k''(x)dx \quad k_\bullet^{0,4} = \int_0^1 k(x)k''''(x)dx \tag{5.33}$$

The value of c which minimizes the asymptotic variance is

$$c = d\frac{\omega^{4/3}}{(t\int_0^t \sigma_u^4 du)^{1/3}} \tag{5.34}$$

where

$$d = \left[\frac{2k^2(0)^2 + k'(1)^2}{k_\bullet^{0,0}}\right]^{1/2} \tag{5.35}$$

The lower bound for the asymptotic variance is $6dk_\bullet^{0,0}\omega^{4/3}(t\int_0^t \sigma_u^4 du)^{2/3}$

But for non-flat-top kernel we can review special case.

Bandwidth for non-flat-top kernel is:

$$H = cn^{1/2} \tag{5.36}$$

In this case the asymptotic distribution is given

$$n^{1/4}\{\tilde{K}(X_\delta) - \int_0^t \sigma_u^2 du\} \xrightarrow{L_s}$$
$$MN\left[0, 4ck_\bullet^{0,0} t \int_0^t \sigma_u^4 du - 8c^{-1}k_\bullet^{0,2}\omega^2\left(\int_0^t \sigma_u^2 du + \frac{\omega^2}{2}\right) + 4\omega^4 c^{-3}\{k'''(0) + k_\bullet^{0,4}\}\right] \tag{5.37}$$

For this class of kernels the value of $\hat{c}$ which minimizes the asymptotic variance is

$$c \approx \sqrt{\frac{1}{k_\bullet^{0,0}}\left\{-k_\bullet^{0,2} + \sqrt{(k_\bullet^{0,2})^2 + 3k_\bullet^{0,0} f}\right\}} \tag{5.38}$$

5.7.4 Properties

Flat-Top Kernel

- Convergence speed: $m^{1/6}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependent noise: no
- Allows for endogenous noise: no

Non-Flat-Top Kernel

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependent noise: yes
- Allows for endogenous noise: yes

5.7.5 Usage

```
variance_krv(estimator,kernelName="ParzenKernel",bandwidth=1)
variance_krvRolling(estimator,kernelName="ParzenKernel",bandwidth=1,wLength=23400)
```

where

- $kernelName$ is Kernel name is one of the following (default:"ParzenKernel")
  - "BartlettKernel"
  - "EpanichnikovKernel"
  - "SecondOrderKernel"
  - "CubicKernel"
  - "ParzenKernel"
  - "TukeyHanningKernel"
  - "TukeyHanningModifiedKernel"
  - "FifthOrderKernel"
  - "SixthOrderKernel"
  - "SeventhOrderKernel"
  - "EighthOrderKernel"
- *bandwidth* "optimal" to compute optimal bandwidth from the data, or the value of bandwidth (default:1)

5.8 IV Estimators Comparison

Type	Method	Unbiased	Consistent	Jump Robust	Time dependence noise	Endogenous noise
Plain	RV	NO	NO	NO	NO	NO
Subsampling	$TSRV$	**YES**	**YES**	NO	NO	NO
	$MSRV$	**YES**	**YES**	NO	**YES**	NO
Kernel	KRV_{FT}	**YES**	**YES**	NO	NO	NO
	KRV_{NFT}	**YES**	**YES**	NO	**YES**	**YES**
Range	MRV	**YES**	**YES**	NO	NO	NO
	MRV_{jump}	**YES**	**YES**	**YES**	NO	NO

6 Price Noise Variance

6.1 RV Noise Variance

Assume that the observed (log) price is contaminated by market microstructure noise u (or measurement error), i.e.:

$$Y_{i/m} = X_{i/m} + u_{i/m}, \quad i = 1, 2, \ldots, m \tag{6.1}$$

where $X_{i/m}$ is the latent true, or so-called efficient, price that follows the semimartingale. In this case, the observed intraday return is given by:

$$r_i^{Y(m)} = r_i^{X(m)} + \epsilon_i^{(m)}, \quad i = 1, 2, \ldots, m, \tag{6.2}$$

i.e. by the efficient intraday return $r_i^{X(m)} = X_{i/m} - X_{(i-1)/m}$ and the intraday noise increment $\epsilon_i^{(m)} = u_{i/m} - u_{(i-1)/m}$.

The noise variance $E\epsilon^2$ can be estimated consistently by normalized realized variance over the whole interval $[0,1]$ for noisy data [3, Zhang et al., 2005]:

$$\hat{\omega}^2 = \frac{1}{2n}\sum_{i=1}^{n}(Y_{t_i} - Y_{t_{i-1}})^2 \tag{6.3}$$

6.1.1 Properties

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Unbiased: no
- Consistent: yes
- Jump Robust: yes
- Allows for time dependent noise: no

6.2 Autocovariance Noise Variance

The noise variance can be estimated as the negative of the first order autocovariance of observed returns [6, Oomen, 2005]:

$$\hat{\omega}^2 = \max\left\{0, -\frac{1}{n-1}\sum_{i=1}^{n}(Y_{t_i} - Y_{t_{i-1}})(Y_{t_{i-1}} - Y_{t_{i-2}})\right\} \tag{6.4}$$

6.2.1 Properties

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Unbiased: no
- Consistent: yes
- Jump Robust: yes
- Allows for time dependent noise: no

6.2.2 Usage

```
noise_acnv(estimator)
```

7 Price Quarticity

7.1 Integrated Quarticity

The integrated quarticity is as described in [1, Pigorsch et al.]:

$$IQ = \int_{t-1}^{t} \sigma^4(s)ds. \tag{7.1}$$

7.2 Realized Quarticity

7.2.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

7.2.2 Estimator

The realized fourth-power variation or realized quarticity, defined as [7, Corsi et al., 2005]:

$$RQ_t = \frac{M}{3}\sum_{j=1}^{M} r_{t,j}^4 \xrightarrow{p} IQ \tag{7.2}$$

where M is sampling frequency.

7.2.3 Usage

```
quarticity_rq(estimator)
```

7.3 Realized Quadpower Quarticity

7.3.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

7.3.2 Estimator

A more robust estimator then 7.2 on p. 22, especially in the presence of jumps, is the realized quad-power quarticity [7, Corsi et al., 2005]:

$$RQQ_t = M\frac{\pi^4}{4}\sum_{j=4}^{M} | r_{t,j} || r_{t,j-1} || r_{t,j-2} || r_{t,j-3} | \xrightarrow{p} IQ \tag{7.3}$$

7.3.3 Usage

```
quarticity_rqq(estimator)
```

7.4 Modulated Realized Quarticity

7.4.1 Assumptions

The microstructure noise.

1. The microstructure noise, $\epsilon_{t,i}$, has zero mean and is an independent and identically distributed random variable.
2. The noise is independent of the price process.
3. The variance of $\nu_{t,i} = \epsilon_{t,i} - \epsilon_{t,i-1}$ is $O(1)$.

7.4.2 Estimator

Modulated Realized Quarticity is written as [4, Podolskij and Vetter, 2009]:

$$MRQ(Y)_n = \frac{(c_1c_2/3)MBV(Y,4,0)_n - 2\nu_1\nu_2\hat{\omega}^2 MRV(Y)_n - \nu_2^2(\hat{\omega}^2)^2}{\nu_1^2} \xrightarrow{p} IQ \tag{7.4}$$

where MBV is written as 5.14 on p. 14,

$$\nu_1 = \frac{c_1(3c_2 - 4 + (2-c_2)^3 \bigvee 0)}{3(c_2-1)^2} \tag{7.5}$$

$$\nu_2 = \frac{2((c_2-1)\bigwedge 1)}{c_1(c_2-1)^2} \tag{7.6}$$

We can choose the constants c_1 and c_2 from specific process:

$$c_1 = 0.25, \quad c_2 = 2. \tag{7.7}$$

7.4.3 Properties

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Unbiased: yes
- Consistent: yes
- Jump Robust: no
- Allows for time dependence noise: no
- Allows for endogenous noise: no

7.4.4 Usage

```
quarticity_mrq(estimator)
```

Contents

[1] U. Pigorsch, C. Pigorsch, I. Popov, ”Volatility estimation based on high-frequency data”, In: Duan, JC, Gentle, JE and Hardle E, Handbook of Computational Finance, New York: Springer, 2010

[2] Y. Zu and Peter Boswijk, ”Estimating spot volatility with high-frequency financial data”, Journal of Econometrics, 181(2), pp. 117-135, 2014

[3] 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, vol. 100, No. 472, pp. 1394-1411, 2005.

[4] M. Podolskij and M. Vetter, ”Estimation of volatility functionals in the simultaneous presence of microstructure noise and jumps”, Bernoulli, vol. 15, No. 3, pp. 634-658, 2009.

[5] O.E.Barndorff-Nielsen, P.Reinhard Hansen, A.Lunde, and N.Shephard, ”Designing realized kernels to measure the ex-post variation of equity prices in the presence of noise”, Economics Series Working Papers 264, University of Oxford, Department of Economics, 2006.

[6] R. C. Oomen, ”Comment on realized variance and market microstructure noise by Peter R. Hansen and Asger Lunde,” pp. 1-15, 2005.

[7] Corsi, F., U. Kretschmer, S. Mittnik and C. Pigorsch, ”The Volatility of Realized Volatility”, Econometric Reviews, pp. 46-78, 2008