Contents

1 Package Installation 4
   1.1 Install Latest JDK/JRE Runtime 4
   1.2 Configure Java Environment (Optional) 4
   1.3 Install Required Packages (Optional) 4

2 API Credentials 5
   2.1 Locate API Credentials 5
   2.2 Set API Credentials in R 5

3 Estimator Construction 6
   3.1 User Data 6
      3.1.1 Create Estimator 6
   3.2 Server Data 6
      3.2.1 Create Estimator 6
      3.2.2 Get Symbols List 7

4 Estimator Settings 8
   4.0.1 Results Sampling Interval 8
   4.0.2 Input Sampling Interval 8
   4.0.3 Jumps/Outliers Model 9

5 Price Variance 10
   5.1 Integrated Variance 10
6 Price Noise Variance

6.1 RV Noise Variance
   6.1.1 Properties

6.2 Autocovariance Noise Variance
   6.2.1 Properties
   6.2.2 Usage

7 Price Quarticity

7.1 Integrated Quarticity

7.2 Realized Quarticity
   7.2.1 Assumptions
   7.2.2 Estimator
   7.2.3 Usage

7.3 Realized Quadpower Quarticity
   7.3.1 Assumptions
   7.3.2 Estimator
   7.3.3 Usage

7.4 Modulated Realized Quarticity
   7.4.1 Assumptions
   7.4.2 Estimator
   7.4.3 Properties
   7.4.4 Usage
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:

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

Apply environment changes:

```bash
source /etc/environment
```

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

```bash
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

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

You are now ready to install the PortfolioEffectEstim package directly from www.portfolioeffect.com downloads section.
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](http://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 your 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 \texttt{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.

<table>
<thead>
<tr>
<th>Time</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1412256601000</td>
<td>99.30</td>
</tr>
<tr>
<td>1412256602000</td>
<td>99.33</td>
</tr>
<tr>
<td>1412256603000</td>
<td>99.30</td>
</tr>
<tr>
<td>1412256604000</td>
<td>99.26</td>
</tr>
<tr>
<td>1412256605000</td>
<td>99.36</td>
</tr>
<tr>
<td>1412256606000</td>
<td>99.36</td>
</tr>
<tr>
<td>1412256607000</td>
<td>99.36</td>
</tr>
<tr>
<td>1412256608000</td>
<td>99.38</td>
</tr>
<tr>
<td>1412256609000</td>
<td>99.40</td>
</tr>
<tr>
<td>1412256610000</td>
<td>99.37</td>
</tr>
</tbody>
</table>

If asset symbol is specified, it is silently ignored.

```python
data(spy.data)
```

```python
# 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 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 \texttt{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

```python
# 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.

```python
estimator_availableSymbols(estimator)
```

<table>
<thead>
<tr>
<th>id</th>
<th>description</th>
<th>exchange</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BBC        &quot;BioShares Biotechnology Clinical Trials Fund&quot;</td>
<td>&quot;NASDAQ&quot;</td>
</tr>
<tr>
<td>2</td>
<td>SCS        &quot;Steelcase Inc. Common Stock&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
<tr>
<td>3</td>
<td>BBD        &quot;Banco Bradesco Sa American Depositary Shares&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
<tr>
<td>4</td>
<td>BBG        &quot;Bill Barrett Corporation Common Stock&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
<tr>
<td>5</td>
<td>STPP       &quot;Barclays PLC - iPath US Treasury Steepener ETN&quot;</td>
<td>&quot;NASDAQ&quot;</td>
</tr>
<tr>
<td>6</td>
<td>BBF        &quot;BlackRock Municipal Income Investment Trust&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
<tr>
<td>7</td>
<td>BBH        &quot;Market Vectors Biotech ETF&quot;</td>
<td>&quot;NYSEARCA&quot;</td>
</tr>
<tr>
<td>8</td>
<td>SCON       &quot;Superconductor Technologies Inc. - Common Stock&quot;</td>
<td>&quot;NASDAQ&quot;</td>
</tr>
<tr>
<td>9</td>
<td>SCX        &quot;L.S. Starrett Company (The) Common Stock&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
<tr>
<td>10</td>
<td>BBK        &quot;Blackrock Municipal Bond Trust&quot;</td>
<td>&quot;NYSE&quot;</td>
</tr>
</tbody>
</table>
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.

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

```python
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

\[ X_t = \int_0^t \mu(s)ds + \int_0^t \sigma(s)dW(s) \] (5.1)

where

- \( W_t \) is a standard Brownian Motion,
- the mean process \( \mu \) 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 \( \Delta \) according to [1, Pigorsch et al.]:

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

5.1.1 Returns

Suppose there exist \( m \) intraday equilibrium returns, the \( i \)th intraday return is then defined as:

\[ r_{i}^{X(m)} = X_{i/m} - X_{(i-1)/m}, \quad i = 1, 2, \ldots, m. \] (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,

\[ p_t^X = \alpha_t + m_t; \] (5.4)

where \( \alpha_t \) (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 \( \sigma_t \) is cadlag and bounded away from zero.

3. The process \( \int_0^t \sigma_s^2 ds \) is bounded almost surely for all \( t < \infty \).
5.2.2 Estimator

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

\[
RV_t = \sum_{i=1}^{n} (X_{t_i} - X_{t_{i-1}})^2 
\]

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

\[
\begin{align*}
\text{variance}_r v(\text{estimator}) \\
\text{variance}_r vR(\text{estimator}, wLength=23400)
\end{align*}
\]

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:

$$[Y,Y]_{t|k}^{avg} = \frac{1}{K} \sum_{i=K}^{n} (Y_{ti} - Y_{t(i-k)})^2$$  \hspace{1cm} (5.6)$$

$$[Y,Y]_{t}^{all} = \sum_{i=1}^{n} (Y_{ti} - Y_{t(i-1)})^2$$  \hspace{1cm} (5.7)$$

$$\bar{n} = \frac{n - K + 1}{K},$$  \hspace{1cm} (5.8)$$

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

$$TSRV_t = [Y,Y]_{t}^{avg} - \frac{\bar{n}}{n} [Y,Y]_{t}^{all} \overset{p}{\rightarrow} IV$$  \hspace{1cm} (5.9)$$

A small sample refinement to the estimator give correction:

$$TSRV_t^{adjust} = \left(1 - \frac{\bar{n}}{n}\right)^{-1} TSVR_t \overset{p}{\rightarrow} IV. \hspace{1cm} (5.10)$$

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

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

which can be consistently estimated from data in past time periods (before time $t_0 = 0$), using $E\epsilon^2$ and an estimator of $\eta^2$. $\eta^2$ 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

```python
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 \[ [4, \text{Podolskij and Vetter, 2009}] \]:

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

\[
[Y, Y]_{T}^{(r)} = \frac{1}{J} \sum_{r=0}^{J-1} [Y, Y]_{T}^{(J,r)} = \frac{1}{J} \sum_{t=0}^{n-J} (Y_{t+i} - Y_{t})^2
\]

(5.12)

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

\[
MSRV_t = [Y, Y]_{T}^{(K)} - \frac{\bar{n}_K}{n_J} [Y, Y]_{T}^{(J)} \overset{p}{\to} IV
\]

(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

\[
\text{variance}_{\text{msrv}}(\text{estimator}, K=2, J=1)
\]
\[
\text{variance}_{\text{msrvRolling}}(\text{estimator}, K=2, J=1, \text{wLength}=23400)
\]

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 [Podolskij and Vetter, 2009]:

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

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

with

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

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

$$c_1 = 0.25, \quad c_2 = 2. \quad (5.17)$$

Modulated Realized Variance is written as:

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

where

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

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

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

\[ \text{variance_mrv(}\text{estimator}\text{)} \]
\[ \text{variance_mrvRolling(}\text{estimator, wLength=23400}\text{)} \]

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].

\[ MBV(Y, r, l)_n = \frac{(c_1c_2/\mu^2)(MBV(Z, 1, 1)_n - \nu_2\omega^2)}{\nu_1} \]

where

\[ \nu_1 = \frac{c_1(3c_2 - 4 + (2 - c_2)^3\sqrt{0})}{3(c_2 - 1)^2} \]  
\[ \nu_2 = \frac{2((c_2 - 1)\wedge 1)}{c_1(c_2 - 1)^2} \]

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

\[ \beta^2 = \frac{2c_1c_2}{\nu_1^2} \int_0^1 (\nu_1\sigma_u^2 + \nu_2\omega^2)^2 du \]  

Consistent estimator of \( \beta \) is:

\[ \beta_n^2 = \frac{2c_1c_2}{3\nu_1^2} MBV(Y, 4, 0)_n \]

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 \( \sigma \) is constant. In that case the conditional variance \( \beta^2 \) is minimised by

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

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

\[ c_1 = 0.25, \quad c_2 = 2. \]
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_jrmv(estimator)
variance_jrmvRolling(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

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

5.7.3 Estimator

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

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

where $k()$ is kernel function and:

$$\tilde{\gamma}_h(X_\delta) = \gamma_h(X_\delta) + \gamma_{-h}(X_\delta)$$ \hspace{1cm} (5.29)

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

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

Bandwidth for all kernel is:

$$H = cn^{2/3}$$ \hspace{1cm} (5.31)
In this case we have the asymptotic distribution given:

\[
n^{1/6}\{\tilde{K}(X_δ) - \int_0^t \sigma_u^2 du\} \overset{L}{\rightarrow} MN \left[ 0, 4ck_0^{0.0} + 4\omega^4 c^{-2}\{k'(0)^2 + k'(1)^2\} \right]
\]

(5.32)

where \(\omega = (1, 1, k(H), \ldots, k(H-1))^T\)

\[
k_0^{0.0} = \int_0^1 k(x)^2 dx, \quad k_0^{0.2} = \int_0^1 k(x)k''(x)dx \quad k_0^{0.4} = \int_0^1 k(x)k''''(x)dx
\]

(5.33)

The value of \(c\) which minimizes the asymptotic variance is

\[
c = d \frac{\omega^{4/3}}{(t\int_0^t \sigma_u^4 du)^{1/3}}
\]

(5.34)

where

\[
d = \left[ \frac{2k^2(0)^2 + k'(1)^2}{k_0^{0.0}} \right]^{1/2}
\]

(5.35)

The lower bound for the asymptotic variance is \(6dk_0^{0.0} \omega^{4/3}(t\int_0^t \sigma_u^4 du)^{2/3}\).

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

Bandwidth for non-flat-top kernel is:

\[H = cn^{1/2}\]

(5.36)

In this case the asymptotic distribution is given

\[
n^{1/4}\{\tilde{K}(X_δ) - \int_0^t \sigma_u^2 du\} \overset{L}{\rightarrow}

MN \left[ 0, 4ck_0^{0.0} + 8c^{-1}k_0^{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_0^{0.4}\} \right]
\]

(5.37)

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

\[
c \approx \sqrt{\frac{1}{k_0^{0.0}}} \left\{ -k_0^{0.2} + \sqrt{(k_0^{0.2})^2 + 3k_0^{0.0} f} \right\}
\]

(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

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Unbiased</th>
<th>Consistent</th>
<th>Jump Robust</th>
<th>Time dependence noise</th>
<th>Endogenous noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>RV</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Subsampling</td>
<td>TSRV</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>MSRV</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Kernel</td>
<td>KRV_FT</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>KRV_NFT</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Range</td>
<td>MRV</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td></td>
<td>MRV_jump</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
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.:

\[
Y_{i/m} = X_{i/m} + u_{i/m}, \quad i = 1, 2, \ldots, m
\] (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:

\[
r_i^{Y(m)} = r_i^{X(m)} + \epsilon_i^{(m)}, \quad i = 1, 2, \ldots, m,
\] (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 \( \hat{\omega}^2 \) can be estimated consistently by normalized realized variance over the whole interval \([0, 1]\) for noisy data \[3, \text{Zhang et al., 2005}\]:

\[
\hat{\omega}^2 = \frac{1}{2n} \sum_{i=1}^{n} (Y_{t_i} - Y_{t_{i-1}})^2
\] (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 \[6, \text{Oomen, 2005}\]:

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

\texttt{noise_acnv(\textit{estimator})}
7  Price Quarticity

7.1  Integrated Quarticity

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

\[
IQ = \int_{t-1}^{t} \sigma^4(s) ds.
\]  (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]:

\[
RQ_t = \frac{M}{3} \sum_{j=1}^{M} r_{t,j}^4 \rightarrow IQ
\]  (7.2)

where \( M \) is sampling frequency.

7.2.3  Usage

\texttt{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 than 7.2 on p. 22 especially in the presence of jumps, is the realized quad-power quarticity [7, Corsi et al., 2005]:

$$RQQ_t = M \frac{4}{4} \sum_{j=4}^{M} |r_{t,j}||r_{t,j-1}||r_{t,j-2}||r_{t,j-3}| \overset{P}{\to} IQ \quad (7.3)$$

### 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]:

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

where $MBV$ is written as 5.14 on p. 14,

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

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

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

$$c_1 = 0.25, \quad c_2 = 2. \quad (7.7)$$
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)
```


