Package ‘PortfolioEffectEstim’

September 17, 2016

Type Package
Title High Frequency Price Estimators by PortfolioEffect
Version 1.4
Date 2016-09-17
Depends methods,PortfolioEffectHFT(>= 1.7)
Imports rJava
LazyData yes
ByteCompile TRUE
Maintainer Andrey Kostin <andrey.kostin@portfolioeffect.com>
Description R interface to PortfolioEffect cloud service for estimating high frequency price variance, quarticity, microstructure noise variance, and other metrics in both aggregate and rolling window flavors. Constructed estimators could use client-side market data or access HF intraday price history for all major US Equities. See <https://www.portfolioeffect.com/> for more information on the PortfolioEffect high frequency portfolio analytics platform.

URL https://www.portfolioeffect.com/
License GPL-3
SystemRequirements Java (>= 1.7)
NeedsCompilation no
Repository CRAN
Author Andrey Kostin [aut, cre], Aleksey Zemnitskiy [aut], Oleg Nechaev [aut]

Date/Publication 2016-09-17 19:54:52

R topics documented:
estimator-class ............................................................ 2
estimator_availableSymbols ........................................ 3
estimator-class

Class for storing java Estimator object.

Slots

java: Object of class "jobjRef"

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

Examples

showClass("estimator")
estimator_availableSymbols

Get All Symbol List

Description

Returns a list of symbols.

Usage

estimator_availableSymbols(estimator)

Arguments

estimator Vector of (time, price) observations for market asset when external market data is used.

Value

List of symbols, exchanges and descriptions

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

Examples

```r
## Not run:
estimator=estimator_create(asset='AAPL',fromTime="2014-09-01 09:00:00",
toTime="2014-09-14 16:00:00")
list=estimator_availableSymbols(estimator)

## End(Not run)
```

estimator_create Creates new estimator

Description

Creates new empty estimator object.

Usage

estimator_create(asset, fromTime, toTime, priceData)
estimator_create

Arguments

- **asset**: Unique identifier of the instrument.
- **fromTime**: Start of market data interval in "yyyyMMdd-MM-dd hh:mm:ss" format when internal market data is used. Offset from last available date/time by N days is denoted as "t-N" (e.g. "t-7" denotes offset by 7 days).
- **toTime**: End of market data interval in "yyyyMMdd-MM-dd hh:mm:ss" format when internal market data is used. Offset from last available date/time by N days is denoted as "t-N" (e.g. "t-7" denotes offset by 7 days).
- **priceData**: Vector of (time, price) observations for asset when external market data is used.

Value

- estimator object

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

See Also

- estimator_settings

Examples

```r
## Not run:
data(goog.data)
estimator=estimator_create(priceData=goog.data)
estimator_settings(estimator,resamplingInterval='60s')
util_plot2d(variance_rv(estimator),title="RV")

estimator=estimator_create(asset='AAPL',fromTime="2014-09-01 09:00:00",
toTime="2014-09-14 16:00:00")
estimator_settings(estimator,resamplingInterval='60s')
util_plot2d(variance_tsrv(estimator,K=2),title="TSRV")

estimator=estimator_create(asset='GOOG',fromTime="t-2", toTime="t")
estimator_settings(estimator,resamplingInterval='60s')
util_plot2d(variance_mrv(estimator),title="MRV")

## End(Not run)
```
**estimator_defaultSettings**

*Estimator Default Settings*

**Description**

Advanced settings that regulate how estimator metrics are computed, returned and stored. Default: jumpsModel = "moments", resultsSamplingInterval = "1s", inputSamplingInterval="none"

**Usage**

```
estimator_defaultSettings(estimator)
```

**Arguments**

- **estimator**: Estimator object created using `estimator_create( )` function

**Value**

Void

**Author(s)**

Kostin Andrey <andrey.kostin@portfolioeffect.com>

**See Also**

- `estimator_create`  
- `estimator_getSettings`

**Examples**

```R
## Not run:  
data(spy.data)  
estimator=estimator_create(priceData=spy.data)  
estimator_settings(estimator,   
  inputSamplingInterval = '10s',   
  resultsSamplingInterval = '10s')  
estimator_getSettings(estimator)  
estimator_defaultSettings(estimator)  
estimator_getSettings(estimator)  
## End(Not run)
```
estimator_getSettings  Get Estimator Settings

Description
Method returns active list of settings of a given estimator.

Usage
estimator_getSettings(estimator)

Arguments
estimator  Estimator object created using estimator_create() function

Value
List with estimator settings.

Author(s)
Kostin Andrey <andrey.kostin@portfolioeffect.com>

Examples
## Not run:
dateStart = "2014-11-17 09:30:00"
dateEnd = "2014-11-17 16:00:00"
estimator=estimator_create('AAPL',dateStart,dateEnd)
estimator_settings(estimator,resultsSamplingInterval='60s')
settings=estimator_getSettings(estimator)

## End(Not run)

estimator_settings  Estimator Settings

Description
Advanced settings that regulate how estimator metrics are computed, returned and stored. Default: jumpsModel = "moments", resultsSamplingInterval = "1s", inputSamplingInterval="none"

Usage
estimator_settings(estimator,...)
estimator_settings

Arguments

estimator Estimator object created using estimator_create() function

... One of the following estimator settings:
"jumpsModel" - 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 moments (variance, skewness, kurtosis) and derived metrics, "all" - price jumps are filtered everywhere. Defaults to "moments", which implies that only return moments and related metrics would be using jump-filtered returns in their calculations.
"resultsSamplingInterval" - 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), "last" - last result in a series is returned, "none" - no sampling. 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.
"inputSamplingInterval" - 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.

Value

Void

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

See Also

estimator_create estimator_getSettings

Examples

## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
                   inputSamplingInterval = '10s',
                   resultsSamplingInterval = '10s')
util_plot2d(variance_mrv(estimator),title="MRV")

dateStart = "2014-11-17 09:30:00"
dateEnd = "2014-11-17 16:00:00"
noise_acnv

estimator=estimator_create('AAPL',dateStart,dateEnd)
estimator_settings(estimator,
                   resultsSamplingInterval = '10s')
util_plot2d(variance_mrv(estimator),title="MRV")

## End(Not run)

---

**noise_acnv**  
**Autocovariance Noise Variance**

**Description**

Autocovariance Noise Variance (ACNV) estimates the noise variance based on the autocovariance of returns, rather than the Rescaled Noise Variance (RNV). It is generally preferred to RNV as it leads to a reduction in MSE and is robust to the presence of rare jumps. Also, this approach can be extended straightforwardly to estimate the parameters of higher order noise dependence.

**Usage**

noise_acnv(estimator)

**Arguments**

- **estimator**  
  Vector of (time, price) observations for market asset when external market data is used.

**Details**

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Accounts for additive noise: **yes**
- Accounts for finite price jumps: **yes**
- Accounts for time dependence in noise: **yes**
- Accounts for endogenous effects in noise: **no**

**Value**

a numeric vector of the same length as data.

**Author(s)**

Kostin Andrey <andrey.kostin@portfolioeffect.com>

**References**

noise_nts

See Also
noise_rnv noise_urnv noise_uznv

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
inputSamplingInterval = '10s',
resultsSamplingInterval = '10s')
util_plot2d(noise_acnv(estimator),title="ACNV")
## End(Not run)
```

Description
Noise to Signal Ratio is a measure that compares the level of noise to the level of a desired signal.

Usage
noise_nts(estimator)

Arguments
- `estimator` Vector of (time, price) observations for market asset when external market data is used.

Value
a numeric vector of the same length as data.

Author(s)
Kostin Andrey <andrey.kostin@portfolioeffect.com>

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
inputSamplingInterval = '10s',
resultsSamplingInterval = '10s')
```
Rescaled Noise Variance (RNV) is an asymptotically consistent estimator of noise volatility when dealing with additive microstructure noise. It is derived based on Realized Variance property of convergence to noise variance with the increase of sampling frequency.

### Usage

```r
noise_rnv( estimator )
```

### Arguments

- **estimator**: Vector of (time, price) observations for market asset when external market data is used.

### Details

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

### Value

a numeric vector of the same length as data.

### Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

### References

**noise_urnv**

**Unbiased Rescaled Noise Variance**

**Description**

Unbiased Rescaled Noise Variance (URNV) corrects for a bias of Rescaled Noise Variance.

**Usage**

```r
noise_urnv( estimator )
```

**Arguments**

- `estimator` Vector of (time, price) observations for market asset when external market data is used.

**Details**

- Convergence speed: \( m^{1/2} \) (\( m \) - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

**Value**

a numeric vector of the same length as input data.

**Author(s)**

Kostin Andrey <andrey.kostin@portfolioeffect.com>
References

See Also
noise_rnv noise_acnv noise_uznv

Examples

## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(noise_urnv(estimator),title="URNV")

## End(Not run)

---

**noise_uznv**

*Uncertainty Zones Noise Variance*

Description
Uncertainty Zones Noise Variance (UZNV) based on the concept of uncertainty zones.

Usage
noise_uznv(estimator)

Arguments

- **estimator**
  Vector of (time, price) observations for market asset when external market data is used.

Details
- Convergence speed: $m^{1/2}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: yes
Value

a numeric vector of the same length as input data.

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

References


See Also

noise_rnv noise_urnv noise_acnv

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
   inputSamplingInterval = '10s',
   resultsSamplingInterval = '10s')
util_plot2d(noise_uznv(estimator),title="UZNV")

## End(Not run)
```

Description

Method returns active list of settings of a given estimator.

Usage

```r
price(estimator)
```

Arguments

- `estimator` Estimator object created using `estimator_create()` function

Value

numeric vector of prices.
**quarticity_mrq**

**Author(s)**
Kostin Andrey <andrey.kostin@portfolioeffect.com>

**Examples**
```r
## Not run:
dateStart = "2014-11-17 09:30:00"
dateEnd = "2014-11-17 16:00:00"
estimator=estimator_create('AAPL',dateStart,dateEnd)
estimator_settings(estimator,resultsSamplingInterval='60s')
AAPL=price(estimator)
util_plot2d(AAPL,title='AAPL')

## End(Not run)
```

**Description**
Modulated Realized Quarticity (MRQ) is an asymptotically unbiased estimator of integrated quarticity in the presence of microstructure noise.

**Usage**
```
quarticity_mrq(estimator)
```

**Arguments**
estimator  Vector of (time, price) observations for market asset when external market data is used.

**Details**
- Convergence speed: $m^{1/4}$ (m - number of observation)
- Accounts for additive noise: **yes**
- Accounts for finite price jumps: **yes**
- Accounts for time dependence in noise: **no**
- Accounts for endogenous effects in noise: **no**

**Value**
a numeric vector of the same length as input data.

**Author(s)**
Kostin Andrey <andrej.kostin@snowfallsystems.com>
References


See Also

quarticity_rq quarticity_rqq quarticity_rtq quarticity_mtq

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(quarticity_mrq(estimator),title="MRQ")

## End(Not run)
```

---

**quarticity_mtq**  
*Modulated Tripower Quarticity*

**Description**

Modulated Tri-power Quarticity (MTQ) is an asymptotically unbiased estimator of integrated quarticity in the presence of microstructure noise. This estimator is also robust to finite activity jumps in price.

**Usage**

```r
quarticity_mtq(estimator)
```

**Arguments**

- `estimator` Vector of (time, price) observations for market asset when external market data is used.

**Details**

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: yes
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no
quarticity_rq

Value

A numeric vector of the same length as input data.

Author(s)

Kostin Andrey <andrei.kostin@snowfallsystems.com>

References


See Also

quarticity_rq quarticity_rqq quarticity_rqqt quarticity_mrqt

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(quarticity_mrqt(estimator),title="MTQ")
## End(Not run)
```

---

quarticity_rq

Realized Quarticity

Description

Realized Quarticity (RQ) is an asymptotically unbiased estimator of integrated quarticity in the absence of microstructure noise.

Usage

quarticity_rq(estimator)

Arguments

estimator  Vector of (time, price) observations for market asset when external market data is used.
quarticity_rqq

Details
- Convergence speed: \( n^{1/4} \) (n - number of observation)
- Accounts for additive noise: no
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

Value
a numeric vector of the same length as input data.

Author(s)
Kostin Andrey <andrei.kostin@snowfallsystems.com>

References

See Also
quarticity_mrq quarticity_rqq quarticity_rtq quarticity_mtq

Examples
```
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(quarticity_rq(estimator),title="RQ")

## End(Not run)
```

---

quarticity_rqq $\text{Realized Quadpower Quarticity}$

Description
Realized Quadpower Quarticity (RQQ) is an asymptotically unbiased estimator of integrated quarticity in the absence of microstructure noise.
Usage

quarticity_rqq(estimator)

Arguments

estimator Vector of (time, price) observations for market asset when external market data is used.

Details

- Convergence speed: \( m^{1/4} \) (m - number of observation)
- Accounts for additive noise: no
- Accounts for finite price jumps: yes
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

Author(s)

Kostin Andrey <andrei.kostin@snowfallsystems.com>

References


See Also

quarticity_rq quarticity_mrq quarticity_rtq quarticity_mtq

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(quarticity_rqq(estimator),title="RQQ")

## End(Not run)
```
Description

Realized Tri-power Quarticity (RTQ) is an asymptotically unbiased estimator of integrated quarticity in the absence of microstructure noise.

Usage

quarticity_rtq(estimator)

Arguments

estimator
Vector of (time, price) observations for market asset when external market data is used.

Details

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Accounts for additive noise: no
- Accounts for finite price jumps: yes
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

Value

a numeric vector of the same length as input data.

Author(s)

Kostin Andrey <andrei.kostin@snowfallsystems.com>

References


See Also

quarticity_rq quarticity_rqq quarticity_mrq quarticity_mtq
Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
   inputSamplingInterval = '10s',
   resultsSamplingInterval = '10s')
util_plot2d(quarticity_rtq(estimator),title="RTQ")

## End(Not run)
```

---

**variance_jrmrv**

*Jump Robust Modulated Realized Variance*

**Description**

Jump Robust Modulated Realized Variance (JMRV) is an integrated variance estimator introduced by Podolskij and Vetter. It is based on the concept of multipower variation, is robust to finite activity jumps and assumes additive noise structure.

**Usage**

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

**Arguments**

- **estimator**: Vector of (time, price) observations for market asset when external market data is used.
- **wLength**: Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)

**Details**

- Converges to integrated variance
  - Convergence speed: \( m^{1/6} \) (\( m \) - number of observation)
  - Accounts for additive noise: yes
  - Accounts for finite price jumps: yes
  - Accounts for time dependence in noise: no
  - Accounts for endogenous effects in noise: no

**Value**

a numeric vector of the same length as input data.
variance_krv

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

References


See Also

variance_rv variance_tsr variance_msr variance_mrv variance_uzrv variance_krv

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
inputSamplingInterval = '10s',
resultsSamplingInterval = '10s')
util_plot2d(variance_jrmrv(estimator),title='JMRV',legend='Simple')+
util_line2d(variance_jrmvRolling(estimator,wLength=3600),legend='Rolling Window')

## End(Not run)
```

---

variance_krv

**Kernel Realized Variance**

Description

Kernel Realized Variance (KRV) is an asymptotically consistent estimator of integrated volatility based on the concept of realized kernels for dealing with additive microstructure noise.

Usage

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

Arguments

- `estimator`: Vector of (time, price) observations for market asset when external market data is used.
- `wLength`: Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)
- `kernelName`: Kernel name is one of the following (default:"ParzenKernel")
  - "BartlettKernel"
- "EpanchikovKernel"
- "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)

**Details**

**Flat Top kernel types:**
(Bartlett, Epanchikov and Second order kernel)
- Convergence speed: $m^{1/6}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

**Non Flat Top kernel types:**
(Cubic, Parzen, Tukey Hanning, Tukey Hanning modified and 5, 6, 7, 8 order kernel)
- Convergence speed: $m^{1/4}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: yes
- Accounts for endogenous effects in noise: yes

**Value**

a numeric vector of the same length as input data.

**Author(s)**

Kostin Andrey <andrey.kostin@portfolioeffect.com>

**References**

**variance_mrv**

**Modulated Realized Variance**

**Description**

Modulated Realized Variance (MRV) is an integrated variance estimator introduced by Podolskij and Vetter. It is based on the concept of multipower variation and assumes additive noise structure.

**Usage**

```r
data(spy.data)  # Not run
estimator = estimator_create(priceData = spy.data)
estimator_settings(estimator,
  inputSamplingInterval = '10s',
  resultsSamplingInterval = '10s')
util_plot2d(variance_krv(estimator, kernelName = "EpanechnikovKernel"),
  title = 'KRV', legend = 'Simple')
util_line2d(variance_krvRolling(estimator, kernelName = "ParzenKernel",
  wLength = 3600), legend = 'Rolling Window')
```

**Arguments**

- **estimator**: Vector of (time, price) observations for market asset when external market data is used.
- **wLength**: Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)

**Details**

- Convergence speed: $m^{1/4}$ (m - number of observation)
- Accounts for additive noise: **yes**
- Accounts for finite price jumps: **no**
- Accounts for time dependence in noise: **no**
- Accounts for endogenous effects in noise: **no**
**Value**

a numeric vector of the same length as input data.

**Author(s)**

Kostin Andrey <andrey.kostin@portfolioeffect.com>

**References**


**See Also**

variance_rv variance_rsv variance_msrv variance_jrmrv variance_uzrv variance_krv

**Examples**

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(variance_mrv(estimator),title='MRV',legend='Simple')+
util_line2d(variance_mrvRolling(estimator,wLength=3600),legend='Rolling Window')

## End(Not run)
```

---

**variance_msrv**  
*Multiple Scales Realized Variance*

**Description**

Multiple Series Realized Variance (MSRV) is a generalization of the TSRV estimator of integrated volatility. It uses multiple time scales to account for the effects of additive market microstructure noise.

**Usage**

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

- estimator: Vector of (time, price) observations for market asset when external market data is used.
- K: number of subsamples in the slow time series (default: 2)
- J: number of subsamples in the fast time series (default: 1)
- wLength: Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)

Details

- Convergence speed: \( m^{1/4} \) (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: yes
- Accounts for endogenous effects in noise: no

Value

a numeric vector of the same length as input data.

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

References


See Also

- variance_rv
- variance_tsr
- variance_jrmrv
- variance_mrv
- variance_uzrv
- variance_krv

Examples

```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
    inputSamplingInterval = '10s',
    resultsSamplingInterval = '10s')
util_plot2d(variance_msrv(estimator),title='MSRV',legend='Simple')+
util_line2d(variance_msrvRolling(estimator,wLength=3600),legend='Rolling Window')
```

## End(Not run)
Realized Variance (RV) is the sum of squared returns. For instance, the RV can be the sum of squared daily returns for a particular month, which would yield a measure of price variation over this month. This variance estimator does not account for market microstructure effects.

Usage

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

Arguments

estimator Vector of (time, price) observations for market asset when external market data is used.
wLength Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)

Details

- Convergence speed: $m^{1/2}$ (m - number of observation)
- Accounts for additive noise: no
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no

Value

A vector of integrated variance estimates

Author(s)

Kostin Andrey <andrey.kostin@portfolioeffect.com>

References

Two Scales Realized Variance

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 additive market microstructure noise.

Usage

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

Arguments

- `estimator`: Vector of (time, price) observations for market asset when external market data is used.
- `K`: number of subsamples in the slow time series (default: 2)
- `wLength`: Length of a rolling window for rolling estimators. Default window length is 23400 (number of seconds in a trading day)

Details

- Convergence speed: $m^{1/6}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: no
Uncertainty Zones Realized Variance (UZRV) is an integrated variance estimator that accounts for stochastic rounding noise like bid-ask bounce effects.

Usage

```r
variance_uzrv(estimator)
```

Arguments

- `estimator` : Vector of (time, price) observations for market asset when external market data is used.
Details
- Convergence speed: $m^{1/2}$ (m - number of observation)
- Accounts for additive noise: yes
- Accounts for finite price jumps: no
- Accounts for time dependence in noise: no
- Accounts for endogenous effects in noise: yes

Value
a numeric vector of the same length as input data.

Author(s)
Kostin Andrey <andrey.kostin@portfolioeffect.com>

References

See Also
variance_rv variance_tsrv variance_msrv variance_mrv variance_jrmrv variance_krv

Examples
```r
## Not run:
data(spy.data)
estimator=estimator_create(priceData=spy.data)
estimator_settings(estimator,
   inputSamplingInterval = '10s',
   resultsSamplingInterval = '10s')
util_plot2d(variance_uzrv(estimator),title="UZRV")

## End(Not run)
```
Index

*Topic **PortfolioEffectEstim**, nonparametric, models
  estimator-class, 2
  estimator_create, 3
  estimator_defaultSettings, 5
  estimator_getSettings, 6
  estimator_settings, 6
  noise_acnv, 8
  noise_nts, 9
  noise_rnv, 10
  noise_urnv, 11
  noise_uznv, 12
  price, 13
  quarticity_mrq, 14
  quarticity_mtq, 15
  quarticity_rq, 16
  quarticity_rqq, 17
  quarticity_rqq, 19
  variance_jrmrv, 20
  variance_krv, 21
  variance_mr, 23
  variance_msrv, 24
  variance_rv, 26
  variance_tsrv, 27
  variance_uzrv, 28
*Topic **PortfolioEffectHFT**
  estimator_availableSymbols, 3
*Topic **classes**
  estimator-class, 2
*Topic **estimator_availableSymbols**
  estimator_availableSymbols, 3
*Topic **estimator_create**
  estimator_create, 3
*Topic **estimator_defaultSettings**
  estimator_defaultSettings, 5
*Topic **estimator_getSettings**
  estimator_getSettings, 6
*Topic **estimator_settings**
  estimator_settings, 6
*Topic **noise_acnv**
  noise_acnv, 8
*Topic **noise_nts**
  noise_nts, 9
*Topic **noise_rnv**
  noise_rnv, 10
*Topic **noise_urnv**
  noise_urnv, 11
*Topic **noise_uznv**
  noise_uznv, 12
*Topic **price**
  price, 13
*Topic **quarticity_mrq**
  quarticity_mrq, 14
*Topic **quarticity_mtq**
  quarticity_mtq, 15
*Topic **quarticity_rq**
  quarticity_rq, 17
*Topic **quarticity_rqq**
  quarticity_rqq, 19
*Topic **quarticity_rqq**
  quarticity_rqq, 19
*Topic **variance_jrmrv**
  variance_jrmrv, 20
*Topic **variance_krv**
  variance_krv, 21
*Topic **variance_mr**
  variance_mr, 23
*Topic **variance_msrv**
  variance_msrv, 24
*Topic **variance_rv**
  variance_rv, 26
*Topic **variance_tsrv**
  variance_tsrv, 27
*Topic **variance_uzrv**
  variance_uzrv, 28

estimator-class, 2
estimator_availableSymbols, 3
INDEX

estimator_create, 3, 5, 7
estimator_create(), 5–7, 13
estimator_defaultSettings, 5
estimator_getSettings, 5, 6, 7
estimator_settings, 4, 6

noise_acnv, 8, 11–13
noise_nts, 9
noise_rnv, 9, 10, 12, 13
noise_urnv, 9, 11, 11, 13
noise_uznv, 9, 11, 12, 12

price, 13

quarticity_mrq, 14, 16–19
quarticity_mtq, 15, 15, 17–19
quarticity_rq, 15, 16, 16, 18, 19
quarticity_rqq, 15–17, 17, 19
quarticity_rtq, 15–18, 19

variance_jrmrv, 20, 23–25, 27–29
variance_jrmrvRolling (variance_jrmrv), 20
variance_krv, 21, 21, 24, 25, 27–29
variance_krvRolling (variance_krv), 21
variance_mrv, 21, 23, 23, 25, 27–29
variance_mrvRolling (variance_mrv), 23
variance_msrv, 21, 23, 24, 24, 27–29
variance_msrvRolling (variance_msrv), 24
variance_rv, 21, 23–25, 26, 28, 29
variance_rvRolling (variance_rv), 26
variance_tsrv, 21, 23–25, 27, 27, 29
variance_tsrvRolling (variance_tsrv), 27
variance_uzrv, 21, 23–25, 27, 28, 28