

# Package ‘highfrequency’

September 28, 2020

**Version** 0.7.0.1

**Date** 2020-09-24

**Title** Tools for Highfrequency Data Analysis

**Maintainer** Kris Boudt

<kris.boudt@ugent.be>

**Description** Provide functionality to manage, clean and match highfrequency trades and quotes data, calculate various liquidity measures, estimate and forecast volatility, detect price jumps and investigate microstructure noise and intraday periodicity.

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/jonathancornelissen/highfrequency>

**Depends** R (>= 3.5.0)

**Imports** xts, zoo, Rcpp, RcppArmadillo, graphics, methods, stats, utils, grDevices, robustbase, cubature, mvtnorm, data.table (>= 1.12.0), RcppRoll, quantmod, sandwich

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** covr, FKF, BMS, rugarch, testthat, knitr, rmarkdown

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Kris Boudt [aut, cre] (<<https://orcid.org/0000-0002-1000-5142>>),  
Jonathan Cornelissen [aut],  
Scott Payseur [aut],  
Giang Nguyen [ctb],  
Onno Kleen [aut] (<<https://orcid.org/0000-0003-4731-4640>>),  
Emil Sjoerup [aut]

**Repository** CRAN

**Date/Publication** 2020-09-28 13:40:02 UTC

**R topics documented:**

highfrequency-package . . . . .	4
aggregateQuotes . . . . .	4
aggregateTrades . . . . .	6
aggregateTS . . . . .	7
AJumpTest . . . . .	9
autoSelectExchangeQuotes . . . . .	11
autoSelectExchangeTrades . . . . .	12
BNSJumpTest . . . . .	13
businessTimeAggregation . . . . .	16
exchangeHoursOnly . . . . .	17
getAlphaVantageData . . . . .	18
getLiquidityMeasures . . . . .	19
getPrice . . . . .	23
getTradeDirection . . . . .	24
HARmodel . . . . .	25
hasQty . . . . .	28
HEAVYmodel . . . . .	29
intradayJumpTest . . . . .	31
ivInference . . . . .	33
JOjumpTest . . . . .	35
knChooseReMeDI . . . . .	37
leadLag . . . . .	39
listAvailableKernels . . . . .	40
listCholCovEstimators . . . . .	41
lltc . . . . .	41
makePsd . . . . .	42
makeReturns . . . . .	43
matchTradesQuotes . . . . .	43
medRQ . . . . .	44
medRV . . . . .	45
mergeQuotesSameTimestamp . . . . .	46
mergeTradesSameTimestamp . . . . .	47
minRQ . . . . .	48
minRV . . . . .	49
MRC . . . . .	50
noZeroPrices . . . . .	52
noZeroQuotes . . . . .	53
quotesCleanup . . . . .	53
rankJumpTest . . . . .	55
rAVGCov . . . . .	57
rBeta . . . . .	58
rBPCov . . . . .	60
rCholCov . . . . .	61
rCov . . . . .	62
realizedLibrary . . . . .	64
refreshTime . . . . .	64

ReMeDI . . . . .	65
rHYCov . . . . .	66
rKernelCov . . . . .	67
rKurt . . . . .	69
rmLargeSpread . . . . .	70
rmNegativeSpread . . . . .	70
rmOutliersQuotes . . . . .	71
rMPV . . . . .	72
rmTradeOutliers . . . . .	74
rmTradeOutliersUsingQuotes . . . . .	74
rOWCov . . . . .	75
rQPVar . . . . .	77
rQuar . . . . .	78
rRTSCov . . . . .	79
rSemiCov . . . . .	82
rSkew . . . . .	83
rSV . . . . .	85
rThresholdCov . . . . .	86
rTPVar . . . . .	87
RTQ . . . . .	88
rTSCov . . . . .	89
RV . . . . .	91
salesCondition . . . . .	92
sample5MinPrices . . . . .	92
sample5MinPricesJumps . . . . .	93
sampleQData . . . . .	94
sampleQDataMicroseconds . . . . .	94
sampleQDataRaw . . . . .	95
sampleQDataRawMicroseconds . . . . .	95
sampleReal5MinPrices . . . . .	96
sampleReturns5Min . . . . .	96
sampleTData . . . . .	96
sampleTDataMicroseconds . . . . .	97
sampleTDataRaw . . . . .	98
sampleTDataRawMicroseconds . . . . .	98
sbux . . . . .	99
selectExchange . . . . .	99
SP500RM . . . . .	100
spotDrift . . . . .	101
spotVol . . . . .	102
tradesCleanup . . . . .	108
tradesCleanupUsingQuotes . . . . .	110

---

highfrequency-package *highfrequency: Tools for Highfrequency Data Analysis*

---

### Description

The highfrequency package provides numerous tools for analyzing financial (highfrequency) data. It has five main goals:

- Provide functionality to manage, clean and match highfrequency trades and quotes data,
- calculate various liquidity measures,
- estimate and forecast volatility,
- detect price jumps and
- investigate microstructure noise and intraday periodicity.

To learn more about highfrequency, start with the vignettes: `'browseVignettes(package = "highfrequency")'`

### Author(s)

Kris Boudt, Jonathan Cornelissen, Scott Payseur

Maintainer: Kris Boudt <Kris.Boudt@ugent.be>

Contributors: Giang Nguyen, Onno Kleen, Emil Sjoerup

Thanks: We would like to thank Brian Peterson, Chris Blakely, Eric Zivot and Maarten Schermer. We are also grateful to Dirk Eddelbuettel for his extraordinary support as a mentor during the Google Summer of Code 2019.

### See Also

Useful links:

- <https://github.com/jonathancornelissen/highfrequency>

---

aggregateQuotes

*Aggregate a data.table or xts object containing quote data*

---

### Description

Function returns a data.table or xts object containing the aggregated quote data with columns "SYMBOL", "EX", "BID", "BIDSIZ", "OFR", "OFRSIZ". See [sampleQData](#) for an example of the argument qData.

**Usage**

```
aggregateQuotes(
  qData,
  on = "minutes",
  k = 5,
  marketOpen = "09:30:00",
  marketClose = "16:00:00",
  tz = "GMT"
)
```

**Arguments**

qData	data.table or xts object to be aggregated, containing the intraday quote data of a stock for one day.
on	character, indicating the time scale in which "k" is expressed. Possible values are: "secs", "seconds", "mins", "minutes", "hours". xts object to the 5 minute frequency, set k=5 and on = "minutes".
k	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate an object to the 5 minute frequency set k = 5 and on = "minutes".
marketOpen	the market opening time, by default: marketOpen = "09:30:00".
marketClose	the market closing time, by default: marketClose = "16:00:00".
tz	time zone used, by default: tz = "GMT".

**Details**

The output "BID" and "OFR" columns are constructed using previous tick aggregation.

The variables "BIDSIZ" and "OFRSIZ" are aggregated by taking the sum of the respective inputs over each interval.

The timestamps of the new time series are the closing times of the intervals.

Please note: Returned objects always contain the first observation (i.e. opening quotes,...).

**Value**

a data.table or xts object containing the aggregated time series.

A data.table or an xts object containing the aggregated quote data.

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen

**Examples**

```
# aggregate quote data to the 30 second frequency
qDataAggregated <- aggregateQuotes(sampleQData, on = "seconds", k = 30)
head(qDataAggregated)
```

---

aggregateTrades	<i>Aggregate a data.table or xts object containing trades data</i>
-----------------	--------------------------------------------------------------------

---

### Description

Function returns new time series as a data.table or xts object where first observation is always the opening price and subsequent observations are the closing prices over the interval.

### Usage

```
aggregateTrades(
  tData,
  on = "minutes",
  k = 5,
  marketOpen = "09:30:00",
  marketClose = "16:00:00",
  tz = "GMT"
)
```

### Arguments

tData	data.table or xts object to be aggregated, containing the intraday price series of a stock for possibly multiple days.
on	character, indicating the time scale in which "k" is expressed. Possible values are: "secs", "seconds", "mins", "minutes", "hours".
k	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate an object to the 5 minute frequency set k = 5 and on = "minutes".
marketOpen	the market opening time, by default: marketOpen = "09:30:00".
marketClose	the market closing time, by default: marketClose = "16:00:00".
tz	time zone used, by default: tz = "GMT".

### Details

The timestamps of the new time series are the closing times and/or days of the intervals.

The output "PRICE" column is constructed using previous tick aggregation.

The variable "SIZE" is aggregated by taking the sum over each interval.

The variable "VWPRICE" is the aggregated price weighted by volume.

The timestamps of the new time series are the closing times of the intervals.

In case of previous tick aggregation or on = "seconds"/"minutes"/"hours", the element of the returned series with e.g. timestamp 09:35:00 contains the last observation up to that point, including the value at 09:35:00 itself.

### Value

A data.table or xts object containing the aggregated time series.

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen.

**Examples**

```
# aggregate trade data to 5 minute frequency
tDataAggregated <- aggregateTrades(sampleTData, on = "minutes", k = 5)
head(tDataAggregated)
```

---

aggregateTS

*Aggregate a time series*

---

**Description**

Function returns aggregated time series as xts object. It can handle irregularly spaced timeseries and returns a regularly spaced one. Use univariate timeseries as input for this function, and check out [aggregateTrades](#) and [aggregateQuotes](#) to aggregate Trade or Quote data objects.

**Usage**

```
aggregateTS(
  ts,
  FUN = "previoustick",
  on = "minutes",
  k = 1,
  weights = NULL,
  dropna = FALSE,
  tz = NULL
)
```

**Arguments**

ts	xts object to aggregate.
FUN	function to apply over each interval. By default, previous tick aggregation is done. Alternatively one can set e.g. FUN = "mean". In case weights are supplied, this argument is ignored and a weighted average is taken.
on	character, indicating the time scale in which "k" is expressed. Possible values are: "secs", "seconds", "mins", "minutes", "hours", "days", "weeks", "ticks".
k	positive integer, indicating the number of periods to aggregate over. For example, to aggregate an xts object to the five-minute frequency set k = 5 and on = "minutes".
weights	By default, no weighting scheme is used. When you assign an xts object with weights to this argument, a weighted mean is taken over each interval. Of course, the weights should have the same timestamps as the supplied time series.

dropna	boolean, which determines whether empty intervals should be dropped. By default, an NA is returned in case an interval is empty, except when the user opts for previous tick aggregation, by setting FUN = "previousstick" (default).
tz	character denoting which timezone the output should be in. Defaults to "GMT"

### Details

The timestamps of the new time series are the closing times and/or days of the intervals. E.g. for a weekly aggregation the new timestamp is the last day in that particular week (namely sunday).

In case of previous tick aggregation, for on = "seconds"/"minutes"/"hours", the element of the returned series with e.g. timestamp 09:35:00 contains the last observation up to that point, including the value at 09:35:00 itself.

Please note: In case an interval is empty, by default an NA is returned.. In case e.g. previous tick aggregation it makes sense to fill these NA's by the function `na.locf` (last observation carried forward) from the zoo package.

In case on = "ticks", the sampling is done such the sampling starts on the first tick, and the last tick is always included For example, if 14 observations are made on one day, and these are 1, 2, 3, ... 14. Then, with on = "ticks" and k = 3, the output will be 1, 4, 7, 10, 13, 14.

### Value

An xts object containing the aggregated time series.

### Author(s)

Jonathan Cornelissen and Kris Boudt

### Examples

```
#load sample price data
ts <- sampleTData$PRICE

#Previous tick aggregation to the 5-minute sampling frequency:
tsagg5min <- aggregateTS(ts, on = "minutes", k = 5)
head(tsagg5min)
#Previous tick aggregation to the 30-second sampling frequency:
tsagg30sec <- aggregateTS(ts, on = "seconds", k = 30)
tail(tsagg30sec)

tsagg3ticks <- aggregateTS(ts, on = "ticks", k = 3)
```



AJumpTest

*Ait-Sahalia and Jacod (2009) tests for the presence of jumps in the price series.*

### Description

This test examines the presence of jumps in highfrequency price series. It is based on the theory of Ait-Sahalia and Jacod (2009) (AJ). It consists in comparing the multipower variation of equispaced returns computed at a fast time scale ( $h$ ),  $r_{t,i}$  ( $i = 1, \dots, N$ ) and those computed at the slower time scale ( $kh$ ),  $y_{t,i}$  ( $i = 1, \dots, N/k$ ).

They found that the limit (for  $N \rightarrow \infty$ ) of the realized power variation is invariant for different sampling scales and that their ratio is 1 in case of jumps and  $k^{p/2} - 1$  if no jumps. Therefore the AJ test detects the presence of jump using the ratio of realized power variation sampled from two scales. The null hypothesis is no jumps.

The function returns three outcomes: 1.z-test value 2.critical value under confidence level of 95% and 3.p-value.

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

And there is  $N/k$  equispaced returns in period  $t$ . Let  $y_{t,i}$  be a return (with  $i = 1, \dots, N/k$ ) in period  $t$ .

Then the AJumpTest is given by:

$$\text{AJumpTest}_{t,N} = \frac{S_t(p, k, h) - k^{p/2-1}}{\sqrt{V_{t,N}}}$$

in which,

$$S_t(p, k, h) = \frac{PV_{t,M}(p, kh)}{PV_{t,M}(p, h)}$$

$$PV_{t,N}(p, kh) = \sum_{i=1}^{N/k} |y_{t,i}|^p$$

$$PV_{t,N}(p, h) = \sum_{i=1}^N |r_{t,i}|^p$$

$$V_{t,N} = \frac{N(p, k)A_{t,N(2p)}}{NA_{t,N(p)}}$$

$$N(p, k) = \left( \frac{1}{\mu_p^2} \right) (k^{p-2}(1+k))\mu_{2p} + k^{p-2}(k-1)\mu_p^2 - 2k^{p/2-1}\mu_{k,p}$$

$$A_{t,n(2p)} = \frac{(1/N)^{(1-p/2)}}{\mu_p} \sum_{i=1}^N |r_{t,i}|^p \text{ for } |r_j| < \alpha(1/N)^w$$

$$\mu_{k,p} = E(|U|^p |U + \sqrt{k-1}V|^p)$$

$U, V$ : independent standard normal random variables;  $h = 1/N$ ;  $p, k, \alpha, w$ : parameters.

### Usage

```
AJjumpTest(
  pData,
  p = 4,
  k = 2,
  alignBy = NULL,
  alignPeriod = NULL,
  alphaMultiplier = 4,
  makeReturns = FALSE,
  ...
)
```

### Arguments

<code>pData</code>	a zoo/xts object containing all prices in period t for one asset.
<code>p</code>	can be chosen among 2 or 3 or 4. The author suggests 4. 4 by default.
<code>k</code>	can be chosen among 2 or 3 or 4. The author suggests 2. 2 by default.
<code>alignBy</code>	a string, align the tick data to "seconds" "minutes" "hours"
<code>alignPeriod</code>	an integer, align the tick data to this many [seconds minutes hours].
<code>alphaMultiplier</code>	alpha multiplier
<code>makeReturns</code>	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.
<code>...</code>	additional arguments.

### Details

The theoretical framework underlying jump test is that the logarithmic price process  $X_t$  belongs to the class of Brownian semimartingales, which can be written as:

$$X_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

where  $a$  is the drift term,  $\sigma$  denotes the spot volatility process,  $W$  is a standard Brownian motion and  $Z$  is a jump process defined by:

$$Z_t = \sum_{j=1}^{N_t} k_j$$

where  $k_j$  are nonzero random variables. The counting process can be either finite or infinite for finite or infinite activity jumps.

The Ait-Sahalia and Jacod test is that: Using the convergence properties of power variation and its dependence on the time scale on which it is measured, Ait-Sahalia and Jacod (2009) define a new variable which converges to 1 in the presence of jumps in the underlying return series, or to another deterministic and known number in the absence of jumps. (Theodosiou& Zikes(2009))

## Value

list or xts in case the input prices span more than one day.

## Author(s)

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

## References

Ait-Sahalia, Y. and Jacod, J. (2009). Testing for jumps in a discretely observed process. *The Annals of Statistics*, 37(1), 184- 222.

Theodosiou, M., & Zikes, F. (2009). A comprehensive comparison of alternative tests for jumps in asset prices. Unpublished manuscript, Graduate School of Business, Imperial College London.

## Examples

```
AJjumpTest(sampleTData$PRICE, p = 2, k = 3, alignBy = "seconds",  
alignPeriod = 5, makeReturns = TRUE)
```

---

autoSelectExchangeQuotes

*Retain only data from the stock exchange with the highest volume*

---

## Description

Function returns an xts object containing only observations of the exchange with highest value for the sum of "BIDSIZ" and "OFRSIZ", i.e. the highest quote volume.

## Usage

```
autoSelectExchangeQuotes(qData)
```

## Arguments

- qData a data.table or xts object with at least a column "EX", indicating the exchange symbol and columns "BIDSIZ" and "OFRSIZ", indicating the volume available at the bid and ask respectively. The chosen exchange is printed on the console. The possible exchanges are:
- A: AMEX
  - N: NYSE
  - B: Boston
  - P: Arca
  - C: NSX
  - T/Q: NASDAQ
  - D: NASD ADF and TRF
  - X: Philadelphia
  - I: ISE
  - M: Chicago
  - W: CBOE
  - Z: BATS

## Value

data.table or xts object depending on input

## Author(s)

Jonathan Cornelissen, Kris Boudt and Onno Kleen

## Examples

```
autoSelectExchangeQuotes(sampleQDataRawMicroseconds)
```

---

```
autoSelectExchangeTrades
```

*Retain only data from the stock exchange with the highest trading volume*

---

## Description

Function returns a data.table or xts object containing only observations of the exchange with the highest value for the variable "SIZE", i.e. the highest trade volume.

## Usage

```
autoSelectExchangeTrades(tData)
```

**Arguments**

tData	<p>an xts object with at least a column "EX", indicating the exchange symbol and "SIZE", indicating the trade volume. The chosen exchange is printed on the console.</p> <ul style="list-style-type: none"> <li>• A: AMEX</li> <li>• N: NYSE</li> <li>• B: Boston</li> <li>• P: Arca</li> <li>• C: NSX</li> <li>• T/Q: NASDAQ</li> <li>• D: NASD ADF and TRF</li> <li>• X: Philadelphia</li> <li>• I: ISE</li> <li>• M: Chicago</li> <li>• W: CBOE</li> <li>• Z: BATS</li> </ul>
-------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Value**

data.table or xts object depending on input

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen

**Examples**

```
autoSelectExchangeTrades(sampleTDataRawMicroseconds)
```

---

BNSjumpTest

*Barndorff-Nielsen and Shephard (2006) tests for the presence of jumps in the price series.*

---

**Description**

This test examines the presence of jumps in highfrequency price series. It is based on theory of Barndorff- Nielsen and Shephard (BNS). The null hypothesis is no jumps. Depending on the choice of estimator (integrated variance (IVestimator), integrated quarticity (IQestimator)), mechanism (linear, ratio) and adjustment (logarithm), the function returns the result. Function returns three outcomes: 1.z-test value 2.critical value(with confidence level of 95%) and 3.pvalue of the test. Assume there is  $N$  equispaced returns in period  $t$ .

Assume the Realized variance (RV), IVestimator and IQestimator are based on  $N$  equispaced returns.

Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then the BNSjumpTest is given by:

$$\text{BNSjumpTest} = \frac{RV - IVestimator}{\sqrt{(\theta - 2) \frac{1}{N} IQestimator}}$$

in which, *IVestimator* can be: bipower variance (BV), minRV, medRV. *IQestimator* can be: tripower quarticity (TP), quadpower quarticity (QP), minRQ, medRQ.

$\theta$ : depends on IVestimator (Huang and Tauchen (2005)).

### Usage

```
BNSjumpTest(
  rData,
  IVestimator = "BV",
  IQestimator = "TP",
  type = "linear",
  logTransform = FALSE,
  max = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE,
  alpha = 0.975
)
```

### Arguments

rData	a zoo/xts object containing all returns in period t for one asset.
IVestimator	can be chosen among jump robust integrated variance estimators: BV, minRV, medRV and corrected threshold bipower variation (CTBV). If CTBV is chosen, an argument of <i>startV</i> , start point of auxiliary estimators in threshold estimation (Corsi et al. (2010) can be included. BV by default.
IQestimator	can be chosen among jump robust integrated quarticity estimators: TP, QP, minRQ and medRQ. TP by default.
type	a method of BNS testing: can be linear or ratio. Linear by default.
logTransform	boolean, should be TRUE when QVestimator and IVestimator are in logarithm form. FALSE by default.
max	boolean, should be TRUE when max adjustment in SE. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.
alpha	numeric of length one with the significance level to use for the jump test(s). Defaults to 0.975.

### Details

The theoretical framework underlying jump test is that the logarithmic price process  $X_t$  belongs to the class of Brownian semimartingales, which can be written as:

$$X_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

where  $a$  is the drift term,  $\sigma$  denotes the spot volatility process,  $W$  is a standard Brownian motion and  $Z$  is a jump process defined by:

$$Z_t = \sum_{j=1}^{N_t} k_j$$

where  $k_j$  are nonzero random variables. The counting process can be either finite or infinite for finite or infinite activity jumps.

Since the realized volatility converges to the sum of integrated variance and jump variation, while the robust IVestimator converges to the integrated variance, it follows that the difference between  $\#^* RV_{t,N}$  and the IVestimator captures the jump part only, and this observation underlines the BNS test for jumps. (Theodosiou& Zikes(2009))

### Value

list or xts in case the input prices span more than one day.

### Author(s)

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

### References

- Barndorff-Nielsen, O. E., & Shephard, N. (2006). Econometrics of testing for jumps in financial economics using bipower variation. *Journal of financial Econometrics*, 4(1), 1-30.
- Corsi, F., Pirino, D., & Reno, R. (2010). Threshold bipower variation and the impact of jumps on volatility forecasting. *Journal of Econometrics*, 159(2), 276-288.
- Huang, X., & Tauchen, G. (2005). The relative contribution of jumps to total price variance. *Journal of financial econometrics*, 3(4), 456-499.
- Theodosiou, M., & Zikes, F. (2009). A comprehensive comparison of alternative tests for jumps in asset prices. Unpublished manuscript, Graduate School of Business, Imperial College London.

### Examples

```
BNSjumpTest(sampleTData$PRICE, IVestimator= "minRV",
             IQestimator = "medRQ", type= "linear", makeReturns = TRUE)
```

---

`businessTimeAggregation`*Business time aggregation*

---

### Description

Time series aggregation based on ‘business time’ statistics. Instead of equidistant sampling based on time during a trading day, business time sampling creates measures and samples equidistantly using these instead. For example when sampling based on volume, business time aggregation will result in a time series that has an equal amount of volume between each observation (if possible).

### Usage

```
businessTimeAggregation(  
  pData,  
  measure = "volume",  
  obs = 390,  
  bandwidth = 0.075,  
  ...  
)
```

### Arguments

<code>pData</code>	xts or data.table containing data to aggregate.
<code>measure</code>	character denoting which measure to use. Valid options are "intensity", "vol", and "volume", denoting the trade intensity process of Oomen (2005), volatility, and volume, respectively. Default is "volume"
<code>obs</code>	integer valued numeric of length 1 denoting how many observations is wanted after the aggregation procedure.
<code>bandwidth</code>	numeric of length one, denoting which bandwidth parameter to use in the trade intensity process estimation of Oomen (2005.)
<code>...</code>	extra arguments passed on to <a href="#">spotVol</a> when measure is "vol"

### Value

A list containing "pData" which is the aggregated data and a list containing the intensity process, split up day by day.

### Author(s)

Emil Sjoerup



## References

Roel C. A. Oomen Properties of realized variance under alternative sampling schemes. (2006) Journal of Business & Economic Statistics 24, pages 219-237

Yingjie Dong and Yiu Kuen Tse. (2017) Business time sampling scheme with applications to testing semi-martingale hypothesis and estimating integrated volatility. Econometrics, 5

## Examples

```
pData <- sampleTData[,c("PRICE", "SIZE")]
storage.mode(pData) <- "numeric"
# Aggregate based on the trade intensity measure. Getting 390 observations.
agged <- businessTimeAggregation(pData, measure = "intensity", obs = 390, bandwidth = 0.075)
# Plot the trade intensity measure
plot.ts(agged$intensityProcess$`2008-01-04`)
rCov(agged$pData[, "PRICE"], makeReturns = TRUE)
rCov(pData[, "PRICE"], makeReturns = TRUE, alignBy = "minutes", alignPeriod = 1)

# Aggregate based on the volume measure. Getting 78 observations.
agged <- businessTimeAggregation(pData, measure = "volume", obs = 78)
rCov(agged$pData[, "PRICE"], makeReturns = TRUE)
rCov(pData[, "PRICE"], makeReturns = TRUE, alignBy = "minutes", alignPeriod = 5)
```

---

exchangeHoursOnly      *Extract data from an xts object for the Exchange Hours Only*

---

## Description

The function returns data within exchange trading hours "dayBegin" and "dayEnd". By default, dayBegin and dayEnd are set to "09:30:00" and "16:00:00" respectively (see Brownlees and Gallo (2006) for more information on good choices for these arguments).

## Usage

```
exchangeHoursOnly(data, dayBegin = "09:30:00", dayEnd = "16:00:00")
```

## Arguments

data	a data.table or xts object containing the time series data. Multiple days of input are allowed.
dayBegin	character in the format of " <code>\\HH:MM:SS\\</code> ", specifying the starting hour, minute and second of an exchange trading day.
dayEnd	character in the format of " <code>\\HH:MM:SS\\^</code> ", specifying the closing hour, minute and second of an exchange trading day.

## Value

xts or data.table object depending on input

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen.

**References**

Brownlees, C.T. and Gallo, G.M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51, pages 2232-2245.

**Examples**

```
exchangeHoursOnly(sampleTDataRawMicroseconds)
```

---

getAlphaVantageData    *Get high frequency data from Alpha Vantage*

---

**Description**

Function to retrieve high frequency data from Alpha Vantage - wrapper around quantMods' get-Symbols.av function

**Usage**

```
getAlphaVantageData(
  symbols = NULL,
  interval = "5min",
  outputType = "xts",
  apiKey = NULL,
  doSleep = TRUE
)
```

**Arguments**

symbols	character vector with the symbols to import.
interval	the sampling interval of the data retrieved. Should be one of one of "1min", "5min", "15min", "30min", or "60min"
outputType	string either "xts" or "DT" to denote the type of output wanted. "xts" will yield an xts object, "DT" will yield a data.table object.
apiKey	string with the api key provided by Alpha Vantage.
doSleep	logical when the length of symbols > 5 the function will sleep for 12 seconds by default.

**Details**

#' The doSleep argument is set to true as default because Alpha Vantage has a limit of five calls per minute. The function does not try to extract when the last API call was made which means that if you made successive calls to get 3 symbols in rapid succession, the function may not retrieve all the data.

**Value**

An object of type xts or data.table in case the length of symbols is 1. If the length of symbols > 1 the xts and data.table objects will be put into a list.

**Author(s)**

Emil Sjoerup (wrapper only) Paul Teetor (for quantMod's getSymbols.av)

**See Also**

The getSymbols.av function in the quantMod package

**Examples**

```
## Not run:
# Get data for SPY at an interval of 1 minute in the standard xts format.
data <- getAlphaVantageData(symbols = "SPY", apiKey = "yourKey", interval = "1min")

# Get data for 3M and Goldman Sachs at a 5 minute interval in the data.table format.
# The data.tables will be put in a list.
data <- getAlphaVantageData(symbols = c("MMM", "GS"), interval = "5min",
                           outputType = "DT", apiKey = 'yourKey')

# Get data for JPM and Citicorp at a 15 minute interval in the xts format.
# The xts objects will be put in a list.
data <- getAlphaVantageData(symbols = c("JPM", "C"), interval = "15min",
                           outputType = "xts", apiKey = 'yourKey')

## End(Not run)
```

---

getLiquidityMeasures *Compute Liquidity Measure Function returns an xts or data.table object containing 23 liquidity measures. Please see details below. Note that this assumes a regular time grid. The Lee + Ready measure uses two lags for the Tick Rule.*

---

**Description**

Compute Liquidity Measure

Function returns an xts or data.table object containing 23 liquidity measures. Please see details below.

Note that this assumes a regular time grid. The Lee + Ready measure uses two lags for the Tick Rule.

**Usage**

```
getLiquidityMeasures(tqData, win = 300, type = NULL)
```

**Arguments**

tqData	A data.table or xts object as in the <b>highfrequency</b> merged trades and quotes data (that is included).
win	A windows length for the forward-prices used for ‘realized’ spread
type	Legacy option. Default is to return all liquidity measures.

**Details**

NOTE: xts or data.table should only contain one day of observations

The respective liquidity measures are defined as follows:

- effectiveSpread

$$\text{effective spread}_t = 2 * D_t * \left( \text{PRICE}_t - \frac{(\text{BID}_t + \text{OFR}_t)}{2} \right),$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)). Note that the input of this function consists of the matched trades and quotes, so this is were the time indication refers to (and thus not to the registered quote timestamp).

- realizedSpread: realized spread

$$\text{realized spread}_t = 2 * D_t * \left( \text{PRICE}_t - \frac{(\text{BID}_{t+300} + \text{OFR}_{t+300})}{2} \right),$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)). Note that the time indication of BID and OFR refers to the registered time of the quote in seconds.

- valueTrade: trade value

$$\text{trade value}_t = \text{SIZE}_t * \text{PRICE}_t.$$

- signedValueTrad: signed trade value

$$\text{signed trade value}_t = D_t * (\text{SIZE}_t * \text{PRICE}_t),$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)).

- depthImbalanceDifference: depth imbalance (as a difference)

$$\text{depth imbalance (as difference)}_t = \frac{D_t * (\text{OFRSIZ}_t - \text{BIDSIZ}_t)}{(\text{OFRSIZ}_t + \text{BIDSIZ}_t)},$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)). Note that the input of this function consists of the matched trades and quotes, so this is were the time indication refers to (and thus not to the registered quote timestamp).

- depthImbalanceRatio: depth imbalance (as ratio)

$$\text{depth imbalance (as ratio)}_t = \left( \frac{\text{OFRSIZ}_t}{\text{BIDSIZ}_t} \right)^{D_t},$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)). Note that the input of this function consists of the matched trades and quotes, so this is were the time indication refers to (and thus not to the registered quote timestamp).

- proportionalEffectiveSpread: proportional effective spread

$$\text{proportional effective spread}_t = \frac{\text{effective spread}_t}{(\text{OFR}_t + \text{BID}_t)/2}$$

(Venkataraman, 2001).

Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- proportionalRealizedSpread: proportional realized spread

$$\text{proportional realized spread}_t = \frac{\text{realized spread}_t}{(\text{OFR}_t + \text{BID}_t)/2}$$

(Venkataraman, 2001).

Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered

- priceImpact: price impact

$$\text{price impact}_t = \frac{\text{effective spread}_t - \text{realized spread}_t}{2}$$

(see Boehmer (2005), Bessembinder (2003)).

- proportionalPriceImpact: proportional price impact

$$\text{proportional price impact}_t = \frac{(\text{effective spread}_t - \text{realized spread}_t) / 2}{\frac{\text{OFR}_t + \text{BID}_t}{2}}$$

(Venkataraman, 2001). Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- halfTradedSpread: half traded spread

$$\text{half traded spread}_t = D_t * (\text{PRICE}_t - \frac{(\text{BID}_t + \text{OFR}_t)}{2}),$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell) (see Boehmer (2005), Bessembinder (2003)). Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- proportionalHalfTradedSpread: proportional half traded spread

$$\text{proportional half traded spread}_t = \frac{\text{half traded spread}_t}{\frac{\text{OFR}_t + \text{BID}_t}{2}}.$$

Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- squaredLogReturn: squared log return on trade prices

$$\text{squared log return on Trade prices}_t = (\log(\text{PRICE}_t) - \log(\text{PRICE}_{t-1}))^2.$$

- absLogReturn: absolute log return on trade prices

$$\text{absolute log return on Trade prices}_t = |\log(\text{PRICE}_t) - \log(\text{PRICE}_{t-1})|.$$

- quotedSpread: quoted spread

$$\text{quoted spread}_t = \text{OFR}_t - \text{BID}_t$$

Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- proportionalQuotedSpread: proportional quoted spread

$$\text{proportional quoted spread}_t = \frac{\text{quoted spread}_t}{\frac{\text{OFR}_t + \text{BID}_t}{2}}$$

(Venkataraman, 2001). Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- logQuotedSpread: log quoted spread

$$\text{log quoted spread}_t = \log\left(\frac{\text{OFR}_t}{\text{BID}_t}\right)$$

(Hasbrouck and Seppi, 2001). Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- logQuotedSize: log quoted size

$$\text{log quoted size}_t = \log(\text{OFRSIZ}_t) - \log(\text{BIDSIZ}_t)$$

(Hasbrouck and Seppi, 2001). Note that the input of this function consists of the matched trades and quotes, so this is where the time indication refers to (and thus not to the registered quote timestamp).

- quotedSlope: quoted slope

$$\text{quoted slope}_t = \frac{\text{quoted spread}_t}{\text{log quoted size}_t}$$

(Hasbrouck and Seppi, 2001).

- logQSlope: log quoted slope

$$\text{log quoted slope}_t = \frac{\log \text{quoted spread}_t}{\text{log quoted size}_t}.$$

- midQuoteSquaredReturn: midquote squared return

$$\text{midquote squared return}_t = (\log(\text{midquote}_t) - \log(\text{midquote}_{t-1}))^2,$$

where  $\text{midquote}_t = \frac{\text{BID}_t + \text{OFR}_t}{2}$ .

- midQuoteAbsReturn: midquote absolute return

$$\text{midquote absolute return}_t = |\log(\text{midquote}_t) - \log(\text{midquote}_{t-1})|,$$

$$\text{where midquote}_t = \frac{\text{BID}_t + \text{OFR}_t}{2}.$$

- signedTradeSize: signed trade size

$$\text{signed trade size}_t = D_t * \text{SIZE}_t,$$

where  $D_t$  is 1 (-1) if  $trade_t$  was buy (sell).

## Value

A modified (enlarged) xts or data.table with the new measures.

## References

Bessembinder, H. (2003). Issues in assessing trade execution costs. *Journal of Financial Markets*, 223-257. Boehmer, E. (2005). Dimensions of execution quality: Recent evidence for US equity markets. *Journal of Financial Economics* 78 (3), 553-582. Hasbrouck, J. and D. J. Seppi (2001). Common factors in prices, order flows and liquidity. *Journal of Financial Economics*, 383-411. Venkataraman, K. (2001). Automated versus floor trading: An analysis of execution costs on the paris and new york exchanges. *The Journal of Finance*, 56, 1445-1485.

## Examples

```
tqData <- matchTradesQuotes(sampleTData, sampleQData)
res <- getLiquidityMeasures(tqData)
head(res)
```

---

getPrice

*Get price column(s) from a timeseries*

---

## Description

Will attempt to locate price column(s) from a time series with rational defaults.

## Usage

```
getPrice(x, symbol = NULL, prefer = NULL)
```

## Arguments

x	A data object with columns containing data to be extracted
symbol	text string containing the symbol to extract
prefer	preference for any particular type of price, see Details

## Details

May be subset by symbol and preference. prefer Preference will be for any commonly used financial time series price description, e.g. 'trade', 'close', 'bid', 'ask' with specific tests and matching for types and column names currently supported in R, but a default grep match will be performed if one of the supported types doesn't match.

The functionality was taken from the quantmod-package

---

getTradeDirection      *Get trade direction*

---

## Description

Function returns a vector with the inferred trade direction which is determined using the Lee and Ready algorithm (Lee and Ready, 1991).

## Usage

```
getTradeDirection(tqData)
```

## Arguments

tqData                  data.table or xts object, containing joined trades and quotes (e.g. using [matchTradesQuotes](#))

## Details

NOTE: The value of the first (and second) observation of the output should be ignored if price == midpoint for the first (second) observation.

## Value

A vector which has values 1 or (-1) if the inferred trade direction is buy or sell respectively.

## Author(s)

Jonathan Cornelissen, Kris Boudt and Onno Kleen. Special thanks to Dirk Eddelbuettel.

## References

Lee, C. M. C. and M. J. Ready (1991). Inferring trade direction from intraday data. Journal of Finance 46, 733-746.

## Examples

```
# generate matched trades and quote data set
tqData <- matchTradesQuotes(sampleTData, sampleQData)
directions <- getTradeDirection(tqData)
head(directions)
```



---

HARmodel	<i>HAR model estimation (Heterogeneous Autoregressive model for Realized volatility)</i>
----------	------------------------------------------------------------------------------------------

---

### Description

Function returns the estimates for the Heterogeneous Autoregressive model for Realized volatility discussed in Andersen et al. (2007) and Corsi (2009). This model is mainly used to forecast the next day's volatility based on the high-frequency returns of the past. Consult the vignette for more information.

### Usage

```
HARmodel(
  data,
  periods = c(1, 5, 22),
  periodsJ = c(1, 5, 22),
  periodsQ = c(1),
  leverage = NULL,
  RVest = c("rCov", "rBPCov", "rQuar"),
  type = "HARRV",
  inputType = "RM",
  jumpTest = "ABDJumptest",
  alpha = 0.05,
  h = 1,
  transform = NULL,
  ...
)
```

### Arguments

data	an xts object containing either: intra-day (log-)returns or realized measures already computed from such returns. In case more than one realized measure is needed, the object should have the as many columns as realized measures needed. The first column should always be the realized variance proxy. In case type is either "HARRVQJ" or "CHARRVQ" the order should be "RV", "BPV", "RQ", or the relevant proxies.
periods	a vector of integers indicating over how days the realized measures in the model should be aggregated. By default periods = c(1,5,22), which corresponds to one day, one week and one month respectively. This default is in line with Andersen et al. (2007).
periodsJ	a vector of integers indicating over what time periods the jump components in the model should be aggregated. By default periodsJ = c(1,5,22), which corresponds to one day, one week and one month respectively.
periodsQ	a vector of integers indicating over what time periods the realized quarticity in the model should be aggregated. By default periodsQ = c(1,5,22), which corresponds to one day, one week and one month respectively.

leverage	a vector of integers indicating over what periods the negative returns should be aggregated. See Corsi and Reno (2012) for more information. By default leverage = NULL and the model assumes the absence of a leverage effect. Set leverage = c(1,5,22) to mimic the analysis in Corsi and Reno (2012).
RVest	a character vector with one, two, or three elements. The first element always refers to the name of the function to estimate the daily integrated variance (non-jump-robust). The second and third element depends on which type of model is estimated: If type = "HARRVJ", type = "HARRVCJ", type = "HARRVQJ" the second element refers to the name of the function to estimate the continuous component of daily volatility (jump robust). If type = "HARRVQ", the second element refers to the name of the function used to estimate the integrated quarticity. If type = "HARRVQJ" the third element always refers to the name of the function used to estimate integrated quarticity. By default RVest = c("rCov","rBPCov","rQuar"), i.e. using the Realized Volatility, Realized Bi-Power Variance, and Realized Quarticity.
type	a string referring to the type of HAR model you would like to estimate. By default type = "HARRV", the most basic model. Other valid options are type = "HARRVJ", type = "HARRVCJ", type = "HARRVQ", type = "HARRVQJ", type = "CHARRV", or type = "CHARRVQ".
inputType	a string denoting if the input data consists of realized measures or high-frequency returns, default "RM" is the only way to denote realized measures and everything else denotes returns.
jumpTest	the function name of a function used to test whether the test statistic which determines whether the jump variability is significant that day. By default jumpTest = "ABDJumptest", hence using the test statistic in Equation or Equation (18) of Andersen et al. (2007).
alpha	a real indicating the confidence level used in testing for jumps. By default alpha = 0.05.
h	an integer indicating the number over how many days the dependent variable should be aggregated. By default, h=1, i.e. no aggregation takes place, you just model the daily realized volatility.
transform	optionally a string referring to a function that transforms both the dependent and explanatory variables in the model. By default transform=NULL, so no transformation is done. Typical other choices in this context would be "log" or "sqrt".
...	extra arguments for jump test.

**Value**

The function outputs an object of class HARmodel and `lm` (so HARmodel is a subclass of `lm`).

**Author(s)**

Jonathan Cornelissen and Kris Boudt

## References

Andersen, T. G., T. Bollerslev, and F. Diebold (2007). Roughing it up: including jump components in the measurement, modelling and forecasting of return volatility. *The Review of Economics and Statistics* 89, 701-720. Corsi, F. (2009). A simple approximate long memory model of realized volatility. *Journal of Financial Econometrics* 7, 174-196. Corsi, F. and Reno R. (2012). Discrete-time volatility forecasting with persistent leverage effect and the link with continuous-time volatility modeling. *Journal of Business and Economic Statistics*, forthcoming. Bollerslev, T., Patton, A., Quaedvlieg, R. 2016, Exploiting the errors: A simple approach for improved volatility forecasting, *Journal of Econometrics*, vol.192, issue 1, 1-18.

## Examples

```
##### Example 1: HARRVCJ #####
dat <- sample5MinPricesJumps$stock1
dat <- makeReturns(dat) #Get the high-frequency return data

x <- HARmodel(dat, periods = c(1,5,10), periodsJ = c(1,5,10),
              RVest = c("rCov","rBPCov"),
              type = "HARRVCJ",transform = "sqrt", inputType = "returns")
# Estimate the HAR model of type HARRVCJ
class(x)
x
# plot(x)
predict(x)

##### Example 2: HARRV #####
# Forecasting daily Realized volatility for the S&P 500 using the basic HARmodel: HARRV
library(xts)
RV_SP500 <- as.xts(realizedLibrary$rv5, order.by = realizedLibrary$date)

x <- HARmodel(data = RV_SP500 , periods = c(1,5,22), RVest = c("rCov"),
              type = "HARRV", h = 1, transform = NULL, inputType = "RM")
class(x)
x
summary(x)
plot(x)
predict(x)

##### Example 3: HARRVQ #####
dat <- sample5MinPricesJumps$stock1
dat <- makeReturns(dat) #Get the high-frequency return data
#
x <- HARmodel(dat, periods = c(1,5,10), periodsJ = c(1,5,10),
              periodsQ = c(1), RVest = c("rCov", "rQuar"),
              type="HARRVQ", inputType = "returns")
## Estimate the HAR model of type HARRVQ
class(x)
x
# plot(x)
```

```

#predict(x)

##### Example 4: HARRVQJ with already computed realized measures #####
dat <- SP500RM[, c("RV", "BPV", "RQ")]
x <- HARmodel(dat, periods = c(1,5,22), periodsJ = c(1),
              periodsQ = c(1), type = "HARRVQJ")
## Estimate the HAR model of type HARRVQJ
class(x)
x
# plot(x)
predict(x)

##### Example 5: CHARRV with already computed realized measures #####
dat <- SP500RM[, c("RV", "BPV")]

x <- HARmodel(dat, periods = c(1, 5, 22), type = "CHARRV")
# Estimate the HAR model of type CHARRV
class(x)
x
# plot(x)
predict(x)

##### Example 6: CHARRVQ with already computed realized measures #####
dat <- SP500RM[, c("RV", "BPV", "RQ")]

x <- HARmodel(dat, periods = c(1,5,22), periodsQ = c(1), type = "CHARRVQ")
# Estimate the HAR model of type CHARRVQ
class(x)
x
# plot(x)
predict(x)

#' ##### Example 7: HARRV #####
# Forecasting weekly Realized volatility for the S&P 500 using the basic HARmodel: HARRV
library(xts)
RV_SP500 <- as.xts(realizedLibrary$rv5, order.by = realizedLibrary$date)

x <- HARmodel(data = RV_SP500 , periods = c(1,5,22), RVest = c("rCov"),
              type = "HARRV", h = 5, transform = NULL, inputType = "RM")

class(x)
x
summary(x)
plot(x)
predict(x)

```

**Description**

A set of functions to check for appropriate TBBO/BBO and price column names within a data object, as well as the availability and position of those columns.

**Usage**

```
hasQty(x, which = FALSE)
```

**Arguments**

x	data object
which	display position of match

---

HEAVYmodel

*HEAVY Model estimation*


---

**Description**

This function calculates the High frEQUENCY bAsed VolatilitY (HEAVY) model proposed in Shephard and Sheppard (2010). This function is used as a predictive volatility model built to exploit highfrequency data.

**Usage**

```
HEAVYmodel(
  data,
  p = matrix(c(0, 0, 1, 1), ncol = 2),
  q = matrix(c(1, 0, 0, 1), ncol = 2),
  startingValues = NULL,
  LB = NULL,
  UB = NULL,
  backCast = NULL,
  compConst = FALSE
)
```

**Arguments**

data	a (T x K) matrix containing the data, with T the number of days. For the traditional HEAVY model: K = 2, the first column contains the squared daily de-meaned returns, the second column contains the realized measures.
p	a (K x K) matrix containing the lag length for the model innovations. Position (i, j) in the matrix indicates the number of lags in equation i of the model for the innovations in data column j. For the traditional heavy model p is given by matrix(c(0,0,1,1), ncol = 2) (default).

q	a (K x K) matrix containing the lag length for the conditional variances. Position (i, j) in the matrix indicates the number of lags in equation i of the model for conditional variances corresponding to series j. For the traditional heavy model introduced above q is given by matrix( c(1,0,0,1),ncol=2 ) (default).
startingValues	a vector containing the starting values to be used in the optimization to find the optimal parameters estimates.
LB	a vector of length K indicating the lower bounds to be used in the estimation. If NULL it is set to a vector of zeros by default.
UB	a vector of length K indicating the upper bounds to be used in the estimation. If NULL it is set to a vector of Inf by default.
backCast	a vector of length K used to initialize the estimation. If NULL the unconditional estimates are taken.
compConst	a boolean variable. In case TRUE, the omega values are estimated in the optimization. In case FALSE, volatility targeting is done and omega is just 1 minus the sum of all relevant alpha's and beta's multiplied by the unconditional variance.

### Details

Assume there are  $T$  daily returns and realized measures in the period  $t$ . Let  $r_i$  and  $RM_i$  be the  $i^{th}$  daily return and daily realized measure respectively (with  $i = 1, \dots, T$ ).

The most basic heavy model is the one with lag matrices p of  $\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$  and q of  $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ . This can be represented by the following equations:

$$\text{var}(r_t) = h_t = w + \alpha RM_{t-1} + \beta h_{t-1}; w, \alpha \geq 0, \beta \in [0, 1]$$

$$E(RM_t) = \mu_t = w_R + \alpha_R RM_{t-1} + \beta_R \mu_{t-1}; w_R, \alpha_R, \beta_R \geq 0, \alpha_R + \beta_R \in [0, 1]$$

Equivalently, they can be presented in terms of matrix notation as below:

$$\begin{pmatrix} h_t \\ \mu_t \end{pmatrix} = \begin{pmatrix} w \\ w_R \end{pmatrix} + \begin{pmatrix} 0 & \alpha \\ 0 & \alpha_R \end{pmatrix} \begin{pmatrix} r_{t-1}^2 \\ RM_{t-1} \end{pmatrix} + \begin{pmatrix} \beta & 0 \\ 0 & \beta_R \end{pmatrix} \begin{pmatrix} h_{t-1} \\ \mu_{t-1} \end{pmatrix}$$

In this version, the parameters vector to be estimated is  $(w, w_R, \alpha, \alpha_R, \beta, \beta_R)$ .

In terms of startingValues, Shephard and Sheppard recommend for this version of the Heavy model to set  $\beta$  be around 0.6 and sum of  $\alpha + \beta$  to be close to but slightly less than one. In general, the lag length for the model innovation and the conditional covariance can be greater than 1. Consider, for example, matrix p is  $\begin{pmatrix} 0 & 2 \\ 0 & 1 \end{pmatrix}$  and matrix q is the same as above. Matrix notation will be as below:

$$\begin{pmatrix} h_t \\ \mu_t \end{pmatrix} = \begin{pmatrix} w \\ w_R \end{pmatrix} + \begin{pmatrix} 0 & \alpha_1 \\ 0 & \alpha_R \end{pmatrix} \begin{pmatrix} r_{t-1}^2 \\ RM_{t-1} \end{pmatrix} + \begin{pmatrix} 0 & \alpha_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} r_{t-2}^2 \\ RM_{t-2} \end{pmatrix} + \begin{pmatrix} \beta & 0 \\ 0 & \beta_R \end{pmatrix} \begin{pmatrix} h_{t-1} \\ \mu_{t-1} \end{pmatrix}$$

In this version, the parameters vector to be estimated is  $(w, w_R, \alpha_1, \alpha_R, \alpha_2, \beta, \beta_R)$ .

**Value**

A list with the following values: (i) loglikelihood: the log likelihood evaluated at the parameter estimates. (ii) likelihoods: an xts object of length T containing the log likelihoods per day. (iii) condvar: a (T x K) xts object containing the conditional variances (iv) estparams: a vector with the parameter estimates. The order in which the parameters are reported is as follows: First the estimates for omega then the estimates for the non-zero alpha's with the most recent lags first in case  $\max(p) > 1$ , then the estimates for the non-zero beta's with the most recent lag first in case  $\max(q) > 1$ . (v) convergence: an integer code indicating the successfulness of the optimization. See `optim` for more information.

**Author(s)**

Giang Nguyen, Jonathan Cornelissen, Kris Boudt and Onno Kleen.

**References**

Shephard, N. and K. Shephard (2010). Realising the future: forecasting with high frequency based volatility (heavy) models. *Journal of Applied Econometrics* 25, 197-231.

**Examples**

```
# Implementation of the heavy model on DJI:
returns <- realizedLibrary$open_to_close
bv      <- realizedLibrary$bv
returns <- returns[!is.na(bv)]
bv <- bv[!is.na(bv)] # Remove NA's
data <- cbind( returns^2, bv) # Make data matrix with returns and realized measures
backCast <- matrix(c(var(returns), mean(bv)), ncol = 1)

#For traditional (default) version:
startvalues <- c(0.004,0.02,0.44,0.41,0.74,0.56) # Initial values
output <- HEAVYmodel(data = as.matrix(data,ncol=2), compConst=FALSE,
                    startingValues = startvalues, backCast=backCast)

#For general version:
startvalues <- c(0.004, 0.02, 0.44, 0.4, 0.41, 0.74, 0.56) # Initial values;
p <- matrix(c(2, 0, 0, 1), ncol = 2)
q <- matrix(c(1, 0, 0, 1), ncol = 2)

heavy_model <- HEAVYmodel(data = as.matrix(data, ncol = 2), p = p, q = q, compConst = FALSE,
                        startingValues = startvalues, backCast = backCast)
```

---

intradayJumpTest

*Intraday jump tests*


---

**Description**

This function can be used to test for jumps in intraday price paths. The tests are of the form  $L(t) = (R(t) - mu(t))/sigma(t)$ .

**Usage**

```
intradayJumpTest(
  pData,
  volEstimator = "RM",
  driftEstimator = "none",
  alpha = 0.95,
  ...,
  on = "minutes",
  k = 5,
  marketOpen = "09:30:00",
  marketClose = "16:00:00",
  tz = "GMT"
)
```

**Arguments**

<code>pData</code>	xts or data.table of the price data in levels. This data can (and should in some cases) be tick-level data. The data can span more than one day.
<code>volEstimator</code>	character denoting which volatility estimator to use for the tests. See <a href="#">spotVol</a> . Default = "RM" denoting realized measures.
<code>driftEstimator</code>	character denoting which drift estimator to use for the tests. See <a href="#">spotDrift</a> . Default = "none" denoting no drift estimation.
<code>alpha</code>	numeric of length one determining what confidence level to use when constructing the critical values.
<code>...</code>	extra arguments passed on to <a href="#">spotVol</a> for the volatility estimation, and to <a href="#">spotDrift</a> .
<code>on</code>	string indicating the time scale in which k is expressed. Possible values are: "secs", "seconds", "mins", "minutes", "hours".
<code>k</code>	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate an xts object to the 5 minute frequency, set k = 5 and on = "minutes".
<code>marketOpen</code>	the market opening time. This should be in the time zone specified by tz. By default, marketOpen = "09:30:00".
<code>marketClose</code>	the market closing time. This should be in the time zone specified by tz. By default, marketClose = "16:00:00".
<code>tz</code>	string specifying the time zone to which the times in data and/or marketOpen/marketClose belong. Default = "GMT".

The null hypothesis of the tests in this function is that there are no jumps in the price series

**Examples**

```
## Not run:
# We can easily make a Lee-Mykland jump test.
LMtest <- intradayJumpTest(pData = sampleTDataMicroseconds[, list(DT, PRICE)],
  volEstimator = "RM", driftEstimator = "none",
  RM = "bipower", lookBackPeriod = 20,
  on = "minutes", k = 5, marketOpen = "09:30:00",
```



```

marketClose = "16:00:00")
plot(LMtest)

# We can just as easily use the pre-averaged version from the "Fact or Friction" paper
FoFtest <- intradayJumpTest(pData = sampleTDataMicroseconds[, list(DT, PRICE)],
                           volEstimator = "PARM", driftEstimator = "none",
                           RM = "bipower", lookBackPeriod = 20, theta = 1.2,
                           marketOpen = "09:30:00", marketClose = "16:00:00")
plot(FoFtest)

## End(Not run)

```

---

ivInference	<i>Function returns the value, the standard error and the confidence band of the integrated variance (IV) estimator.</i>
-------------	--------------------------------------------------------------------------------------------------------------------------

---

### Description

This function supplies information about standard error and confidence band of integrated variance (IV) estimators under Brownian semimartingales model such as: bipower variation, minRV, medRV. Depending on users' choices of estimator (integrated variance (IVestimator), integrated quarticity (IQestimator)) and confidence level, the function returns the result.(Barndorff (2002)) Function returns three outcomes: 1.value of IV estimator 2.standard error of IV estimator and 3.confidence band of IV estimator.

Assume there is  $N$  equispaced returns in period  $t$ .

Then the ivInference is given by:

$$\text{standard error} = \frac{1}{\sqrt{N}} * sd$$

$$\text{confidence band} = \hat{IV} \pm cv * se$$

in which,

$$sd = \sqrt{\theta \times \hat{IQ}}$$

$cv$  : critical value.

$se$  : standard error.

$\theta$  : depending on IQestimator,  $\theta$  can take different value (Andersen et al. (2012)).

$\hat{IQ}$  integrated quarticity estimator.

**Usage**

```

ivInference(
  rData,
  IVestimator = "RV",
  IQestimator = "rQuar",
  confidence = 0.95,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE,
  ...
)

```

**Arguments**

rData	zoo/xts object containing all returns in period t for one asset.
IVestimator	can be chosen among integrated variance estimators: RV, BV, TV, minRV or medRV. RV by default.
IQestimator	can be chosen among integrated quarticity estimators: rQuar, realized tri-power quarticity (TPQ), quad-power quarticity (QPQ), minRQ or medRQ. TPQ by default.
confidence	confidence level set by users. 0.95 by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours"
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.
...	additional arguments.

**Details**

The theoretical framework is the logarithmic price process  $X_t$  belongs to the class of Brownian semimartingales, which can be written as:

$$X_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u$$

where  $a$  is the drift term,  $\sigma$  denotes the spot volatility process,  $W$  is a standard Brownian motion (assume that there are no jumps).

**Value**

list

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

## References

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

Barndorff-Nielsen, O. E. (2002). Econometric analysis of realized volatility and its use in estimating stochastic volatility models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(2), 253-280.

## Examples

```
ivInference(sampleTData$PRICE, IVestimator= "minRV", IQestimator = "medRQ",
            confidence = 0.95, makeReturns = TRUE)
```

---

JOjumpTest	<i>Jiang and Oomen (2008) tests for the presence of jumps in the price series.</i>
------------	------------------------------------------------------------------------------------

---

## Description

This test examines the jump in highfrequency data. It is based on theory of Jiang and Oomen (JO). They found that the difference of simple return and logarithmic return can capture one half of integrated variance if there is no jump in the underlying sample path. The null hypothesis is no jumps.

Function returns three outcomes: 1.z-test value 2.critical value under confidence level of 95% and 3.p-value.

Assume there is  $N$  equispaced returns in period  $t$ .

Let  $r_{t,i}$  be a logarithmic return (with  $i = 1, \dots, N$ ) in period  $t$ .

Let  $R_{t,i}$  be a simple return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then the JOjumpTest is given by:

$$\text{JOjumpTest}_{t,N} = \frac{NBV_t}{\sqrt{\Omega_{SwV}} \left(1 - \frac{RV_t}{SwV_t}\right)}$$

in which,  $BV$ : bipower variance;  $RV$ : realized variance (defined by Andersen et al. (2012));

$$\text{SwV}_t = 2 \sum_{i=1}^N (R_{t,i} - r_{t,i})$$

$$\Omega_{SwV} = \frac{\mu_6}{9} \frac{N^3 \mu_{6/p}^{-p}}{N-p-1} \sum_{i=0}^{N-p} \prod_{k=1}^p |r_{t,i+k}|^{6/p}$$

$$\mu_p = \mathbb{E}[|U|^p] = 2^{p/2} \frac{\Gamma(1/2(p+1))}{\Gamma(1/2)}$$

$U$ : independent standard normal random variables

$p$ : parameter (power).

**Usage**

```
JOjumpTest(
  pData,
  power = 4,
  alignBy = NULL,
  alignPeriod = NULL,
  alpha = 0.975,
  ...
)
```

**Arguments**

<code>pData</code>	a zoo/xts object containing all prices in period t for one asset.
<code>power</code>	can be chosen among 4 or 6. 4 by default.
<code>alignBy</code>	a string, align the tick data to "seconds" "minutes" "hours". Defaults to NULL, denoting testing by using tick by tick returns
<code>alignPeriod</code>	an integer, align the tick data to this many [seconds minutes hours]. Defaults to NULL, denoting testing by using tick by tick returns or if the returns are already aligned as
<code>alpha</code>	numeric of length one with the significance level to use for the jump test(s). Defaults to 0.975.
<code>...</code>	additional arguments. (Currently unused)

**Details**

The theoretical framework underlying jump test is that the logarithmic price process  $X_t$  belongs to the class of Brownian semimartingales, which can be written as:

$$X_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

where  $a$  is the drift term,  $\sigma$  denotes the spot volatility process,  $W$  is a standard Brownian motion and  $Z$  is a jump process defined by:

$$Z_t = \sum_{j=1}^{N_t} k_j$$

where  $k_j$  are nonzero random variables. The counting process can be either finite or infinite for finite or infinite activity jumps.

The Jiang and Oomen test is that: in the absence of jumps, the accumulated difference between the simple return and the log return captures one half of the integrated variance. (Theodosiou& Zikes(2009))

**Value**

list

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

Jiang, J.G. and Oomen R.C.A (2008). Testing for jumps when asset prices are observed with noise- a "swap variance" approach. *Journal of Econometrics*,144(2), 352-370.

Theodosiou, M., & Zikes, F. (2009). A comprehensive comparison of alternative tests for jumps in asset prices. Unpublished manuscript, Graduate School of Business, Imperial College London.

**Examples**

```
JOjumpTest(sample5MinPricesJumps$stock1, power = 6)
```

---

knChooseReMeDI	<i>ReMeDI tuning parameter function to choose the tuning parameter, kn in ReMeDI estimation</i>
----------------	-------------------------------------------------------------------------------------------------

---

**Description**

ReMeDI tuning parameter function to choose the tuning parameter, kn in ReMeDI estimation

**Usage**

```
knChooseReMeDI(  
  pData,  
  knEqual = FALSE,  
  knMax = 10,  
  tol = 0.05,  
  size = 3,  
  lower = 2,  
  upper = 5,  
  plot = FALSE  
)
```

**Arguments**

pData	xts or data.table containing the log-prices of the asset.
knEqual	Use an altered version of the ReMeDI estimator, where we instead use equal kn, instead of kn and 2*kn for the windows. See Figure 1 of paper in reference section.
knMax	max value of kn to be considered

tol	tolerance for the minimizing value. If tol is high, the algorithm will choose a lower optimal value.
size	size of the local window
lower	lower boundary for the method if it fails to find an optimal value. If this is the case, the best kn between lower and upper is returned
upper	upper boundary for the method if it fails to find an optimal value. If this is the case, the best kn between lower and upper is returned
plot	logical whether to plot the errors.

**Details**

This is the algorithm B.2 in the appendix of the Li and Linton (2019) working paper

**Value**

integer containing the optimal kn

**Author(s)**

Emil Sjoerup

**References**

A ReMeDI for Microstructure Noise

**Examples**

```

optimalKn <- knChooseReMeDI(sampleTDataMicroseconds[as.Date(DT) == "2018-01-02",],
                           knMax = 10, tol = 0.05, size = 3,
                           lower = 2, upper = 5, plot = TRUE)

optimalKn
## Not run:
# We can also have a much larger search-space
optimalKn <- knChooseReMeDI(sampleTDataMicroseconds[, as.Date(DT) == "2018-01-02"],
                           knMax = 50, tol = 0.05,
                           size = 3, lower = 2, upper = 5, plot = TRUE)

optimalKn

## End(Not run)

```

---

leadLag	<i>Lead-Lag estimation</i>
---------	----------------------------

---

### Description

Function that estimates whether one series leads (or lags) another.

### Usage

```
leadLag(
  price1 = NULL,
  price2 = NULL,
  lags = NULL,
  resolution = "seconds",
  normalize = TRUE,
  parallelize = FALSE,
  nCores = NA
)
```

### Arguments

price1	xts or data.table containing prices in levels, in case of data.table, use a column DT to denote the date-time in POSIXct format, and a column PRICE to denote the price
price2	xts or data.table containing prices in levels, in case of data.table, use a column DT to denote the date-time in POSIXct format, and a column PRICE to denote the price
lags	a numeric denoting which lags (in units of resolution) should be tested as leading or lagging
resolution	the resolution at which the lags is measured. The default is "seconds", use this argument to gain 1000 times resolution by setting it to either "ms", "milliseconds", or "milli-seconds".
normalize	logical denoting whether the contrasts should be normalized by the product of the L2 norms of both the prices. Default = TRUE. This does not change the value of the lead-lag-ratio.
parallelize	logical denoting whether to use a parallelized version of the C++ code (parallelized using OPENMP). Default = FALSE
nCores	integer valued numeric denoting how many cores to use for the lead-lag estimation procedure in case parallelize is TRUE. Default is NA, which does not parallelize the code.

### Details

The lead-lag-ratio (LLR) can be used to see if one asset leads the other. If  $LLR < 1$ , then price1 MAY be leading price2 and vice versa if  $LLR > 1$ .

**Value**

An list with class "leadLag" which containing 'contrasts', 'lead-lag-ratio', and 'lags', denoting the estimated values for each lag calculated, the lead-lag-ratio, and the tested lags respectively.

**References**

M. Hoffmann, M. Rosenbaum, and N. Yoshida (2013) : Estimation of the lead-lag parameter from non-synchronous data. *Bernoulli* 19, pages 1-37

**Examples**

```
ll <- leadLag(100 + cumsum(sbox), 100 + cumsum(lltc), lags = seq(-50,50))
plot(ll)
```

---

listAvailableKernels    *Available Kernels*

---

**Description**

Returns a vector of the available kernels.

**Usage**

```
listAvailableKernels()
```

**Value**

character vector

**Author(s)**

Scott Payseur

**References**

Ole E. Barndorff-Nielsen, Peter Reinhard Hansen, Asger Lunde, and Neil Shephard (2008). Designing Realized Kernels to Measure the ex post Variation of Equity Prices in the Presence of Noise. *Econometrica*, 76, pp. 1481-1536.

**Examples**

```
listAvailableKernels
```



---

listCholCovEstimators *Utility function listing the available estimators for the CholCov estimation*

---

**Description**

Utility function listing the available estimators for the CholCov estimation

**Usage**

```
listCholCovEstimators()
```

**Value**

This function returns a character vector containing the available estimators.

---

lltc *LLTC Data*

---

**Description**

Tick data for LLTC 2011/07/01, cleaned with tradesCleanup.

**Usage**

```
lltc
```

**Format**

xts object

**Examples**

```
data(lltc)  
plot(lltc)
```

---

makePsd	<i>Returns the positive semidefinite projection of a symmetric matrix using the eigenvalue method</i>
---------	-------------------------------------------------------------------------------------------------------

---

### Description

Function returns the positive semidefinite projection of a symmetric matrix using the eigenvalue method.

### Usage

```
makePsd(S, method = "covariance")
```

### Arguments

S	matrix.
method	character, indicating whether the negative eigenvalues of the correlation or covariance should be replaced by zero. Possible values are "covariance" and "correlation".

### Details

We use the eigenvalue method to transform  $S$  into a positive semidefinite covariance matrix (see e.g. Barndorff-Nielsen and Shephard, 2004, and Rousseeuw and Molenberghs, 1993). Let  $\Gamma$  be the orthogonal matrix consisting of the  $p$  eigenvectors of  $S$ . Denote  $\lambda_1^+, \dots, \lambda_p^+$  its  $p$  eigenvalues, whereby the negative eigenvalues have been replaced by zeroes. Under this approach, the positive semi-definite projection of  $S$  is  $S^+ = \Gamma' \text{diag}(\lambda_1^+, \dots, \lambda_p^+) \Gamma$ .

If `method = "correlation"`, the eigenvalues of the correlation matrix corresponding to the matrix  $S$  are transformed. See Fan et al (2010).

### Value

An xts object containing the aggregated trade data.

### Author(s)

Jonathan Cornelissen and Kris Boudt

### References

Barndorff-Nielsen, O. and N. Shephard (2004). Measuring the impact of jumps in multivariate price processes using bipower covariation. Discussion paper, Nuffield College, Oxford University.  
 Fan, J., Y. Li, and K. Yu (2010). Vast volatility matrix estimation using high frequency data for portfolio selection. Working paper.  
 Rousseeuw, P. and G. Molenberghs (1993). Transformation of non positive semidefinite correlation matrices. Communications in Statistics - Theory and Methods 22, 965-984.

---

makeReturns	<i>Compute log returns</i>
-------------	----------------------------

---

**Description**

Function returns an xts object with the log returns as xts object.

$$\log \text{ return}_t = (\log(\text{PRICE}_t) - \log(\text{PRICE}_{t-1})).$$

**Usage**

```
makeReturns(ts)
```

**Arguments**

ts	xts object
----	------------

**Details**

Note: the first (row of) observation(s) is set to zero.

**Value**

an xts object containing the log returns.

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

matchTradesQuotes	<i>Match trade and quote data</i>
-------------------	-----------------------------------

---

**Description**

Function matches the trades and quotes and returns an xts-object containing both.

**Usage**

```
matchTradesQuotes(tData, qData, adjustment = 2)
```

**Arguments**

tData	data.table or xts-object containing the trade data (multiple days possible).
qData	data.table or xts-object containing the quote data (multiple days possible).
adjustment	numeric, number of seconds the quotes are registered faster than the trades (should be round and positive). Based on the research of Vergote (2005), we set 2 seconds as the default.

**Value**

data.table or xts-object containing the matched trade and quote data

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen

**References**

Vergote, O. (2005). How to match trades and quotes for NYSE stocks? K.U.Leuven working paper.

**Examples**

```
# match the trade and quote data
tqData <- matchTradesQuotes(sampleTData, sampleQData)
head(tqData)
# multi-day input allowed
tqData <- matchTradesQuotes(sampleTDataMicroseconds, sampleQDataMicroseconds)
```

---

medRQ

*An estimator of integrated quarticity from applying the median operator on blocks of three returns.*

---

**Description**

Function returns the medRQ, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the medRQ is given by

$$\text{medRQ}_t = \frac{3\pi N}{9\pi + 72 - 52\sqrt{3}} \left( \frac{N}{N-2} \right) \sum_{i=2}^{N-1} \text{med}(|r_{t,i-1}|, |r_{t,i}|, |r_{t,i+1}|)^4$$

**Usage**

```
medRQ(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

**rData** a zoo/xts object containing all returns in period t for one asset.

**alignBy** a string, align the tick data to "seconds"|"minutes"|"hours".

**alignPeriod** an integer, align the tick data to this many [seconds|minutes|hours].

**makeReturns** boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

**Examples**

```
## Not run:
medRQ(rData = sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
medRQ

## End(Not run)
```

---

medRV

---

*medRV*


---

**Description**

Function returns the medRV, defined in Andersen et al. (2009).

Let  $r_{t,i}$  be a return (with  $i = 1, \dots, M$ ) in period  $t$ .

Then, the medRV is given by

$$\text{medRV}_t = \frac{\pi}{6 - 4\sqrt{3} + \pi} \left( \frac{M}{M-2} \right) \sum_{i=2}^{M-1} \text{med}(|r_{t,i-1}|, |r_{t,i}|, |r_{t,i+1}|)^2$$

**Usage**

```
medRV(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Details**

The medRV belongs to the class of realized volatility measures in this package that use the series of high-frequency returns  $r_{t,i}$  of a day  $t$  to produce an ex post estimate of the realized volatility of that day  $t$ . medRV is designed to be robust to price jumps. The difference between RV and medRV is an estimate of the realized jump variability. Disentangling the continuous and jump components in RV can lead to more precise volatility forecasts, as shown in Andersen et al. (2007) and Corsi et al. (2010).

**Value**

numeric

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169 (1), 75-93.

Andersen, T.G., T. Bollerslev, and F. Diebold (2007). Roughing it up: including jump components in the measurement, modelling and forecasting of return volatility. *The Review of Economics and Statistics* 89 (4), 701-720.

Corsi, F., D. Pirino, and R. Reno (2010). Threshold Bipower Variation and the Impact of Jumps on Volatility Forecasting. *Journal of Econometrics* 159 (2), 276-288.

**Examples**

```
data(sampleTData);
medrv <- medRV(rData = sampleTData$PRICE, alignBy = "minutes",
              alignPeriod = 5, makeReturns = TRUE)
medrv
```

---

```
mergeQuotesSameTimestamp
```

*Merge multiple quote entries with the same time stamp*

---

**Description**

Function replaces multiple quote entries that have the same time stamp by a single one and returns an xts object with unique time stamps only.

**Usage**

```
mergeQuotesSameTimestamp(qData, selection = "median")
```

**Arguments**

- |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| qData     | an xts object or data.table containing the time series data, with at least two columns named "BID" and "OFR" indicating the bid and ask price and two columns "BIDSIZ", "OFRSIZ" indicating the number of round lots available at these prices. For data.table an additional column "DT" is necessary that stores the date/time information.                                                                                                                                                                        |
| selection | <p>indicates how the bid and ask price for a certain time stamp should be calculated in case of multiple observation for a certain time stamp. By default, selection = "median", and the median price is taken. Alternatively:</p> <ul style="list-style-type: none"> <li>• selection = "max.volume": use the (bid/ask) price of the entry with largest (bid/ask) volume.</li> <li>• selection = "weighted.average": take the weighted average of all bid (ask) prices, weighted by "BIDSIZ" ("OFRSIZ").</li> </ul> |

**Value**

xts or data.table object depending on input

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

mergeTradesSameTimestamp

*Merge multiple transactions with the same time stamp*

---

**Description**

Function replaces multiple transactions that have the same time stamp by a single one and returns an xts or data.table object with unique time stamps only.

**Usage**

```
mergeTradesSameTimestamp(tData, selection = "median")
```

**Arguments**

- |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tData     | an xts object containing the time series data, with one column named "PRICE" indicating the transaction price and one column "SIZE" indicating the number of shares traded.                                                                                                                                                                                                                                                                    |
| selection | <p>indicates how the price for a certain time stamp should be calculated in case of multiple observation for a certain time stamp. By default, selection = "median", and the median price is taken. Alternatively:</p> <ul style="list-style-type: none"> <li>• selection = "max.volume": use the price of the transaction with largest volume.</li> <li>• selection = "weighted.average": take the weighted average of all prices.</li> </ul> |

**Value**

data.table or xts object depending on input

**Note**

previously this function returned the mean of the size of the merged trades (pre version 0.7 and when not using max.volume as the criterion), now it returns the sum.

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

minRQ	<i>An estimator of integrated quarticity from applying the minimum operator on blocks of two returns.</i>
-------	-----------------------------------------------------------------------------------------------------------

---

**Description**

Function returns the minRQ, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the minRQ is given by

$$\text{minRQ}_t = \frac{\pi N}{3\pi - 8} \left( \frac{N}{N-1} \right) \sum_{i=1}^{N-1} \min(|r_{t,i}|, |r_{t,i+1}|)^4$$

**Usage**

```
minRQ(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours"
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt



## References

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

## Examples

```
## Not run:
minRQ(rData = sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
minRQ

## End(Not run)
```

---

minRV	<i>minRV</i>
-------	--------------

---

## Description

Function returns the minRV, defined in Andersen et al. (2009).

Let  $r_{t,i}$  be a return (with  $i = 1, \dots, M$ ) in period  $t$ .

Then, the minRV is given by

$$\text{minRV}_t = \frac{\pi}{\pi - 2} \left( \frac{M}{M - 1} \right) \sum_{i=1}^{M-1} \min(|r_{t,i}|, |r_{t,i+1}|)^2$$

## Usage

```
minRV(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

## Arguments

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

## Value

numeric

## Author(s)

Jonathan Cornelissen and Kris Boudt

## References

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169 (1), 75-93.

## Examples

```
data(sampleTData)
minrv <- minRV(rData = sampleTData$PRICE, alignBy = "minutes",
              alignPeriod = 5, makeReturns = TRUE)
minrv
```

---

MRC

*Modulated Realized Covariance (MRC): Return univariate or multivariate preaveraged estimator.*

---

## Description

Function returns univariate or multivariate preaveraged estimator, as defined in Hautsch and Podolskij (2013).

## Usage

```
MRC(pData, pairwise = FALSE, makePsd = FALSE)
```

## Arguments

pData	a list. Each list-item contains an xts object with the intraday price data of a stock.
pairwise	boolean, should be TRUE when refresh times are based on pairs of assets. FALSE by default.
makePsd	boolean, in case it is TRUE, the positive definite version of MRC is returned. FALSE by default.

## Details

In practice, market microstructure noise leads to a departure from the pure semimartingale model. We consider the process  $Y$  in period  $\tau$ :

$$Y_\tau = X_\tau + \epsilon_\tau$$

where, the observed  $d$  dimensional log-prices are the sum of underlying Brownian semimartingale process  $X$  and a noise term  $\epsilon_\tau$ .

$\epsilon_\tau$  is an i.i.d process with  $X$ .

It is intuitive that under mean zero i.i.d. microstructure noise some form of smoothing of the observed log-price should tend to diminish the impact of the noise. Effectively, we are going to

approximate a continuous function by an average of observations of  $Y$  in a neighborhood, the noise being averaged away.

Assume there is  $N$  equispaced returns in period  $\tau$  of a list (after refreshing data). Let  $r_{\tau_i}$  be a return (with  $i = 1, \dots, N$ ) of an asset in period  $\tau$ . Assume there is  $d$  assets.

In order to define the univariate pre-averaging estimator, we first define the pre-averaged returns as

$$\bar{r}_{\tau_j}^{(k)} = \sum_{h=1}^{k_N-1} g\left(\frac{h}{k_N}\right) r_{\tau_{j+h}}^{(k)}$$

where  $g$  is a non-zero real-valued function  $g : [0, 1] \rightarrow \mathbb{R}$  given by  $g(x) = \min(x, 1 - x)$ .  $k_N$  is a sequence of integers satisfying  $k_N = \lfloor \theta N^{1/2} \rfloor$ . We use  $\theta = 0.8$  as recommended in Hautsch & Podolskij (2013). The pre-averaged returns are simply a weighted average over the returns in a local window. This averaging diminishes the influence of the noise. The order of the window size  $k_n$  is chosen to lead to optimal convergence rates. The pre-averaging estimator is then simply the analogue of the Realized Variance but based on pre-averaged returns and an additional term to remove bias due to noise

$$\hat{C} = \frac{N^{-1/2}}{\theta\psi_2} \sum_{i=0}^{N-k_N+1} (\bar{r}_{\tau_i})^2 - \frac{\psi_1^{k_N} N^{-1}}{2\theta^2\psi_2^{k_N}} \sum_{i=0}^N r_{\tau_i}^2$$

with

$$\psi_1^{k_N} = k_N \sum_{j=1}^{k_N} \left( g\left(\frac{j+1}{k_N}\right) - g\left(\frac{j}{k_N}\right) \right)^2,$$

$$\psi_2^{k_N} = \frac{1}{k_N} \sum_{j=1}^{k_N-1} g^2\left(\frac{j}{k_N}\right).$$

$$\psi_2 = \frac{1}{12}$$

The multivariate counterpart is very similar. The estimator is called the Modulated Realized Covariance (MRC) and is defined as

$$\text{MRC} = \frac{N}{N - k_N + 2} \frac{1}{\psi_2 k_N} \sum_{i=0}^{N-k_N+1} \bar{\mathbf{r}}_{\tau_i} \cdot \bar{\mathbf{r}}_{\tau_i}' - \frac{\psi_1^{k_N}}{\theta^2 \psi_2^{k_N}} \hat{\Psi}$$

where  $\hat{\Psi}_N = \frac{1}{2N} \sum_{i=1}^N \mathbf{r}_{\tau_i} (\mathbf{r}_{\tau_i})'$ . It is a bias correction to make it consistent. However, due to this correction, the estimator is not ensured PSD. An alternative is to slightly enlarge the bandwidth such that  $k_N = \lfloor \theta N^{1/2+\delta} \rfloor$ .  $\delta = 0.1$  results in a consistent estimate without the bias correction and a PSD estimate, in which case:

$$\text{MRC}^\delta = \frac{N}{N - k_N + 2} \frac{1}{\psi_2 k_N} \sum_{i=0}^{N-k_N+1} \bar{\mathbf{r}}_i \cdot \bar{\mathbf{r}}_i'$$

## Value

an  $d \times d$  matrix

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Hautsch, N., & Podolskij, M. (2013). Preaveraging-Based Estimation of Quadratic Variation in the Presence of Noise and Jumps: Theory, Implementation, and Empirical Evidence. *Journal of Business & Economic Statistics*, 31(2), 165-183.

**Examples**

```
a <- list(sample5MinPricesJumps["2010-01-04",1], sample5MinPricesJumps["2010-01-04",2])
MRC(a, pairwise = TRUE, makePsd = TRUE)
```

---

noZeroPrices

*Delete the observations where the price is zero*

---

**Description**

Function deletes the observations where the price is zero.

**Usage**

```
noZeroPrices(tData)
```

**Arguments**

tData            an xts or data.table object at least containing a column "PRICE".

**Value**

an xts or data.table object depending on input

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

noZeroQuotes	<i>Delete the observations where the bid or ask is zero</i>
--------------	-------------------------------------------------------------

---

**Description**

Function deletes the observations where the bid or ask is zero.

**Usage**

```
noZeroQuotes(qData)
```

**Arguments**

qData            an xts or data.table object at least containing the columns "BID" and "OFR".

**Value**

xts object or data.table depending on type of input

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

quotesCleanup	<i>Cleans quote data</i>
---------------	--------------------------

---

**Description**

This is a wrapper function for cleaning the quote data in the entire folder dataSource. The result is saved in the folder dataDestination.

In case you supply the argument "qDataRaw", the on-disk functionality is ignored and the function returns the cleaned quotes as xts or data.table object (see examples).

The following cleaning steps are performed sequentially: [noZeroQuotes](#), [selectExchange](#), [rmLargeSpread](#), [mergeQuotesSameTimestamp](#), [rmOutliersQuotes](#).

**Usage**

```
quotesCleanup(  
  dataSource = NULL,  
  dataDestination = NULL,  
  exchanges,  
  qDataRaw = NULL,  
  report = TRUE,  
  selection = "median",  
  maxi = 50,  
)
```

```

window = 50,
type = "advanced",
rmoutliersmaxi = 10,
saveAsXTS = TRUE,
tz = "EST"
)

```

## Arguments

dataSource	character indicating the folder in which the original data is stored.
dataDestination	character indicating the folder in which the cleaned data is stored.
exchanges	vector of stock exchange symbols for all data in dataSource, e.g. exchanges = c("T","N") retrieves all stock market data from both NYSE and NASDAQ. The possible exchange symbols are: <ul style="list-style-type: none"> <li>• A: AMEX</li> <li>• N: NYSE</li> <li>• B: Boston</li> <li>• P: Arca</li> <li>• C: NSX</li> <li>• T/Q: NASDAQ</li> <li>• D: NASD ADF and TRF</li> <li>• X: Philadelphia</li> <li>• I: ISE</li> <li>• M: Chicago</li> <li>• W: CBOE</li> <li>• Z: BATS</li> </ul>
qDataRaw	xts or data.table object containing (ONE stock only) raw quote data. This argument is NULL by default. Enabling it means the arguments from, to, dataSource and dataDestination will be ignored. (only advisable for small chunks of data)
report	boolean and TRUE by default. In case it is true the function returns (also) a vector indicating how many quotes remained after each cleaning step.
selection	argument to be passed on to the cleaning routine <a href="#">mergeQuotesSameTimestamp</a> . The default is "median".
maxi	spreads which are greater than median(spreads of day) times maxi are excluded.
window	argument to be passed on to the cleaning routine <a href="#">rmOutliersQuotes</a> .
type	argument to be passed on to the cleaning routine <a href="#">rmOutliersQuotes</a> .
rmoutliersmaxi	argument to be passed on to the cleaning routine <a href="#">rmOutliersQuotes</a> .
saveAsXTS	indicates whether data should be saved in xts format instead of data.table when using on-disk functionality. TRUE by default.
tz	timezone to use

**Value**

The function converts every csv file in `dataSource` into multiple xts or `data.table` files. In `dataDestination`, there will be one folder for each symbol containing `.rds` files with cleaned data stored either in `data.table` or xts format.

In case you supply the argument `"qDataRaw"`, the on-disk functionality is ignored and the function returns a list with the cleaned quotes as an xts or `data.table` object depending on input (see examples).

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen.

**References**

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2009). Realized kernels in practice: Trades and quotes. *Econometrics Journal* 12, C1-C32. Brownlees, C.T. and Gallo, G.M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51, pages 2232-2245. Falkenberry, T.N. (2002). High frequency data filtering. Unpublished technical report.

**Examples**

```
# Consider you have raw quote data for 1 stock for 2 days
head(sampleQDataRawMicroseconds)
dim(sampleQDataRawMicroseconds)
qDataAfterCleaning <- quotesCleanup(qDataRaw = sampleQDataRawMicroseconds, exchanges = "N")
qDataAfterCleaning$report
dim(qDataAfterCleaning$qData)

# In case you have more data it is advised to use the on-disk functionality
#via "dataSource" and "dataDestination" arguments
```

---

rankJumpTest

*Rank jump test*

---

**Description**

Rank jump test

**Usage**

```
rankJumpTest(
  marketPrice,
  stockPrices,
  alpha = c(5, 3),
  coarseFreq = 10,
```

```

localWindow = 30,
rank = 1,
BoxCox = 1,
nBoot = 1000,
dontTestAtBoundaries = TRUE,
on = "minutes",
k = 5,
marketOpen = "09:30:00",
marketClose = "16:00:00",
tz = "GMT"
)

```

### Arguments

marketPrice	data.table or xts containing the market prices in levels
stockPrices	list containing the individual stock prices in either data.table or xts format. The format should be the same as marketPrice
alpha	significance level (in standard deviations) to use for the jump detections. Default is c(5, 3) for 5 and 3 in the market and stocks respectively.
coarseFreq	numeric denoting the coarse sampling frequency. Default is 10
localWindow	numeric denoting the local window for the bootstrap algorithm. Default is 30
rank	rank of the jump matrix under the null hypothesis. Default is 1
BoxCox	numeric of exponents for the Box-Cox transformation, default is 1
nBoot	numeric denoting how many replications to be used for the bootstrap algorithm. Default is 1000
dontTestAtBoundaries	logical determining whether to exclude data across different days. Default is TRUE
on	string indicating the time scale in which k is expressed. Possible values are: "secs", "seconds", "mins", "minutes", "hours".
k	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate an xts object to the 5 minute frequency, set k = 5 and on = "minutes".
marketOpen	the market opening time. This should be in the time zone specified by tz. By default, marketOpen = "09:30:00".
marketClose	the market closing time. This should be in the time zone specified by tz. By default, marketClose = "16:00:00".
tz	string specifying the time zone to which the times in data and/or marketOpen/marketClose belong. Default = "GMT". This parameter will also help determine the testing times as the test is done on non-overlapping pre-averaged returns.

### Value

A list containing "criticalValues" which are the bootstrapped critical values, "testStatistic" the test statistic of the jump test, "dimensions" which are the dimensions of the jump matrix "marketJumpDetections" the jumps detected in the market prices, "stockJumpDetections" the co-jumps detected in the individual stock prices, and "jumpIndices" which are the indices of the detected jumps.



**Examples**

```

## Not run:
##Rank jump test using simulated sample data that includes jumps
## pretend that the marketPrice is the first asset in the data:
marketPrice <- sample5MinPricesJumps[,1]
## construct stockPrice as a list:
stockPrice <- list()
for (i in 1:(ncol(sample5MinPricesJumps)-1)) {
  stockPrice[[i]] <- sample5MinPricesJumps[,i+1]
}
## This can take a long time due to the bootstrapping
rankTest <- rankJumpTest(marketPrice, stockPrice, coarseFreq = 10, k = 1, alpha = c(5,3),
                        tz = "GMT", marketOpen = "09:30:00", marketClose = "16:00:00")
# Plot the detected stock jump detections
plot(rankTest$stockJumpDetections)

## End(Not run)

```

rAVGCov

*Realized Covariance: Average Subsample***Description**

Realized Covariance using average subsample.

**Usage**

```

rAVGCov(
  rData,
  cor = FALSE,
  alignBy = "minutes",
  alignPeriod = 5,
  k = 1,
  makeReturns = FALSE
)

```

**Arguments**

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	Align the tick data to seconds minutes hours. Default is "minutes"
alignPeriod	Align the tick data to this many [seconds minutes hours]. This can be a fraction. Default is 5
k	numeric denoting which horizon to use for the subsamples. This can be a fraction as long as k is a divisor of alignPeriod default is 1
makeReturns	Prices are passed make them into log returns

**Value**

Realized covariance using average subsample.

**Author(s)**

Scott Payseur

**References**

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.

Michiel de Pooter, Martin Martens, and Dick van Dijk. Predicting the daily covariance matrix for S&P100 stocks using intraday data - but which frequency to use? *Econometric Reviews*, 2008.

**Examples**

```
# Average subsampled realized variance/covariance aligned at one minute returns at
# 5 subgrids (5 minutes).

# Univariate
rvSub <- rAVGCov(rData = sampleTData$PRICE, alignBy = "minutes",
                alignPeriod = 5, makeReturns = TRUE)
rvSub

# Multivariate:
rcovSub <- rAVGCov(rData = cbind(1l1tc, sbux, fill = 0), alignBy = "minutes",
                  alignPeriod = 5, makeReturns = FALSE)
rcovSub

# Multivariate with a 30 second fast aggregation and a 2.5 minute slow aggregation.
rcovSub <- rAVGCov(rData = cbind(1l1tc, sbux, fill = 0),
                  alignBy = "minutes", alignPeriod = 2.5, k = 0.5, makeReturns = FALSE)
rcovSub
```

---

rBeta

*Realized beta: a tool in measuring risk with respect to the market.*

---

**Description**

Depending on users' choices of estimator (realized covariance (RCOVestimator) and realized variance (RVestimator)), the function returns the realized beta, defined as the ratio between both.

The realized beta is given by

$$\beta_{jm} = \frac{RCOVestimator_{jm}}{RVestimator_m}$$

in which

*RCOVestimator* : Realized covariance of asset j and market index m.

*RVestimator* : Realized variance of market index m.

**Usage**

```
rBeta(
  rData,
  rIndex,
  RCOVestimator = "rCov",
  RVestimator = "RV",
  makeReturns = FALSE
)
```

**Arguments**

**rData** a zoo/xts object containing all returns in period t for one asset.

**rIndex** a zoo/xts object containing return in period t for an index.

**RCOVestimator** can be chosen among realized covariance estimators: rCov, rAVGCov, rBPCov, rHYCov, rKernelCov, rOWCov, rRTSCov, rThresholdCov and rTSCov. rCov by default.

**RVestimator** can be chosen among realized variance estimators: RV, minRV and medRV. RV by default. In case of missing RVestimator, RCOVestimator function applying for rIndex will be used.

**makeReturns** boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Details**

Suppose there are  $N$  equispaced returns on day  $t$  for the asset  $j$  and the index  $m$ . Denote  $r_{(j)i,t}$ ,  $r_{(m)i,t}$  as the  $i$ th return on day  $t$  for asset  $j$  and index  $m$  (with  $i = 1, \dots, N$ ).

By default, the RCov is used and the realized beta coefficient is computed as:

$$\hat{\beta}_{(jm)t} = \frac{\sum_{i=1}^N r_{(j)i,t} r_{(m)i,t}}{\sum_{i=1}^N r_{(m)i,t}^2}$$

(Barndorff & Shephard (2004)).

Note: It is worth to note that the function does not support to calculate for data of multiple days.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O. E., & Shephard, N. (2004). Econometric analysis of realized covariation: High frequency based covariance, regression, and correlation in # financial economics. *Econometrica*, 72(3), 885-925.

**Examples**

```
a <- sample5MinPricesJumps['2010-01-04', 1]
b <- sample5MinPricesJumps['2010-01-04', 2]
rBeta(a, b, RCOVestimator = "rBPCov", RVestimator = "minRV", makeReturns = TRUE)
```

rBPCov

*Realized BiPower Covariance***Description**

Function returns the Realized BiPower Covariance (rBPCov), defined in Barndorff-Nielsen and Shephard (2004).

Let  $r_{t,i}$  be an intraday  $N \times 1$  return vector and  $i = 1, \dots, M$  the number of intraday returns.

The rBPCov is defined as the process whose value at time  $t$  is the  $N$ -dimensional square matrix with  $k, q$ -th element equal to

$$\text{rBPCov}[k, q]_t = \frac{\pi}{8} \left( \sum_{i=2}^M |r_{(k)t,i} + r_{(q)t,i}| |r_{(k)t,i-1} + r_{(q)t,i-1}| - |r_{(k)t,i} - r_{(q)t,i}| |r_{(k)t,i-1} - r_{(q)t,i-1}| \right),$$

where  $r_{(k)t,i}$  is the  $k$ -th component of the return vector  $r_{i,t}$ .

**Usage**

```
rBPCov(
  rData,
  cor = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE,
  makePsd = FALSE
)
```

**Arguments**

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.
makePsd	boolean, in case it is TRUE, the positive definite version of rBPCov is returned. FALSE by default.

**Value**

an  $N \times N$  matrix or a list of matrices if the time period spans multiple days

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O. and N. Shephard (2004). Measuring the impact of jumps in multivariate price processes using bipower covariation. Discussion paper, Nuffield College, Oxford University.

**Examples**

```
# Realized Bipower Variance/Covariance for a price series aligned
# at 5 minutes.

# Univariate:
rbpv <- rBPCov(rData = sampleTData$PRICE, alignBy = "minutes",
              alignPeriod = 5, makeReturns = TRUE)
rbpv

# Multivariate:
rbpc <- rBPCov(rData = sample5MinPricesJumps['2010-01-04'], makeReturns = TRUE, makePsd = TRUE)
rbpc
```

---

rCholCov

*rCholCov positive semi-definite covariance estimation using the Chol-Cov algorithm*

---

**Description**

Function that estimates the integrated covariance matrix using the CholCov algorithm. The algorithm estimates the integrated covariance matrix by sequentially adding series and using ‘refresh-Time’ to synchronize the observations. This is done in order of liquidity, which means that the algorithm uses more data points than most other estimation techniques.

**Usage**

```
rCholCov(
  pData,
  IVest = "MRC",
  COVest = "MRC",
  criterion = "squared duration",
  ...
)
```

**Arguments**

pData	a list. Each list-item $i$ contains an xts object with the intraday price data of stock $i$ for day $t$ . The order of the data does not matter as it will be sorted according to the criterion specified in the criterion argument
IVest	integrated variance estimator, default is "MRC". For a list of implemented estimators, use listCholCovEstimators().
COVest	covariance estimator, default is "MRC". For a list of implemented estimators, use listCholCovEstimators().
criterion	criterion to use for sorting the data according to liquidity. Possible values are ["squared duration" "duration" "count"], defaults to "squared duration".
...	additional arguments to pass to IVest and COVest. See details.

**Details**

additional arguments for IVest and COVest should be passed in the ... argument. For the MRC estimator, which is the default, the theta and delta parameters can be set. These default to 1 and 0.1 respectively.

**Value**

a list containing the covariance matrix "CholCov", and the Cholesky decomposition "L" and "G" such that  $L * G * L' = CholCov$

**Author(s)**

Emil Sjoerup

**References**

Boudt, Laurent, Lunde, Quaadvlieg, Sauri(2017) Positive semidefinite integrated covariance estimation, factorizations and asynchronicity. Journal of Econometrics 196, 347-367

---

rCov

*Realized Covariance*

---

**Description**

Function returns the Realized Covariation (rCov). Let  $r_{t,i}$  be an intraday  $N \times 1$  return vector and  $i = 1, \dots, M$  the number of intraday returns.

Then, the rCov is given by

$$rCov_t = \sum_{i=1}^M r_{t,i} r'_{t,i}.$$

**Usage**

```
rCov(
  rData,
  cor = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE
)
```

**Arguments**

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ . In case of a matrix, no multi-day adjustment is possible.
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

an  $N \times N$  matrix

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**Examples**

```
# Realized Variance/Covariance for prices aligned
# at 5 minutes.
data(sampleTData)
data(sample5MinPricesJumps)

# Univariate:
rv = rCov(rData = sampleTData$PRICE, alignBy = "minutes",
          alignPeriod = 5, makeReturns = TRUE)
rv

# Multivariate:
rc = rCov(rData = sample5MinPricesJumps['2010-01-04'], makeReturns=TRUE)
rc
```

---

realizedLibrary	<i>The realized library from the Oxford-Man Institute of Quantitative Finance</i>
-----------------	-----------------------------------------------------------------------------------

---

### Description

A data.frame object containing the daily open-close returns, daily realized variances based on five-minute intraday returns and daily realized kernels ranging from 2000-01-03 up to 2019-06-10 for the S&P 500. Use `colnames(realizedLibrary)` to see which realized measures. The full library of the Oxford-Man Institute of Quantitative Finance can be found on their website: <https://realized.oxford-man.ox.ac.uk/>.

### Usage

```
realizedLibrary
```

### Format

```
data.frame
```

### References

Gerd Heber, Asger Lunde, Neil Shephard, and Kevin Sheppard (2009) "Oxford-Man Institute's realized library, version 0.3", Oxford-Man Institute, University of Oxford.

---

refreshTime	<i>Synchronize (multiple) irregular timeseries by refresh time</i>
-------------	--------------------------------------------------------------------

---

### Description

This function implements the refresh time synchronization scheme proposed by Harris et al. (1995). It picks the so-called refresh times at which all assets have traded at least once since the last refresh time point. For example, the first refresh time corresponds to the first time at which all stocks have traded. The subsequent refresh time is defined as the first time when all stocks have traded again. This process is repeated until the end of one time series is reached.

### Usage

```
refreshTime(pData, sort = FALSE, criterion = "squared duration")
```

### Arguments

pData	a list. Each list-item contains an xts object containing the original time series (one day only and typically a price series).
sort	logical determining whether to sort the index based on a criterion (will only sort descending (i.e. most liquid first)). Default is FALSE
criterion	character determining which criterion used. Currently supports "squared duration" and "duration". Default is "squared duration".



**Value**

An xts object containing the synchronized time series.

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Harris, F., T. McNish, G. Shoesmith, and R. Wood (1995). Cointegration, error correction, and price discovery on informationally linked security markets. *Journal of Financial and Quantitative Analysis* 30, 563-581.

**Examples**

```
# Suppose irregular timepoints:
start <- as.POSIXct("2010-01-01 09:30:00")
ta <- start + c(1,2,4,5,9)
tb <- start + c(1,3,6,7,8,9,10,11)

# Yielding the following timeseries:
a <- xts::as.xts(1:length(ta), order.by = ta)
b <- xts::as.xts(1:length(tb), order.by = tb)

# Calculate the synchronized timeseries:
refreshTime(list(a,b))
```

---

ReMeDI

*ReMeDI This function estimates the auto-covariance of market-microstructure noise*

---

**Description**

ReMeDI This function estimates the auto-covariance of market-microstructure noise

**Usage**

```
ReMeDI(pData, kn = 1, lags = 1, knEqual = FALSE, makeCorrelation = FALSE)
```

**Arguments**

pData	xts or data.table containing the log-prices of the asset
kn	numeric of length 1 determining the tuning parameter kn this controls the lengths of the non-overlapping interval in the ReMeDI estimation
lags	numeric containing integer values indicating

knEqual            Use an altered version of the ReMeDI estimator, where we instead use equal kn, instead of kn and 2\*kn for the windows. See Figure 1 of paper in reference section.

makeCorrelation    logical indicating whether to transform the autocovariances into autocorrelations. The estimate of variance is imprecise and thus, constructing the correlation like this may show correlations that fall outside (-1,1)

### Author(s)

Emil Sjoerup

### References

Li and Linton (2019) (Working paper): "A ReMeDI for microstructure noise."

### Examples

```
remed <- ReMeDI(sampleTDataMicroseconds[as.Date(DT) == "2018-01-02", ], kn = 2, lags = 1:8)
# We can also use the algorithm for choosing the kn to
optimalKn <- knChooseReMeDI(sampleTDataMicroseconds[as.Date(DT) == "2018-01-02", ],
                             knMax = 10, tol = 0.05, size = 3,
                             lower = 2, upper = 5, plot = TRUE)
optimalKn
remed <- ReMeDI(sampleTDataMicroseconds[as.Date(DT) == "2018-01-02", ], kn = optimalKn, lags = 1:8)
```

---

rHYCov

*Hayashi-Yoshida Covariance*

---

### Description

Calculates the Hayashi-Yoshida Covariance estimator

### Usage

```
rHYCov(
  rData,
  cor = FALSE,
  period = 1,
  alignBy = "seconds",
  alignPeriod = 1,
  makeReturns = FALSE,
  makePsd = TRUE
)
```

**Arguments**

rData	a possibly multivariate xts object.
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
period	Sampling period
alignBy	Align the tick data to seconds minutes hours
alignPeriod	Align the tick data to this many [seconds minutes hours]
makeReturns	Prices are passed make them into log returns
makePsd	boolean, in case it is TRUE, the positive definite version of rHYCov is returned. FALSE by default.

**Author(s)**

Scott Payseur

# Average Hayashi-Yoshida Covariance estimator is calculated on five-minute returns

# Multivariate: # realized\_cov <- rHYCov(rData = cbind(l1tc, sbux, fill = 0), period = 5, alignBy = "minutes", # alignPeriod = 5, makeReturns = FALSE) # realized\_cov Note: for the diagonal elements the rCov is used.

**References**

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

---

rKernelCov

*Realized Covariance: Kernel*


---

**Description**

Realized covariance calculation using a kernel estimator.

**Usage**

```
rKernelCov(
  rData,
  cor = FALSE,
  alignBy = "seconds",
  alignPeriod = 1,
  makeReturns = FALSE,
  kernelType = "rectangular",
  kernelParam = 1,
  kernelDOFadj = TRUE
)
```

**Arguments**

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	Align the tick data to seconds minutes hours
alignPeriod	Align the tick data to this many [seconds minutes hours]
makeReturns	Convert to Returns
kernelType	Kernel name (or number)
kernelParam	Kernel parameter (usually lags)
kernelDOFadj	Kernel Degree of freedom adjustment

**Details**

The different types of kernels can be found using [listAvailableKernels](#).

**Value**

Kernel estimate of realized covariance.

**Author(s)**

Scott Payseur and Onno Kleen

**References**

- Ole E. Barndorff-Nielsen, Peter Reinhard Hansen, Asger Lunde, and Neil Shephard (2008). Designing Realized Kernels to Measure the ex post Variation of Equity Prices in the Presence of Noise. *Econometrica*, 76, pp. 1481-1536.
- B. Zhou. High-frequency data and volatility in foreign-exchange rates. *Journal of Business & Economic Statistics*, 14:45-52, 1996.
- P. Hansen and A. Lunde. Realized variance and market microstructure noise. *Journal of Business and Economic Statistics*, 24:127-218, 2006.

**Examples**

```
# Univariate:
rvKernel <- rKernelCov(rData = sampleTData$PRICE, alignBy = "minutes",
                      alignPeriod = 5, makeReturns = TRUE)
rvKernel

# Multivariate:
rcKernel <- rKernelCov(rData = cbind(1l1tc, sbux, fill = 0), alignBy = "minutes",
                      alignPeriod = 5, makeReturns = FALSE)
rcKernel
```

---

rKurt	<i>Realized kurtosis of highfrequency return series.</i>
-------	----------------------------------------------------------

---

**Description**

Function returns Realized kurtosis, defined in Amaya et al. (2011).

**Usage**

```
rKurt(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Details**

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rKurt is given by

$$\text{rKurt}_t = \frac{N \sum_{i=1}^N (r_{t,i})^4}{RV_t^2}$$

in which  $RV_t$  : realized variance

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Amaya, D., Christoffersen, P., Jacobs, K. and Vasquez, A. (2011). Do realized skewness and kurtosis predict the cross-section of equity returns?. CREATES research paper. p. 3-7.

**Examples**

```
data(sampleTData)
rKurt(sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
```

---

rmLargeSpread	<i>Delete entries for which the spread is more than "maxi" times the median spread</i>
---------------	----------------------------------------------------------------------------------------

---

**Description**

Function deletes entries for which the spread is more than "maxi" times the median spread on that day.

**Usage**

```
rmLargeSpread(qData, maxi = 50)
```

**Arguments**

qData	an xts or data.table object at least containing the columns "BID" and "OFR".
maxi	an integer. By default maxi = "50", which means that entries are deleted if the spread is more than 50 times the median spread on that day.

**Value**

xts or data.table object depending on input.

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

rmNegativeSpread	<i>Delete entries for which the spread is negative</i>
------------------	--------------------------------------------------------

---

**Description**

Function deletes entries for which the spread is negative.

**Usage**

```
rmNegativeSpread(qData)
```

**Arguments**

qData	an xts object at least containing the columns "BID" and "OFR".
-------	----------------------------------------------------------------

**Value**

data.table or xts object

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen

**Examples**

```
rmNegativeSpread(sampleQDataRawMicroseconds)
```

---

rmOutliersQuotes	<i>Delete entries for which the mid-quote is outlying with respect to surrounding entries</i>
------------------	-----------------------------------------------------------------------------------------------

---

**Description**

If type = "standard": Function deletes entries for which the mid-quote deviated by more than "maxi" median absolute deviations from a rolling centered median (excluding the observation under consideration) of "window" observations.

If type = "advanced": Function deletes entries for which the mid-quote deviates by more than "maxi" median absolute deviations from the value closest to the mid-quote of these three options:

1. Rolling centered median (excluding the observation under consideration)
2. Rolling median of the following "window" observations
3. Rolling median of the previous "window" observations

The advantage of this procedure compared to the "standard" proposed by Barndorff-Nielsen et al. (2010) is that it will not incorrectly remove large price jumps. Therefore this procedure has been set as the default for removing outliers.

Note that the median absolute deviation is taken over the entire day. In case it is zero (which can happen if mid-quotes don't change much), the median absolute deviation is taken over a subsample without constant mid-quotes.

**Usage**

```
rmOutliersQuotes(qData, maxi = 10, window = 50, type = "advanced")
```

**Arguments**

qData	a data.table or xts object at least containing the columns "BID" and "OFR".
maxi	an integer, indicating the maximum number of median absolute deviations allowed.
window	an integer, indicating the time window for which the "outlyingness" is considered.
type	should be "standard" or "advanced" (see description).

**Details**

NOTE: This function works only correct if supplied input data consists of 1 day.

**Value**

xts object or data.table depending on type of input

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2009). Realized kernels in practice: Trades and quotes. *Econometrics Journal* 12, C1-C32.

Brownlees, C.T. and Gallo, G.M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51, pages 2232-2245.

---

rMPV

*Realized multipower variation (MPV), an estimator of integrated power variation.*

---

**Description**

Function returns the rMPV, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rMPV is given by

$$\text{rMPV}_N(m, p) = d_{m,p} \frac{N^{p/2}}{N - m + 1} \sum_{i=1}^{N-m+1} |r_{t,i}|^{p/m} \dots |r_{t,i+m-1}|^{p/m}$$

in which

$$d_{m,p} = \mu_p^{-m};$$

$m$ : the window size of return blocks;

$p$ : the power of the variation;

and  $m > p/2$ .



**Usage**

```
rMPV(
  rData,
  m = 2,
  p = 2,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE
)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
m	the window size of return blocks. 2 by default.
p	the power of the variation. 2 by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

**Examples**

```
data(sampleTData)
rMPV(sampleTData$PRICE, m = 2, p = 3, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
```

---

rmTradeOutliers	<i>Delete transactions with unlikely transaction prices</i>
-----------------	-------------------------------------------------------------

---

**Description**

Deprecated - use rmTradeOutliers instead.

**Usage**

```
rmTradeOutliers(tData, qData)
```

**Arguments**

tData	a data.table or xts object containing the time series data, with at least the column "PRICE", containing the transaction price (ONE DAY ONLY).
qData	a data.table or xts object containing the time series data with at least the columns "BID" and "OFR", containing the bid and ask prices (ONE DAY ONLY).

**Value**

xts or data.table object depending on input

---

rmTradeOutliersUsingQuotes	<i>Delete transactions with unlikely transaction prices</i>
----------------------------	-------------------------------------------------------------

---

**Description**

Function deletes entries with prices that are above the ask plus the bid-ask spread. Similar for entries with prices below the bid minus the bid-ask spread.

**Usage**

```
rmTradeOutliersUsingQuotes(tData, qData, lagQuotes = 2)
```

**Arguments**

tData	a data.table or xts object containing the time series data, with at least the column "PRICE", containing the transaction price.
qData	a data.table or xts object containing the time series data with at least the columns "BID" and "OFR", containing the bid and ask prices.
lagQuotes	a numeric of length 1 that denotes how many seconds to lag the quotes. Default is 2 seconds. See Details.

**Details**

Note: in order to work correctly, the input data of this function should be cleaned trade (tData) and quote (qData) data respectively. In older high frequency datasets the trades frequently lag the quotes. In newer datasets this tends to happen only during extreme market activity when exchange networks are at maximum capacity.

**Value**

xts or data.table object depending on input

**Author(s)**

Jonathan Cornelissen, Kris Boudt and Onno Kleen

**References**

Vergote, O. (2005). How to match trades and quotes for NYSE stocks? K.U.Leuven working paper.

---

rOWCov

*Realized Outlyingness Weighted Covariance*


---

**Description**

Function returns the Realized Outlyingness Weighted Covariance, defined in Boudt et al. (2008).

Let  $r_{t,i}$ , for  $i = 1, \dots, M$  be a sample of  $M$  high-frequency ( $N \times 1$ ) return vectors and  $d_{t,i}$  their outlyingness given by the squared Mahalanobis distance between the return vector and zero in terms of the reweighted MCD covariance estimate based on these returns.

Then, the rOWCov is given by

$$\text{rOWCov}_t = c_w \frac{\sum_{i=1}^M w(d_{t,i}) r_{t,i} r'_{t,i}}{\frac{1}{M} \sum_{i=1}^M w(d_{t,i})},$$

The weight  $w_{i,\Delta}$  is one if the multivariate jump test statistic for  $r_{i,\Delta}$  in Boudt et al. (2008) is less than the 99.9% percentile of the chi-square distribution with  $N$  degrees of freedom and zero otherwise. The scalar  $c_w$  is a correction factor ensuring consistency of the rOWCov for the Integrated Covariance, under the Brownian Semimartingale with Finite Activity Jumps model.

**Usage**

```
rOWCov(
  rData,
  cor = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE,
  seasadjR = NULL,
```

```
wFunction = "HR",
alphaMCD = 0.75,
alpha = 0.001
)
```

### Arguments

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.
seasadjR	a ( $M \times N$ ) matrix/zoo/xts object containing the seasonally adjusted returns. This is an optional argument.
wFunction	determines whether a zero-one weight function (one if no jump is detected based on $d_{t,i}$ and 0 otherwise) or Soft Rejection ("SR") weight function is to be used. By default a zero-one weight function (wFunction = "HR") is used.
alphaMCD	a numeric parameter, controlling the size of the subsets over which the determinant is minimized. Allowed values are between 0.5 and 1 and the default is 0.75. See Boudt et al. (2008) or the covMcd function in the robustbase package.
alpha	is a parameter between 0 en 0.5, that determines the rejection threshold value (see Boudt et al. (2008) for details).

### Details

Advantages of the rOWCov compared to the rBPCov include a higher statistical efficiency, positive semidefiniteness and affine equivariance. However, the rOWCov suffers from a curse of dimensionality. Indeed, the rOWCov gives a zero weight to a return vector if at least one of the components is affected by a jump. In the case of independent jump occurrences, the average proportion of observations with at least one component being affected by jumps increases fast with the dimension of the series. This means that a potentially large proportion of the returns receives a zero weight, due to which the rOWCov can have a low finite sample efficiency in higher dimensions

### Value

an  $N \times N$  matrix

### Author(s)

Jonathan Cornelissen and Kris Boudt

### References

Boudt, K., C. Croux, and S. Laurent (2008). Outlyingness weighted covariation. Mimeo.

**Examples**

```
# Realized Outlyingness Weighted Variance/Covariance for prices aligned
# at 5 minutes.

# Univariate:
rvoutw <- rOWCov(rData = sampleTData$PRICE, alignBy = "minutes",
                 alignPeriod = 5, makeReturns = TRUE)
rvoutw

# Multivariate:
rcoutw <- rOWCov(rData = sample5MinPricesJumps['2010-01-04'], makeReturns = TRUE)
rcoutw
```

rQPVar

*Realized quad-power variation of highfrequency return series.***Description**

Function returns the realized quad-power variation, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rQPVar is given by

$$\text{rQPVar}_t = N * \frac{N}{N-3} \left( \frac{\pi^2}{4} \right)^{-4} (|r_{t,i}| |r_{t,i-1}| |r_{t,i-2}| |r_{t,i-3}|)$$

**Usage**

```
rQPVar(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

## References

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

## Examples

```
data(sampleTData)
rQPVar(rData= sampleTData$PRICE, alignBy= "minutes", alignPeriod =5, makeReturns= TRUE)
rQPVar
```

---

rQuar

*Realized quarticity of highfrequency return series.*

---

## Description

Function returns the rQuar, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rQuar is given by

$$\text{rQuar}_t = \frac{N}{3} \sum_{i=1}^N (r_{t,i}^4)$$

## Usage

```
rQuar(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

## Arguments

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

## Value

numeric

## Author(s)

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

## References

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

**Examples**

```
## Not run:
data(sampleTData)
rQuar(rData = sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)

## End(Not run)
```

rRTSCov

*Robust two time scale covariance estimation***Description**

Function returns the robust two time scale covariance matrix proposed in Boudt and Zhang (2010). Unlike the `rOWCov`, but similarly to the `rThresholdCov`, the `rRTSCov` uses univariate jump detection rules to truncate the effect of jumps on the covariance estimate. By the use of two time scales, this covariance estimate is not only robust to price jumps, but also to microstructure noise and non-synchronous trading.

**Usage**

```
rRTSCov(
  pData,
  cor = FALSE,
  startIV = NULL,
  noisevar = NULL,
  K = 300,
  J = 1,
  K_cov = NULL,
  J_cov = NULL,
  K_var = NULL,
  J_var = NULL,
  eta = 9,
  makePsd = FALSE
)
```

**Arguments**

<code>pData</code>	a list. Each list-item <i>i</i> contains an <code>xts</code> object with the intraday price data of stock <i>i</i> for day <i>t</i> .
<code>cor</code>	boolean, in case it is <code>TRUE</code> , the correlation is returned. <code>FALSE</code> by default.
<code>startIV</code>	vector containing the first step estimates of the integrated variance of the assets, needed in the truncation. Is <code>NULL</code> by default.
<code>noisevar</code>	vector containing the estimates of the noise variance of the assets, needed in the truncation. Is <code>NULL</code> by default.
<code>K</code>	positive integer, slow time scale returns are computed on prices that are <i>K</i> steps apart.

J	positive integer, fast time scale returns are computed on prices that are J steps apart.
K_cov	positive integer, for the extradiagonal covariance elements the slow time scale returns are computed on prices that are K steps apart.
J_cov	positive integer, for the extradiagonal covariance elements the fast time scale returns are computed on prices that are J steps apart.
K_var	vector of positive integers, for the diagonal variance elements the slow time scale returns are computed on prices that are K steps apart.
J_var	vector of positive integers, for the diagonal variance elements the fast time scale returns are computed on prices that are J steps apart.
eta	positive real number, squared standardized high-frequency returns that exceed eta are detected as jumps.
makePsd	boolean, in case it is TRUE, the positive definite version of rRTSCov is returned. FALSE by default.

### Details

The rRTSCov requires the tick-by-tick transaction prices. (Co)variances are then computed using log-returns calculated on a rolling basis on stock prices that are  $K$  (slow time scale) and  $J$  (fast time scale) steps apart.

The diagonal elements of the rRTSCov matrix are the variances, computed for log-price series  $X$  with  $n$  price observations at times  $\tau_1, \tau_2, \dots, \tau_n$  as follows:

$$\left(1 - \frac{\bar{n}_K}{\bar{n}_J}\right)^{-1} \left( \{X, X\}_T^{(K)*} - \frac{\bar{n}_K}{\bar{n}_J} \{X, X\}_T^{(J)*} \right),$$

where  $\bar{n}_K = (n - K + 1)/K$ ,  $\bar{n}_J = (n - J + 1)/J$  and

$$\{X, X\}_T^{(K)*} = \frac{c_\eta^* \sum_{i=1}^{n-K+1} (X_{t_{i+K}} - X_{t_i})^2 I_X^K(i; \eta)}{K \frac{1}{n-K+1} \sum_{i=1}^{n-K+1} I_X^K(i; \eta)}.$$

The constant  $c_\eta$  adjusts for the bias due to the thresholding and  $I_X^K(i; \eta)$  is a jump indicator function that is one if

$$\frac{(X_{t_{i+K}} - X_{t_i})^2}{\left(\int_{t_i}^{t_{i+K}} \sigma_s^2 ds + 2\sigma_{\varepsilon_X}^2\right)} \leq \eta$$

and zero otherwise. The elements in the denominator are the integrated variance (estimated recursively) and noise variance (estimated by the method in Zhang et al, 2005).

The extradiagonal elements of the rRTSCov are the covariances. For their calculation, the data is first synchronized by the refresh time method proposed by Harris et al (1995). It uses the function `refreshTime` to collect first the so-called refresh times at which all assets have traded at least once since the last refresh time point. Suppose we have two log-price series:  $X$  and  $Y$ . Let  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_{N_\tau^X}\}$  and  $\Theta = \{\theta_1, \theta_2, \dots, \theta_{N_\tau^Y}\}$  be the set of transaction times of these assets. The first refresh time corresponds to the first time at which both stocks have traded, i.e.  $\phi_1 = \max(\tau_1, \theta_1)$ . The subsequent refresh time is defined as the first time when both stocks have again traded, i.e.  $\phi_{j+1} = \max(\tau_{N_{\phi_j}^X} + 1, \theta_{N_{\phi_j}^Y} + 1)$ . The complete refresh time sample grid is



$\Phi = \{\phi_1, \phi_2, \dots, \phi_{M_N+1}\}$ , where  $M_N$  is the total number of paired returns. The sampling points of asset  $X$  and  $Y$  are defined to be  $t_i = \max\{\tau \in \Gamma : \tau \leq \phi_i\}$  and  $s_i = \max\{\theta \in \Theta : \theta \leq \phi_i\}$ .

Given these refresh times, the covariance is computed as follows:

$$c_N(\{X, Y\}_T^{(K)} - \frac{\bar{n}_K}{\bar{n}_J} \{X, Y\}_T^{(J)}),$$

where

$$\{X, Y\}_T^{(K)} = \frac{1}{K} \frac{\sum_{i=1}^{M_N-K+1} c_i (X_{t_{i+K}} - X_{t_i})(Y_{s_{i+K}} - Y_{s_i}) I_X^K(i; \eta) I_Y^K(i; \eta)}{\frac{1}{M_N-K+1} \sum_{i=1}^{M_N-K+1} I_X^K(i; \eta) I_Y^K(i; \eta)},$$

with  $I_X^K(i; \eta)$  the same jump indicator function as for the variance and  $c_N$  a constant to adjust for the bias due to the thresholding.

Unfortunately, the rRTSCov is not always positive semidefinite. By setting the argument `makePsd = TRUE`, the function `makePsd` is used to return a positive semidefinite matrix. This function replaces the negative eigenvalues with zeroes.

### Value

an  $N \times N$  matrix

### Author(s)

Jonathan Cornelissen and Kris Boudt

### References

Boudt K. and Zhang, J. 2010. Jump robust two time scale covariance estimation and realized volatility budgets. Mimeo.

Harris, F., T. McInish, G. Shoesmith, and R. Wood (1995). Cointegration, error correction, and price discovery on informationally linked security markets. *Journal of Financial and Quantitative Analysis* 30, 563-581.

Zhang, L., P. A. Mykland, and Y. Ait-Sahalia (2005). A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association* 100, 1394-1411.

### Examples

```
# Robust Realized two timescales Variance/Covariance
data(sampleTData)
# Univariate:
rvRTS <- rRTSCov(pData = sampleTData$PRICE)
# Note: Prices as input
rvRTS

# Multivariate:
rcRTS <- rRTSCov(pData = list(cumsum(1l1tc) + 100, cumsum(sbox) + 100))
# Note: List of prices as input
rcRTS
```

rSemiCov

*Realized Semicovariance***Description**

Function returns the Realized Semicovariances (rSemiCov). Let  $r_{t,i}$  be an intraday  $M \times N$  return matrix and  $i = 1, \dots, M$  the number of intraday returns. Then, let  $p = \max(r_{t,i}, 0)$  and  $n = \min(r_{t,i}, 0)$ .

Then, the realized semicovariance is given by the following three matrices:

$$\text{pos}_t = \sum_{i=1}^M p_{t,i} p'_{t,i}$$

$$\text{neg}_t = \sum_{i=1}^M n_{t,i} n'_{t,i}$$

$$\text{mixed}_t = \sum_{i=1}^M (p_{t,i} n'_{t,i} + n_{t,i} p'_{t,i})$$

The mixed covariance matrix will have 0 on the diagonal. From these three matrices, the realized covariance can be constructed as  $\text{pos} + \text{neg} + \text{mixed}$ . The concordant semicovariance matrix is  $\text{pos} + \text{neg}$ . The off-diagonals of the concordant matrix is always positive, while for the mixed matrix, it is always negative.

**Usage**

```
rSemiCov(
  rData,
  cor = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE
)
```

**Arguments**

rData	a ( $M \times N$ ) matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ . In case of a matrix, no multi-day adjustment is possible.
cor	logical, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	logical, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Details**

In the case that cor is TRUE, the mixed matrix will be an  $N \times N$  matrix filled with NA as mapping the mixed covariance matrix into correlation space is impossible due to the 0-diagonal.

**Value**

In case the data consists of one day a list of four  $N \times N$  matrices are returned. These matrices are the mixed, positive, negative and concordant In case the data spans more than one day, the list for each day will be put into another list named according to the date of the estimates.

**Author(s)**

Emil Sjoerup

Emil Sjoerup

**Examples**

```
# Realized semi-variance/semi-covariance for prices aligned
# at 5 minutes.
data(sampleTData)
data(sample5MinPricesJumps)

# Univariate:
rSV = rSemiCov(rData = sampleTData$PRICE, alignBy = "minutes",
              alignPeriod = 5, makeReturns = TRUE)

# Multivariate single day:
rSC = rSemiCov(rData = sample5MinPricesJumps['2010-01-04'], makeReturns=TRUE)

## Not run:
# Multivariate multi day:
rSC <- rSemiCov(sample5MinPrices, makeReturns = TRUE) # rSC is a list of lists
# We extract the covariance between stock 1 and stock 2 for all three covariances.
mixed <- do.call(rbind, lapply(rSC, function(x) x[["mixed"]][1,2]))
neg <- do.call(rbind, lapply(rSC, function(x) x[["negative"]][1,2]))
pos <- do.call(rbind, lapply(rSC, function(x) x[["positive"]][1,2]))
covariances <- xts(cbind(mixed, neg, pos), as.Date(rownames(pos)))
colnames(covariances) <- c("mixed", "neg", "pos")
# We make a quick plot of the different covariances
plot(covariances)
addLegend(lty = 1) # Add legend so we can distinguish the series.

## End(Not run)
```

**Description**

Function returns Realized skewness, defined in Amaya et al. (2011).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rSkew is given by

$$\text{rSkew}_t = \frac{\sqrt{N} \sum_{i=1}^N (r_{t,i})^3}{RV_t^{3/2}}$$

in which  $RV_t$  : realized variance

**Usage**

```
rSkew(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Amaya, D., Christoffersen, P., Jacobs, K. and Vasquez, A. (2011). Do realized skewness and kurtosis predict the cross-section of equity returns?. CREATES research paper. p. 3-7.

**Examples**

```
data(sampleTData)
rSkew(sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
```

rSV

*Realized semivariance of highfrequency return series.***Description**

Function returns realized semivariances, defined in Barndorff-Nielsen et al. (2008).

Function returns two outcomes: 1.Downside realized semivariance and 2.Upside realized semivariance.

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rSV is given by

$$\text{rSVdownside}_t = \sum_{i=1}^N (r_{t,i})^2 \times I[r_{t,i} < 0]$$

$$\text{rSVupside}_t = \sum_{i=1}^N (r_{t,i})^2 \times I[r_{t,i} > 0]$$

**Usage**

```
rSV(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

rData	a zoo/xts object containing all returns in period t for one asset.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

list with two arguments. The realized positive and negative semivariance.

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O.E., Kinnebrock, S. and Shephard N. (2008). Measuring downside risk - realized semivariance. CREATES research paper. p. 3-5.

**Examples**

```
## Not run:
data(sampleTData)
rSV(sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)

## End(Not run)
```

rThresholdCov

*Threshold Covariance***Description**

Function returns the threshold covariance matrix proposed in Gobbi and Mancini (2009). Unlike the `rOWCov`, the `rThresholdCov` uses univariate jump detection rules to truncate the effect of jumps on the covariance estimate. As such, it remains feasible in high dimensions, but it is less robust to small cojumps.

Let  $r_{t,i}$  be an intraday  $N \times 1$  return vector and  $i = 1, \dots, M$  the number of intraday returns.

Then, the  $k, q$ -th element of the threshold covariance matrix is defined as

$$\text{thresholdcov}[k, q]_t = \sum_{i=1}^M r^{(k)t,i} 1_{\{r_{(k)t,i}^2 \leq TR_M\}} r^{(q)t,i} 1_{\{r_{(q)t,i}^2 \leq TR_M\}},$$

with the threshold value  $TR_M$  set to  $9\Delta^{-1}$  times the daily realized bi-power variation of asset  $k$ , as suggested in Jacod and Todorov (2009).

**Usage**

```
rThresholdCov(
  rData,
  cor = FALSE,
  alignBy = NULL,
  alignPeriod = NULL,
  makeReturns = FALSE
)
```

**Arguments**

rData	a $(M \times N)$ matrix/zoo/xts object containing the $N$ return series over period $t$ , with $M$ observations during $t$ .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
alignBy	a string, align the tick data to "seconds" "minutes" "hours".
alignPeriod	an integer, align the tick data to this many [seconds minutes hours].
makeReturns	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

an  $N \times N$  matrix

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O. and N. Shephard (2004). Measuring the impact of jumps in multivariate price processes using bipower covariation. Discussion paper, Nuffield College, Oxford University.

Jacod, J. and V. Todorov (2009). Testing for common arrival of jumps in discretely-observed multidimensional processes. *Annals of Statistics* 37, 1792-1838.

Mancini, C. and F. Gobbi (2009). Identifying the covariation between the diffusion parts and the co-jumps given discrete observations. Mimeo.

**Examples**

```
# Realized threshold Variance/Covariance:
# Multivariate:
rcThreshold <- rThresholdCov(cbind(1l1tc, sbux), alignBy = "minutes", alignPeriod = 1)
rcThreshold
```

---

rTPVar

*Realized tri-power variation estimator of quarticity for a high-frequency return series.*

---

**Description**

Function returns the rTPVar, defined in Andersen et al. (2012).

Assume there is  $N$  equispaced returns in period  $t$ . Let  $r_{t,i}$  be a return (with  $i = 1, \dots, N$ ) in period  $t$ .

Then, the rTPVar is given by

$$\text{rTPVar}_t = N \frac{N}{N-2} \left( \frac{\Gamma(0.5)}{2^{2/3} \Gamma(7/6)} \right)^3 \sum_{i=3}^N (|r_{t,i}|^{4/3} |r_{t,i-1}|^{4/3} |r_{t,i-2}|^{4/3})$$

**Usage**

```
rTPVar(rData, alignBy = NULL, alignPeriod = NULL, makeReturns = FALSE)
```

**Arguments**

<code>rData</code>	a zoo/xts object containing all returns in period t for one asset.
<code>alignBy</code>	a string, align the tick data to "seconds" "minutes" "hours".
<code>alignPeriod</code>	an integer, align the tick data to this many [seconds minutes hours].
<code>makeReturns</code>	boolean, should be TRUE when rData contains prices instead of returns. FALSE by default.

**Value**

numeric

**Author(s)**

Giang Nguyen, Jonathan Cornelissen and Kris Boudt

**References**

Andersen, T. G., D. Dobrev, and E. Schaumburg (2012). Jump-robust volatility estimation using nearest neighbor truncation. *Journal of Econometrics*, 169(1), 75- 93.

**Examples**

```
data(sampleTData)
rTPVar(rData = sampleTData$PRICE, alignBy = "minutes", alignPeriod = 5, makeReturns = TRUE)
rTPVar
```

---

RTQ

*Calculate the realized tripower quarticity*

---

**Description**

Calculate the realized tripower quarticity

**Usage**

```
RTQ(rData)
```

**Arguments**

<code>rData</code>	a zoo/xts object containing all returns in period t for one asset.
--------------------	--------------------------------------------------------------------

**Value**

numeric



---

rTSCov

*Two time scale covariance estimation*


---

### Description

Function returns the two time scale covariance matrix proposed in Zhang et al (2005) and Zhang (2010). By the use of two time scales, this covariance estimate is robust to microstructure noise and non-synchronous trading.

### Usage

```
rTSCov(
  pData,
  cor = FALSE,
  K = 300,
  J = 1,
  K_cov = NULL,
  J_cov = NULL,
  K_var = NULL,
  J_var = NULL,
  makePsd = FALSE
)
```

### Arguments

pData	a list. Each list-item <i>i</i> contains an xts object with the intraday price data of stock <i>i</i> for day <i>t</i> .
cor	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
K	positive integer, slow time scale returns are computed on prices that are <i>K</i> steps apart.
J	positive integer, fast time scale returns are computed on prices that are <i>J</i> steps apart.
K_cov	positive integer, for the extradiagonal covariance elements the slow time scale returns are computed on prices that are <i>K</i> steps apart.
J_cov	positive integer, for the extradiagonal covariance elements the fast time scale returns are computed on prices that are <i>J</i> steps apart.
K_var	vector of positive integers, for the diagonal variance elements the slow time scale returns are computed on prices that are <i>K</i> steps apart.
J_var	vector of positive integers, for the diagonal variance elements the fast time scale returns are computed on prices that are <i>J</i> steps apart.
makePsd	boolean, in case it is TRUE, the positive definite version of rTSCov is returned. FALSE by default.

### Details

The rTSCov requires the tick-by-tick transaction prices. (Co)variances are then computed using log-returns calculated on a rolling basis on stock prices that are  $K$  (slow time scale) and  $J$  (fast time scale) steps apart.

The diagonal elements of the rTSCov matrix are the variances, computed for log-price series  $X$  with  $n$  price observations at times  $\tau_1, \tau_2, \dots, \tau_n$  as follows:

$$\left(1 - \frac{\bar{n}_K}{\bar{n}_J}\right)^{-1} ([X, X]_T^{(K)} - \frac{\bar{n}_K}{\bar{n}_J} [X, X]_T^{(J)})$$

where  $\bar{n}_K = (n - K + 1)/K$ ,  $\bar{n}_J = (n - J + 1)/J$  and

$$[X, X]_T^{(K)} = \frac{1}{K} \sum_{i=1}^{n-K+1} (X_{t_{i+K}} - X_{t_i})^2.$$

The extradiagonal elements of the rTSCov are the covariances. For their calculation, the data is first synchronized by the refresh time method proposed by Harris et al (1995). It uses the function `refreshTime` to collect first the so-called refresh times at which all assets have traded at least once since the last refresh time point. Suppose we have two log-price series:  $X$  and  $Y$ . Let  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_{N_X}^X\}$  and  $\Theta = \{\theta_1, \theta_2, \dots, \theta_{N_Y}^Y\}$  be the set of transaction times of these assets. The first refresh time corresponds to the first time at which both stocks have traded, i.e.  $\phi_1 = \max(\tau_1, \theta_1)$ . The subsequent refresh time is defined as the first time when both stocks have again traded, i.e.  $\phi_{j+1} = \max(\tau_{N_X^X}^X + 1, \theta_{N_Y^Y}^Y + 1)$ . The complete refresh time sample grid is  $\Phi = \{\phi_1, \phi_2, \dots, \phi_{M_N+1}\}$ , where  $M_N$  is the total number of paired returns. The sampling points of asset  $X$  and  $Y$  are defined to be  $t_i = \max\{\tau \in \Gamma : \tau \leq \phi_i\}$  and  $s_i = \max\{\theta \in \Theta : \theta \leq \phi_i\}$ .

Given these refresh times, the covariance is computed as follows:

$$c_N([X, Y]_T^{(K)} - \frac{\bar{n}_K}{\bar{n}_J} [X, Y]_T^{(J)}),$$

where

$$[X, Y]_T^{(K)} = \frac{1}{K} \sum_{i=1}^{M_N-K+1} (X_{t_{i+K}} - X_{t_i})(Y_{s_{i+K}} - Y_{s_i}).$$

Unfortunately, the rTSCov is not always positive semidefinite. By setting the argument `makePsd = TRUE`, the function `makePsd` is used to return a positive semidefinite matrix. This function replaces the negative eigenvalues with zeroes.

### Value

an  $N \times N$  matrix

### Author(s)

Jonathan Cornelissen and Kris Boudt

## References

Harris, F., T. McInish, G. Shoesmith, and R. Wood (1995). Cointegration, error correction, and price discovery on informationally linked security markets. *Journal of Financial and Quantitative Analysis* 30, 563-581. Zhang, L., P. A. Mykland, and Y. Ait-Sahalia (2005). A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association* 100, 1394-1411. Zhang, L. (2011). Estimating covariation: Epps effect, microstructure noise. *Journal of Econometrics* 160, 33-47.

## Examples

```
# Robust Realized two timescales Variance/Covariance

# Univariate:
rvts <- rTSCov(pData = sampleTData$PRICE)
# Note: Prices as input
rvts

# Multivariate:
rcovts <- rTSCov(pData = list(cumsum(1l1tc) + 100, cumsum(sbox) + 100))
# Note: List of prices as input
rcovts
```

---

 RV

*An estimator of realized variance.*

---

## Description

An estimator of realized variance.

## Usage

```
RV(rData)
```

## Arguments

rData            a zoo/xts object containing all returns in period t for one asset.

## Value

numeric

---

salesCondition	<i>Delete entries with abnormal Sale Condition.</i>
----------------	-----------------------------------------------------

---

### Description

Function deletes entries with abnormal Sale Condition: trades where column "COND" has a letter code, except for "E" and "F".

### Usage

```
salesCondition(
  tData,
  validConds = c("", "@", "E", "@E", "F", "FI", "@F", "@FI", "I", "@I")
)
```

### Arguments

tData	an xts or data.table object containing the time series data, with one column named "COND" indicating the Sale Condition.
validConds	a character vector containing valid sales conditions defaults to c(' ', '@', 'E', '@E', 'F', 'FI', '@F', '@FI', 'I', '@I') See details.

### Details

To get more information on the sales conditions, see the NYSE documentation. Section about Daily TAQ Trades File. The current version (as of May 2020) can be found online at [NYSE's webpage](#)

### Value

xts or data.table object depending on input

### Author(s)

Jonathan Cornelissen and Kris Boudt

---

sample5MinPrices	<i>Ten artificial time series for the NYSE trading days during January 2010</i>
------------------	---------------------------------------------------------------------------------

---

**Description**

Ten simulated price series for the 19 trading days in January 2010:

Ten hypothetical price series were simulated according to the factor diffusion process discussed in Barndorff-Nielsen et al. We assume that prices are only observed when a transaction takes place. The intensity of transactions follows a Poisson process and consequently, the inter-transaction times are exponentially distributed. Therefore, we generated the inter transaction times of the price series by an independent exponential distributions with  $\lambda = 0.1$ , which we keep constant over time. This means we expect one transaction every ten seconds. In a final step, the time series were aggregated to the 5-minute frequency by previous tick aggregation.

**Usage**

```
sample5MinPrices
```

**Format**

xts object

**References**

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde and N. Shephard (2011). Multivariate realised kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. *Journal of Econometrics*, 162, 149-169.

---

sample5MinPricesJumps *Ten artificial time series (including jumps) for the NYSE trading days during January 2010*

---

**Description**

Ten simulated price series for the 19 trading days in January 2010: Ten hypothetical price series were simulated according to the factor diffusion process discussed in Barndorff-Nielsen et al. On top of this process we added a jump process, with jump occurrences governed by the Poisson process with 1 expected jump per day and jump magnitude modelled as in Boudt et al. (2008). We assume that prices are only observed when a transaction takes place. The intensity of transactions follows a Poisson process and consequently, the inter transaction times are exponentially distributed. Therefore, we generated the inter transaction times of the price series by an independent exponential distributions with  $\lambda = 0.1$ , which we keep constant over time. This means we expect one transaction every ten seconds. In a final step, the time series were aggregated to the 5-minute frequency by previous tick aggregation.

**Usage**

```
sample5MinPricesJumps
```

**Format**

xts object

## References

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde and N. Shephard (2011). Multivariate realised kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. *Journal of Econometrics*, 162, 149-169.

Boudt, K., C. Croux, and S. Laurent (2008). Outlyingness weighted covariation. Mimeo.

---

sampleQData

*Sample of cleaned quotes for stock XXX for 1 day*

---

## Description

An xts object containing the raw quotes for the imaginary stock XXX for 1 day, in the typical NYSE TAQ database format. This is the cleaned version of the data sample [sampleQDataRaw](#), using `quotesCleanup`.

## Usage

```
sampleQData
```

## Format

xts object

## Examples

```
## Not run:
#The code to create sampleQData from raw data is
sampleQData <- quotesCleanup(
  qDataRaw = sampleQDataRaw, exchanges = "N", type = "standard",
  report = FALSE)[, c("SYMBOL", "BID", "BIDSIZ", "OFR", "OFRSIZ", "MODE")]
# These are the historically included columns.

## End(Not run)
```

---

sampleQDataMicroseconds

*Sample of cleaned quotes for stock XXX for 2 days measured in microseconds*

---

## Description

A data.table object containing the raw quotes for the imaginary stock XXX for 2 days. This is the cleaned version of the data sample [sampleQDataRawMicroseconds](#), using `quotesCleanup`.

**Usage**

```
sampleQDataMicroseconds
```

**Format**

```
data.table object
```

**Examples**

```
## Not run:
# The code to create the sampleQDataMicroseconds dataset from raw data is
sampleQDataMicroseconds <- quotesCleanup(qDataRaw = sampleQDataRawMicroseconds,
                                          exchanges = "N", type = "standard", report = FALSE)

## End(Not run)
```

---

```
sampleQDataRaw           Sample of raw quotes for stock XXX for 1 day
```

---

**Description**

An imaginary xts object containing the raw quotes for stock XXX for 1 day, in the typical NYSE TAQ database format.

**Usage**

```
sampleQDataRaw
```

**Format**

```
xts object
```

---

```
sampleQDataRawMicroseconds
           Sample of raw quotes for stock XXX for 2 days measured in microseconds
```

---

**Description**

An imaginary xts object containing the raw quotes for stock XXX for 2 days, in the typical NYSE TAQ database format.

**Usage**

```
sampleQDataRawMicroseconds
```

**Format**

```
data.table object
```

---

sampleReal5MinPrices    *Sample of imaginary price data for 61 days*

---

**Description**

An xts object containing the 5-min aggregated imaginary price series for the trading days between 2005-03-04 and 2005-06-01.

**Usage**

```
sampleReal5MinPrices
```

**Format**

xts object

---

sampleReturns5Min    *Sample returns data*

---

**Description**

EUR/USD returns from January to September 2004

**Usage**

```
sampleReturns5Min
```

**Format**

A large xts object.

---

sampleTData    *Sample of cleaned trades for stock XXX for 1 day*

---

**Description**

An xts object containing the trades for the imaginary stock XXX for 1 day, in the typical NYSE TAQ database format. This is the cleaned version of the data sample `sampleTDataRaw`, using `tradesCleanup` and `tradesCleanupUsingQuotes`.

**Usage**

```
sampleTData
```



**Format**

A large xts object.

**Examples**

```
## Not run:
#The code to create sampleTData from raw data is
tradesAfterFirstCleaning <- tradesCleanup(tDataRaw = sampleTDataRaw,
                                         exchanges = "N", report = FALSE)

cleanedQuotes <- quotesCleanup(qDataRaw = sampleQDataRaw, exchanges = "N",
                              type = "standard", report = FALSE)
sampleTData <- tradesCleanupUsingQuotes(tData = tradesAfterFirstCleaning,
                                       qData = cleanedQuotes, lagQuotes = 2)
sampleTData <- sampleTData[,c("SYMBOL", "EX", "PRICE", "SIZE", "COND", "CORR", "G127")]
# These are the historically included columns.

## End(Not run)
```

---

sampleTDataMicroseconds

*Sample of cleaned trades for stock XXX for 2 days*

---

**Description**

An `data.table` object containing the trades for the imaginary stock XXX for 2 days, in the typical NYSE TAQ database format. This is the cleaned version of the data sample [sampleTDataRawMicroseconds](#), using `tradesCleanup`.

**Usage**

```
sampleTDataMicroseconds
```

**Format**

A `data.table` object.

**Examples**

```
## Not run:
# The code to create the sampleTDataMicroseconds dataset from raw data is
sampleQDataMicroseconds <- quotesCleanup(qDataRaw = sampleQDataRawMicroseconds,
                                         exchanges = "N", type = "standard", report = FALSE)

tradesAfterFirstCleaning <- tradesCleanup(tDataRaw = sampleTDataRawMicroseconds,
                                         exchanges = "N", report = FALSE)

sampleTDataMicroseconds <- tradesCleanupUsingQuotes(
```

```

tData = tradesAfterFirstCleaning,
qData = sampleQDataMicroseconds,
lagQuotes = 0)[, c("DT", "SYMBOL", "PRICE", "SIZE")]
# Only some columns are included. These are the ones that were historically included.

## End(Not run)

```

---

sampleTDataRaw

*Sample of raw trades for stock XXX for 1 day*

---

### Description

An imaginary xts object containing the raw trades for stock XXX for 1 day, in the typical NYSE TAQ database format.

### Usage

```
sampleTDataRaw
```

### Format

A large xts object.

---

sampleTDataRawMicroseconds

*Sample of raw trades for stock XXX for 2 days*

---

### Description

An imaginary data.table object containing the raw trades for stock XXX for 2 days, in the typical NYSE TAQ database format.

### Usage

```
sampleTDataRawMicroseconds
```

### Format

A data.table object.

---

sbux	<i>Starbucks Data</i>
------	-----------------------

---

**Description**

Tick data for Starbucks 2011/07/01, cleaned with tradesCleanup.

**Usage**

```
sbux
```

**Format**

```
xts object
```

**Examples**

```
data(sbux)
plot(sbux)
```

---

selectExchange	<i>Retain only data from a single stock exchange</i>
----------------	------------------------------------------------------

---

**Description**

Function returns an xts object containing the data of only 1 stock exchange.

**Usage**

```
selectExchange(data, exch = "N")
```

**Arguments**

data	an xts or data.table object containing the time series data. The object should have a column "EX", indicating the exchange by its symbol.
exch	The (vector of) symbol(s) of the stock exchange(s) that should be selected. By default the NYSE is chosen (exch = "N"). Other exchange symbols are: <ul style="list-style-type: none"> <li>• A: AMEX</li> <li>• N: NYSE</li> <li>• B: Boston</li> <li>• P: Arca</li> <li>• C: NSX</li> <li>• T/Q: NASDAQ</li> <li>• D: NASD ADF and TRF</li> </ul>

- X: Philadelphia
- I: ISE
- M: Chicago
- W: CBOE
- Z: BATS

**Value**

xts or data.table object depending on input

**Author(s)**

Jonathan Cornelissen and Kris Boudt

---

SP500RM

*SP500 Realized Measures calculated with 5 minute sampling*

---

**Description**

Realized measures from the SP500 index from April 1997 to August 2013.

**Usage**

SP500RM

**Format**

A large xts object.

**Source**

<http://public.econ.duke.edu/~ap172/code.html>

**References**

Bollerslev, T., A. J. Patton, and R. Quaedvlieg, 2016, Exploiting the Errors: A Simple Approach for Improved Volatility Forecasting, *Journal of Econometrics*, 192, 1-18.

---

spotDrift	<i>Spot Drift Estimation</i>
-----------	------------------------------

---

## Description

Function used to estimate the spot drift of intraday (tick) stock prices/returns

## Usage

```
spotDrift(
  data,
  method = "driftMean",
  ...,
  on = "minutes",
  k = 5,
  marketOpen = "09:30:00",
  marketClose = "16:00:00",
  tz = "GMT"
)
```

## Arguments

data	Can be one of two input types, xts or data.table. It is assumed that the input comprises prices in levels.
method	Which method to be used to estimate the spot-drift. Currently, three methods are available, rolling mean and median as well as the kernel method of Christensen et al. 2018. The kernel is a left hand exponential kernel that will weigh newer observations more heavily than older observations.
...	Additional arguments for the individual methods. See details
on	What time-frame should the estimator be applied? Accepted inputs are "milliseconds", "seconds" and "secs" for seconds, "minutes" and "mins" for minutes, and "hours" for hours. Standard is minutes
k	How often should the estimation take place? If k is 5 the estimation will be done every fifth unit of on.
marketOpen	Opening time of the market, standard is "09:30:00"
marketClose	Closing time of the market, standard is "16:00:00"
tz	Time zone, standard is "GMT"

## Details

The additional arguments for the mean and median methods are: periods for the rolling window length which is 5 by standard and align to allow for control of the alignment, should one wish to do so, the standard is "right". For the kernel mean estimator, the arguments meanBandwidth can be used to control the bandwidth of the drift estimator and the preAverage argument, which can be used to control the pre-averaging horizon. These arguments default to 300 and 5 respectively.

**Value**

An object of class "spotDrift" containing at least the estimated spot drift process. Input on what this class should contain and methods for it is welcome.

**Author(s)**

Emil Sjoerup

**References**

Christensen, Oomen and Reno (2018): The drift burst hypothesis (working paper) <DOI:10.2139/ssrn.2842535>.

**Examples**

```
# Example 1: Rolling mean and median estimators for 2 days
meandrift <- spotDrift(data = sampleTDataMicroseconds, k = 1, tz = "EST")
mediandrift <- spotDrift(data = sampleTDataMicroseconds, method = "driftMedian",
                        on = "seconds", k = 30, tz = "EST")

plot(meandrift)
plot(mediandrift)

# Example 2: Kernel based estimator for one day
price <- sampleTData$PRICE
storage.mode(price) <- "numeric"
#kerneldrift <- spotDrift(price, method = "driftKernel", on = "minutes", k = 1)
#plot(kerneldrift)
```

---

spotVol

*Spot volatility estimation*

---

**Description**

Spot volatility estimation

**Usage**

```
spotVol(
  data,
  method = "detPer",
  ...,
  on = "minutes",
  k = 5,
  marketOpen = "09:30:00",
  marketClose = "16:00:00",
  tz = "GMT"
)
```

**Arguments**

data	Can be one of two input types, <code>xts</code> or <code>data.table</code> . It is assumed that the input comprises prices in levels. Irregularly spaced observations are allowed. They will be aggregated to the level specified by parameters <code>on</code> and <code>k</code> .
method	specifies which method will be used to estimate the spot volatility. Options include <code>"detPer"</code> and <code>"stochper"</code> . See 'Details'.
...	method-specific parameters (see 'Details').
on	string indicating the time scale in which <code>k</code> is expressed. Possible values are: <code>"secs"</code> , <code>"seconds"</code> , <code>"mins"</code> , <code>"minutes"</code> , <code>"hours"</code> .
k	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate an <code>xts</code> object to the 5 minute frequency, set <code>k = 5</code> and <code>on = "minutes"</code> .
marketOpen	the market opening time. This should be in the time zone specified by <code>tz</code> . By default, <code>marketOpen = "09:30:00"</code> .
marketClose	the market closing time. This should be in the time zone specified by <code>tz</code> . By default, <code>marketClose = "16:00:00"</code> .
tz	string specifying the time zone to which the times in <code>data</code> and/or <code>marketOpen</code> / <code>marketClose</code> belong. Default = <code>"GMT"</code> .

**Details**

The following estimation methods can be specified in `method`:

**Deterministic periodicity method ("detPer")**

Parameters:

dailyvol	A string specifying the estimation method for the daily component $s_t$ . Possible values are <code>"bipower"</code> , <code>"rv"</code> , <code>"periodic"</code> .
periodicvol	A string specifying the estimation method for the component of intraday volatility, that depends in a deterministic way on the time of day.
P1	A positive integer corresponding to the number of cosinus terms used in the flexible Fourier specification of the periodic component.
P2	Same as P1, but for the sinus terms. Default = 5.
dummies	Boolean: in case it is TRUE, the parametric estimator of periodic standard deviation specifies the periodicity function.

Outputs (see 'Value' for a full description of each component):

- spot
- daily
- periodic

The spot volatility is decomposed into a deterministic periodic factor  $f_i$  (identical for every day in the sample) and a daily factor  $s_t$  (identical for all observations within a day). Both components are then estimated separately. For more details, see Taylor and Xu (1997) and Andersen and Bollerslev (1997). The jump robust versions by Boudt et al. (2011) have also been implemented.

**Stochastic periodicity method ("stochper") Parameters:**

P1	A positive integer corresponding to the number of cosinus terms used in the flexible Fourier specification of the periodic component.
P2	Same as P1, but for the sinus terms. Default = 5.
init	A named list of initial values to be used in the optimization routine ( <code>"BFGS"</code> in <code>optim</code> ). Default = <code>list(sigma = 0.001)</code> .
control	A list of options to be passed down to <code>optim</code> .

Outputs (see 'Value' for a full description of each component):

- spot
- par

This method by Beltratti and Morana (2001) assumes the periodicity factor to be stochastic. The spot volatility estimation is split into four components: a random walk, an autoregressive process, a stochastic cyclical process and a deterministic cyclical process. The model is estimated using a quasi-maximum likelihood method based on the Kalman Filter. The package FKF is used to apply the Kalman filter. In addition to the spot volatility estimates, all parameter estimates are returned.

#### **Nonparametric filtering ("kernel")**

Parameters:

type	String specifying the type of kernel to be used. Options include "gaussian", "epanechnikov", "beta". Default =
h	Scalar or vector specifying bandwidth(s) to be used in kernel. If h is a scalar, it will be assumed equal throughout the
est	String specifying the bandwidth estimation method. Possible values include "cv", "quarticity". Method "cv" equ
lower	Lower bound to be used in bandwidth optimization routine, when using cross-validation method. Default is $0.1n^{-0.2}$
upper	Upper bound to be used in bandwidth optimization routine, when using cross-validation method. Default is $n^{-0.2}$ .

Outputs (see 'Value' for a full description of each component):

- spot
- par

This method by Kristensen (2010) filters the spot volatility in a nonparametric way by applying kernel weights to the standard realized volatility estimator. Different kernels and bandwidths can be used to focus on specific characteristics of the volatility process.

Estimation results heavily depend on the bandwidth parameter  $h$ , so it is important that this parameter is well chosen. However, it is difficult to come up with a method that determines the optimal bandwidth for any kind of data or kernel that can be used. Although some estimation methods are provided, it is advised that you specify  $h$  yourself, or make sure that the estimation results are appropriate.

One way to estimate  $h$ , is by using cross-validation. For each day in the sample,  $h$  is chosen as to minimize the Integrated Square Error, which is a function of  $h$ . However, this function often has multiple local minima, or no minima at all ( $h \rightarrow \infty$ ). To ensure a reasonable optimum is reached, strict boundaries have to be imposed on  $h$ . These can be specified by lower and upper, which by default are  $0.1n^{-0.2}$  and  $n^{-0.2}$  respectively, where  $n$  is the number of observations in a day.

When using the method "kernel", in addition to the spot volatility estimates, all used values of the bandwidth  $h$  are returned.

#### **Piecewise constant volatility ("piecewise")**

Parameters:

type	String specifying the type of test to be used. Options include "MDa", "MDb", "DM". See Fried (2012) for details. De
m	Number of observations to include in reference window. Default = 40.
n	Number of observations to include in test window. Default = 20.
alpha	Significance level to be used in tests. Note that the test will be executed many times (roughly equal to the total num





- spot
- RM
- lookBackPeriod

This method returns the estimates of the spot volatility, a string containing the realized measure used, and the lookBackPeriod.

### (Non-overlapping) Pre-Averaged Realized Measures ("PARM")

Parameters:

RM String denoting which realized measure to use to estimate the local volatility. Possible values are: "bipow", "kernel", "stochper", "detPer", "par", "cp".  
lookBackPeriod positive integer denoting the amount of sub-sampled returns to use for the estimation of the local volatility.

Outputs (see 'Value' for a full description of each component):

- spot
- RM
- lookBackPeriod
- kn

### Value

A spotVol object, which is a list containing one or more of the following outputs, depending on the method used:

spot

An xts or matrix object (depending on the input) containing spot volatility estimates  $\sigma_{t,i}$ , reported for each interval  $i$  between marketOpen and marketClose for every day  $t$  in data. The length of the intervals is specified by k and on. Methods that provide this output: All.

daily An xts or numeric object (depending on the input) containing estimates of the daily volatility levels for each day  $t$  in data, if the used method decomposed spot volatility into a daily and an intraday component. Methods that provide this output: "detPer".

periodic

An xts or numeric object (depending on the input) containing estimates of the intraday periodicity factor for each day interval  $i$  between marketOpen and marketClose, if the spot volatility was decomposed into a daily and an intraday component. If the output is in xts format, this periodicity factor will be dated to the first day of the input data, but it is identical for each day in the sample. Methods that provide this output: "detPer".

par

A named list containing parameter estimates, for methods that estimate one or more parameters. Methods that provide this output: "stochper", "kernel".

cp

A vector containing the change points in the volatility, i.e. the observation indices after which the volatility level changed, according to the applied tests. The vector starts with a 0. Methods that provide this output: "piecewise".

`ugarchfit`

A `ugarchfit` object, as used by the `rugarch` package, containing all output from fitting the GARCH model to the data. Methods that provide this output: `"garch"`.

The `spotVol` function offers several methods to estimate spot volatility and its intraday seasonality, using high-frequency data. It returns an object of class `spotVol`, which can contain various outputs, depending on the method used. See 'Details' for a description of each method. In any case, the output will contain the spot volatility estimates.

The input can consist of price data or return data, either tick by tick or sampled at set intervals. The data will be converted to equispaced high-frequency returns  $r_{t,i}$  (read: the  $i$ th return on day  $t$ ).

**References**

Andersen, T. G. and T. Bollerslev (1997). Intraday periodicity and volatility persistence in financial markets. *Journal of Empirical Finance* 4, 115-158.

Beltratti, A. and C. Morana (2001). Deterministic and stochastic methods for estimation of intraday seasonal components with high frequency data. *Economic Notes* 30, 205-234.

Boudt K., Croux C. and Laurent S. (2011). Robust estimation of intraweek periodicity in volatility and jump detection. *Journal of Empirical Finance* 18, 353-367.

Fried, Roland (2012). On the online estimation of local constant volatilities. *Computational Statistics and Data Analysis* 56, 3080-3090.

Kristensen, Dennis (2010). Nonparametric filtering of the realized spot volatility: A kernel-based approach. *Econometric Theory* 26, 60-93.

Taylor, S. J. and X. Xu (1997). The incremental volatility information in one million foreign exchange quotations. *Journal of Empirical Finance* 4, 317-340.

**Examples**

```
# Default method, deterministic periodicity

vol1 <- spotVol(sampleReal5MinPrices)
plot(vol1)

# Compare to stochastic periodicity
## Not run:
init <- list(sigma = 0.03, sigma_mu = 0.005, sigma_h = 0.007,
             sigma_k = 0.06, phi = 0.194, rho = 0.986, mu = c(1.87, -0.42),
             delta_c = c(0.25, -0.05, -0.2, 0.13, 0.02),
             delta_s = c(-1.2, 0.11, 0.26, -0.03, 0.08))

# next method will take around 110 iterations
vol2 <- spotVol(sampleReal5MinPrices, method = "stochper", init = init)
plot(as.numeric(vol1$spot[1:780]), type="l")
lines(as.numeric(vol2$spot[1:780]), col="red")
legend("topright", c("detPer", "stochper"), col = c("black", "red"), lty=1)
## End(Not run)

# Various kernel estimates
## Not run:
```

```

h1 <- bw.nrd0((1:nrow(sampleReal5MinPrices))*(5*60))
vol3 <- spotVol(sampleReal5MinPrices, method = "kernel", h = h1)
vol4 <- spotVol(sampleReal5MinPrices, method = "kernel", est = "quarticity")
vol5 <- spotVol(sampleReal5MinPrices, method = "kernel", est = "cv")
plot(vol3, length = 2880)
lines(as.numeric(t(vol4$spot))[1:2880], col = "red")
lines(as.numeric(t(vol5$spot))[1:2880], col = "blue")
legend("topright", c("h = simple estimate", "h = quarticity corrected",
                    "h = crossvalidated"), col = c("black", "red", "blue"), lty=1)

## End(Not run)

# Piecewise constant volatility
## Not run:
vol6 <- spotVol(sampleReal5MinPrices, method = "piecewise", m = 200, n = 100,
                online = FALSE)
plot(vol6)
## End(Not run)

# Compare regular GARCH(1,1) model to eGARCH, both with external regressors
## Not run:
vol7 <- spotVol(sampleReal5MinPrices, method = "garch", model = "sGARCH")
vol8 <- spotVol(sampleReal5MinPrices, method = "garch", model = "eGARCH")
plot(as.numeric(t(vol7$spot)), type = "l")
lines(as.numeric(t(vol8$spot)), col = "red")
legend("topleft", c("GARCH", "eGARCH"), col = c("black", "red"), lty=1)

## End(Not run)

```

---

tradesCleanup

*Cleans trade data*


---

## Description

This is a wrapper function for cleaning the trade data of all stock data inside the folder `dataSource`. The result is saved in the folder `dataDestination`.

In case you supply the argument `"rawtData"`, the on-disk functionality is ignored. The function returns a vector indicating how many trades were removed at each cleaning step in this case. and the function returns an `xts` or `data.table` object.

The following cleaning functions are performed sequentially: [noZeroPrices](#), [selectExchange](#), [salesCondition](#), [mergeTradesSameTimestamp](#).

Since the function [rmTradeOutliersUsingQuotes](#) also requires cleaned quote data as input, it is not incorporated here and there is a separate wrapper called [tradesCleanupUsingQuotes](#).

## Usage

```
tradesCleanup(
  dataSource = NULL,
```

```

dataDestination = NULL,
exchanges,
tDataRaw = NULL,
report = TRUE,
selection = "median",
validConds = c("", "@", "E", "@E", "F", "FI", "@F", "@FI", "I", "@I"),
saveAsXTS = TRUE,
tz = "EST"
)

```

### Arguments

dataSource	character indicating the folder in which the original data is stored.
dataDestination	character indicating the folder in which the cleaned data is stored.
exchanges	vector of stock exchange symbols for all data in dataSource, e.g. exchanges = c("T","N") retrieves all stock market data from both NYSE and NASDAQ. The possible exchange symbols are: <ul style="list-style-type: none"> <li>• A: AMEX</li> <li>• N: NYSE</li> <li>• B: Boston</li> <li>• P: Arca</li> <li>• C: NSX</li> <li>• T/Q: NASDAQ</li> <li>• D: NASD ADF and TRF</li> <li>• X: Philadelphia</li> <li>• I: ISE</li> <li>• M: Chicago</li> <li>• W: CBOE</li> <li>• Z: BATS</li> </ul>
tDataRaw	xts object containing (for ONE stock only) raw trade data. This argument is NULL by default. Enabling it means the arguments from, to, dataSource and dataDestination will be ignored. (only advisable for small chunks of data)
report	boolean and TRUE by default. In case it is true the function returns (also) a vector indicating how many trades remained after each cleaning step.
selection	argument to be passed on to the cleaning routine <a href="#">mergeTradesSameTimestamp</a> . The default is "median".
validConds	character vector containing valid sales conditions. Passed through to <a href="#">salesCondition</a> .
saveAsXTS	indicates whether data should be saved in xts format instead of data.table when using on-disk functionality. TRUE by default.
tz	timezone to use

**Value**

For each day an xts or data.table object is saved into the folder of that date, containing the cleaned data. This procedure is performed for each stock in "ticker". The function returns a vector indicating how many trades remained after each cleaning step.

In case you supply the argument "rawtData", the on-disk functionality is ignored and the function returns a list with the cleaned trades as xts object (see examples).

**Author(s)**

Jonathan Cornelissen and Kris Boudt

**References**

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2009). Realized kernels in practice: Trades and quotes. *Econometrics Journal* 12, C1-C32.

Brownlees, C.T. and Gallo, G.M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51, pp. 2232-2245.

**Examples**

```
# Consider you have raw trade data for 1 stock for 2 days
head(sampleTDataRawMicroseconds)
dim(sampleTDataRawMicroseconds)
tDataAfterFirstCleaning <- tradesCleanup(tDataRaw = sampleTDataRaw, exchanges = list("N"))
tDataAfterFirstCleaning$report
dim(tDataAfterFirstCleaning$tData)

#In case you have more data it is advised to use the on-disk functionality
#via "dataSource" and "dataDestination" arguments
```

---

```
tradesCleanupUsingQuotes
```

*Perform a final cleaning procedure on trade data*

---

**Description**

Function performs cleaning procedure [rmTradeOutliersUsingQuotes](#) for the trades of all stocks data in "dataDestination". Note that preferably the input data for this function is trade and quote data cleaned by respectively e.g. [tradesCleanup](#) and [quotesCleanup](#).

**Usage**

```
tradesCleanupUsingQuotes(
  tradeDataSource = NULL,
  quoteDataSource = NULL,
  dataDestination = NULL,
```

```

    tData = NULL,
    qData = NULL,
    lagQuotes = 2
)

```

### Arguments

tradeDataSource	character indicating the folder in which the original trade data is stored.
quoteDataSource	character indicating the folder in which the original quote data is stored.
dataDestination	character indicating the folder in which the cleaned data is stored, folder of dataSource by default.
tData	data.table or xts object containing (ONE day and for ONE stock only) trade data cleaned by <code>tradesCleanup</code> . This argument is NULL by default. Enabling it, means the arguments from, to, dataSource and dataDestination will be ignored. (only advisable for small chunks of data)
qData	data.table or xts object containing (ONE day and for ONE stock only) cleaned quote data. This argument is NULL by default. Enabling it means the arguments from, to, dataSource, dataDestination will be ignored. (only advisable for small chunks of data)
lagQuotes	passed through to <code>rmTradeOutliersUsingQuotes</code> . <a href="#">rmTradeOutliersUsingQuotes</a>

### Details

In case you supply the arguments "tData" and "qData", the on-disk functionality is ignored and the function returns cleaned trades as a data.table or xts object (see examples).

When using the on-disk functionality and tradeDataSource and quoteDataSource are the same, the quote files are all files in the folder that contains 'quote', and the rest are treated as containing trade data.

### Value

For each day an xts object is saved into the folder of that date, containing the cleaned data.

### Author(s)

Jonathan Cornelissen, Kris Boudt and Onno Kleen.

### References

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde, and N. Shephard (2009). Realized kernels in practice: Trades and quotes. *Econometrics Journal* 12, C1-C32.

Brownlees, C.T. and Gallo, G.M. (2006). Financial econometric analysis at ultra-high frequency: Data handling concerns. *Computational Statistics & Data Analysis*, 51, pages 2232-2245.

**Examples**

```
# Consider you have raw trade data for 1 stock for 2 days
tDataAfterFirstCleaning <- tradesCleanup(tDataRow = sampleTDataRowMicroseconds,
                                         exchanges = "N", report = FALSE)

#
qData <- quotesCleanup(qDataRow = sampleQDataRowMicroseconds,
                      exchanges = "N", report = FALSE)
dim(tDataAfterFirstCleaning)
tDataAfterFinalCleaning <-
  tradesCleanupUsingQuotes(qData = qData[as.Date(DT) == "2018-01-02"],
                          tData = tDataAfterFirstCleaning[as.Date(DT) == "2018-01-02"])
dim(tDataAfterFinalCleaning)
#In case you have more data it is advised to use the on-disk functionality
#via the "tradeDataSource", "quoteDataSource", and "dataDestination" arguments
```



# Index

- \* **AJumpTest**
  - AJumpTest, [9](#)
- \* **BNSjumpTest**
  - BNSjumpTest, [13](#)
- \* **Drift**
  - spotDrift, [101](#)
- \* **JOjumpTest**
  - JOjumpTest, [35](#)
- \* **RV**
  - RV, [91](#)
- \* **autocorrelation**
  - ReMeDI, [65](#)
- \* **autocovariance**
  - ReMeDI, [65](#)
- \* **cleaning**
  - autoSelectExchangeQuotes, [11](#)
  - autoSelectExchangeTrades, [12](#)
  - exchangeHoursOnly, [17](#)
  - mergeQuotesSameTimestamp, [46](#)
  - mergeTradesSameTimestamp, [47](#)
  - noZeroPrices, [52](#)
  - noZeroQuotes, [53](#)
  - quotesCleanup, [53](#)
  - rmLargeSpread, [70](#)
  - rmNegativeSpread, [70](#)
  - rmOutliersQuotes, [71](#)
  - rmTradeOutliersUsingQuotes, [74](#)
  - selectExchange, [99](#)
  - tradesCleanup, [108](#)
  - tradesCleanupUsingQuotes, [110](#)
- \* **datasets**
  - lltc, [41](#)
  - realizedLibrary, [64](#)
  - sample5MinPrices, [92](#)
  - sample5MinPricesJumps, [93](#)
  - sampleQData, [94](#)
  - sampleQDataMicroseconds, [94](#)
  - sampleQDataRaw, [95](#)
  - sampleQDataRawMicroseconds, [95](#)
  - sampleReal5MinPrices, [96](#)
  - sampleReturns5Min, [96](#)
  - sampleTData, [96](#)
  - sampleTDataMicroseconds, [97](#)
  - sampleTDataRaw, [98](#)
  - sampleTDataRawMicroseconds, [98](#)
  - sbux, [99](#)
  - SP500RM, [100](#)
- \* **data**
  - aggregateQuotes, [4](#)
  - aggregateTrades, [6](#)
  - aggregateTS, [7](#)
  - getAlphaVantageData, [18](#)
  - makePsd, [42](#)
  - matchTradesQuotes, [43](#)
  - refreshTime, [64](#)
- \* **forecasting**
  - HARmodel, [25](#)
- \* **highfrequency**
  - AJumpTest, [9](#)
  - BNSjumpTest, [13](#)
  - ivInference, [33](#)
  - JOjumpTest, [35](#)
  - medRQ, [44](#)
  - MRC, [50](#)
  - rBeta, [58](#)
  - rKurt, [69](#)
  - rMPV, [72](#)
  - rQPVar, [77](#)
  - rQuar, [78](#)
  - rSkew, [83](#)
  - rSV, [85](#)
  - rTPVar, [87](#)
  - RV, [91](#)
- \* **ivInference**
  - ivInference, [33](#)
- \* **leaning**
  - salesCondition, [92](#)
- \* **liquidity**

- getTradeDirection, 24
  - \* **manipulation**
    - aggregateQuotes, 4
    - aggregateTrades, 6
    - aggregateTS, 7
    - makePsd, 42
    - matchTradesQuotes, 43
    - refreshTime, 64
  - \* **medRQ**
    - medRQ, 44
  - \* **microstructure**
    - ReMeDI, 65
  - \* **noise**
    - ReMeDI, 65
  - \* **preaveraging**
    - MRC, 50
  - \* **rBeta**
    - rBeta, 58
  - \* **rKurt**
    - rKurt, 69
  - \* **rMPV**
    - rMPV, 72
  - \* **rQPVar**
    - rQPVar, 77
  - \* **rQuar**
    - rQuar, 78
  - \* **rSV**
    - rSV, 85
  - \* **rSkew**
    - rSkew, 83
  - \* **rTPVar**
    - rTPVar, 87
  - \* **volatility**
    - listAvailableKernels, 40
    - medRV, 45
    - minRV, 49
    - rAVGCov, 57
    - rBPCov, 60
    - rCov, 62
    - rHYCov, 66
    - rKernelCov, 67
    - rOWCov, 75
    - rRTSCov, 79
    - rSemiCov, 82
    - rThresholdCov, 86
    - rTSCov, 89
- aggregateQuotes, 4, 7
- aggregateTrades, 6, 7
- aggregateTS, 7
- AJjumpTest, 9
- autoSelectExchangeQuotes, 11
- autoSelectExchangeTrades, 12
- BNSjumpTest, 13
- businessTimeAggregation, 16
- exchangeHoursOnly, 17
- getAlphaVantageData, 18
- getLiquidityMeasures, 19
- getPrice, 23
- getTradeDirection, 24
- HARmodel, 25
- hasQty, 28
- HEAVYmodel, 29
- highfrequency (highfrequency-package), 4
- highfrequency-package, 4
- intradayJumpTest, 31
- ivInference, 33
- JOjumpTest, 35
- knChooseReMeDI, 37
- leadLag, 39
- listAvailableKernels, 40, 68
- listCholCovEstimators, 41
- lltc, 41
- lm, 26
- makePsd, 42, 81, 90
- makeReturns, 43
- matchTradesQuotes, 24, 43
- medRQ, 44
- medRV, 45
- mergeQuotesSameTimestamp, 46, 53, 54
- mergeTradesSameTimestamp, 47, 108, 109
- minRQ, 48
- minRV, 49
- MRC, 50
- noZeroPrices, 52, 108
- noZeroQuotes, 53, 53
- quotesCleanup, 53, 110
- rankJumpTest, 55

- rAVGCov, [57](#)
- rBeta, [58](#)
- rBPCov, [60](#), [76](#)
- rCholCov, [61](#)
- rCov, [62](#)
- realizedLibrary, [64](#)
- refreshTime, [64](#), [80](#), [90](#)
- ReMeDI, [65](#)
- rHYCov, [66](#)
- rKernelCov, [67](#)
- rKurt, [69](#)
- rmLargeSpread, [70](#)
- rmNegativeSpread, [70](#)
- rmOutliersQuotes, [53](#), [54](#), [71](#)
- rMPV, [72](#)
- rmTradeOutliers, [74](#)
- rmTradeOutliersUsingQuotes, [74](#), [108](#), [110](#),  
[111](#)
- rOWCov, [75](#), [79](#), [86](#)
- rQPVar, [77](#)
- rQuar, [78](#)
- rRTSCov, [79](#)
- rSemiCov, [82](#)
- rSkew, [83](#)
- rSV, [85](#)
- rThresholdCov, [79](#), [86](#)
- rTPVar, [87](#)
- RTQ, [88](#)
- rTSCov, [89](#)
- RV, [91](#)
  
- salesCondition, [92](#), [108](#), [109](#)
- sample5MinPrices, [92](#)
- sample5MinPricesJumps, [93](#)
- sampleQData, [4](#), [94](#)
- sampleQDataMicroseconds, [94](#)
- sampleQDataRaw, [94](#), [95](#)
- sampleQDataRawMicroseconds, [94](#), [95](#)
- sampleReal5MinPrices, [96](#)
- sampleReturns5Min, [96](#)
- sampleTData, [96](#)
- sampleTDataMicroseconds, [97](#)
- sampleTDataRaw, [96](#), [98](#)
- sampleTDataRawMicroseconds, [97](#), [98](#)
- sbux, [99](#)
- selectExchange, [53](#), [99](#), [108](#)
- SP500RM, [100](#)
- spotDrift, [32](#), [101](#)
- spotVol, [16](#), [32](#), [102](#)
- tradesCleanup, [108](#), [110](#), [111](#)
- tradesCleanupUsingQuotes, [108](#), [110](#)