Package ‘highfrequency’

February 20, 2015

Version 0.4

Date 2014-11-25

Title Tools For Highfrequency Data Analysis

Author Kris Boudt [aut, cre], Jonathan Cornelissen [aut], Scott Payseur [aut], Giang Nguyen [ctb], Maarten Schermer [ctb]

Maintainer Kris Boudt

`<Kris.Boudt@econ.kuleuven.be>`

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

License GPL (>= 2)

Depends R (>= 2.12.0), xts, zoo

Suggests robustbase, cubature, mvtnorm, chron, timeDate, quantmod, MASS, sandwich, numDeriv, FKF, BMS, rugarch

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-11-28 11:27:28

R topics documented:

`highfrequency-package` *Tools For Highfrequency Data Analysis*

Description

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

Details

Package:	highfrequency
Type:	Package
Version:	0.4
Date:	2014-08-31
License:	GPL (>= 2)

Author(s)

Jonathan Cornelissen, Kris Boudt, Scott Payseur

Maintainer: Kris Boudt <Kris.Boudt@econ.kuleuven.be>

Contributors: Giang Nguyen, Maarten Schermers.

Thanks: A special thanks for additional contributions from Chris Blakely, Brian Peterson, Eric Zivot and GSoC.

Examples

```
# see users manual
```

`aggregatePrice` *Aggregate a time series but keep first and last observation*

Description

Function returns new time series as xts object where first observation is always the opening price and subsequent observations are the closing prices over the interval with as endpoint the timestamp of the result.

Usage

```
aggregatePrice(ts,FUN = previoustick,on="minutes",k=1,
                   marketopen = "09:30:00", marketclose = "16:00:00", tz = "GMT")
```

Arguments

`ts`	xts object to be aggregated, containing the intraday price series of a stock for one day.
`FUN`	function to apply over each interval. By default, previous tick aggregation is done.
`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 a xts 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".

Value

An xts object containing the aggregated time series.

Details

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

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, excluding the value at 09:35:00 itself. An exception is marketclose (i.e. 16:00:00 ET by default), where the observation at 16:00:00 is included in the interval, since this is the end of a trading day at the NYSE.

Please input an object containing ONE day of data.

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
#load data
data("sample_tdata");
#aggregate price data to the 30 second frequency
head(sample_tdata$PRICE)
head(aggregatePrice(sample_tdata$PRICE,on="secs",k=30));
```

`aggregateQuotes` *Aggregate an xts object containing quote data*

Description

Function returns an xts object containing the aggregated quote data with columns "SYMBOL", "EX", "BID","BIDSIZ","OFR","OFRSIZ". See `sample_qdata` for an example of the argument qdata.

Usage

```
aggregateQuotes(qdata,on="minutes",k=5,marketopen,marketclose)
```

Arguments

`qdata`	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".
`k`	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate a xts 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".

Value

An xts object containing the aggregated quote data.

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.

For the variables "SYMBOL" and "EX" aggregation doesn't really make sense, thus the first value of the input is taken as the value for the complete output.

Column "MODE" is dropped because aggregating makes absolutely no sense.

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,...).

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
#load data
data("sample_qdata");
#aggregate quote data to the 30 second frequency
xx = aggregateQuotes(sample_qdata,on="seconds",k=30);
head(xx);
```

`aggregateTrades` *Aggregate an xts object containing trade data*

Description

Function returns an xts object containing the aggregated trade data with columns "SYMBOL", "EX", "PRICE", "SIZE". See `sample_tdata` for an example of the argument tdata.

Usage

```
aggregateTrades(tdata,on="minutes",k=5,marketopen,marketclose)
```

Arguments

`tdata`	xts object to be aggregated, containing the intraday trade 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".
`k`	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate a xts 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".

Value

An xts object containing the aggregated trade data.

Details

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

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

For the variables "SYMBOL" and "EX" aggregation doesn't really make sense, thus the first value of the input is taken as the value for the complete output.

Columns "COND", "CORR", "G127" are dropped because aggregating them makes absolutely no sense.

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 price,...).

Please input an object containing ONE day of data.

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
data("sample_tdata");
#aggregate trade data to 5 minute frequency
x = aggregateTrades(sample_tdata,on="minutes",k=5)
head(x);
```

`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)
```

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".
`k`	positive integer, indicating the number of periods to aggregate over. E.g. to aggregate a xts object to the 5 minute frequency set k=5 and on="minutes".
`weights`	By default, no weighting scheme is used. When you assign an xts object with wheights 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="previoustick" (default).

Value

An xts object containing the aggregated time series.

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, excluding the value at 09:35:00 itself.

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
#load sample price data
data("sample_tdata");
ts = sample_tdata$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);
```

`AJjumptest` *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), $r_{t,i}$ ($i = 1, \ldots, N$) and those computed at the slower time scale (kh), $y_{t,i}$($i = 1, \ldots,$ N/k).

They found that the limit (for $N \to \infty$) 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.

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 $i = 1, \ldots, N$) in period t.

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

Then the AJjumptest is given by:

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

in which,

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

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

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

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

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

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

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

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

Usage

```
AJjumptest(pdata, p=4 , k=2, align.by= NULL, align.period = NULL, makeReturns= FALSE, ...)
```

Arguments

`pdata`	a zoo/xts object containing all prices in period t for one asset.
`p`	can be chosen among 2 or 3 or 4. The author suggests 4. 4 by default.
`k`	can be chosen among 2 or 3 or 4. The author suggests 2. 2 by default.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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 underlying jump test is that the logarithmic price process X_t belongs to the class of Brownian semimartingales, which can be written as:

$$\mathrm{X}_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

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

$$\mathrm{Z}_t = \sum_{j=1}^{N_t} k_j$$

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

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

```
data(sample_tdata)
AJjumptest(sample_tdata$PRICE, p= 2, k= 3, align.by= "seconds", align.period= 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`	an 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

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

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

Description

Function returns an 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`	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. 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

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

`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 users' choices of estimator (integrated variance (IVestimator), integrated quarticity (IQestimator)), mechanism (linear, ratio) and adjustment (logarith), 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 $i = 1, \ldots, N$) 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.

θ: depends on IVestimator. (Huang and Tauchen (2005))

Usage

```
BNSjumptest(rdata, IVestimator= "BV", IQestimator= "TP", type= "linear",
            logtransform= FALSE, max= FALSE, align.by= NULL, align.period= NULL,
             makeReturns = FALSE, ...)
```

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 $startV$, 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 logarith form. FALSE by default.
`max`	boolean, should be TRUE when max adjustment in SE. FALSE by default.

`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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 underlying jump test is that the logarithmic price process X_t belongs to the class of Brownian semimartingales, which can be written as:

$$\mathrm{X}_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

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

$$\mathrm{Z}_t = \sum_{j=1}^{N_t} k_j$$

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

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

```
data(sample_tdata)
BNSjumptest(sample_tdata$PRICE, QVestimator= "RV", IVestimator= "minRV",
            IQestimator = "medRQ", type= "linear", makeReturns = TRUE)
```

---

convert *Convert trade or quote data into xts object saved in the RData format*

Description

Function converts both trade and quote data stored as "txt" or "csv" and structured as illustrated in the pdf documentation into xts-objects and stores them in the "RData" format. Subsequently, the data can be loaded into the R console by `TAQLoad`.

Usage

```
convert(from,to,datasource,datadestination,trades=TRUE,quotes=TRUE,
        ticker,dir=FALSE,extension="txt",header=FALSE,
        tradecolnames=NULL,quotecolnames=NULL,
        format="%Y%m%d %H:%M:%S",onefile=FALSE)
```

Arguments

`from`	first date to convert e.g. "2008-01-30".
`to`	last date to convert e.g. "2008-01-31".
`datasource`	folder in which the original data is stored.
`datadestination`	folder in which the converted data should be stored.
`trades`	boolean, determines whether trades are converted.
`quotes`	boolean, determines whether quotes are converted.
`ticker`	vector with tickers to be converted.
`dir`	boolean, if TRUE the datadestination folder and subfolders will be created automatically.
`extension`	character, indicating the data format of the original data. Can be either "txt" or "csv". In case your data comes from "www.tickdata.com", set extension="tickdatacom".
`header`	boolean, indicating whether each data file contains a header.
`tradecolnames`	vector containing column names of your trade data. By default, the standard NYSE data format is taken, see pdf documentation for more details.
`quotecolnames`	vector containing column names of your quote data. By default, the standard NYSE data format is taken, see pdf documentation for more details.
`format`	character, indicates in what format TIME and DATE are recorded in the original data. By default, %Y%m%d %H:%M:%S" is taken, which means the date is denoted by e.g. "20080130" and the time by e.g. "09:30:00".

`onefile`	indicates the way the source data is stored. By default FALSE, which means the function expects that the source data is saved in a folder sturcture with a folder for each date (see vignette for more info). In case the data for multiple days is stored in one file, set onefile=TRUE. The naming convention for files is, e.g. for ticker="AAPL", "AAPL_trades.extension" and "AAPL_quotes.extension" for trades and quotes respectively.

Value

For each day an xts object is saved into the folder of that date, containing the converted data. This procedure is performed for each stock in "ticker". More information can be found in the pdf documentation.

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
#In order for these examples to work, the folder datasource
#should contain two folders named 2008-01-02 and 2008-01-03.
#These folder contain the files with the trade data,
#which are named "AAPL_trades.txt" or "AA_trades.txt".

from="2008-01-02";
to = "2008-01-03";
## Not run: datasource=datadestination="C:\data"

### txt files from NYSE:

convert(from,to,datasource,datadestination,trades=TRUE,
          quotes=FALSE,ticker=c("AA","AAPL"),dir=FALSE,extension="txt",
          header=FALSE,tradecolnames=NULL,quotecolnames=NULL,
          format="%Y%m%d %H:%M:%S");

#Now, the folder datadestination will contain two folders
#named 2008-01-02 and 2008-01-03 containing
#the files AAPL_trades.RData and AAPL_trades.RData containing the trades.
#The data can now be loaded with the TAQLoad function.

########## Csv file from WRDS
#Suppose the datasource folder contains one csv file from WRDS
#with data on IBM for multiple days.
#The file should be named "IBM_trades.csv" and can be easily converted into xts
#and then saved in RData format by:

convert(from=from, to=to, datasource=datasource, datadestination=datadestination,
    trades = T, quotes = T, ticker="IBM", dir = FALSE, extension = "csv",
    header = TRUE, tradecolnames = NULL, quotecolnames = NULL, format = format,
    onefile = TRUE )

####### ASC file from www.tickdata.com
```

```
#Suppose the datasource folder contains asc files for trades and quotes
#from "www.tickdata.com" for GLP.
#The files "GLP_quotes.asc" and "GLP_trades.asc" should be saved in datasource folder.

 convert(from=from, to=to, datasource=datasource, datadestination=datadestination,
    trades = T, quotes = T, ticker="GLP", dir = TRUE, extension = "tickdatacom",
    header = TRUE, tradecolnames = NULL, quotecolnames = NULL, format = "
    onefile = TRUE );

## End(Not run)
```

---

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`	an xts object containing the time series data.
`daybegin`	character in the format of "HH:MM:SS", specifying the starting hour, minute and second of an exhange trading day.
`dayend`	character in the format of "HH:MM:SS", specifying the closing hour, minute and second of an exhange trading day.

Value

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

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.

`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
`...`	any other passthrough parameters

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.

`getTradeDirection` *Get trade direction*

Description

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

Usage

```
getTradeDirection(tqdata,...);
```

Arguments

`tqdata`	xts object, containing joined trades and quotes (e.g. using `matchTradesQuotes`)
`...`	additional arguments.

Value

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

Details

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

Author(s)

Jonathan Cornelissen and Kris Boudt

References

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

---

harModel *HAR model estimation (Heterogeneous Autoregressive model for Realized volatility)*

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 days'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), leverage=NULL,
 RVest = c("rCov", "rBPCov"), type = "HARRV", jumptest = "ABDJumptest",
 alpha = 0.05, h = 1, transform = NULL, ...)
```

Arguments

`data`	an xts-object containing the intraday (log-)returns.
`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.
`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 or two elements. The first element refers to the name of the function to estimate the daily integrated variance (non-jump-robust), while the second element refers to the name of the function to estimate the continuous component of daily volatility (jump-robust). By default RVest = c("rCov","rBPCov"), i.e. using the Realized Volatility and Realized Bi-Power Variance.
`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" or type = "HARRVCJ".
`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

Value

The function outputs an object of class `harModel` and `lm` (so `harModel` is a subclass of `lm`). So far I only added a print method as you can see in the examples. Input here is welcome, what should a plot of an "harmodel" object look like? What other methods are useful?

Details

See vignette.

Author(s)

Jonathan Cornelissen and Kris Boudt

References

Andersen, T. G., T. Bollerslev, and F. Diebold (2007). Roughing it up: includ- ing 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.

Examples

```
##### Example 1: HARRVCJ #####
data("sample_5minprices_jumps");
data = sample_5minprices_jumps[,1];
data = makeReturns(data); #Get the high-frequency return data

x = harModel(data, periods = c(1,5,10), periodsJ=c(1,5,10), RVest = c("rCov","rBPCov"),
      type="HARRVCJ",transform="sqrt");
# Estimate the HAR model of type HARRVCJ
class(x);
x

##### Example 2:  #####
# Forecasting daily Realized volatility for DJI 2008 using the basic harModel: HARRV
data(realized_library); #Get sample daily Realized Volatility data
DJI_RV = realized_library$Dow.Jones.Industrials.Realized.Variance; #Select DJI
DJI_RV = DJI_RV[!is.na(DJI_RV)]; #Remove NA's
DJI_RV = DJI_RV['2008'];

x = harModel(data=DJI_RV , periods = c(1,5,22), RVest = c("rCov"),
   type="HARRV",h=1,transform=NULL);
class(x);
x;
summary(x);
plot(x);
```

`has.Qty` *check for Trade, Bid, and Ask/Offer (BBO/TBBO), Quantity, and Price data*

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

```
has.Qty(x, which = FALSE)
```

Arguments

`x`	data object
`which`	disply position of match

`heavyModel` *HEAVY Model estimation*

Description

This function calculatest the High frEquency bAsed VolatilitY (HEAVY) model proposed in Shephard and Sheppard (2010).

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 demeaned 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 traditionalheavy 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 optimizat ion to find the optimal parameters estimates.
`LB`	a vector of length K indicating the lower bounds to be used in the esti- mation. 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.

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.

Details

See vignette. NOTE: The implementation of the heavyModel is not completely finished. For the moment only bound constraints on the parameters are imposed in the optimization. Future developments also include outputting standard errors, and a c-implementation of the likelihood function to speed up the QML estimation.

Author(s)

Jonathan Cornelissen and Kris Boudt

References

Shephard, N. and K. Sheppard (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:
 data("realized_library");
 returns =  realized_library$Dow.Jones.Industrials.Returns;
 rk      =  realized_library$Dow.Jones.Industrials.Realized.Kernel;
 returns = returns[!is.na(rk)];  rk = rk[!is.na(rk)]; # Remove NA's
 data = cbind( returns^2, rk ); # Make data matrix with returns and realized measures
 backcast = matrix( c(var(returns),mean(rk)) ,ncol=1);

 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);
```

`heavyModelC` *HEAVY Model estimation using C code*

Description

This function is the same as heavyModel function, except for using C code to speed up the calculation process.

This function calculatest 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

```
heavyModelC(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 demeaned 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 traditionalheavy 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 $i = 1, \ldots, T$).

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 reprensented by the following equations:

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

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

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 β be around 0.6 and sum of α+β 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:

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

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 and Kris Boudt

References

Shephard, N. and K. Sheppard (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:
data("realized_library");
returns =  realized_library$Dow.Jones.Industrials.Returns;
rk      =  realized_library$Dow.Jones.Industrials.Realized.Kernel;
returns = returns[!is.na(rk)];  rk = rk[!is.na(rk)]; # Remove NA's
data = cbind( returns^2, rk ); # Make data matrix with returns and realized measures
backcast = matrix( c(var(returns),mean(rk)) ,ncol=1);

#For traditional (default) version:
startvalues = c(0.004,0.02,0.44,0.41,0.74,0.56); # Initial values;
output = heavyModelC( 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);

output = heavyModelC( data = as.matrix(data,ncol=2), p=p, q=q, compconst=FALSE,
                  startingvalues = startvalues, backcast=backcast);
```

ivInference — *Function returns the value, the standard error and the confidence band of the integrated variance (IV) estimator.*

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,

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

cv : critical value.

se : standard error.

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

$\hat{IQ}$ integrated quarticity estimator.

Usage

```
ivInference (rdata, IVestimator="RV", IQestimator="rQuar", confidence=0.95,
           align.by= NULL, align.period = NULL, makeReturns = FALSE, ...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`IVestimator`	can be chosen among integrated variance estimators: RV, BV, minRV or medRV. RV by default.

`IQestimator`	can be chosen among integrated quarticity estimators: rQuar, TP, QP, minRQ or medRQ. rQuar by default.
`confidence`	confidence level set by users. 0.95 by default.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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:

$$\mathrm{X}_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u$$

where a is the drift term, σ 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

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

`JOjumptest` *Jiang and Oomen (2008) tests for the presence of jumps in the price series.*

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 $i = 1, \ldots, N$) in period t.

Let $R_{t,i}$ be a simple return (with $i = 1, \ldots, N$) 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));

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

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

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

U: independent standard normal random variables

p: parameter (power).

Usage

```
JOjumptest(pdata, power=4,...)
```

Arguments

`pdata`	a zoo/xts object containing all prices in period t for one asset.
`power`	can be chosen among 4 or 6. 4 by default.
`...`	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:

$$\mathrm{X}_t = \int_0^t a_u du + \int_0^t \sigma_u dW_u + Z_t$$

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

$$\mathrm{Z}_t = \sum_{j=1}^{N_t} k_j$$

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

```
data(sample_5minprices_jumps)
JOjumptest(sample_5minprices_jumps[,1], power= 6)
```

lltc.xts *LLTC Data*

Description

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

Usage

```
data(lltc.xts)
```

Examples

```
data(lltc.xts)
plot(lltc.xts)
```

makePsd *Returns the positive semidinite projection of a symmetric matrix using the eigenvalue method*

Description

Function returns the positive semidinite 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".

Value

An xts object containing the aggregated trade data.

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 Γ be the orthogonal matrix consisting of the p eigenvectors of S. Denote $\lambda_1^+, \ldots, \lambda_p^+$ its p eigenvalues, whereby the negative eigenvalues have been replaced by zeroes. Under this approach, the positive semi-definite projection of S is $S^+ = \Gamma' \text{diag}(\lambda_1^+, \ldots, \lambda_p^+)\Gamma$.

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

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` *Compute log returns*

Description

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

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

Usage

```
makeReturns(ts);
```

Arguments

`ts`	xts object

Value

an xts object containing the log returns.

Details

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

Author(s)

Jonathan Cornelissen and Kris Boudt

---

matchTradesQuotes *Match trade and quote data*

Description

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

Usage

```
matchTradesQuotes(tdata,qdata,adjustment=2)
```

Arguments

`tdata`	xts-object containing the trade data.
`qdata`	xts-object containing the quote data.
`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

xts-object containing the matched trade and quote data

Author(s)

Jonathan Cornelissen and Kris Boudt

References

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

Examples

```
#load data samples
data("sample_tdata");
data("sample_qdata");
#match the trade and quote data
tqdata = matchTradesQuotes(sample_tdata,sample_qdata);
#have a look
head(tqdata);
```

`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 $i = 1, \ldots, N$) 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, align.by=NULL, align.period=NULL, makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
medRQ(rdata= sample_tdata$PRICE, align.by= "minutes", align.period =5, makeReturns= TRUE)
medRQ
```

`medRV` *medRV*

Description

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

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

Then, the medRV is given by

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

Usage

```
medRV(rdata,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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.

Value

numeric

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).

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(sample_tdata);
medrv = medRV( rdata = sample_tdata$PRICE, align.by ="minutes",
          align.period =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 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.
`selection`	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: • selection = "maxvolume": use the (bid/ask) price of the entry with largest (bid/ask) volume. • selection = "weightedaverage": take the weighted average of all bid (ask) prices, weighted by "BIDSIZ" ("OFRSIZ").

Value

xts object

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 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`	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: • selection = "maxvolume": use the price of the transaction with largest volume. • selection = "weightedaverage": take the weighted average of all prices.

Value

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

minRQ *An estimator of integrated quarticity from applying the minimum operator on blocks of two returns.*

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 $i = 1, \ldots, N$) 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,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
minRQ(rdata= sample_tdata$PRICE, align.by= "minutes", align.period =5, makeReturns= TRUE)
minRQ
```

`minRV` *minRV*

Description

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

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

Then, the minRV is given by

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

Usage

```
minRV(rdata,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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.

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(sample_tdata);

minrv = minRV( rdata = sample_tdata$PRICE, align.by ="minutes",
           align.period =5, makeReturns=TRUE);
minrv
```

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

Description

Function returns univariate or multivariate preaveraged estimator, as difined 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.
`...`	additional arguments.

Details

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

$$\mathrm{Y}_\tau = X_\tau + \epsilon_\tau$$

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

ϵ_τ 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 τ of a list (after refeshing data). Let r_{τ_i} be a return (with $i = 1, \ldots, N$) of an asset in period τ. 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 $g : [0,1] \rightarrow R$ given by $g(x)$ = $\min(x, 1-x)$. k_N is a sequence of integers satisfying $\mathrm{k}_N = \lfloor \theta N^{1/2} \rfloor$. We use $\theta = 0.8$ as recommendations 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

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

where $\hat{\Psi}_N = \frac{1}{2N} \sum_{i=1}^{N} \boldsymbol{r}_{\tau_i} (\boldsymbol{r}_{\tau_i})'$. 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 $\text{k}_N = \lfloor \theta N^{1/2+\delta} \rfloor$. $\delta = 0.1$ results in a consistent estimate without the bias correction and a PSD estimate, in which case:

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

Value

an dxd 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

```
data(sample_5minprices_jumps)
a= list (sample_5minprices_jumps["2010-01-04",1],
         sample_5minprices_jumps["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 object at least containing a column "PRICE".

Value

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

`noZeroQuotes` *Delete the observations where the bid or ask is zero*

Description

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

Usage

```
noZeroQuotes(qdata)
```

Arguments

`qdata`	an xts object at least containing the columns "BID" and "OFR".

Value

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

`previoustick` *previoustick (internal function)*

Description

See `aggregatets`.

`quotesCleanup` *Cleans quote data*

Description

This is a wrapper function for cleaning the quote data of all stocks in "ticker" over the interval [from,to]. The result is saved in the folder datadestination.

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

The following cleaning functions are performed sequentially: `noZeroQuotes`, `selectExchange`, `rmLargeSpread`, `mergeQuotesSameTimestamp`, `rmOutliers`.

Usage

```
quotesCleanup(from,to,datasource,datadestination,ticker,exchanges,
              qdataraw,report,selection,maxi,window,type,
              rmoutliersmaxi,...)
```

Arguments

`from` character indicating first date to clean, e.g. "2008-01-30".

`to` character indicating last date to clean, e.g. "2008-01-31".

`datasource` character indicating the folder in which the original data is stored.

`datadestination`
character indicating the folder in which the cleaned data is stored.

`ticker` vector of tickers for which the data should be cleaned, e.g. ticker = c("AAPL","AIG").

`exchanges` vector of stock exchange symbols for all tickers in vector "ticker". It thus should have the same length as the vector ticker. Only data from one exchanges will be retained for each stock respectively, e.g. exchanges = c("T","N"). The possible exchange symbols 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

`qdataraw`	xts object containing (ONE day and for 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 `mergeQuotesSameTimestamp`. The default is "median".
`maxi`	argument to be passed on to the cleaning routine `rmLargeSpread`.
`window`	argument to be passed on to the cleaning routine `rmOutliers`.
`type`	argument to be passed on to the cleaning routine `rmOutliers`.
`rmoutliersmaxi`	argument to be passed on to the cleaning routine `rmOutliers`.
`...`	additional arguments.

Value

For each day an xts 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 quotes remained after each cleaning step.

In case you supply the argument "rawqdata", the on-disk functionality is ignored and the function returns a list with the cleaned quotes 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, 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 1 day
#data("sample_qdataraw");
#head(sample_qdataraw);
#dim(sample_qdataraw);
#qdata_aftercleaning = quotesCleanup(qdataraw=sample_qdataraw,exchanges="N");
#qdata_aftercleaning$report;
#barplot(qdata_aftercleaning$report);
#dim(qdata_aftercleaning$qdata);

#In case you have more data it is advised to use the on-disk functionality
#via "from","to","datasource",etc. arguments
```

`rAccumulation` *Realized Accumulation Plot*

Description

Plots the realized estimate as it accumulates over a time interval.

Usage

```
rAccumulation(x, period = 1, y = NULL, align.by="seconds", align.period = 1,
   plotit = FALSE, cts = TRUE, makeReturns = FALSE)
```

Arguments

`x`	Tick data in xts object.
`y`	Tick data in xts object.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the returns to this period first
`plotit`	T for plot
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns

Details

Plots the realized estimate as it accumulates over a time interval. This is a good tool to determine what obersations are adding (possibly subtracting for covariance) to the estimate.

Value

Realized accumulation vector if plotit = F

Author(s)

Scott Payseur <scott.payseur@gmail.com>

References

S. W. Payseur. A One Day Comparison of Realized Variance and Covariance Estimators. *Working Paper: University of Washington*, 2007

See Also

`rMarginal`

Examples

```
data(sbux.xts)

cumm <- list()
cumm[[1]] <- rCumSum(sbux.xts, period=1, align.by="seconds", align.period=60)
cumm[[2]] <- rCumSum(sbux.xts, period=10, align.by="seconds", align.period=60)
cumm[[3]] <- rCumSum(sbux.xts, period=20, align.by="seconds", align.period=60)
cumm[[4]] <- rCumSum(sbux.xts, period=30, align.by="seconds", align.period=60)
accum <- list()
accum[[1]] <- rAccumulation(sbux.xts, period=10, align.by="seconds", align.period=60)
accum[[2]] <- rAccumulation(sbux.xts, period=20, align.by="seconds", align.period=60)
accum[[3]] <- rAccumulation(sbux.xts, period=30, align.by="seconds", align.period=60)

par(mfrow=c(2,1))
plot(cumm[[1]], xlab="", ylab="Cumulative Ruturns", main="Starbucks (SBUX)",
     sub='20110701', type="p", col=16, lwd=2)
lines(cumm[[2]], col=2, lwd=2)
lines(cumm[[3]], col=3, lwd=2)
lines(cumm[[4]], col=4, lwd=2)
plot(accum[[1]], xlab="", ylab="Realized Accumulation", type="l",main="Starbucks (SBUX)",
      sub='20110701', col=2, lwd=2)
lines(accum[[2]], col=3, lwd=2)
lines(accum[[3]], col=4, lwd=2)
```

`rAVGCov` *Realized Covariance: Average Subsample*

Description

Realized Covariance using average subsample.

Usage

```
rAVGCov(rdata, cor = FALSE, period = 1, align.by = "seconds",
         align.period = 1, cts = TRUE, makeReturns = FALSE, ...)
```

Arguments

`rdata`	In the multivariate case: a list. Each list-item i contains an xts object with the intraday data of stock i for day t. In the univariate case: an xts object containing the (tick) data for one day.
`cor`	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the tick data to this many [seconds|minutes|hours]
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns
`...`	...

Value

Realized covariance using average subsample.

Author(s)

Scott Payseur <scott.payseur@gmail.com>

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 sp100 stocks using intraday data - but which frequency to use? *Econometric Reviews*, 2008.

Examples

```
# Average subsampled realized variance/covariance for CTS aligned at one minute returns at
# 5 subgrids (5 minutes).
data(sample_tdata);
data(lltc.xts);
data(sbux.xts);

# Univariate
rvSub = rAVGCov( rdata = sample_tdata$PRICE, period = 5, align.by ="minutes",
                  align.period=5, makeReturns=TRUE);
rvSub

# Multivariate:
rcSub = rAVGCov( rdata = list(lltc.xts,sbux.xts), period = 5, align.by ="minutes",
                  align.period=5, makeReturns=FALSE);
rcSub
```

`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.
`...`	additional arguments.

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 ith return on day t for asset j and index m (with $i = 1, \ldots, N$).

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

```
data(sample_5minprices_jumps)
a=sample_5minprices_jumps['2010-01-04',1]
b=sample_5minprices_jumps['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 $Nx1$ return vector and $i = 1, ..., M$ 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

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

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

Usage

```
rBPCov(rdata, cor = FALSE, align.by = NULL, align.period = NULL,
        makeReturns = FALSE, makePsd = FALSE, ...)
```

Arguments

`rdata`	a (MxN) 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.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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.
`...`	additional arguments.

Value

an NxN 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.

Examples

```
# Realized Bipower Variance/Covariance for CTS aligned
# at 5 minutes.
data(sample_tdata);
data(sample_5minprices_jumps);

# Univariate:
rbpv = rBPCov( rdata = sample_tdata$PRICE, align.by ="minutes",
                   align.period =5, makeReturns=TRUE);
rbpv

# Multivariate:
rbpc = rBPCov( rdata = sample_5minprices_jumps['2010-01-04'], makeReturns=TRUE,makePsd=TRUE);
rbpc
```

rCov *Realized Covariance*

Description

Function returns the Realized Covariation (rCov).

Let $r_{t,i}$ be an intraday $Nx1$ return vector and $i = 1, ..., M$ the number of intraday returns.

Then, the rCov is given by

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

Usage

```
rCov(rdata,cor = FALSE, align.by = NULL, align.period = NULL, makeReturns = FALSE, ...)
```

Arguments

`rdata`	a (MxN) 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.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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.

Value

an NxN matrix

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
# Realized Variance/Covariance for CTS aligned
# at 5 minutes.
data(sample_tdata);
data(sample_5minprices_jumps);

# Univariate:
rv = rCov( rdata = sample_tdata$PRICE, align.by ="minutes",
                   align.period =5, makeReturns=TRUE);
rv
```

```
# Multivariate:
rc = rCov( rdata = sample_5minprices_jumps['2010-01-04'], makeReturns=TRUE);
rc
```

`rCumSum` *Plot cummulative returns*

Description

Plots cummulative returns at a certain alignment given a return series.

Usage

```
rCumSum(x, period = 1, align.by="seconds",align.period = 1,
  plotit = FALSE, type = "l", cts = TRUE, makeReturns = FALSE)
```

Arguments

`x`	Tick data in xts object.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the returns to this period first
`plotit`	T for plot
`type`	Line or points
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns

Value

Cummulative return vector if plotit = F

Author(s)

Scott Payseur <scott.payseur@gmail.com>

Examples

```
data(sbux.xts)

cumm <- list()
cumm[[1]] <- rCumSum(sbux.xts, period=1, align.by="seconds", align.period=60)
cumm[[2]] <- rCumSum(sbux.xts, period=10, align.by="seconds", align.period=60)
cumm[[3]] <- rCumSum(sbux.xts, period=20, align.by="seconds", align.period=60)
cumm[[4]] <- rCumSum(sbux.xts, period=30, align.by="seconds", align.period=60)
plot(cumm[[1]], xlab="", ylab="Cumulative Ruturns", main="Starbucks (SBUX)",
   sub='20110701', type="p", col=16, lwd=2)
```

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

---

realized_library *The realized library from the Oxford-Man Institute of Quantitative Finance*

Description

An xts object containing the daily returns, daily Realized Variance and daily Realized Kernels ranging from 1996-01-03 up to 2009-03-01 for several indices and exchange rates. Use `colnames(realized_library)` to see which assets are included. The full library of the Oxford-Man Institute of Quantitative Finance can be found on their website: http://realized.oxford-man.ox.ac.uk.

Usage

```
data("realized_library")
```

Format

xts object

References

Gerd Heber, Asger Lunde, Neil Shephard, and Kevin Sheppard (2009) "Oxford-Man Institute's realized library, version 0.1", Oxford-Man Institute, University of Oxford.

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

---

refreshTime *Synchronize (multiple) irregular timeseries by refresh time*

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 again traded. This process is repeated untill the end of one time series is reached.

Usage

```
refreshTime(pdata)
```

Arguments

`pdata`	a list. Each list-item contains an xts object containing the original time series (one day only and typically a price series).

Value

An xts object containing the synchronized time series.

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 infomationally 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 = as.xts(1:length(ta),order.by=ta);
b = as.xts(1:length(tb),order.by=tb);

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

`rHYCov` *Hayashi-Yoshida Covariance*

Description

Hayashi-Yoshida Covariance

Usage

```
rHYCov(rdata, cor = FALSE, period = 1, align.by = "seconds",
       align.period = 1, cts = TRUE, makeReturns = FALSE, makePsd = TRUE, ...)
```

Arguments

`rdata`	a list. Each list-item i contains an xts object with the intraday data of stock i for day t.
`cor`	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the tick data to this many [seconds|minutes|hours]
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns
`makePsd`	boolean, in case it is TRUE, the positive definite version of rTSCov is returned. FALSE by default.
`...`	...

Author(s)

Scott Payseur

References

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

Examples

```
# Average Realized Kernel Variance/Covariance for CTS aligned at one minute returns at
# 5 subgrids (5 minutes).
data(lltc.xts);
data(sbux.xts);
 # Multivariate:
rHYCov = rHYCov( rdata = list(lltc.xts,sbux.xts), period = 5, align.by ="minutes",
                  align.period=5, makeReturns=FALSE);
rHYCov
#Note: for the diagonal elements the rCov is used.
```

`rKernel.available` *Available Kernels*

Description

Returns a vector of the available kernels.

Usage

```
rKernel.available()
```

Author(s)

Scott Payseur <scott.payseur@gmail.com>

References

Ole E. Barndorff-Nielsen, Peter Reinhard Hansen, Asger Lunde, and Neil Shephard. Regular and modified kernel-based estimators of integrated variance: The case with independent noise. *Working Paper*, 2004.

See Also

`rKernelCov`

Examples

```
rKernel.available()
```

`rKernelCov` *Realized Covariance: Kernel*

Description

Realized covariance calculation using a kernel estimator.

Usage

```
rKernelCov(rdata, cor = FALSE, kernel.type = "rectangular", kernel.param = 1,
            kernel.dofadj = TRUE, align.by = "seconds", align.period = 1,
            cts = TRUE, makeReturns = FALSE, type = NULL, adj = NULL,
            q = NULL, ...)
```

Arguments

`rdata`	In the multivariate case: a list. Each list-item i contains an xts object with the intraday data of stock i for day t. In the univariate case: an xts object containing the (tick) data for one day.
`cor`	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
`kernel.type`	Kernel name (or number)
`kernel.param`	Kernel parameter (usually lags)
`kernel.dofadj`	Kernel Degree of freedom adjustment
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the tick data to this many [seconds|minutes|hours]
`cts`	Calendar Time Sampling is used
`makeReturns`	Convert to Returns

`type`	Deprecated, use kernel.type
`adj`	Deprecated, use kernel.dofadj
`q`	Deprecated, use kernel.param
`...`	...

Details

The different types of kernels can be found using `rKernel.available`.

Value

Kernel estimate of realized covariance.

Author(s)

Scott Payseur <scott.payseur@gmail.com>

References

Ole E. Barndorff-Nielsen, Peter Reinhard Hansen, Asger Lunde, and Neil Shephard. Regular and modified kernel-based estimators of integrated variance: The case with independent noise. *Working Paper*, 2004.

B. Zhou. High-frequency data and volatility in foreign-exchange rates. *Journal of Buiness & 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.

See Also

`rKernel.available`

Examples

```
# Average Realized Kernel Variance/Covariance for CTS aligned at one minute returns at
# 5 subgrids (5 minutes).
data(sample_tdata);
data(lltc.xts);
data(sbux.xts);

# Univariate:
rvKernel = rKernelCov( rdata = sample_tdata$PRICE, period = 5, align.by ="minutes",
                  align.period=5, makeReturns=TRUE);
rvKernel

# Multivariate:
rcKernel = rKernelCov( rdata = list(lltc.xts,sbux.xts), period = 5, align.by ="minutes",
                  align.period=5, makeReturns=FALSE);
rcKernel
```

`rKurt` *Realized kurtosis of highfrequency return series.*

Description

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

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

Usage

```
rKurt (rdata,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rKurt(sample_tdata$PRICE,align.by ="minutes", align.period =5, makeReturns = TRUE)
```

`rMarginal` *Maginal Contribution to Realized Estimate*

Description

Plots the marginal contribution to the realized estimate.

Usage

```
rMarginal(x, y = NULL, period,align.by="seconds", align.period = 1,
  plotit = FALSE, cts = TRUE,makeReturns = FALSE)
```

Arguments

`x`	RealizedObject or TimeSeries for S+
`y`	RealizedObject or TimeSeries for S+
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the returns to this period first
`plotit`	T for plot
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns

Details

Plots the marginal contribution to the realized estimate. This is a good tool to determine what obersations are adding (possibly subtracting for covariance) to the estimate.

Value

Marginal contribution vector if plotit = F

Author(s)

Scott Payseur <scott.payseur@gmail.com

References

S. W. Payseur. A One Day Comparison of Realized Variance and Covariance Estimators. *Working Paper: University of Washington*, 2007

See Also

`rAccumulation`

Examples

```
data(sbux.xts)
par(mfrow=c(2,1))
plot(rCumSum(sbux.xts, period=10, align.by="seconds", align.period=60),
   xlab="", ylab="Cumulative Ruturns", main="Starbucks (SBUX)",
    sub='20110701', type="p")
barplot(rMarginal(sbux.xts, period=10, align.by="seconds", align.period=60)$y,
    main="Marginal Contribution Plot")
```

`rmLargeSpread` *Delete entries for which the spread is more than "maxi" times the median spread*

Description

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

Usage

```
rmLargeSpread(qdata,maxi)
```

Arguments

`qdata`	an xts 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 object

Details

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

Author(s)

Jonathan Cornelissen and Kris Boudt

`rmNegativeSpread` *Delete entries for which the spread is negative*

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

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

`rmOutliers` *Delete entries for which the mid-quote is outlying with respect to surrounding entries*

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 sample. 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

```
rmOutliers(qdata,maxi=10,window=50,type="advanced")
```

Arguments

`qdata`	an 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).

Value

xts object

Details

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

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 $i = 1, \ldots, N$) in period t.

Then, the rMPV is given by

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

in which

$d_{m,p} = \mu_{p/m}^{-m}$:

m: the window size of return blocks;

p: the power of the variation;

and $m > p/2$.

Usage

```
rMPV(rdata, m= 2, p=2, align.by= NULL, align.period= 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.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rMPV(sample_tdata$PRICE, m=2, p=3, align.by= "minutes", align.period=5,makeReturns= TRUE)
```

`rmTradeOutliers` *Delete transactions with unlikely transaction prices*

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

```
rmTradeOutliers(tdata,qdata)
```

Arguments

`tdata`	an xts object containing the time series data, with at least the column "PRICE", containing the transaction price.
`qdata`	an xts object containing the time series data, with at least the columns "BID" and "OFR", containing the bid and ask prices.

Value

xts object

Details

Note: in order to work correctly, the input data of this function should be cleaned trade (tdata) and quote (qdata) data respectively.

Author(s)

Jonathan Cornelissen and Kris Boudt

`rOWCov` *Realized Outlyingness Weighted Covariance*

Description

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

Let $r_{t,i}$, for $i = 1, ..., M$ be a sample of M high-frequency $(Nx1)$ 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, align.by = NULL, align.period = NULL,
       makeReturns = FALSE, seasadjR = NULL, wfunction = "HR", alphaMCD = 0.75,
       alpha = 0.001, ...)
```

Arguments

`rdata`	a (MxN) 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.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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 (MxN) matrix/zoo/xts object containing the seasonaly 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).
`...`	additional arguments.

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 NxN 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 CTS aligned
# at 5 minutes.
data(sample_tdata);
data(sample_5minprices_jumps);

# Univariate:
rvoutw = rOWCov( rdata = sample_tdata$PRICE, align.by ="minutes",
                   align.period =5, makeReturns=TRUE);
rvoutw

# Multivariate:
rcoutw = rOWCov( rdata = sample_5minprices_jumps['2010-01-04'], makeReturns=TRUE);
rcoutw
```

`rQPVar` *Realized quadpower variation of highfrequency return series.*

Description

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

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

Then, the rQPVar is given by

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

Usage

```
rQPVar (rdata, align.by=NULL, align.period=NULL, makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rQPVar(rdata= sample_tdata$PRICE, align.by= "minutes", align.period =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 $i = 1, \ldots, N$) 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, align.by=NULL, align.period=NULL, makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rQuar(rdata= sample_tdata$PRICE, align.by= "minutes", align.period =5, makeReturns= TRUE)
rQuar
```

`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-synchronic 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

`pdata`	a list. Each list-item i contains an xts object with the intraday price data of stock i for day t.
`cor`	boolean, in case it is TRUE, the correlation is returned. FALSE by default.
`startIV`	vector containing the first step estimates of the integrated variance of the assets, needed in the truncation. Is NULL by default.
`noisevar`	vector containing the estimates of the noise variance of the assets, needed in the truncation. Is NULL by default.
`K`	positive integer, slow time scale returns are computed on prices that are K 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.
makePsd	boolean, in case it is TRUE, the positive definite version of rRTSCov is returned. FALSE by default.
eta	positive real number, squared standardized high-frequency returns that exceed eta are detected as jumps.

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 $\tau_1, \tau_2, \ldots, \tau_n$ as follows:

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

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

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

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

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

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 $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_{N_T^X}\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{N_T^Y}\}$ 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 $\Phi = \{\phi_1, \phi_2, ..., \phi_{M_N+1}\}$, 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:

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

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 NxN 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 infomationally 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 for CTS
data(sample_tdata);
data(lltc.xts);
data(sbux.xts);

# Univariate:
rvRTS = rRTSCov( pdata = sample_tdata$PRICE);
# Note: Prices as input
rvRTS

# Multivariate:
rcRTS = rRTSCov( pdata = list(cumsum(lltc.xts)+100,cumsum(sbux.xts)+100) );
# Note: List of prices as input
rcRTS
```

rScatterReturns *Scatterplot of aligned returns*

Description

Creates a scatterplot of cross returns.

Usage

```
rScatterReturns(x,y, period, align.by="seconds", align.period=1,
   numbers=FALSE,xlim= NULL, ylim=NULL,
  plotit=TRUE, pch=NULL, cts=TRUE, makeReturns=FALSE,
   scale.size=0, col.change=FALSE,...)
```

Arguments

`x`	Tick data in xts object.
`y`	Tick data in xts object.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the returns to this period first
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns
`plotit`	T for plot
`numbers`	T for count
`pch`	type of point
`ylim`	ylimit
`xlim`	xlimit
`scale.size`	.
`col.change`	.
`...`	...

Details

Scatterplot of returns.

Author(s)

Scott Payseur <scott.payseur@gmail.com>

References

S. W. Payseur. A One Day Comparison of Realized Variance and Covariance Estimators. *Working Paper: University of Washington*, 2007

Examples

```
data(sbux.xts)
data(lltc.xts)
par(mfrow=c(2,1))
rScatterReturns(sbux.xts,y=lltc.xts, period=1, align.period=20,
    ylab="LLTC",xlab="SBUX",numbers=FALSE)
rScatterReturns(sbux.xts,y=lltc.xts, period=1, align.period=20,
   ylab="LLTC",xlab="SBUX",numbers=TRUE)
```

`rSkew` *Realized skewness of highfrequency return series.*

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 $i = 1, \ldots, N$) 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,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rSkew(sample_tdata$PRICE,align.by ="minutes", align.period =5, makeReturns = TRUE)
```

`rSV` *Realized semivariance of highfrequency return series.*

Description

Function returns Realized semivariance, 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 $i = 1, \ldots, N$) 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,align.by=NULL,align.period=NULL,makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

Value

list

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

```
data(sample_tdata)
rSV(sample_tdata$PRICE,align.by ="minutes", align.period =5, makeReturns = TRUE)
```

`rThresholdCov` *Threshold Covariance*

Description

Function returns the treshold covariance matrix proposed in Gobbi and Mancini (2009). Unlike the `rOWCov`, the THRESCov 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 $Nx1$ return vector and $i = 1, ..., M$ the number of intraday returns.

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

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

with the treshold 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, align.by = NULL, align.period = NULL,
              makeReturns = FALSE, ...)
```

Arguments

`rdata`	a (MxN) 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.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours".
`align.period`	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.

Value

an NxN 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:
data(lltc.xts);
data(sbux.xts);

# Multivariate:
rcThreshold = rThresholdCov(cbind(lltc.xts,sbux.xts), align.by="minutes",align.period=1);
rcThreshold
```

`rTPVar` *Realized tripower variation of highfrequency 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 $i = 1, \ldots, N$) in period t.

Then, the rTPVar is given by

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

Usage

```
rTPVar (rdata, align.by=NULL, align.period=NULL, makeReturns=FALSE,...)
```

Arguments

`rdata`	a zoo/xts object containing all returns in period t for one asset.
`align.by`	a string, align the tick data to "seconds"|"minutes"|"hours"
`align.period`	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.

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(sample_tdata)
rTPVar(rdata= sample_tdata$PRICE, align.by= "minutes", align.period =5, makeReturns= TRUE)
rTPVar
```

`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-synchronic 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 contains an xts object with the intraday price data of stock i for day t.
`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 K 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.
`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 $\tau_1, \tau_2, \ldots, \tau_n$ as follows:

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

where $\overline{n}_K = (n-K+1)/K$, $\overline{n}_J = (n-J+1)/J$ 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 $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_{N_T^X}\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{N_T^Y}\}$ 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. $\phi_{j+1} = \max(\tau_{N^X_{\phi_j}+1}, \theta_{N^Y_{\phi_j}+1})$. The complete refresh time sample grid is $\Phi = \{\phi_1, \phi_2, ..., \phi_{M_N+1}\}$, 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:

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

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 NxN 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 infomationally 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 for CTS
data(sample_tdata);
data(lltc.xts);
data(sbux.xts);

# Univariate:
rvTS = rTSCov( pdata = sample_tdata$PRICE);
# Note: Prices as input
rvTS

# Multivariate:
rcTS = rTSCov( pdata = list(cumsum(lltc.xts)+100,cumsum(sbux.xts)+100) );
# Note: List of prices as input
rcTS
```

rZero *Calculates the percentage of co-zero returns at a specified sampling period*

Description

Calculates the percentage of co-zero returns at a specified sampling period.

Usage

```
rZero(rdata, period=1, align.by="seconds", align.period = 1, cts = TRUE,
   makeReturns = FALSE, ...)
```

Arguments

`rdata`	an xts object containing the tick data or a list. In case of list: each list-item i contains an xts object with the intraday data of stock i for day t.
`period`	Sampling period
`align.by`	Align the tick data to seconds|minutes|hours
`align.period`	Align the returns to this period first
`cts`	Create calendar time sampling if a non realizedObject is passed
`makeReturns`	Prices are passed make them into log returns
`...`	...

Value

Percentage of co-zero returns.

Author(s)

Scott Payseur <scott.payseur@gmail.com>

References

S. W. Payseur. A One Day Comparison of Realized Variance and Covariance Estimators. *Working Paper: University of Washington*, 2007

Examples

```
data(sbux.xts)
data(lltc.xts)
rZero( rdata = list(sbux.xts, lltc.xts) , period = 60, align.by ="seconds", align.period=1)
```

salesCondition *Delete entries with abnormal Sale Condition.*

Description

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

Usage

```
salesCondition(tdata)
```

Arguments

tdata	an xts object containing the time series data, with one column named "COND" indicating the Sale Condition.

Value

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

sample_5minprices *Ten artificial time series 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. 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

```
data("sample_5minprices")
```

Format

xts object

References

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde and N. Shephard (2009). Multivariate realised kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. Journal of Econometrics, forthcoming.

`sample_5minprices_jumps` *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

```
data("sample_5minprices_jumps")
```

Format

xts object

References

Barndorff-Nielsen, O. E., P. R. Hansen, A. Lunde and N. Shephard (2009). Multivariate realised kernels: consistent positive semi-definite estimators of the covariation of equity prices with noise and non-synchronous trading. Journal of Econometrics, forthcoming.

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

`sample_qdata` *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 `sample_qdataraw`, using `quotesCleanup`.

Usage

```
data("sample_qdata")
```

Format

xts object

`sample_qdataraw` *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

```
data("sample_qdataraw")
```

Format

xts object

`sample_real5minprices` *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

```
data("sample_real5minprices")
```

Format

xts object

`sample_returns_5min` *Sample returns data*

Description

EUR/USD returns from January to September 2004

Usage

```
data(sample_returns_5min)
```

Format

matrix

`sample_tdata` *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 `sample_tdataraw`, using `tradesCleanup`.

Usage

```
data("sample_tdata")
```

Format

xts object

`sample_tdataraw` *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

```
data("sample_tdataraw")
```

Format

xts object

`sbux.xts` *Starbucks Data*

Description

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

Usage

```
data(sbux.xts)
```

Examples

```
data(sbux.xts)
plot(sbux.xts)
```

`selectExchange` *Retain only data from a single stock exchange*

Description

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

Usage

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

Arguments

`data`	an xts 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:

- 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

xts object

Author(s)

Jonathan Cornelissen and Kris Boudt

`spotvol` *Spot volatility estimation*

Description

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 ith return on day t).

Usage

```
spotvol(data, method = "detper", ..., on = "minutes", k = 5,
  marketopen = "09:30:00", marketclose = "16:00:00", tz = "GMT")
```

Arguments

`data`	Either an `xts` object, containing price data, or a `matrix` containing returns. For price data, irregularly spaced observations are allowed. They will be aggregated to the level specified by parameters `on` and `k`. For return data, the observations are assumed to be equispaced, with the time between them specified by `on` and `k`. Return data should be in matrix form, where each row corresponds to a day, and each column to an intraday period. The output will be in the same form as the input (`xts` or `matrix`/`numeric`).
`method`	specifies which method will be used to estimate the spot volatility. Options include `"detper"` and `"stochper"`. See 'Details'.
`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"`.
`...`	method-specific parameters (see 'Details').

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 `"bipower", "rv"`
`periodicvol`	A string specifying the estimation method for the component of intraday volatility, that depends in a determin
`P1`	A positive integer corresponding to the number of cosinus terms used in the flexible Fourier specification of t
`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 fu

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 pe
`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 (`"BFGS"` in `optim`). Default = `list(sigma = 0`
`control`	A list of options to be passed down to `optim`.

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 specifiying the bandwidth estimation method. Possible values include `"cv", "quarticity"`. Method `"cv"` eq
`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 appropiate.

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 ($h- > \infty$). 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. D
`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 numl
`volest`	String specifying the realized volatility estimator to be used in local windows. Possible values are "bipower", "rv
`online`	Boolean indicating whether estimations at a certain point t should be done online (using only information available

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

- `spot`
- `cp`

This nonparametric method by Fried (2012) assumes the volatility to be piecewise constant over local windows. Robust two-sample tests are applied to detect changes in variability between subsequent windows. The spot volatility can then be estimated by evaluating regular realized volatility estimators within each local window.

Along with the spot volatility estimates, this method will return the detected change points in the volatility level. When plotting a `spotvol` object containing `cp`, these change points will be visualized.

GARCH models with intraday seasonality ("garch")

Parameters:

`model`	String specifying the type of test to be used. Options include "sGARCH", "eGARCH". See `ugarchspec` in t
`garchorder`	Numeric value of length 2, containing the order of the GARCH model to be estimated. Default = `c(1,1)`.
`dist`	String specifying the distribution to be assumed on the innovations. See `distribution.model` in `ugarch`
`solver.control`	List containing solver options. See `ugarchfit` for possible values. Default = `list()`.
`P1`	A positive integer corresponding to the number of cosinus terms used in the flexible Fourier specification
`P2`	Same as `P1`, but for the sinus terms. Default = 5.

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

- `spot`
- `ugarchfit`

This method generates the external regressors needed to model the intraday seasonality with a Flexible Fourier form. The `rugarch` package is then employed to estimate the specified GARCH(1,1) model.

Along with the spot volatility estimates, this method will return the `ugarchfit` object used by the `rugarch` package.

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 specifiedby `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"`.

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

```
# Load sample data

data(sample_real5minprices)
data(sample_returns_5min)

# Default method, deterministic periodicity

vol1 <- spotvol(sample_real5minprices)
plot(vol1)

# Compare to stochastic periodicity

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(sample_real5minprices, 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)

# Various kernel estimates

h1 = bw.nrd0((1:nrow(sample_returns_5min))*(5*60))
vol3 <- spotvol(sample_returns_5min, method = "kernel", h = h1)
vol4 <- spotvol(sample_returns_5min, method = "kernel", est = "quarticity")
vol5 <- spotvol(sample_returns_5min, 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)

# Piecewise constant volatility, using an example from Fried (2012)

simdata <- matrix(sqrt(5/3)*rt(3000, df = 5), ncol = 500, byrow = TRUE)
simdata <- c(1, 1, 1.5, 1.5, 2, 1)*simdata
# the volatility of the simulated now changes at 1000, 2000 and 2500
vol6 <- spotvol(simdata, method = "piecewise", m = 200, n  = 100,
                online = FALSE)
plot(vol6)

# Compare regular GARCH(1,1) model to eGARCH, both with external regressors

vol7 <- spotvol(sample_returns_5min, method = "garch", model = "sGARCH")
vol8 <- spotvol(sample_returns_5min, 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)
```

TAQLoad *Load trade or quote data into R*

Description

Function to load the taq data into R. If only the trades (or quotes) should be loaded the function returns them directly as xts object. If both trades and quotes should be extracted, the function returns a list with two items named "trades" and "quotes" respectively.

The function assumes that the files are ordered in folders per day, and the folders contain: ticker_trades.RData or ticker_quotes.RData. In these files the xts object is stored. If you used the function `convert`, this shall be the case. Please find more information in the pdf documentation.

Usage

```
TAQLoad(tickers,from,to,trades=TRUE,quotes=FALSE,
          datasource=NULL,variables=NULL)
```

Arguments

`tickers`	the ticker(s) to be loaded. It is recommended that you use only 1 ticker as input in case of non-synchronic observations. For synchronic data a vector of tickers can be used.
`from`	first day to load e.g. "2008-01-30".
`to`	last day to load e.g. "2008-01-30".
`trades`	boolean, determines whether trades are extracted.
`quotes`	boolean, determines whether quotes are extracted.
`datasource`	path to folder in which the files are contained.
`variables`	a character (or character vector) containing the name(s) of the variable(s) that should be loaded, e.g. c("SYMBOL","PRICE"). By default all data is loaded.

Value

see section `description`

Author(s)

Jonathan Cornelissen and Kris Boudt

Examples

```
#In order for these examples to work, the folder datasource
#should contain two folders named 2008-01-02 and 2008-01-03.
#These folder contain the files with the trade data,
#which are named "AAPL_trades.RData" or "AA_trades.RData".

from="2008-01-02";
```

```
to = "2008-01-03";
## Not run: datasource="C:\data";

#TAQLoad: load data for stock AAPL
## Not run: xx = TAQLoad(tickers="AAPL",from,to,trades=TRUE,quotes=FALSE,
datasource=datasource,variables=NULL)
## End(Not run)

#Load only price data for stocks AA and AAPL
## Not run: xx = TAQLoad(tickers=c("AA","AAPL"),from,to,trades=TRUE,
quotes=FALSE,datasource=datasource,variables="PRICE")
## End(Not run)
```

---

tqLiquidity *Calculate numerous (23) liquidity measures*

Description

Function returns an xts object containing one of the following liquidity measures:

es, rs, value_trade, signed_value_trade, di_diff, di_div, pes, prs, price_impact, prop_price_impact, tspread, pts, p_return_sqr, p_return_abs, qs, pqs, logqs, logsize, qslope, logqslope, mq_return_sqr, mq_return_abs.

Usage

```
tqLiquidity(tqdata,tdata,qdata,type,...)
```

Arguments

`tqdata`	xts object, containing joined trades and quotes (e.g. using `matchTradesQuotes`)
`tdata`	xts-object containing the trade data.
`qdata`	xts-object containing the quote data.
`type`	a character vector containing the name of one of the above mentioned liquidity measures, e.g. type="es" to get the effective spread.
`...`	additional arguments.

Value

an xts object containing one of the above mentioned liquidity measures.

Details

The respective liquidity measures are defined as follows:

- es: effective spread

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

  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 were the time indication refers to (and thus not to the registered quote timestamp).
- rs: realized spread

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

  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.
- value_trade: trade value

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

- signed_value_trade: 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)).
- di_diff: depth imbalance (as a difference)

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

  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).
- di_div: depth imbalace (as ratio)

$$\text{depth imbalace (as ratio)}_t = (\frac{D_t * \text{OFRSIZ}_t}{\text{BIDSIZ}_t})^{D_t},$$

  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).
- pes: 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 were the time indication refers to (and thus not to the registered quote timestamp).
- prs: 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 were the time indication refers to (and thus not to the registered

- price_impact: price impact

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

  (see Boehmer (2005), Bessembinder (2003)).
- prop_price_impact: proportional price impact

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

  (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).
- tspread: 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 were the time indication refers to (and thus not to the registered quote timestamp).
- pts: 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 were the time indication refers to (and thus not to the registered quote timestamp).
- p_return_sqr: squared log return on trade prices

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

- p_return_abs: absolute log return on trade prices

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

- qs: 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).
- pqs: 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).

- logqs: log quoted spread
$$\text{log quoted spread}_t = \log(\frac{\text{OFR}_t}{\text{BID}_t})$$
(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).
- logsize: 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).
- qslope: quoted slope
$$\text{quoted slope}_t = \frac{\text{quoted spread}_t}{\text{log quoted size}_t}$$
(Hasbrouck and Seppi, 2001).
- logqslope: log quoted slope
$$\text{log quoted slope}_t = \frac{\text{log quoted spread}_t}{\text{log quoted size}_t}.$$
- mq_return_sqr: midquote squared return
$$\text{midquote squared return}_t = (\log(\text{midquote}_t) - \log(\text{midquote}_{t-1}))^2,$$
where $\text{midquote}_t = \frac{\text{BID}_t + \text{OFR}_t}{2}$.
- mq_return_abs: midquote absolute return
$$\text{midquote absolute return}_t = |\log(\text{midquote}_t) - \log(\text{midquote}_{t-1})|,$$
where $\text{midquote}_t = \frac{\text{BID}_t + \text{OFR}_t}{2}$.
- signed_trade_size: 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).

Author(s)

Jonathan Cornelissen and Kris Boudt

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

```
#load data samples
data("sample_tdata");
data("sample_qdata");
tdata = sample_tdata;
qdata = sample_qdata;
#match the trade and quote data
tqdata = matchTradesQuotes(tdata,qdata);

#calculate the proportional realized spread:
prs = tqLiquidity(tqdata,tdata,qdata,type="prs");

#calculate the effective spread:
es = tqLiquidity(tqdata,type="es");
```

`tradesCleanup` *Cleans trade data*

Description

This is a wrapper function for cleaning the trade data of all stocks in "ticker" over the interval [from,to]. The result is saved in the folder datadestination. The function returns a vector indicating how many trades were removed at 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).

The following cleaning functions are performed sequentially: `noZeroPrices`, `selectExchange`, `salesCondition`, `mergeTradesSameTimestamp`.

Since the function `rmTradeOutliers` also requires cleaned quote data as input, it is not incorporated here and there is a seperate wrapper called `tradesCleanupFinal`.

Usage

```
tradesCleanup(from,to,datasource,datadestination,ticker,exchanges,
                 tdataraw,report,selection,...)
```

Arguments

`from`	character indicating first date to clean, e.g. "2008-01-30".
`to`	character indicating last date to clean, e.g. "2008-01-31".
`datasource`	character indicating the folder in which the original data is stored.
`datadestination`	character indicating the folder in which the cleaned data is stored.
`ticker`	vector of tickers for which the data should be cleaned, e.g. ticker = c("AAPL","AIG")
`exchanges`	list of vectors of stock exchange(s) for all tickers in vector "ticker". It thus should have the same length as the vector ticker. E.g. in case of two stocks; exchanges = list("N", c("Q","T")). The possible exchange symbols 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

`tdataraw`	xts object containing (ONE day and 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 `mergeTradesSameTimestamp`. The default is "median".
`...`	additional arguments.

Value

For each day an xts 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, pages 2232-2245.

Falkenberry, T.N. (2002). High frequency data filtering. Unpublished technical report.

Examples

```
#Consider you have raw trade data for 1 stock for 1 day
data("sample_tdataraw");
head(sample_tdataraw);
dim(sample_tdataraw);
tdata_afterfirstcleaning = tradesCleanup(tdataraw=sample_tdataraw,exchanges=list("N") );
tdata_afterfirstcleaning$report;
barplot(tdata_afterfirstcleaning$report);
dim(tdata_afterfirstcleaning$tdata);

#In case you have more data it is advised to use the on-disk functionality
#via "from","to","datasource",etc. arguments
```

`tradesCleanupFinal` *Perform a final cleaning procedure on trade data*

Description

Function performs cleaning procedure `rmTradeOutliers` for the trades of all stocks in "ticker" over the interval [from,to] and saves the result 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

```
tradesCleanupFinal(from,to,datasource,datadestination,ticker,
                      tdata,qdata,...)
```

Arguments

`from`	character indicating first date to clean, e.g. "2008-01-30".
`to`	character indicating last date to clean, e.g. "2008-01-31".
`datasource`	character indicating the folder in which the original data is stored.
`datadestination`	character indicating the folder in which the cleaned data is stored.
`ticker`	vector of tickers for which the data should be cleaned.
`tdata`	xts object containing (ONE day and for ONE stock only) trade data cleaned by `tradesCleanup`. 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`	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)
`...`	additional arguments.

Value

For each day an xts object is saved into the folder of that date, containing the cleaned data. This procedure is performed for each stock in "ticker".

In case you supply the arguments "tdata" and "qdata", 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, pages 2232-2245.

Examples

```
#Consider you have raw trade data for 1 stock for 1 day
#data("sample_qdata");    #load cleaned quote data
#data("sample_tdataraw"); #load raw trade data
#tdata_afterfirstcleaning = tradesCleanup(tdataraw=sample_tdataraw,
#exchange="N",report=FALSE);
#dim(tdata_afterfirstcleaning);
#tdata_afterfinalcleaning = tradesCleanupFinal(qdata=sample_qdata,
#tdata=tdata_afterfirstcleaning);
#dim(tdata_afterfinalcleaning);
#In case you have more data it is advised to use the on-disk functionality
#via "from","to","datasource",etc. arguments
```

Index