Package 'ldhmm'

August 3, 2017

Type Package

Title Hidden Markov Model for Financial Time-Series Based on Lambda Distribution

Version 0.4.2

Date 2017-08-03

Author Stephen H-T. Lihn [aut, cre]

Maintainer Stephen H-T. Lihn <stevelihn@gmail.com>

Description Hidden Markov Model (HMM) based on symmetric lambda distribution framework is implemented for the study of return time-series in the financial market. Major features in the S&P500 index, such as regime identification, volatility clustering, and anti-correlation between return and volatility, can be extracted from HMM cleanly. Univariate symmetric lambda distribution is essentially a location-scale family of exponential power distribution. Such distribution is suitable for describing highly leptokurtic time series obtained from the financial market. It provides a theoretically solid foundation to explore such data where the normal distribution is not adequate. The HMM implementation follows closely the book: ``Hidden Markov Models for Time Series", by Zucchini, MacDonald, Langrock (2016).

URL https://ssrn.com/abstract=2979516

Depends R (>= 3.3.3)

Imports stats, utils, ecd, optimx, xts (>= 0.10-0), zoo, moments, parallel, graphics, scales, ggplot2, grid, methods

Suggests knitr, testthat, depmixS4, roxygen2, R.rsp, shape

VignetteBuilder R.rsp

License Artistic-2.0

Encoding latin1

LazyData true

RoxygenNote 6.0.1

Collate 'ldhmm-calc_stats_from_obs.R' 'ldhmm-numericOrNull-class.R' 'ldhmm-package.R' 'ldhmm-class.R' 'ldhmm-conditional_prob.R' 'ldhmm-constructor.R' 'ldhmm-decode_stats_history.R' 'ldhmm-decoding.R' 'ldhmm-forecast_prob.R' 'ldhmm-forecast_state.R' 'ldhmm-forecast_volatility.R' 'ldhmm-fred_data.R' 'ldhmm-gamma_init.R' 'ldhmm-ld_stats.R' 'ldhmm-log_forward.R' 'ldhmm-mle.R' 'ldhmm-mllk.R'

'ldhmm-n2w.R' 'ldhmm-oxford_man_index_list.R'
'ldhmm-oxford_man_plot_obs.R'
'ldhmm-oxford_man_realized_data.R' 'ldhmm-oxford_man_ts.R'
'ldhmm-plot_spx_vix_obs.R' 'ldhmm-pseudo_residuals.R'
'ldhmm-read_sample_object.R' 'ldhmm-simulate_abs_acf.R'
'ldhmm-simulate_state_transition.R' 'ldhmm-sma.R'
'ldhmm-state_ld.R' 'ldhmm-state_pdf.R' 'ldhmm-ts_abs_acf.R'
'ldhmm-ts_log_rtn.R' 'ldhmm-viterbi.R' 'ldhmm-w2n.R'

NeedsCompilation no

R topics documented:

`ldhmm-package` *ldhmm: A package for HMM using lambda distribution.*

Description

The ldhmm package provides the core class and functions to calculate Hidden Markov Model (HMM) using lambda distribution framework. The main goal is to provide a theoretically solid foundation to explore the return time-series in the financial market, where the normal distribution is not adequate due to the leptokurtic nature of the data. Major features in the S&P 500 index, such as regime identification, volatility clustering, and anti-correlation between return and volatility, can be extracted from HMM cleanly. Univariate symmetric lambda distribution is essentially a location-scale family of power-exponential distribution. Such distribution is suitable for describing highly leptokurtic time series obtained from the financial market.

Details

The main change compared to a normal-distribution based HMM is to add the third paramter `lambda` to describe the kurtosis level of the distribution. When `lambda` is one, the model converges back to a normal-distribution based HMM (e.g. using depmixS4 package). The ability to optimize kurtosis brings the model output to be more consistent with the data. In particular, for daily data, the level of kurtosis is quite high. This puts the normal distribution in great disadvantage. This problem is solved by using the lambda distribution.

Author(s)

Stephen H-T. Lihn

References

Walter Zucchini, Iain L. MacDonald, Roland Langrock (2016). "Hidden Markov Models for Time Series, An Introduction Using R." Second Edition. CRC Press.

`ldhmm` *Constructor of ldhmm class*

Description

Construct an ldhmm class by providing the required parameters.

Usage

```
ldhmm(m, param, gamma, delta = NULL, stationary = TRUE,
  mle.optimizer = "nlm")
```

Arguments

m	numeric, number of states
param	matrix, the ecld parameters of states.
gamma	numeric or matrix, the transition probability matrix, must be conformed to m by m. if provided as vector, it will be converted to a matrix with byrow=TRUE.
delta	numeric, the initial distribution for each state, default is NULL.
stationary	logical, specify whether the initial distribution is stationary or not, default is TRUE.
mle.optimizer	character, specify alternative optimizer, default is nlm.

Value

An object of ldhmm class

Author(s)

Stephen H. Lihn

Examples

```
param0 <- matrix(c(0.003, 0.02, 1, -0.006, 0.03, 1.3), 2, 3, byrow=TRUE)
gamma0 <- matrix(c(0.9, 0.1, 0.1, 0.9), 2, 2, byrow=TRUE)
d <- ldhmm(m=2, param=param0, gamma=gamma0)
```

---

ldhmm-class *The ldhmm class*

Description

This S4 class is the major object class for ldhmm package

Slots

`call` The match.call slot

`m` numeric, length 1, number of states

`param.nbr` numeric, number of parameters (2 or 3) for each ecld object

`param` matrix, natural parameters for ecld objects, size of states times param.nbr. Each row can be 2-parameter sequences, or 3-parameter sequences. Three-parameter unit `(mu, sigma, lambda)` forms an ecld object representing a leptokurtic symmetric lambda distribution. On the other hand, to provide compatibility to a normal distribution HMM, two-parameter unit `(mu, sigma)` forms an ecld object with lambda=1.

`gamma` matrix, the transition probability matrix, must be m by m.

`delta` numeric, the initial distribution for each state, default is NULL.

`stationary` logical, specify whether the initial distribution is stationary or not, default is TRUE.

`mle.optimizer` character, the MLE optimizer. Currently it is just set to "nlm".

`return.code` numeric, the return code from the MLE optimizer.

`iterations` numeric, number of iterations MLE optimizer takes.

`mllk` numeric, the final mllk value.

`AIC` numeric, the final AIC.

`BIC` numeric, the final BIC.

`observations` numeric, stores the observations post optimization

`states.prob` matrix, stores the state probabilities post optimization

`states.local` numeric, stores the local decoding states post optimization

`states.global` numeric, stores the global decoding states post optimization (Viterbi)

`states.local.stats` matrix, stores the statistics of local states post optimization

`states.global.stats` matrix, stores the statistics of global states post optimization

---

`ldhmm.calc_stats_from_obs`

Computing the statistics for each state

Description

This utility computes the statistics (mean, sd, kurtosis, length) for each state. It can be based on the local or global decoding result. The concept of asymptotic statistics can be applied by which the largest N observations (in absolute term) can be dropped to avoid distortion from outliers. It is assumed the object already has come with filled data in `observations, states.prob, states.local, states.global` slots.

Usage

```
ldhmm.calc_stats_from_obs(object, drop = 0, use.local = TRUE)

ldhmm.drop_outliers(x, drop = 1)
```

Arguments

`object`	an ldhmm object that contains the observations.
`drop`	numeric, an integer to drop the largest N observations, default is zero.
`use.local`	logical, use local decoding result, default is `TURE`. Otherwise, use global decoding result.
`x`	numeric, the observations.

Value

an ldhmm object containing results of decoding

Author(s)

Stephen H. Lihn

`ldhmm.conditional_prob` *Computing the conditional probabilities*

Description

This utility computes the conditional probabilities that observation at time t equals xc, given all observations other than that at time t being the same.

Usage

```
ldhmm.conditional_prob(object, x, xc)
```

Arguments

`object`	an ldhmm object
`x`	numeric, the observations.
`xc`	numeric, the conditional observations.

Value

matrix of probabilities, size of xc times size of x.

Author(s)

Stephen H. Lihn

`ldhmm.decode_stats_history` *Estimating historical statistics (mean, volatility and kurtosis)*

Description

This utility estimates historical statistics (mean, volatility and kurtosis) according to the state probabilities. The ldhmm object must have been decoded by running through `ldhmm.decoding` function. Note that kurtosis is naively implemented as the linear sum from each state weighted by state probabilities. It is subject to change to more rigorous formula in future releases.

Usage

```
ldhmm.decode_stats_history(object, ma.order = 0, annualize = FALSE,
  days.pa = 252)
```

Arguments

`object`	a decoded ldhmm object
`ma.order`	a positive integer or zero, specifying order of moving average. Default is zero.
`annualize`	logical, to annaulize the sd and mean to V (xsqrt(days.pa)x100) and R (xdays.pa). Default is `FALSE`.
`days.pa`	a positive integer, specifying number of days per year, default is 252.

Value

an matrix of statistics history, size of observations times size of 3

Author(s)

Stephen H. Lihn

---

ldhmm.decoding *Computing the minus log-likelihood (MLLK)*

Description

This utility computes the state probabilities, uses local and global decoding to calculate the states. The results are saved to the returned ldhmm object.

Usage

```
ldhmm.decoding(object, x)
```

Arguments

object	an ldhmm object
x	numeric, the observations.

Value

an ldhmm object containing results of decoding

Author(s)

Stephen H. Lihn

---

ldhmm.forecast_prob *Computing the forecast probability distribution*

Description

This utility computes the forecast probability distribution (Zucchini, 5.3)

Usage

```
ldhmm.forecast_prob(object, x, xf, h = 1)
```

Arguments

object	an ldhmm object
x	numeric, the observations.
xf	numeric, the future observations to be forecasted.
h	integer, time steps to forecast.

Value

matrix of probabilities, size of h times size of xf.

Author(s)

Stephen H. Lihn

`ldhmm.forecast_state` *Computing the state forecast*

Description

This utility computes the state forecast.

Usage

```
ldhmm.forecast_state(object, x, h = 1)
```

Arguments

`object`	an ldhmm object
`x`	numeric, the observations.
`h`	integer, time steps to forecast.

Value

matrix of probabilities per state (even if h=1), number of states times size of h

Author(s)

Stephen H. Lihn

`ldhmm.forecast_volatility`
Computing the volatility forecast for next one period

Description

This utility computes the volatility forecast based on the given future observations for next one period.

Usage

```
ldhmm.forecast_volatility(object, x, xf, ma.order = 0, days.pa = 252)
```

Arguments

object	an ldhmm object
x	numeric, the observations.
xf	numeric, the future observations to be forecasted.
ma.order	a positive integer or zero, specifying order of moving average. Default is zero.
days.pa	a positive integer specifying trading days per year, default is 252.

Value

matrix of future observations and volatilities, size of 2 times length of xf.

Author(s)

Stephen H. Lihn

---

ldhmm.fred_data *Utility to download time series from FRED*

Description

This utility downloads time series from FRED. It serves as a data source for daily data, e.g. SP500 for S&P 500, and VIXCLS for CBOE VIX index. This can be concatenated to the static data to provide daily updates.

Usage

```
ldhmm.fred_data(symbol, col_out = "Close", do.logr = TRUE)
```

Arguments

symbol	character, the name of the time series
col_out	character, the name of the output closing price column. Default: "Close"
do.logr	logical, if TRUE (default), produce xts object of logr; otherwise, just the col_out column. Be aware that, because logr uses diff, the first day close will be deleted.

Value

The xts object for the time series

Examples

```
## Not run:
ldhmm.fred_data("VIXCLS")

## End(Not run)
```

`ldhmm.gamma_init` *Initializing tansition probability paramter*

Description

This utility has multiple purposes. It can generate a simple transition probability matrix, using p1 and p2, if prob is left as NULL. The generated gamma is raw and not normalized. If prob is provided as a vector, the utility converts it into a matrix as gamma. Furthermore, if prob is provided as a vector or matrix, the utility applies `min.gamma`, and normalize the sum of t.p.m. rows to 1. This is mainly an internal function used by MLE, not be concerned by external users.

Usage

```
ldhmm.gamma_init(m, p1 = 0.04, p2 = 0.01, prob = NULL, min.gamma = 0)
```

Arguments

`m`	numeric, number of states
`p1`	numeric, the first-neighbor transition probability, default is 0.4.
`p2`	numeric, the second-neighbor transition probability, default is 0.1.
`prob`	numeric or matrix, a full specified transition probability by user, default is `NULL`. If this is specified, p1, p2 would be ignored.
`min.gamma`	numeric, a minimum transition probability added to gamma to avoid singularity, default is 0. This is only used when `prob` is not `NULL`.

Value

a matrix as gamma

Author(s)

Stephen H. Lihn

Examples

```
gamma0 <- ldhmm.gamma_init(m=3)
prob=c(0.9, 0.1, 0.1,
       0.1, 0.9, 0.0,
       0.1, 0.1, 0.8)
gamma1 <- ldhmm.gamma_init(m=3, prob=prob)
gamma2 <- ldhmm.gamma_init(m=2, prob=gamma1, min.gamma=1e-6)
```

ldhmm.ld_stats *Computes the theoretical statistics per state*

Description

This utility computes the statistics (mean, sd, kurtosis) based on the lambda distribution. This is used to compare to the statistics from observations for each state.

Usage

```
ldhmm.ld_stats(object, annualize = FALSE, days.pa = 252)
```

Arguments

object	an ldhmm object
annualize	logical, to annaulize the sd and mean to V (xsqrt(days.pa)x100) and R (xdays.pa). Default is FALSE.
days.pa	a positive integer, specifying number of days per year, default is 252.

Value

a matrix of statistics for each state, size of states times 3

Author(s)

Stephen H. Lihn

ldhmm.log_forward *Computing the log forward and backward probabilities*

Description

This utility computes the logarithms of the forward and backward probabilities, aka alpha and beta. The logarithm keeps the computation away from floating point under/over-flow. (Zucchini, 5.4)

Usage

```
ldhmm.log_forward(object, x)

ldhmm.log_backward(object, x)
```

Arguments

object	an ldhmm object
x	numeric, the observations.

Value

numeric, the log probabilities

Author(s)

Stephen H. Lihn

`ldhmm.mle` *Computing the MLEs*

Description

Computing the MLEs using `nlm` package

Usage

```
ldhmm.mle(object, x, min.gamma = 1e-06, decode = FALSE, plot.fn = NULL,
  plot.interval = 200, ssm.fn = NULL, print.level = 0, iterlim = 1000,
  ...)
```

Arguments

`object`	an ldhmm object that can supply m, param.nbr and stationary.
`x`	numeric, the observations.
`min.gamma`	numeric, a minimum transition probability added to gamma to avoid singularity, default is `1e-6`.
`decode`	logical, run decoding after optimization, default is `FALSE`.
`plot.fn`	name of the function that takes ldhmm object. It will be called occasionally to track the progress of the fit, mainly by plotting the time series and states. E.g. When one fits the SPX index, the function `ldhmm.oxford_man_plot_obs` can be used to show the expected volatility vs Oxford-Man realized volatility. Default is `NULL`.
`plot.interval`	a positive integer, specifying how often to invoke plot function, default is 200 iterations.
`ssm.fn`	name of the function that takes ldhmm object. This function is called after the MLLK call. The purpose is to generate an additional score for optimization. E.g. It can be used to separate the states into predefined intervals, modeling a state space model. Default is `NULL`.
`print.level`	numeric, this argument determines the level of printing which is done during the minimization process. The default value of 0 means that no printing occurs, a value of 1 means that initial and final details are printed and a value of 2 means that full tracing information is printed.
`iterlim`	numeric, a positive integer specifying the maximum number of iterations to be performed before the program is terminated.
`...`	additional parameters passed to the MLE optimizer

Value

an ldhmm object containg results of MLE optimization

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
    param0 <- matrix(c(0.003, 0.02, 1, -0.006, 0.03, 1.3), 2, 3, byrow=TRUE)
    gamma0 <- ldhmm.gamma_init(m=2, prob=c(0.9, 0.1, 0.1, 0.9))
    h <- ldhmm(m=2, param=param0, gamma=gamma0)
    spx <- ldhmm.ts_log_rtn()
    ldhmm.mle(h, spx$x)

## End(Not run)
```

`ldhmm.mllk` *Computing the minus log-likelihood (MLLK)*

Description

This utility computes the MLLK. It is typically invoked by the MLE optimizer. (Zucchini, 3.2)

Usage

```
ldhmm.mllk(object, x, mllk.print.level = 0)
```

Arguments

`object`	an input ldhmm object to provide static reference, such as m, param.nbr, stationary.
`x`	numeric, the observations.
`mllk.print.level`	numeric, this argument determines the level of printing which is done during the minimization process. The default value of 0 means that no printing occurs, a value of 1 or greater means some tracing information is printed.

Value

an ldhmm object containing results of MLE optimization

Author(s)

Stephen H. Lihn

ldhmm.n2w *Transforming natural parameters to a linear working parameter array*

Description

This utility linearizes the natural parameters and transforms the contrained parameters to unconstrained parameters. (Zucchini, 3.3.1)

Usage

```
ldhmm.n2w(object, mu.scale = 1)
```

Arguments

object	an ldhmm object
mu.scale	numeric, if provided, e.g. mean(abs(x)), it is used to scale up mu so that the scale is more friendly to the optimizer. Default is 1.

Value

numeric, linear working parameter array

Author(s)

Stephen H. Lihn

Examples

```
param0 <- matrix(c(0.003, 0.02, 1, -0.006, 0.03, 1.3), 2, 3, byrow=TRUE)
gamma0 <- matrix(c(0.9, 0.1, 0.1, 0.9), 2, 2, byrow=TRUE)
d <- ldhmm(m=2, param=param0, gamma=gamma0)
v <- ldhmm.n2w(d)
```

ldhmm.oxford_man_index_list *Get the index list from Oxford-Man*

Description

This utility shows the index list within the Oxford-Man data set.

Usage

```
ldhmm.oxford_man_index_list()
```

Value

character, list of indices

Author(s)

Stephen H. Lihn

References

Oxford-Man Institute of Quantitative Finance. Realized Library: http://realized.oxford-man.ox.ac.uk

Examples

```
## Not run:
    ldhmm.oxford_man_index_list()

## End(Not run)
```

---

ldhmm.oxford_man_plot_obs

Plotting Oxford-Man realized volatility overlaid with HMM expected volatility

---

Description

This utility plots the Oxford-Man realized volatility (from SPX2.rv) and overlays with the HMM expected volatility with the observations set up SPX2.r. This graph is to show that the HMM is capable of reproducing the realized volatility. Optionally the insert shows the relation between the return and volatility indicated by each state. This plot is also called "volatility yield curve".

Usage

```
ldhmm.oxford_man_plot_obs(object, days.pa = 252, start.date = NULL,
  end.date = NULL, index.r = "SPX2.r", index.rv = "SPX2.rv",
  index.px = "SPX2.closeprice", index.px.scale = 15,
  index.px.origin = NULL, index.vol.ma.order = 5, index.symbol = NULL,
  vix.adj.ratio = NULL, insert.plot = TRUE, insert.viewport = NULL)
```

Arguments

object	an ldhmm object with a stationary solution. If this is set to NULL, an internal 10-state HMM object will be used.
days.pa	a positive integer specifying trading days per year, default is 252.
start.date	Date or character of ISO format (YYYY-MM-DD), specifying the start date of the plot, default is NULL.
end.date	Date or character of ISO format (YYYY-MM-DD), specifying the end date of the plot, default is NULL.
index.r	character, specifying index return column, default is SPX2.r.
index.rv	character, specifying index realized variance column, default is SPX2.rv.
index.px	character, specifying index closing price column, default to "SPX2.closeprice". Set this to NULL if you don't wish to see the price line.
index.px.scale	numeric, specifying the scaling factor when plotting price trend, default is 15. The closing price is converted to cumulative return by the price of the first date. Then plot from the mid-point of volatility axis with this scale.

index.px.origin
: numeric, specifying the starting value of the index price line, the default is NULL, which will start the index price line from the middle of y-axis.

index.vol.ma.order
: a positive integer specifying the simple moving average of the realized volatility, default is 5. This is needed because the realized volatility is very noisy at the daily level.

index.symbol
: character, used as a shortcut for index.r, index.rv, index.px. Once the root symbol is specified, the three time series can be derived by appending suffix .r, .rv, .closeprice. This mechanism will supercede those three components if and only if a value is detected. Default is NULL.

vix.adj.ratio
: numeric, if specified, VIX index is adjusted and plotted, default is NULL. The long-term ratio between VIX and 10-state HMM is 0.79. The VIX data is cached when the Oxford data is downloaded.

insert.plot
: logical, if true, also plot the volatility-return as insert in upper-right corner, default is TRUE.

insert.viewport
: optional viewport for the insert, default is NULL, which is internally set to grid::viewport(.8, .75,

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
    ldhmm.oxford_man_plot_obs(h)

## End(Not run)
```

---

ldhmm.oxford_man_realized_data

Get the realized data from Oxford-Man

Description

This utility fetches the realized data from Oxford-Man and stores the data frame in local memory. It can be retrieved as getOption("ldhmm.oxford.rv"). Since the data is updated on a daily basis. The user can optionally force the utility to fetch the new file in the same R-session. Note that the download is network intensive. The size of file is about 10-20 MB and growing daily. In addition, VIX daily data is downloaded as getOption("ldhmm.oxford.vix").

Usage

```
ldhmm.oxford_man_realized_data(force = FALSE, debug = FALSE)
```

Arguments

force
: logical, force the utility to fetch the new file. Default is FALSE.

debug
: logical, print debug information. Could be very verbose. Default is FALSE.

Value

data.frame containing the raw data from Oxford-Man.

Author(s)

Stephen H. Lihn

References

Oxford-Man Institute of Quantitative Finance. Realized Library: http://realized.oxford-man.ox.ac.uk

Examples

```
## Not run:
    ldhmm.oxford_man_realized_data()

## End(Not run)
```

---

ldhmm.oxford_man_ts *Get time series from Oxford-Man Realized data set*

Description

This utility returns the time series from the specific column in Oxford-Man Realized data set.

Usage

```
ldhmm.oxford_man_ts(symbol, log = FALSE, to.vol = FALSE, days.pa = 252)
```

Arguments

symbol	character, specify the column name. E.g. SPX2.r for the daily returns of SPX, SPX2.rv for the daily realized variances of SPX.
log	logical, take one plus log to convert return to log-return. Default is FALSE.
to.vol	logical, take sqrt(x*252)*100 to convert variance to annualized volatility. Default is FALSE.
days.pa	a positive integer specifying number of days to annualize volatility. Default is 252.

Value

an xts object containing the time series, with dates as index

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
    rtn <- ldhmm.oxford_man_ts("SPX2.r", log=TRUE)
    vol <- ldhmm.oxford_man_ts("SPX2.rv", to.vol=TRUE)

## End(Not run)
```

`ldhmm.plot_spx_vix_obs` *Plotting HMM expected volatility for SPX overlaid with adjusted VIX*

Description

This utility plots the HMM expected volatility of SPX overlaid with the VIX index adjusted by a ratio. The expected volatility is shown to have a long-term ratio of 0.79 relative to the VIX index. This plot will show how HMM deviates from VIX in a shorter time window. Optionally the insert shows the relation between the return and volatility indicated by each state. This plot is also called "volatility yield curve".

Usage

```
ldhmm.plot_spx_vix_obs(object, days.pa = 252, start.date = NULL,
  end.date = NULL, px.origin = NULL, px.scale = NULL,
  vix.adj.ratio = NULL, insert.plot = TRUE, insert.viewport = NULL)
```

Arguments

`object`	an ldhmm object with a stationary solution. If this is set to `NULL`, an internal 10-state HMM object will be used.
`days.pa`	a positive integer specifying trading days per year, default is 252.
`start.date`	Date or character of ISO format (YYYY-MM-DD), specifying the start date of the plot, default is `NULL`, which is converted to 1.5 years ago.
`end.date`	Date or character of ISO format (YYYY-MM-DD), specifying the end date of the plot, default is `NULL`, which means the latest date.
`px.origin`	numeric, specifying the starting value of the index price line, the default is `NULL`, which will start the index price line from the middle of y-axis.
`px.scale`	numeric, specifying the scaling factor when plotting price trend, default is 15. The closing price is converted to cumulative return by the price of the first date. Then plot from the mid-point of volatility axis with this scale.
`vix.adj.ratio`	numeric, if specified, VIX index is adjusted and plotted, default is `NULL`. Default is to use the long-term ratio between VIX and 10-state HMM, which is about 0.79.
`insert.plot`	logical, if true, also plot the volatility-return as insert in upper-right corner, default is `TRUE`.
`insert.viewport`	optional viewport for the insert, default is `NULL`, which is internally set to `grid::viewport(.8, .75,`

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
    ldhmm.plot_spx_vix_obs(h)

## End(Not run)
```

`ldhmm.pseudo_residuals` *Computing pseudo-residuals*

Description

This utility computes pseudo-residuals. (Zucchini, 6.2)

Usage

```
ldhmm.pseudo_residuals(object, x, xc.length = 1000)
```

Arguments

`object`	an ldhmm object
`x`	numeric, the observations.
`xc.length`	a positive integer specifying the length of `xc` when calculating conditional probabilities, default is 1000.

Value

a vector of normal quantiles

Author(s)

Stephen H. Lihn

Examples

```
## Not run:
  sr <- ldhmm.pseudo_residuals(object, x)
  hist(sr)
  acf(sr)
  qqnorm(sr, cex=0.5)
  L <- seq(-3,3,length.out=100)
  lines(L,L,col="red",lwd=2, lty=2)

## End(Not run)
```

ldhmm.read_sample_object

Read sample ldhmm object

Description

This utility is used to read sample ldhmm object so that the user doesn't need to go through lengthy optimization process to obtain a trained HMM for advanced features.

Usage

```
ldhmm.read_sample_object(symbol = "spx-daily-m10", extdata_dir = NULL)
```

Arguments

symbol	Character for the symbol of the time series. Default is spx-daily-m10
extdata_dir	optionally specify user's own extdata folder

Value

The ldhmm object

Author(s)

Stephen H-T. Lihn

Examples

```
hs <- ldhmm.read_sample_object() # SPX daily 10-state HMM
```

ldhmm.simulate_abs_acf

Simulating auto-correlation (ACF)

Description

This utility simulates the auto-correlation. The first few lag of ACF should match the ACF from the market data fairly well. This is a major validation of a successful HMM. Be aware this is a CPU intensive calculation. It uses the multi-core functionality.

Usage

```
ldhmm.simulate_abs_acf(object, n = 10000, lag.max = 5, debug = FALSE)
```

Arguments

object	an ldhmm object that can supply m, param.nbr and stationary.
n	a positive integer specifying number of observations to simulate.
lag.max	a positive integer, specifying number of lags to be computed.
debug	logical, specifying to print progress message or not. Default is FALSE.

Value

a vector of ACF

Author(s)

Stephen H. Lihn

`ldhmm.simulate_state_transition`
Simulating state transition

Description

This utility allows to simulate the states and obervations over time. Be aware this is a CPU intensive calculation. It uses the multi-core functionality.

Usage

```
ldhmm.simulate_state_transition(object, init = NULL)
```

Arguments

`object`	an ldhmm object that can supply m, param.nbr and stationary.
`init`	a positive integer specifying number of observations to simulate initially. The default is NULL, indicating that the simulation should use the (local) states and observations from within the object, and simulate the next set of random states and observations according to gamma. When init is an integer, the utility will generate random states and observations according to delta.

Value

an ldhmm object containing the simulated states and observations. The observations are stored in the `observations` slot. The states are stored in the `states.local` slot.

Author(s)

Stephen H. Lihn

ldhmm.sma *Simple moving average of a time series*

Description

This utility calculates simple moving average, with option to backfill for NA.

Usage

```
ldhmm.sma(x, order, na.backfill = TRUE)
```

Arguments

x	numeric, the time series.
order	a positive integer to specify order of moving average.
na.backfill	logical, specify whether to backfill for NA. Default is TRUE.

Value

numeric, simple moving average, same length as x.

Author(s)

Stephen H. Lihn

Examples

```
x <- 1:100
a <- ldhmm.sma(x, 10)
```

ldhmm.state_ld *Constructing the ecld objects per state*

Description

This utility constructs the ecld objects per state and return them in a list of easy query.

Usage

```
ldhmm.state_ld(object, state = NULL)
```

Arguments

object	an ldhmm object
state	numeric, the states.

Value

a list of ecld objects

Author(s)

Stephen H. Lihn

`ldhmm.state_pdf` *Computing the PDF per state given the observations*

Description

Computing the PDF per state given the observations. Only one of state or x can be a vector per call.

Usage

```
ldhmm.state_pdf(object, state, x)
```

Arguments

`object`	an ldhmm object
`state`	numeric, the states.
`x`	numeric, the observations.

Value

a vector or matrix of PDF. The dimension of matrix is state times x

Author(s)

Stephen H. Lihn

`ldhmm.ts_abs_acf` *Computing ACF of the absolute value of a time series*

Description

This utility computes the ACF of the absolute value of a time series as a proxy of the auto-correlation of the volatility. It allows to drop the largest N outliers so that they would not skew the ACF calculation.

Usage

```
ldhmm.ts_abs_acf(x, drop = 0, lag.max = 100)
```

Arguments

`x`	numeric, the observations.
`drop`	a positive integer, specifying number of outliers to be dropped.
`lag.max`	a positive integer, specifying number of lags to be computed.

Value

a vector of ACF

Author(s)

Stephen H. Lihn

`ldhmm.ts_log_rtn` *Get log-returns from historic prices of an index*

Description

This utility returns the dates and log-returns of an index available in ecd package. Note that the data from ecd package is static. A limited set of live daily time series can be appended from FRED, e.g. SPX, VIX, DJIA.

Usage

```
ldhmm.ts_log_rtn(symbol = "spx", start.date = "1950-01-01",
  end.date = "2015-12-31", on = "weeks", fred.data = FALSE)
```

Arguments

`symbol`	character, specify the symbol of the index, default is `spx`.
`start.date, end.date`	Date or character of ISO format (YYYY-MM-DD), to specify the date range, default is from 1950-01-01 to 2016-12-31. Set start.date and end.date to NULL or "" if you wish to get the entire time series.
`on`	character, specify the interval, days, weeks, months. Default is `weeks`.
`fred.data`	logical, specify whether to append daily time series data from FRED, default is `FALSE`.

Value

list of three vectors: `d` is the dates and `x` is log-returns and `p` is prices

Author(s)

Stephen H. Lihn

Examples

```
a <- ldhmm.ts_log_rtn()
```

ldhmm.viterbi — *Computing the global decoding by the Viterbi algorithm*

Description

This utility computes the global decoding by the Viterbi algorithm.

Usage

```
ldhmm.viterbi(object, x)
```

Arguments

object	an ldhmm object
x	numeric, the observations.

Value

a vector of states

Author(s)

Stephen H. Lihn

ldhmm.w2n — *Transforming working parameter array to natural parameters*

Description

This utility transforms the working parameter array back to the vectors and matrix of the contrained parameters. (Zucchini, 3.3.1)

Usage

```
ldhmm.w2n(object, par.vector, mu.scale = 1)
```

Arguments

object	an ldhmm object that can supply m, param.nbr and stationary.
par.vector	numeric, linear working parameter array. See ldhmm.n2w.
mu.scale	numeric, it should mirror what is provided to ldhmm.n2w. Default is 1.

Value

an ldhmm object

Author(s)

Stephen H. Lihn

`numericOrNull-class` *The numericOrNull class*

Description

The S4 class union of numeric and NULL, primarily used for detla

Index