Package 'ldhmm'

April 13, 2017

Type Package

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

Version 0.1.0

Date 2017-04-13

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 power-exponential 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).

Depends R (>= 3.3.1)

Imports stats, utils, ecd, xts, zoo, moments, parallel, graphics, methods

Suggests knitr, testthat, depmixS4, roxygen2, scales, shape

License Artistic-2.0

Encoding latin1

LazyData true

RoxygenNote 5.0.1

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-decoding.R' 'ldhmm-forecast_prob.R' 'ldhmm-forecast_state.R' 'ldhmm-ld_stats.R' 'ldhmm-log_forward.R' 'ldhmm-mle.R' 'ldhmm-mllk.R' 'ldhmm-n2w.R' 'ldhmm-pseudo_residuals.R' 'ldhmm-state_ld.R' 'ldhmm-state_pdf.R' 'ldhmm-ts_abs_acf.R' 'ldhmm-viterbi.R' 'ldhmm-w2n.R' 'ldhmm.ts_log_rtn.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)
```

Arguments

`m`	numeric, number of states
`param`	matrix, the ecld parameters of states.
`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`.

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)
```

Arguments

object	an ldhmm object
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.

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.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

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

Author(s)

Stephen H. Lihn

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)
```

Arguments

object	an ldhmm object

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, print.level = 0, iterlim = 1000, ...)
```

Arguments

object	an ldhmm object that can supply m, param.nbr and stationary.
x	numeric, the observations.
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 <- matrix(c(0.9, 0.1, 0.1, 0.9), 2, 2, byrow=TRUE)
    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(par.vector, x, ldhmm, mllk.print.level = 0)
```

Arguments

par.vector	numeric, the working parameter vector
x	numeric, the observations.
ldhmm	an input ldhmm object to provide static reference, such as m, param.nbr, stationary.
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)
```

Arguments

object	an ldhmm object

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.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.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.

Usage

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

Arguments

symbol	character, specify the symbol of the index, default is spx.
start.date, end.date	character, specify the date range in ISO-format, default is from 1950-01-01 to 2016-12-31.
on	character, specify the interval, days, weeks, months. Default is weeks.

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)
```

Arguments

`object`	an ldhmm object that can supply m, param.nbr and stationary.
`par.vector`	numeric, linear working parameter array. See `ldhmm.n2w`.

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