Package 'SwitchingVolatility'

July 26, 2013

Type Package

Title Investigate normal Hidden Markov Models with switching volatility

Version 1.0

Date 2010-11-23

Author Joern Dannemann, Hajo Holzmann, Florian Ketterer, Florian Schwaiger

Maintainer Florian Schwaiger <schwaige@mathematik.uni-marburg.de>

Description Methods for estimating parameters and testing the number of regimes.

License GPL-2

LazyLoad yes

Depends boot, MASS, quadprog, methods

R topics documented:

SwitchingVolatility-package

Investigate normal Hidden Markov Models with switching volatility

Description

We assume a Hidden Markov Model (HMM) to be the model of a given time series. The components are normally distributed with mean=0 and state dependent standard deviations. `em_test_normal_sigma` tests the Hypothesis of m_0 regimes of the HMM (the fuction `calc_weights` is a help function to calculate the weights of the asymptotic chi^2 mixture of the test statistic).

`norm.HMM.generate_sample` generates a realization af a HMM for given parameters. The function `norm.HMM.mle` estimates the parameters for a given dataset by (full) mle. Given a dataset and (estimated) parameters the function `norm.HMM.forecast.prop` calculates the distribution of (X_t+h|X_t,X_t-1,...). Further the function `norm.HMM.viterbi` implements the Viterbi algorithm to compute the most likely sequence of states for given parameters and observations.

Finally the functions are applied to a real daily stock log-return dataset (see examples below and `stock_returns`). For computing log-returns we provide the function `price2logreturn`.

Details

Package:	SwitchingVolatility
Type:	Package
Version:	1.0
Date:	2010-11-23
License:	GPL-2
LazyLoad:	yes

Author(s)

Joern Dannemann, Hajo Holzmann, Florian Ketterer, Florian Schwaiger

Maintainer: Florian Schwaiger <schwaige@mathematik.uni-marburg.de>

References

Walter Zucchini: Hidden Markov Models for Time Series

Li, P. and Chen, J. (2009): Testing the order of a Finite mixture, Journal of the American Statistical Association. ahead of print.

Examples

```
#-----------------------------------------------------------#
#Fit: HMM's, sdf: normal distribution (as scale family)     #
#EXAMPLE                                                    #
#-----------------------------------------------------------#
```

```
data(stock_returns)
x=stock_returns

split.screen(c(1,2))
plot.ts(x)
abline("h"=mean(x),lty=2,col=2)
screen(2)
plot(density(x),main="")

mean(x)
x=x-mean(x)

#----------------------------------#
#             1. STOP              #
#----------------------------------#

dev.off()

#testing the hypothesis of 1 regimes
a1 = em_test_normal_sigma(dataset=x, m0=1)
a1 #p-value is zero, so we reject the hypothesis


#testing the hypothesis of 2 regimes
a2 = em_test_normal_sigma(dataset=x, m0=2)
a2 #small p-value, so we reject the hypothesis

#testing the hypothesis of 3 regimes
a3 = em_test_normal_sigma(dataset=x, m0=3)
a3 #we maintain the hypothesis

plot(a3)


#----------------------------------#
#             2. STOP              #
#----------------------------------#

#now we fit the parameters of the HMM with the full likelihood
m0=3
sigma0 = a3@mle_hyp[4:6]
HMM=norm.HMM.mle(x=x,m=m0,sigma0=sigma0)
round(HMM$sigma,digits=3) #without permuting the states s.t. sigma_1<sigma_2<sigma_3
round(HMM$gamma,digits=3)

#----------------------------------#
#             3. STOP              #
#----------------------------------#


#after permuting the states
p.hat.sigma=sort(HMM$sigma)
p.hat.sigma
HILF=HMM$gamma
hilf=rank(HMM$sigma)
HILF1=HMM$gamma[hilf,]
HILF2=HILF1[,hilf]
```

```
p.hat.gamma=HILF2
round(p.hat.gamma,6)


#we fit the most likely sequence of states, given the parameters via Viterbi algorithm
estimate.states=norm.HMM.viterbi(x,m=3,HMM$sigma,HMM$gamma)
estimate.states


#-----------------------------------#
#             4. STOP               #
#-----------------------------------#


#permute the states s.t. sigma_1<sigma_2<sigma_3
hilf=rank(HMM$sigma)
p.estimate.states=hilf[estimate.states]

y=x
s=p.estimate.states
par(fig=c(0,1,0.3,1))
par(las=1)
par(mar=c(0,5,2,2))
plot.ts(y,xlab="",xaxt="n",ylab="log Returns")
par(fig=c(0,1,0,0.3),new=TRUE)
par(mar=c(5,5,0,2))
plot(1:length(y),s,col=2,type="h",yaxt="n",xlab="",ylim=c(0,5),ylab="state")
axis(2,at=c(1,2,3),labels=c(1,2,3))


#-----------------------------------#
#             5. STOP               #
#-----------------------------------#


#simulate with estimated parameters
m0=3
n0=1000
p.hat.sigma #estimators after permuting the states s.t. sigma_1<sigma_2<sigma_3
p.hat.gamma

simHMM=norm.HMM.generate_sample(n=n0,m=m0,p.hat.sigma,p.hat.gamma)
simHMM$x
simHMM$s

y=simHMM$x
s=simHMM$s
par(fig=c(0,1,0.3,1))
par(las=1)
par(mar=c(0,5,2,2))
plot.ts(y,xlab="",xaxt="n",ylab="Simulated log Returns")
par(fig=c(0,1,0,0.3),new=TRUE)
par(mar=c(5,5,0,2))
plot(1:length(y),s,col=2,type="h",yaxt="n",xlab="",ylim=c(0,5),ylab="state")
axis(2,at=c(1,2,3),labels=c(1,2,3))
```

```
#-----------------------------------#
#             6. STOP               #
#-----------------------------------#

#---------------------
#Forecast Distribution
#---------------------

#-------------------------------------------------------------#
#comparison of forecast distribution and marginal distribution #
#->should be equal if time horizon tends to infinity           #
#-------------------------------------------------------------#

dev.off()
FORECAST.HMM=norm.HMM.forecast.prop(x,m=3,sigma=HMM$sigma,gamma=HMM$gamma,H=1000)

alpha = HMM$delta

sdeviation= HMM$sigma

horizon=100
x.pred=seq(-10,10,length=horizon)
for(i in 1:horizon){
 alpha1=FORECAST.HMM$xi[,i]
 y=alpha[1]*dnorm(x.pred,sd=sdeviation[1])+alpha[2]*dnorm(x.pred,sd=sdeviation[2])+alpha[
 plot(x.pred,y,type="l",main=paste("Horizon:",i))
 yy=alpha1[1]*dnorm(x.pred,sd=sdeviation[1])+alpha1[2]*dnorm(x.pred,sd=sdeviation[2])+alp
 lines(x.pred,yy,col=2,lty=3)
 Sys.sleep(0.1)
}
```

`calc_weights` *Estimating the weights of the asymptotic χ^2-mixture of the test statistic*

Description

The asymptotic distribution of the likelihood ratio is (under null hypothesis) a mixture of m0+1 χ^2 distributions and depends on the parameters of the null distribution of the data. This function calculates these weights. Since we can estimate the parameters consistently, we can replace them by the mle under the hypothesis.

Usage

```
calc_weights(p0, theta0)
```

Arguments

`p0`	true or estimated weights of the normal mixture
`theta0`	true or estimated standard deviations of the normal mixture

Value

weights of the asymptotic χ^2-mixture of the test statistic

Examples

```
calc_weights(p0=c(0.5,0.5),theta0=c(1,5))
```

`dnorm.scale` *Application of the normal distribution as a scale family*

Description

This function returns the density of a normal distribution with mean=0 and standard deviation sigma.

Usage

```
dnorm.scale(x, sigma)
```

Arguments

`x`	x value for the density to evaluate
`sigma`	given standard deviation

`EM-Test-class` *Class "EM-Test"*

Description

Objects of this class are the returned value of `em_test_normal_sigma`.

Slots

`data`**:** the given dataset

`mean_data`**:** the mean of the original dataset

`m0`**:** the number of regimes unter the hypothesis

`p_value`**:** p-value of the test

`LR`**:** the likelihood ratio value

`loglikelihood_hyp`**:** value of the likelihoodfunction under the hypothesis

`loglikelihood_alter`**:** value of the likelihoodfunction under the alternative

`mle_hyp`**:** estimated stationary mixing proportions and standard deviations of the regimes

`mle_alter`**:** same as mle_hyp, but for 2*m0 regimes

Methods

We provide the following methods for a EM-Test object:

see `plot-methods`

plot the visualisation of an object

em_test_normal_sigma

Testing the number of regimes in a Hidden Markov Model

Description

The function implements a likelihood ratio based test for the number of regimes of a Hidden Markov Model (HMM). It tests the hypothesis of m0 regimes versus the alternative of 2*m0 regmies by calculating the mle's with the likelihood function under independence. The test statistic converges (under null hypothesis) in distribution to

$$\alpha_0\chi_0^2 + \ldots + \alpha_{m0}\chi_{m0}^2,$$

where χ_0^2 is a point mass a zero and χ_k^2 a chi-square distribution with k degrees of freedom. The weights α_k a calculated by the function `calc_weights` in dependece of the estimation under the hypothesis. For further details see the paper below.

If you use return time series you should rescale them to percentage values.

Usage

```
em_test_normal_sigma(dataset, m0, start_param = NA)
```

Arguments

`dataset`	the given dataset
`m0`	number of regimes under the hypothesis
`start_param`	optionally a starting value for the optimisation c(p_1,...p_(m0-1),sigma_1,...,sigma_m0)

Value

returns an object of class EM-Test (see: `EM-Test-class`)

References

cite paper EM-Test for HMM

Examples

```
x = c(rnorm(500,sd=1),rnorm(500,sd=5),rnorm(500,sd=10))
erg = em_test_normal_sigma(x,2)
erg
plot(erg)
```

`norm.HMM.forecast.prop`
Forecast

Description

The forecast distribution can be written as a mixture of the state-dependent pdfs.

Usage

```
norm.HMM.forecast.prop(x, m, sigma, gamma, delta = NULL, xrange = NULL, H = 1, .
```

Arguments

`x`	sample
`m`	number of states of the hidden Markov chain
`sigma`	state dependent variances (estimated via MLE)
`gamma`	transition probability matrix (tpm) of the hidden Markov chain (usually estimated via MLE)
`delta`	initial distribution of the hidden Markov chain (DEFAULT: stationary distribution corresponding to the tpm gamma)
`xrange`	...
`H`	time horizon
`...`	...

Value

a mxH matrix xi where xi[,h] corresponds to the weights of the forecast distribution of X_t+h|X_t,X_t-1,...

`norm.HMM.generate_sample`
Generates a realization of a Normal Hidden Markov Model

Description

This function generates a realization of a normal HMM with m states, where the state dependent distributions have a fixed mean of zero and switching standard deviations. The returned value is a list with the observable and the non-observable process.

Usage

```
norm.HMM.generate_sample(n, m, sigma, gamma, delta = NULL)
```

Arguments

n	length of the sample to be generated
m	number of states of the hidden Markov chain
sigma	m component vector with standard deviations
gamma	transition probability matrix of the hidden Markov chain
delta	initial distribution of the hidden Markov chain (DEFAULT: stationary distribution of the hidden Markov chain)

Value

The returned value is a list with components:

x	realisations of the observable process (length n)
state	realisations of the non-observable process (length n)

Examples

```
#set tpm
gamma=matrix(ncol=3,nrow=3)
gamma[1,1]=0.6
gamma[1,2]=0.1
gamma[1,3]=1-sum(gamma[1,1:2])
gamma[2,1]=0.1
gamma[2,2]=0.8
gamma[2,3]=1-sum(gamma[2,1:2])
gamma[3,1]=0.2
gamma[3,2]=0.05
gamma[3,3]=1-sum(gamma[3,1:2])
gamma

#set length
n=1000

#set number of states
m=3

#set regime dependant standard deviations
sigma=c(0.2,1,10)

#genrate realization
HMM=norm.HMM.generate_sample(n,m,sigma,gamma)

y=HMM$x
s=HMM$state
```

---

norm.HMM.mle *Fittig the parameters of a Normal Hidden Markov Model*

Description

Estimates the parameters of the model using numerical minimization of the negative log likelihood.

Usage

```
norm.HMM.mle(x, m, sigma0, gamma0, ...)
```

Arguments

x	sample
m	number of states of the hidden Markov chain
sigma0	starting parameters for the minimization (vector of length m)
gamma0	starting parameters for the minimization (mxm-Matrix)
...	...

Value

The returned value is a list with components:

sigma	MLE for the state dependent variances
gamma	MLE for the tpm
delta	MLE for the initianl distribution of the hidden Markov chain
code	an integer indicating why the optimization process in nlm() terminated
mllk	(negative) maximum value of the log likelihood
AIC	Information criterion (AIC)
BIC	Information criterion (BIC)

---

norm.HMM.pn2pw *Natural parameters to working parameters*

Description

Transforms the natural parameters of an m-state Hidden Markov Model into a vector of unconstrained working parameters.

Usage

```
norm.HMM.pn2pw(m, sigma, gamma)
```

Arguments

m	number of states
sigma	m component vector with standard deviations
gamma	tpm

Value

A list with unconstrained working parameters.

norm.HMM.pw2pn *Working parameters to natural parameters*

Description

Performs the inverse transformation to `norm.HMM.pn2pw`

Usage

```
norm.HMM.pw2pn(m, parvect)
```

Arguments

`m`	number of states
`parvect`	the parameters as a vector -> c(sigma,as.vector(tmp))

Value

A list with natural parameters.

norm.HMM.viterbi *Viterbi algorithm*

Description

Computes the most likely sequence of states, given the parameters and observations.

Usage

```
norm.HMM.viterbi(x, m, sigma, gamma, delta = NULL, ...)
```

Arguments

`x`	sample
`m`	number of states of the hidden Markov chain
`sigma`	state dependent variances (usually estimated via MLE)
`gamma`	transition probability matrix (tpm) of the hidden Markov chain (usually estimated via MLE)
`delta`	initial distribution of the hidden Markov chain (DEFAULT: stationary distribution of the Markov chain corresponding to the tpm gamma)
`...`	...

Value

Vector vi of length n, where vi[i] gives the state at time i.

plot-methods *Plot method for EM-Test objects*

Description

This method allows you to use the plot function for an object of the EM-Test-class.

Methods

x = "EM-Test" The function plots three densities: the empirical, the density of the estimation under the hypothesis and the the density of the estimation under the alternative.

Examples

```
x = c(rnorm(500,sd=1),rnorm(500,sd=5),rnorm(500,sd=10))
erg = em_test_normal_sigma(x,2)
erg
plot(erg)
```

price2logreturn *Calculates log-returns*

Description

Either for a vector of stock prices (use variable prices) or a link directing to csv file with prices in a column called Adj.Close (use variable link) this function calculates the corresponding logarithmic returns.

Usage

```
price2logreturn(link = NA, prices = NA)
```

Arguments

link	link (as a string) directing to csv file with prices in a column called Adj.Close with values from new to old
prices	vector of prices with values from old to new.

Value

vector of log-returns in percent

`stock_returns` *Dataset of daily stock returns*

Description

The dataset contains the daily log-returns of a german blue chip (14.07.2006 - 14.07.2010), including the 4th quarter of 2008 ($i = 550, \ldots, 610$).

Usage

```
data(stock_returns)
```

Format

The format is: num [1:1013] 0.567 2.279 1.27 -0.638 1.362 ...

Examples

```
data(stock_returns)
plot(density(stock_returns))
```

Index