Package 'MitISEM'

July 10, 2017

Type Package

Title Mixture of Student t Distributions using Importance Sampling and Expectation Maximization

Version 1.2

Date 2017-07-10

Author N. Basturk, L.F. Hoogerheide, A. Opschoor, H.K. van Dijk

Maintainer N. Basturk <n.basturk@maastrichtuniversity.nl>

Description Flexible multivariate function approximation using adapted Mixture of Student t Distributions. Mixture of t distribution is obtained using Importance Sampling weighted Expectation Maximization algorithm.

License GPL (>= 3)

LazyLoad yes

Imports mvtnorm

Suggests AdMit

URL https://www.maastrichtuniversity.nl/n.basturk

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-10 18:36:41 UTC

R topics documented:

MitISEM-package — *Mixture of Student t Distributions using Importance Sampling and Expectation Maximization*

Description

Approximates a univariate or multivariate density using mixture of Student `t` distributions, achieved by Importance Sampling and Expectation Maximization algorithms

Details

Package:	MitISEM
Type:	Package
Version:	1.2
Date:	2017-07-10
License:	GPL (>= 3)

Flexible multivariate function approximation using adapted Mixture of Student t Distributions. Mixture of t distribution is obtained using Importance Sampling weighted Expectation Maximization algorithm.

Author(s)

N. Basturk, L.F. Hoogerheide, A. Opschoor, H.K. van Dijk

Maintainer: N. Basturk

References

Basturk, N., Grassi, S., Hoogerheide, L., Opschoor, A. and Van Dijk, H. K. (2017) The R Package MitISEM: Efficient and Robust Simulation Procedures for Bayesian Inference. *Journal of Statistical Software*, 79(1): 1-39. doi: 10.18637/jss.v079.i01.

Hoogerheide L., Opschoor, A. and Van Dijk, H. K. (2012) A Class of Adaptive Importance Sampling Weighted EM Algorithms for Efficient and Robust Posterior and Predictive Simulation. *Journal of Econometrics*, 171(2): 101-120. `http://www.sciencedirect.com/science/article/pii/S0304407612001583`.

MargLik — *Marginal Likelihood calculation using Importance Sampling and mixture of Student-t densities as candidate*

Description

Calculation of marginal likelihoods using Importance Sampling, with a Mixture of Student-t candidate density. Calculated marginal likelihoods from two data samples can be used to get predictive likelihoods using `PredLik`.

Usage

```
MargLik(N=1e4,mit,KERNEL,...)
```

Arguments

`N`	integer > 100 number of draws for Importance Sampling
`mit`	Mixture of Student-t density for the full sample, list describing the mixture of Student-t. See `isMit`. The mixture density can be obtained from `MitISEM` or `SeqMitISEM`
`KERNEL`	Posterior kernel to be approximated. See *Details*. A `log` argument must exist. The function must return log-density if `log=TRUE`.
`...`	other arguments to be passed to `KERNEL`

Details

If `MargLik` is used to get the Marginal Likelihood of a single model, `KERNEL` must be the exact posterior density (including the scaling constant) of parameters.

If `MargLik` is used as an intermediate step, for instance for calculating predictive likelihoods, `KERNEL` can be a posterior kernel or the exact posterior density of parameters. See `PredLik`.

Value

list containing:

`ML.mean`	Marginal likelihood (posterior mean) $\times 10^{scale}$
`ML.NSE`	Numerical Standard Error for mean Marginal likelihood $\times 10^{scale}$
`scale`	integer > 0 providing the scaling for predictive likelihood. (scaling may be necessary for numerical accuracy)

See Also

`isMit`,`PredLik`,`MitISEM`,`SeqMitISEM`

Examples

```
mit  <- list(p=1,mu=matrix(1),Sigma=matrix(0.1),df=5)
data <- rnorm(100,1)
KERNEL <- function(theta,data,log=TRUE){
  if(is.vector(theta))
    theta = matrix(theta,nrow=1)
  r <- apply(theta,1,function(x,data)(sum(dnorm(data,x,log=log))),data=data)
}
MargLik(N=1000,mit=mit,KERNEL=KERNEL,data=data)
```

Mit *The 'mit' object*

Description

Function to check if mit generalized mixture of t densities is well-defined. The mit object is designed to be used in the rest of the MitISEM package functions

Usage

```
isMit(mit)
```

Arguments

mit	an object to be tested

Details

Argument mit is a list describing the mixture of Student-t distributions with the following components:

p vector (of length H) of mixture probabilities.

mu matrix (of size $H \times d$) containing the vectors of modes (in row) of the mixture components.

Sigma matrix (of size $H \times d^2$) containing the scale matrices (in row) of the mixture components.

df vector (of length H) degree of freedom parameters for each Student-t component (double > 0).

Value

logical, TRUE if mit definition is correct, FALSE otherwise

Examples

```
# a correct Mit definition returns 'TRUE'
H      <- 2
p      <- runif(H)
p      =  p / sum(p)
mu     <- matrix(seq(1:H),H,1)
Sigma  <- matrix(runif(H^2),H,H)
df     <- seq(1:H)
isMit(mit=list(p=p,mu=mu,Sigma=Sigma,df=df))

# an incorrect Mit definition returns 'FALSE'
mu   = t(mu)
isMit(mit=list(p=p,mu=mu,Sigma=Sigma,df=df))
```

MitISEM — *Mixture of Student-t distributions using Importance Sampling weighted Expectation Maximization steps*

Description

Approximates a k dimensional function/kernel by a mixture of student-t distributions using Importance Sampling weighted Expectation Maximization steps.

Usage

```
MitISEM(KERNEL,mu0,Sigma0=NULL,df0=1,mit0=NULL,control=list(),...)
```

Arguments

`KERNEL`	Function to be approximated. First argument should be the parameter matrix. A `log` argument should exist such that the function returns log-density if `log=TRUE`
`mu0`	vector of length k, starting points for approximation
`Sigma0`	(optional) $k \times k$ dimensional positive definite symmetric initial scale matrix. Default: matrix is obtained by the `BFGS` algorithm.
`df0`	(optional) double > 0 initial degree of freedom of the Student-t density. Default: $df0 = 1$.
`mit0`	(optional) initial mixture density defined. See *Details*.
`control`	(optional) list of control parameters for IS and EM optimization and stopping rule of the `H` component mixture of `t` densities. See *Details*
`...`	other arguments to be passed to `KERNEL`

Details

Providing `mit0` argument makes arguments `mu0`, `Sigma0` and `df0` obsolete. Argument `mit0` (if provided) should include the following components (see `isMit`):

- `p` vector (of length H) of mixture probabilities.
- `mu` matrix (of size $H \times k$) containing the vectors of modes (in row) of the mixture components.
- `Sigma` matrix (of size $H \times k^2$) containing the scale matrices (in row) of the mixture components.
- `df` vector (of length H) degree of freedom parameters of the Student-t components. Each element should be above 0

Value `mit` has the same structure as `mit0`, where H and parameters of the mixture density are optimized.

Optional argument `control` can provide several optimization parameters:

- `N` integer (>100) number of draws used in the simulation. Default: `N=1e5`.
- `robust.N` logical indicating if robust draws are used if `robust.N=TRUE` (default), simulations are repeated to get `N` draws with finite `KERNEL` values.

`Hmax` integer>0 maximum number of components. Default: `H=10`.

`StopMethod` string, `CV` (default) or `AR` defining type of stopping criteria for MitISEM approximation. `CV` method stops the algorithm if the coefficient of variation in IS weights converges. `AR` method stops the algorithm if the (expected) acceptance rate given the current MitISEM approximation and function `KERNEL` converges.

`CVtol` double in $(0,1)$ convergence criteria for `CV` method. Higher values lead to faster convergence but worse approximation. Default: $CVtol = 0.1$, algorithm stops if $StopMethod = CV$ and the change in coefficient of variation is below 10%.

`ARtol` double in $(0,1)$ convergence criteria similar to `CVtol`, used if $StopMethod = AR$. Default: $ARtol = 0.1$.

`trace` logical to print partial output. Default: $trace = FALSE$, no tracing information.

`trace.init` logical to print output of the first student-t optimization. Default: $trace = FALSE$, no tracing information.

`maxit.init` double, maximum number of iterations in the first student-t optimization. Default: $maxit.init = 1e4$.

`reltol.init` double, relative tolerance in the first student-t optimization. Default: $reltol.init = 1e-8$.

`maxit.EM` integer>0, maximum number of iterations for the EM algorithm. Default: $maxit.EM = 1000$.

`tol.EM` double>0, tolerance for EM steps' convergence, Default: $tol.EM = 0.001$.

`trace.EM` logical to print partial output during the IS-EM algorithm. Default: $trace.EM = FALSE$, no tracing information.

`optim.df` logical to optimize degrees of freedom of the Student-t components. Default: $optim.df = TRUE$ df are optimized. Note: Keeping degrees of freedom in low values may be desired if the approximation is used in a rejection sampling. If $optim.df = FALSE$, degree of freedom of all student t components are fixed at `df0`.

`inter.df` increasing vector of length 2 range of search values for df optimization, active if $optim.df = TRUE$. Default: $inter.df = (0.01, 30)$.

`tol.df` double >0, tolerance for degree of freedom optimization, active if $optim.df = TRUE$. Default: $tol.df = 0.0001$

`maxit.df` integer >0 maximum number of iterations for degree of freedom optimization, active if $optim.df = TRUE$. Default: $maxit.df = 1e3$.

`trace.df` logical to print partial output during degree of freedom optimization, active if $optim.df = TRUE$. Default: $trace.df = FALSE$.

`tol.pr` double in [0,1), minimum probability required to keep mixture components. Default: $tol.pr = 0$ all mixture components are kept.

`ISpc` double in (0,1), fraction of draws to construct new component. Default: $ISpc = 0.1$.

`Pnc` double in (0,1), initial probability of the new component. Default: $Pnc = 0.1$.

Value

list containing:

`mit` (list) optimal mixture density with H mixture components. See *Details*.

`CV`	vector of length H with coefficient of variation obtained from each addition of mixture components.
`time`	(double) processed time.
`summary`	(data.frame) summary information on construction of components, processed time and CV.

References

Basturk, N., Grassi, S., Hoogerheide, L., Opschoor, A. and Van Dijk, H. K. (2017) The R Package MitISEM: Efficient and Robust Simulation Procedures for Bayesian Inference. *Journal of Statistical Software*, 79(1): 1-39. doi: 10.18637/jss.v079.i01.

Hoogerheide L., Opschoor, A. and Van Dijk, H. K. (2012) A Class of Adaptive Importance Sampling Weighted EM Algorithms for Efficient and Robust Posterior and Predictive Simulation. *Journal of Econometrics*, 171(2): 101-120. `http://www.sciencedirect.com/science/article/pii/S0304407612001583`.

See Also

`isMit`

Examples

```
require(graphics)
set.seed(1234);
# define Gelman Meng Kernel
GelmanMeng <- function(x, A = 1, B = 0, C1 = 3, C2 = 3, log = TRUE){
    if (is.vector(x))
    x <- matrix(x, nrow = 1)
    r <- -.5 * (A * x[,1]^2 * x[,2]^2 + x[,1]^2 + x[,2]^2
                - 2 * B * x[,1] * x[,2] - 2 * C1 * x[,1] - 2 * C2 * x[,2])
    if (!log)
    r <- exp(r)
    as.vector(r)
}
# get MitISEM approximation
mu0 <-c(3,4)
app.MitISEM <- MitISEM(KERNEL=GelmanMeng,mu0=mu0,control=list(N=2000,trace=TRUE))
mit=app.MitISEM$mit

# plot approximation (components and full approximation)
MitISEM.plot.comps <- function(mit,x1,x2,log=FALSE){
  Mitcontour <- function(x1,x2,mit,log=FALSE){
    dmvgt(cbind(x1,x2),mit=mit,log=log)
  }
  H <- length(mit$p)
  K <- ncol(mit$mu)
  cols <- 1:H
  for (h in 1:H){
    mit.h  <-list(p=1,mu=matrix(mit$mu[h,],1,K),
                Sigma=matrix(mit$Sigma[h,],1,(K^2)),df=mit$df[h])
    z      <- outer(x1,x2,FUN=Mitcontour,mit=mit.h)
    contour(x1,x2,z,col=h,lty=h,labels="",add=(h!=1),
            xlab="x1",ylab="x2",main='MitISEM approximation components')
  }
  legend("topright",paste("component ",1:H),lty=cols,col=cols,bty='n')
  z <- outer(x1,x2, FUN=Mitcontour,mit=mit)
  image(x1,x2,z,las=1,col=gray((20:0)/20),main='MitISEM approximation')
}
x1 <- seq(-2,6,0.05)
x2 <- seq(-2,7,0.05)
MitISEM.plot.comps(mit,x1,x2)

## Not run:
  # Bayesian inference of the GARCH model using MitISEM and Importance Sampling
  library(AdMit) # required for Importance Sampling
  library(tseries) # required for loading the data
  # load data : downloaded on 2013/01/18
 prices <- as.vector(get.hist.quote("^GSPC",quote="AdjClose",start="1998-01-02",end="2002-12-26"))
  data  <- 100 * (prices[-1] - prices[-length(prices)]) / (prices[-length(prices)])
  prior.GARCH<-function(omega,beta,alpha,
                        mu,log=TRUE){
    c1 <- (omega>0 & omega <1 & beta>=0 & alpha>=0)
    c2 <- (beta + alpha< 1)
    c3 <- (mu>-1 & mu<1)
    r1 <- c1 & c2 & c3
    r2 <- rep.int(-Inf,length(omega))
    r2[r1==TRUE] <- 0
    if (!log)
      r2 <- exp(r2)
    cbind(r1,r2)
  }
  post.GARCH <- function(theta,data,h1,log=TRUE){
    if (is.vector(theta))
      theta <- matrix(theta, nrow = 1)
    omega <- theta[,1]
    beta <- theta[,2]
    alpha <- theta[,3]
    mu <- theta[,4]
    N <- nrow(theta)
    pos <- 2:length(data)
    prior <- prior.GARCH(omega=omega,beta=beta,alpha=alpha,mu=mu)
    d <- rep.int(-Inf,N)
    for (i in 1:N){
      if (prior[i,1] == TRUE){
        h <- c(h1, omega[i] + alpha[i] * (data[pos-1]-mu[i])^2)
        for (j in pos){
          h[j] <- h[j] + beta[i] * h[j-1]
        }
        tmp <- dnorm(data[pos],mu[i],sqrt(h[pos]),log=TRUE)
        d[i] <- sum(tmp) + prior[i,2]
      }
    }
    if (!log) d <- exp(d)
    as.numeric(d)
  }
  theta <- c(.08, .86, .02, .03) # initial parameters for MitISEM
  names(theta)<-c("omega","beta","alpha","mu")
  h1 <- var(data) # initial data variance
  # MitISEM GARCH approximation
  cat("MitISEM GARCH results",fill=TRUE)
  cat('--------------------------',fill=TRUE)
  set.seed(1111)
  app.GARCH <- MitISEM(KERNEL=post.GARCH,
                       mu0=theta, control=list(trace=TRUE),h1=h1,
                       data=data)
  print(app.GARCH$summary)
  # Importance Sampling using MitISEM candidate
  cat('Importance Sampling result from MitISEM candidate',fill=TRUE)
  cat('--------------------------------------------------',fill=TRUE)
  set.seed(1111)
  IS.MitISEM.GARCH <- AdMitIS(N = 10e4,data=data,h1=h1,
                              KERNEL=post.GARCH,mit=app.GARCH$mit)
  print(IS.MitISEM.GARCH)

## End(Not run)
```

Mvgt *General student t distribution*

Description

Density and random generation for the general studen t distribution

Usage

```
dmvgt(theta, mit = list(), log=TRUE)
rmvgt(N,mit)
```

Arguments

theta	Vector of lenght N or $N \times k$ matrix of quantiles. If `theta` is a matrix, each row is taken to be a quantile.
N	number of observations
mit	list defining the mixture components. See `isMit` for how it should be defined.
log	logical; if TRUE (default), probabilities p are given as $\ln(p)$.

Value

`dmvgt` returns vector of size `N` with density values for each row of `theta`

`rmvgt` returns an $N \times k$ matrix of draws from the k-variate mixture of student t densities

See Also

isMit

Examples

```
H      <- 2
p      <- runif(H)
p      =  p / sum(p)
mu     <- matrix(seq(1:H),H,1)
Sigma  <- matrix(runif(H^2),H,H)
df     <- seq(1:H)
Ndraws <- rmvgt(N=10,mit=list(p=p,mu=mu,Sigma=Sigma,df=df))
pdraws <- dmvgt(theta=Ndraws,mit=list(p=p,mu=mu,Sigma=Sigma,df=df))
```

PredLik — *Predictive Likelihood calculation using Importance Sampling and mixture of Student-t densities as candidate*

Description

Calculation of predictive likelihoods using Importance Sampling, given subsample and full data sample and Mixture of Student-t candidate density. Predictive likelihood is calculated using the marginal likelihood from full sample and subsample. See MargLik.

Usage

```
PredLik(N=1e4,mit.fs,mit.ss,KERNEL,data.fs,data.ss,...)
```

Arguments

N	integer > 100 number of draws for Importance Sampling
mit.fs	Mixture of Student-t density for the full sample, list describing the mixture of Student-t. See isMit. The mixture density can be obtained from MitISEM or SeqMitISEM
mit.ss	Mixture of Student-t density for subsample. Must be defined as `mit.fs`.
KERNEL	Function/posterior to be approximated. `data` and `log` arguments must exist. The function must return log-density if `log=TRUE`. All data should be loaded in argument `data`
data.fs	Full data, vector (length $T1$) or matrix (size $T1 \times m$) with data values, $T1$ observations and m data series.
data.ss	Sample of data, vector (length $T2$) or matrix (size $T2 \times m$) with data values, $T2$ observations and m data series. $T2 < T1$.
...	other arguments to be passed to `KERNEL`

Details

Argument `KERNEL`

Value

list containing:

`PL`	Predictive likelihood $\times 10^{scale}$
`scale`	integer $>$ 0 providing the scaling for predictive likelihood. (scaling may be necessary for numerical accuracy)

References

Eklund, J. and Karlsson, S. (2007). Forecast combination and model averaging using predictive measures. *Econometric Reviews*, 26, 329-363.

Min, C. and Zellner, A. (1993). Bayesian and non-Bayesian methods for combining models and forecasts with applications to forecasting international growth rates. *Journal of Econometrics*, 56, 89-118.

See Also

`isMit`,`MargLik`,`MitISEM`,`SeqMitISEM`

`SeqMitISEM` *Sequential approximation using Mixture of Student-t distributions using Importance Sampling weighted Expectation Maximization steps*

Description

Approximates a k dimensional function/kernel using mixture of student-t distributions for the initial data sample and updated data samples sequentially

Usage

```
SeqMitISEM(data,KERNEL,mu0,Sigma0=NULL,df0=1,control.MitISEM=list(),control.seq=list(),
...)
```

Arguments

`data`	matrix (size $T \times m$) with data values, T observations and m data series
`KERNEL`	Function/posterior to be approximated. `data` and `log` arguments should exis. The function must return log-density if `log=TRUE`. All data should be loaded in argument `data`
`mu0`	vector of length k starting points. They should be defined as in `MitISEM`
`Sigma0,df0`	(optional) initial scale and degrees of freedom for the student t density. They should be defined as in `MitISEM`

control.MitISEM
(optional) control parameters passed to MitISEM. See MitISEM.

control.seq
control parameters for sequential updating of the MitISEM approximation. See *Details*.

...
other arguments to be passed to KERNEL.

Details

The optional argument control.seq can provide several optimization parameters:

T0 integer ($< T$) number of observations. Default: round(T/2).

tau vector of length t with iterative number of observations to add to the sample. Its elements should be positive integers, and $T0 + max(tau) <= T$ should hold. Default: tau=1, one single observation is added to the sample for Sequential MitISEM.

tol.seq double in $(0, 1)$ convergence criteria for sequential Coefficient of Variation convergence Default: tol.seq=0.2.

method $0, 1, 2$ method to select initial data sample. if method=0 initial sample is randomly selected. if method=1 first T0 observations are taken as initial sample (default). if method=2 last T0 observations are taken as initial sample.

trace logical to print partial output. Default: $trace = FALSE$, no tracing information.

References

Basturk, N., Grassi, S., Hoogerheide, L., Opschoor, A. and Van Dijk, H. K. (2017) The R Package MitISEM: Efficient and Robust Simulation Procedures for Bayesian Inference. *Journal of Statistical Software*, 79(1): 1-39. doi: 10.18637/jss.v079.i01.

Hoogerheide L., Opschoor, A. and Van Dijk, H. K. (2012) A Class of Adaptive Importance Sampling Weighted EM Algorithms for Efficient and Robust Posterior and Predictive Simulation. *Journal of Econometrics*, 171(2): 101-120. http://www.sciencedirect.com/science/article/pii/S0304407612001583.

See Also

MitISEM

Examples

```
## Not run:
  # Sequential MitISEM application for SP500 data
  # Calculates 50 predictive likelihoods for the mGARCH(1,1) model, SP500 data
  # For details see: 'The R package MitISEM: Efficient and Robust Simulation Procedures
  # for Bayesian Inference' by N. Basturk, S. Grassi, L. Hoogerheide, A. Opschoor,
  # H.K. van Dijk.
  library(tseries)
  source("PostmGARCH.R") # posterior of the model under flat priors

  # load data
  prices <- as.vector(get.hist.quote("^GSPC",quote="AdjClose",start="1998-01-02",
    end="2002-12-26"))
```

```
y <- 100 * (prices[-1] - prices[-length(prices)]) /  (prices[-length(prices)])
# Prior and posterior densities for the mixture of GARCH(1,1) model with
# 2 mixture components
prior.mGARCH<-function(omega, lambda, beta, alpha, p, mu, log=TRUE){
  c1 <- (omega>0 & omega<1 & beta>=0 & alpha>=0)
  c2 <- (beta + alpha< 1)
  c3 <- (lambda>=0 & lambda<=1)
  c4 <- (p>0.5 & p<1)
  c5 <- (mu>-1 & mu<1)
  r1 <- c1 & c2 & c3 & c4 & c5
  r2 <- rep.int(-Inf,length(omega))
  tmp <- log(2) # ln(1 / ( p(beta,alpha) * p(p) * p(mu))
  r2[r1==TRUE] <- tmp
  if (!log)
    r2 <- exp(r2)
  cbind(r1,r2)
}
post.mGARCH <- function(theta, data, h1, log = TRUE){
  if (is.vector(theta))
    theta <- matrix(theta, nrow = 1)
  omega <- theta[,1]
  lambda <- theta[,2]
  beta <- theta[,3]
  alpha <- theta[,4]
  p <- theta[,5]
  mu <- theta[,6]
  N <- nrow(theta)
  pos <- 2:length(data) # # observation index (removing 1st)
  prior <- prior.mGARCH(omega=omega,lambda=lambda,beta=beta,alpha=alpha,
    p=p,mu=mu)
  d <- rep.int(-500000,N)#fixme
  for (i in 1:N){
    if (prior[i,1] == TRUE){
      h <- c(h1, omega[i] + alpha[i] * (data[pos-1]-mu[i])^2)
      for (j in pos){
        h[j] <- h[j] + beta[i] * h[j-1]
      }
      sigma <- 1 / (p[i] + ((1-p[i]) / lambda[i]))
      tmp1 <- dnorm(data[pos],mu[i],sqrt(h[pos]*sigma),log=T)
      tmp2 <- dnorm(data[pos],mu[i],sqrt(h[pos]*sigma/lambda[i]),log=T)
      tmp <- log(p[i] * exp(tmp1) + (1-p[i]) * exp(tmp2))
      d[i] <- sum(tmp) + prior[i,2]
    }
  }
  if (!log)
    d <- exp(d)
  as.numeric(d)
}
# define data subsample
y.ss <- y[1:626]
# initial data variance
h1   <- var(y) # initial variance
N <- 1e3 # number of draws for predictive likelihood
```

```
  mu0 <- c(0.08, 0.37, 0.86, 0.03, 0.82, 0.03)  # initial parameters for MitISEM
  names(mu0) <- c("omega","lambda","beta","alpha","p","mu")
  set.seed(1234)
  cat("starting training subsample estimation", fill=TRUE)
  mit.ss <- MitISEM(KERNEL = post.mGARCH, mu0 = mu0, data = y.ss, h1 = h1,
    control=list(trace=TRUE))$mit
  cat("starting full sample estimation", fill=TRUE)
  mit.fs <- MitISEM(KERNEL = post.mGARCH, mu0 = mu0, data = y,    h1 = h1,
    control=list(trace=TRUE))$mit
  cat("starting predictive likelihood calculation", fill=TRUE)
  N <- 1000  # number of simulations for IS
  rep <- 50  # times to replicate application
  set.seed(1111)
  Mcompare.MitISEM <- PredLik(N,mit.fs,mit.ss,post.mGARCH,y,y.ss,h1=h1)
  # REPLICATE PRED LIKELIHOOD CALCULATION SEVERAL TIMES
  for(i in 2:rep){
    tmp <- PredLik(N,mit.fs,mit.ss,post.mGARCH,y,y.ss,h1=h1)
    Mcompare.MitISEM=mapply(rbind,Mcompare.MitISEM,tmp,SIMPLIFY=FALSE)
    if(i
      cat("rep MitISEM",i,fill=TRUE);
  }
  # REPORT MEAN AND STANDARD DEVIATION
  Means.MitISEM <- mapply(colMeans,Mcompare.MitISEM,SIMPLIFY=FALSE)
  scales <- rep(0,2)
  tmp <- Means.MitISEM[[1]]
  while(floor(tmp)==0){
    scales[i] = scales[i]+1
    tmp = tmp * 10
  }
  # average predictive likelihood and NSE from 50 repetitions
  Adj.Mcompare.MitISEM = Mcompare.MitISEM
  NSE.MitISEM <- sqrt(apply(Adj.Mcompare.MitISEM[[1]],2,var)/rep)
  table1 <- c(colMeans(Adj.Mcompare.MitISEM$PL),NSE.MitISEM)
  table1 =  rbind(rep(Adj.Mcompare.MitISEM$scale[1],2),table1)
  rownames(table1) = c("scale (10^scale)","value")
  colnames(table1) = c("Pred Lik","NSE")
  cat("Pred. Likelihood and NSE values are multiplied by 10^(scale)", fill = TRUE)
  print(round(table1,4))
  cat("number of student t components for full sample and training sample estimation",
    fill = TRUE)
  table2 <- cbind(length(mit.ss$p), length(mit.fs$p))
  colnames(table2) <- c("full sample", "training sample")
  print(round(table2,0))

## End(Not run)
```

Index