Package ‘NHMM’

June 30, 2020

Type Package

Title Bayesian Non-Homogeneous Markov and Mixture Models for Multiple Time Series

Version 3.11

Date 2020-07-01

Author Tracy Holsclaw

Acknowledgements This work was supported by a grant from the U.S. Department of Energy, (through the Earth System Models (EaSM) program.(This package was developed under the direction of Padhraic Smyth at the Department of Computer Science and Statistics at the University of California, Irvine and in collaboration with Andrew Robertson and Arthur Greene of the International Research Institute for Climate and Society at The Earth Institute at Columbia University.

Maintainer Tracy Holsclaw <iamrandom@iamrandom.com>

Description Holsclaw, Greene, Robertson, and Smyth (2017) <doi:10.1214/16-AOAS1009>. Bayesian HMM and NHMM modeling for multiple time series. The emission distribution can be mixtures of Exponential, Gamma, Poisson, or Normal distributions, and zero inflation is possible.

License GPL (>= 3)

Imports Rcpp (>= 0.11.0)

LinkingTo Rcpp

Depends BayesLogit, msm, MCMCpack, MASS

URL http://iamrandom.com/nhmm-package

RoxygenNote 7.1.0

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-06-30 19:30:02 UTC

R topics documented:

HMM *Bayesian Homogeneous Markov Model (NHMM)*

Description

HMM calculates an HMM for multiple sequences of data. The sequences can actually be short sets of equal length sequences (subseq). A set of input variables (W) can be included to influence the mixture proportions of the emission distributions. The HMM follows the general weather state formulation of Hughes and Guttorp but in a Bayesian fashion. All parameters are sampled via Gibbs steps (latent variables such that no tuning is needed.) The W variable coefficients are sampled through an ordered Mulinomial probit (Albert and Chib). The X variable coefficients are sampled through an unordered Multinomial logit model Polya-Gamma formulation (Polson, Scott, Windle). The hidden states are sampled through a blocked Gibbs sampler.

Usage

```
HMM(
  y,
  subseq = NULL,
  dirprior = NULL,
  K = 2,
  iters = 1000,
  burnin = 200,
  emdist = "normal",
  nmix = 1,
  delta = FALSE,
  W = NULL,
  psipriorm = NULL,
  psipriorp = NULL,
  priors = NULL,
  outdir = NULL,
  ymiss = FALSE,
  yrep = 0,
  ypred = 0,
  Wp = NULL,
  pT = NULL,
  yhold = NULL
)
```

Arguments

`y` T by J matrix of data (J=1 is sufficient) - missing data is denoted with NA

`subseq` [optional] if y is actually a set of subsequences then give the length of those sequences (122 for JJAS) (365 is not it!). Default is subseq=T.

`dirprior` [optional] prior for Dirichlet prior on the rows of the transition matrix only for the HMM, must be size KxK. If not supplied a flat prior is used.

`K` number of states (default=2)

`iters` number of iterations to keep after burn in (default=1000)

`burnin` the number of burn in (default=200)

`emdist` emission distribution: "normal", "poisson", "gamma" actual choices are Normal, Poisson, Gamma, Exponential, or finite mixtures of mixtures or zero inflated version of any of these

`nmix` [optional] number of mixture components for emdist, default is one (do not include delta)

`delta` [TRUE/FALSE] TRUE-if we are using a zero inflated distribution (adds a delta function at zero as the first mixture component)

`W` [optional] is an A by T by J array of emission input data (A different inputs), missing values are not allowed, do not include an intercept term here The mixture components of emission depend on W.

`psipriorm` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the mean of the Normal prior for the beta coefficients.

`psipriorp` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the precision prior(1/sig^2) of the Normal prior for the beta coefficients.

`priors` priors for emission components, each state can have a different prior dimension 5 by nmix by K by J (some of the 5 dimensions are filled with zeros for some distributions) Normal(mu,sig2) or reference prior: mu~Normal(a.mu, 1/b.mu^2), sig2~IG(a.sig,b.sig) -the first 4 rows are used [a.mu, b.mu, a.sig, b.sig] but the 5th row is not used but must be present. -reference prior: priors=NULL

Poisson(lambda): lambda~Gamma(a.lam,b.lam) -the first 2 rows are used [a.lam, b.lam] but the 3rd-5th rows are not used but must be present.

Gamma(alpha,beta): alpha~Gamma(a.alpha,b.alpha) beta~Gamma(a.beta,b.beta) -if the second row is supplied as NA then the first row supplies fixed first parameters for the Gamma this is the suggested parametrization of the Gamma. Estimating the first parameter of the Gamma distribution switches to an MCMC that is not as stable. -to create an Exponential distribution the first row is set to 1 and the second is set to NA then the third and fourth rows contain the (a.lam, b.lam) priors for the parameter of the Exponential if prior=NULL then these are set to ones. setting these parameters to zeros is not a good idea. -a non-informative prior may not work, use a matrix of ones for a lowly informative prior instead of zeros. -the 5th row is for the tuning parameter for the MCMC (try 0.5) if a two parameter Gamma is desired. A 2 parameter Gamma needs an informative prior and results may be more unstable. -default prior=NULL is the Exponential distribution with lowly informative prior of ones

`outdir` [optional] can output each set of parameters to output files in a directory use this with larger dimension data sets or with large number of iterations the output will be written line by line and not overburden the memory limit outdir needs to end with a slash or double slash depending on OS.

`ymiss` [optional-TRUE/FALSE] if outdir is specified then draws for any missing data points will be saved to ymiss-J*.txt. There will be one data point for each sequence of length Tmiss (the amount missing from that sequence). Each row of the output file will be an iteration of the algorithm after burnin is removed.

`yrep` [optional] number (ie. 100,200,or 500) of output replicate data sets to print to outdir. The replicates will be the same dimension as y and start after burn in period. Default is zero. These replicate data sets are generated from the same input variables. - must be shorter than iters

`ypred` number of predictive sets (ie. 0,100, 200, 500) - must be shorter than iters Predicted chains: will produce ypred set of predictive [pT by J] to print to outdir. yrep uses the same inputs to make replications, this uses new inputs to make predictions over a different time span (pT) Also outputs a set of predictive z values or length (pT by ypred)

`Wp` predictive set of Ws of length pT [A by pT by J] -missing valus are not allowed -ensure that the Wp inputs are in the same order as W

`pT` the length of the new sequence

`yhold` [optioinal] a sequence of y observed values [pT by J], held out data that is of length ypred that is used to compute the predictive log score (PLS) which is a metric like BIC (ie. hold out last 10 missing data values are filled in with mean PLS value. PLS is the average PLS across sequences.

Value

my.hmm object

Examples

```
## Gamma or Exponential
### because we do not supply "priors" as an input it fits an Exponetial distribution
  ## Not run:
data(NHMMdata)
attach(NHMMdata)

my.hmm1=HMM(y=ygamma,  K=3, iters=100, burnin=10, emdist="gamma",
            nmix=3, delta=TRUE)
OBIC(my.hmm1)
zz=Oz(my.hmm1)  #compare with the truth zgamma
qq=OQQ(my.hmm1)
pp=OWcoef(my.hmm1,FALSE)
```

```
tt=Oemparams(my.hmm1,FALSE)


 ## Normal
 my.hmm2=HMM(y=ynormal, subseq=100,  K=3, iters=100, burnin=10,
             emdist="normal", nmix=2, delta=FALSE)
 OBIC(my.hmm2)

 ## Poisson
 my.hmm3=HMM(y=ypoisson, K=3, iters=100, burnin=10, emdist="poisson",
             nmix=2, delta=FALSE)
 OBIC(my.hmm3)

 ## Predictive estimation - make 15 predictive data sets (new X) and 20 replicate data sets (same X)
 filelocation="C:\\Users\\iamrandom\\Desktop\\here\\"
 my.hmm5=HMM(y=ygamma,  W=tW, K=3, iters=100, burnin=10,
           emdist="gamma", nmix=3, delta=TRUE,
         outdir=filelocation, pT=200, yrep=20,  Wp=Wp1, ypred=15)
 OBIC(my.hmm5)
 pp=OWcoef(my.hmm5,filelocation)

 ## Gamma with fixed first variables nmix=2
 nmix=2; K=3; J=dim(ygamma)[2]
 prior1=array(1,dim=c(5,nmix,K,J));  prior1[1,1,,]=1;  prior1[1,2,,]=2;  prior1[2,,,]=NA
 my.hmm6=HMM(y=ygamma, priors=prior1, K=3, iters=100, burnin=10,
             emdist="gamma", nmix=2, delta=TRUE)
 OBIC(my.hmm6)
 Oemparams(my.hmm6)


 ### my.nhmm7 (K=3)  (yhold is the last 10% of the data)
 filelocation="C:\\Users\\iamrandom\\Desktop\\here\\"
 my.hmm7=HMM(y=ygamma[1:1800,],   W=array(tW[,1:1800,],
             dim=c(1,1800,15)), K=3, iters=50, burnin=10,
            emdist="gamma", nmix=3, delta=TRUE, outdir=filelocation,
             ymiss=TRUE, yrep=10, pT=200,
            Wp=array(tW[,1801:2000,],dim=c(1,200,15)), ypred=10,
             yhold=ygamma[1801:2000,])
 OBIC(my.hmm7)

 # run it with K=3 and then K=1 and compare using both BIC and PLS

## End(Not run)
```

NHMM *Bayesian Non-homogeneous Markov Model (NHMM)*

Description

NHMM calculates an NHMM for multiple sequences of data. The sequences can actually be short sets of equal length sequences (subseq). The traditional input variables (X) influence the non-homogenous transition probabiities of the model. An additional set of input variables (W) can be included to influence the mixture proportions of the emission distributions. The NHMM follows the general weather state formulation of Hughes and Guttorp but in a Bayesian fashion. All parameters are sampled via Gibbs steps (latent variables such that no tuning is needed.) The W variable coefficients are sampled through an ordered Mulinomial probit (Albert and Chib). The X variable coefficients are sampled through an unordered Multinomial logit model Polya-Gamma formulation (Polson, Scott, Windle). The hidden states are sampled through a blocked Gibbs sampler.

Usage

```
NHMM(
  y,
  subseq = NULL,
  X = NULL,
  betapriorm = NULL,
  betapriorp = NULL,
  K = 2,
  iters = 1000,
  burnin = 200,
  emdist = "normal",
  nmix = 1,
  delta = FALSE,
  W = NULL,
  psipriorm = NULL,
  psipriorp = NULL,
  priors = NULL,
  outdir = NULL,
  ymiss = FALSE,
  yrep = 0,
  ypred = 0,
  Xp = NULL,
  Wp = NULL,
  yhold = NULL
)
```

Arguments

`y` T by J matrix of data (J=1 is sufficient) - missing data is denoted with NA

`subseq` [optional] if y is actually a set of subsequences then give the length of those sequences (122 for JJAS) (365 is not it!). Default is subseq=T.

`X` B by T matrix for the transition input data (B different inputs) Missing values are not allowed. If there are no Xs then use HMM function.

`betapriorm` [optional] default=NULL which is reference prior. Or a [K+B by K matrix] for the mean of the Normal prior for the beta coefficients.

`betapriorp` [optional] default=NULL which is reference prior. Or a [K+B by K+B by K array] for the precision prior(1/sig^2) of the Normal prior for the beta coefficients.

`K` number of states (default=2)

`iters` number of iterations to keep after burn in (default=1000)

`burnin` the number of burn in (default=200)

`emdist` emission distribution: "normal", "poisson", "gamma" actual choices are Normal, Poisson, Gamma, Exponential, or finite mixtures of mixtures or zero inflated version of any of these

`nmix` [optional] number of mixture components for emdist, default is one (do not include delta)

`delta` [TRUE/FALSE] TRUE-if we are using a zero inflated distribution (adds a delta function at zero as the first mixture component)

`W` [optional] is an A by T by J array of emission input data (A different inputs), missing values are not allowed, do not include an intercept term here The mixture components of emission depend on W.

`psipriorm` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the mean of the Normal prior for the beta coefficients.

`psipriorp` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the precision prior(1/sig^2) of the Normal prior for the beta coefficients.

`priors` priors for emission components, each state can have a different prior dimension 5 by nmix by K by J (some of the 5 dimensions are filled with zeros for some distributions) Normal(mu,sig2) or reference prior: mu~Normal(a.mu, 1/b.mu^2), sig2~IG(a.sig,b.sig) -the first 4 rows are used [a.mu, b.mu, a.sig, b.sig] but the 5th row is not used but must be present. -reference prior: priors=NULL

Poisson(lambda): lambda~Gamma(a.lam,b.lam) -the first 2 rows are used [a.lam, b.lam] but the 3rd-5th rows are not used but must be present.

Gamma(alpha,beta): alpha~Gamma(a.alpha,b.alpha) beta~Gamma(a.beta,b.beta) -if the second row is supplied as NA then the first row supplies fixed first parameters for the Gamma this is the suggested parametrization of the Gamma. Estimating the first parameter of the Gamma distribution switches to an MCMC that is not as stable. -to create an Exponential distribution the first row is set to 1 and the second is set to NA then the third and fourth rows contain the (a.lam, b.lam) priors for the parameter of the Exponential if prior=NULL then these are set to ones. setting these parameters to zeros is not a good idea. -a non-informative prior may not work, use a matrix of ones for a lowly informative prior instead of zeros. -the 5th row is for the tuning parameter for the MCMC (try 0.5) if a two parameter Gamma is desired. A 2 parameter Gamma needs an informative prior and results may be more unstable. -default prior=NULL is the Exponential distribution with lowly informative prior of ones

`outdir` [optional] can output each set of parameters to output files in a directory use this with larger dimension data sets or with large number of iterations the output will be written line by line and not overburden the memory limit outdir needs to end with a slash or double slash depending on OS.

`ymiss` [optional-TRUE/FALSE] if outdir is specified then draws for any missing data points will be saved to ymiss-J*.txt. There will be one data point for each sequence of length Tmiss (the amount missing from that sequence). Each row of the output file will be an iteration of the algorithm after burnin is removed.

`yrep` [optional] number (ie. 100,200,or 500) of output replicate data sets to print to outdir. The replicates will be the same dimension as y and start after burn in period. default is zero. These replicate data sets are generated from the same input variables. - must be shorter than iters

`ypred` number of predictive sets (ie. 0,100, 200, 500) - must be shorter than iters. Predicted chains: will produce ypred sets of predictives [pT by J] to print to outdir. yrep uses the same inputs to make replications, this uses new inputs to make predictions over a different time span (pT) Also outputs a set of predictive z values or length (pT by ypred)

`Xp` predictive set of Xs of length pT [B by pT] -missing values is not allowed -ensure that the Xp inputs are in the same order as X

`Wp` predictive set of Ws of length pT [A by pT by J] -missing valus are not allowed -ensure that the Wp inputs are in the same order as W

`yhold` [optioinal] a sequence of y observed values [pT by J], held out data that is of length pT that is used to compute the predictive log score (PLS) which is a metric like BIC (ie. hold out last 10 missing data values are filled in with mean PLS value. PLS is the average PLS across sequences.

Value

my.nhmm object

Examples

```
## Gamma or Exponential
### if "priors" is not specified, this is an Exponetial distribution
data(NHMMdata)
attach(NHMMdata)

## Set to iters=40 for example only this should be in the thousands
my.nhmm=NHMM(y=ygamma[1:200,1:3],  X=matrix(tX[,1:200],1,200),
    K=3, iters=40, burnin=2, emdist="gamma", nmix=3, delta=TRUE)

OBIC(my.nhmm)
Oz(my.nhmm)  #compare with the truth: tz1
OQQ(my.nhmm) #transition probabilities
## Not run:
bb=OXcoef(my.nhmm)
pp=OWcoef(my.nhmm,FALSE)
tt=Oemparams(my.nhmm,FALSE)

 ## Normal - X is not used to create this data, so it should not be significant
 my.nhmm2=NHMM(y=ynormal, subseq=1000, X=tX, K=3, iters=100,
           burnin=10, emdist="normal", nmix=2, delta=FALSE)
 OBIC(my.nhmm2)

 ## Poisson
 my.nhmm3=NHMM(y=ypoisson, X=tX, K=3, iters=100, burnin=10,
              emdist="poisson", nmix=2, delta=FALSE)
 OBIC(my.nhmm3)
 ## Predictive estimation - make 15 predictive data sets (new X) and 20 replicate data sets (same X)
 #filelocation="C:\Users\iamrandom\Desktop\here\"
 #my.nhmm4=NHMM(y=ygamma,  X=tX, K=3, iters=100, burnin=10,
 #              emdist="gamma", nmix=3, delta=TRUE,
 #              outdir=filelocation, yrep=20, Xp=Xp1, ypred=15)
 #OBIC(my.nhmm4)  #needed larger burnin
 #tt=Oemparams(my.nhmm4,TRUE,filelocation)

 ## Exponential with W variable
 #filelocation="C:\Users\iamrandom\Desktop\here\"
 #my.nhmm5=NHMM(y=ygamma,  X=tX, W=tW1, K=3, iters=50, burnin=10,
 #                emdist="gamma", nmix=3, delta=TRUE,
 #              outdir=filelocation, yrep=20, Xp=Xp1, Wp=Wp1,ypred=35)
 #OBIC(my.nhmm5)
 #pp=OWcoef(my.nhmm5,filelocation)

 ## Gamma with fixed first variables nmix=2
 nmix=2; K=3; J=dim(ygamma)[2]
 prior1=array(1,dim=c(5,nmix,K,J));  prior1[1,1,,]=1;
               prior1[1,2,,]=2;  prior1[2,,,]=NA
 my.nhmm6=NHMM(y=ygamma,  X=tX, priors=prior1, K=3, iters=100,
               burnin=10, emdist="gamma", nmix=2, delta=TRUE)


 ## One dimensional y vector case (J=1)
 #my.nhmm=NHMM(y=matrix(ygamma[1:200,1],200,1),  X=matrix(tX[,1:200],1,200),
 #K=3, iters=40, burnin=2, emdist="gamma", nmix=3, delta=TRUE)

 ### Compare my.nhmm6 (K=3) and my.nhmm7 (K=1) using both BIC
 ###     and PLS (yhold is the last 10% of the data)
 #ygamma2=ygamma
 #ygamma2[1600,10]=NA  #add some missingness
 #ygamma2[1840,10]=NA  #add some missingness to yhold
 #filelocation="C:\Users\iamrandom\Desktop\here\"
 #my.nhmm7=NHMM(y=ygamma2[1:1800,],  X=matrix(tX[,1:1800],1,1800),
 #              W=array(tW[,1:1800,],dim=c(1,1800,15)),
 #              K=3, iters=50, burnin=10, emdist="gamma", nmix=3,
 #               delta=TRUE, outdir=filelocation, ymiss=TRUE, yrep=10,
 #           Xp=matrix(tX[,1801:2000],1,200), Wp=array(tW[,1801:2000,],dim=c(1,200,15)),
 #              ypred=10, yhold=ygamma2[1801:2000,])
 #OBIC(my.nhmm7)

 #compare K=1 and K=3

## End(Not run)
```

NHMMdata *Simulated Datasets*

Description

The NHMMdata is a list containing data sets for Gamma (Exponential), Poisson, Normal, and multivariate Normal emission distributions.

Format

A list containing all of the matrices and vectors.

Details

Global settings: T=2000 time steps. J=15 time series. K=3 hidden states. pT=200 predictive chain length.

tX B=1 by T matrix of inputs. A single input of increasing linear drift from 1/2000 to 1.

Xp1 A=1 by pT. Increasing linear trend from 1/200 to 1. To be used with models already using tX.

tW A=1 by T by J array. A single input of decreasing linear drift from 1 to 1/2000. Was not used to build any of the emission distrbutions but works with ygamma, ynormal, ypoisson, ymvn but should result as "not significant."

tW2 A=2 by T by J array. 2 harmonic sin/cos terms of period 100 (20 periods per time series). Was used to build the ymvn and should come up significant. It can also be used with the other emissions but should be "not significant."

Wp1 B=1 by pT by J. Decreasing linear trend from 1 to 1/200. To be used with models already using tW.

zt1 a sequence of hidden states that uses tX as an input (NHMM), has no Markov property. Was used to create ygamma and ypoisson. tX should be significant for these data sets but not for ynormal and ymvn.

zt2 a sequence of hidden states that has no input X variable (HMM) but has a Markov property. Was used to create ynormal and ymvn. tX should not be significant but the Markov terms should be.

ygamma T by J. zt1 and tX. no tW or tW2

ypoisson T by J. zt1 and tX. no tW or tW2

ynormal T by J. zt2. no tW or tW2 or tX.

ymvn T by J. zt2 and tW2 as the mean. no tW or tX.

NHMM_MVN *Bayesian Non-homogeneous Markov Model with Multivariate Normal emission distribution (NHMMMVN)*

Description

NHMM_MVN calculates an NHMM for multiple sequences of data. The sequences can actually be short sets of equal length sequences (subseq). The traditional input variables (X) influence the non-homogenous transition probabiities of the model. An additional set of input variables (W) can be included to influence the mean of the emission distribution. All parameters are sampled via Gibbs steps (latent variables such that no tuning is needed.) The X variable coefficients are sampled through an unordered Multinomial logit model Polya-Gamma formulation (Polson, Scott, Windle). The hidden states are sampled through a blocked Gibbs sampler.

Usage

```
NHMM_MVN(
  y,
  subseq = NULL,
  X = NULL,
  betapriorm = NULL,
  betapriorp = NULL,
  K = 2,
  iters = 1000,
  burnin = 200,
  W = NULL,
  psipriorm = NULL,
  psipriorp = NULL,
  priors1 = NULL,
  priors2 = NULL,
  outdir = NULL,
  ymiss = FALSE,
  yrep = 0,
  ypred = 0,
  Xp = NULL,
  Wp = NULL,
  yhold = NULL
)
```

Arguments

`y` T by J matrix of data (J=1 is sufficient) - missing data is denoted with NA

`subseq` [optional] if y is actually a set of subsequences then give the length of those sequences (122 for JJAS) (365 is not it!). Default is subseq=T.

`X` B by T matrix for the transition input data (B different inputs) Missing values are not allowed. If there are no Xs then use HMM function.

`betapriorm` [optional] default=NULL which is reference prior. Or a [K+B by K matrix] for the mean of the Normal prior for the beta coefficients.

`betapriorp` [optional] default=NULL which is reference prior. Or a [K+B by K+B by K array] for the precision prior(1/sig^2) of the Normal prior for the beta coefficients.

`K` number of states (default=2)

`iters` number of iterations to keep after burn in (default=1000)

`burnin` the number of burn in (default=200)

`W` [optional] is an A by T by J array of emission input data (A different inputs), missing values are not allowed, do not include an intercept term here The mean function depends on W.

`psipriorm` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the mean of the Normal prior for the beta coefficients.

`psipriorp` [optional] default=NULL which is reference prior. Or a [K+A by J matrix] for the precision prior(1/sig^2) of the Normal prior for the beta coefficients.

priors1 [optional] scale parameter for each state (vector of length K) of the Wishart prior for Sigma priors1=NULL is set to 1

priors2 [optional] Covariance matrix [J by J by K] for the MVN for each state, second parameter of the Wishart for the Sigma matrix. priors2=NULL will be a diagonal matrix with diagonal set to 1

outdir [optional] can output each set of parameters to output files in a directory use this with larger dimension data sets or with large number of iterations the output will be written line by line and not overburden the memory limit outdir needs to end with a slash or double slash depending on OS.

ymiss [optional-TRUE/FALSE] if outdir is specified then draws for any missing data points will be saved to ymiss-J*.txt. There will be one data point for each sequence of length Tmiss (the amount missing from that sequence). Each row of the output file will be an iteration of the algorithm after burnin is removed.

yrep [optional] number (ie. 100,200,or 500) of output replicate data sets to print to outdir. The replicates will be the same dimension as y and start after burn in period. default is zero. These replicate data sets are generated from the same input variables. - must be shorter than iters

ypred number of predictive sets (ie. 0,100, 200, 500) - must be shorter than iters Predicted chains: will produce ypred set of predictives [pT by J] to print to outdir. yrep uses the same inputs to make replications but ypred uses new input values of X and W to make predictions over a different time span (pT) Also outputs a set of predictive z values or length (pT by ypred)

Xp predictive set of Xs of length pT [B by pT] -missing values is not allowed -ensure that the Xp inputs are in the same order as X

Wp predictive set of Ws of length pT [A by pT by J] -missing valus are not allowed -ensure that the Wp inputs are in the same order as W

yhold [optioinal] a sequence of y observed values [pT by J], held out data that is of length ypred that is used to compute the predictive log score (PLS) which is a metric like BIC (ie. hold out last 10 missing data values are filled in with mean PLS value. PLS is the average PLS across sequences.

Value

my.nhmm object

Examples

```
## Multivariate Normal data
## Not run:
data(NHMMdata)
attach(NHMMdata)

my.nhmm1=NHMM_MVN(y=ymvn,  X=tX, W=tW2, K=3, iters=50, burnin=10,
                  priors1=rep(2,3))
OBIC(my.nhmm1)
zz=Oz(my.nhmm1)  #compare with the truth zgamma
qq=OQQ(my.nhmm1)
```

```
bb=OXcoef(my.nhmm1)
pp=OWcoef(my.nhmm1,FALSE)
tt=Oemparams(my.nhmm1,FALSE)  #just Sigma matrix for MVN, returns mean of Sigma


 #filelocation="C:\Users\iamrandom\Desktop\here\"
 #my.nhmm6=NHMM_MVN(y=ymvn[1:1800,],   X=matrix(tX[,1:1800],1,1800),
 #          W=array(tW2[,1:1800,],dim=c(2,1800,15)), K=3, iters=50,
 #            burnin=10,outdir=filelocation, ymiss=TRUE, yrep=10,
 #            Xp=matrix(tX[,1801:2000],1,200),
 #           Wp=array(tW2[,1801:2000,],dim=c(2,200,15)), ypred=10,
 #           yhold=ymvn[1801:2000,])
 #OBIC(my.nhmm6)

 #Could try it with K=1, to compare K=1 to K=3

## End(Not run)
```

`OBIC` *Calculates BIC, AIC, PLS, log-likelihood*

Description

`OBIC` calculates BIC, AIC, approximate log-likelihood and plots the log-likelihood for all iterations. The log-likelihood plot should be flat to show convergence to a stationary distribution. Minimize the AIC and BIC for the *best* model and maximize PLS. The log likelihood is approximate in that it is calculated by marginalizing over the current chain of hidden states instead of using a recurrsive algorithm to compute it; every iterations produces an estimation of the log-likelihood. If yhold is provided the preditive log score (PLS) is also given.

Usage

```
OBIC(nhmmobj, outfile = NULL)
```

Arguments

`nhmmobj`	an object created from the NHMM function
`outfile`	a directory to put the .png plot

Details

Predictive Log Score: mean(log(E(p(yhold|...))) The expectation is over all of the iterations of the algorithm. And the mean is over the pT count of yhold. The scale of the PLS is in the unit of t (usually days).

Value

BIC

output: AIC, BIC, PLS [if yhold data was provided], log-likelihood to the GUI and a plot of the log-likelihood

Examples

```
#OBIC(my.nhmm)
```

`Oemparams` *Emission Parameters*

Description

`Oemparams` calculates emission parameters 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations for each parameter. Each of the J sequences and K states each has 2 parameters (Gamma and Normal) or 1 parameter (Poisson) per mixture component (nmix+delta)

Usage

```
Oemparams(nhmmobj, plots = FALSE, outfile = NULL)
```

Arguments

`nhmmobj` an object created from the NHMM or HMM function

`plots` TRUE/FALSE- default is FALSE because the plot window can grow quite large depending on the number of parameters. [J by K*nmix] panes if outboo=TRUE and [K*nmix by J] panes if outboo=FALSE, where outboo is a parameter from NHMM which determines if output was written to a file (TRUE) or to a variable (FALSE). If outfile is used then there will be K*J .png files containing dimension 2 parameters by nmix panes of trace plots. Exception: NHMM_MVN, plots is always FALSE.

`outfile` a directory to put the .png plot

Value

params [2 by nmix by K by J] by 6. There are six values returned: 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations. (0.025, 0.975) are used to construct 95 likewise (0.05, 0.95) can be used to construct 90 but the one-parameter distributions like the Exponential will only have meaningful data in the first row. Exception for NHMM_MVN then the parameters returned is the mean covariance matrix over all iterations [J by J by K]

output: plot window can grow quite large depending on the number of parameters. [J by K*nmix] panes if outboo=TRUE and [K*nmix by J] panes if outboo=FALSE, where outboo is a parameter from NHMM which determines if output was written to a file (TRUE) or to a variable (FALSE). Exception: NHMM_MVN object does not plot the covariance.

Examples

```
#thetas=Oemparams(my.nhmm, FALSE);
#thetas[,,,,3]  #mean values
```

OQQ *Calculates the mean of the transition matrix*

Description

OQQ calculates the mean of the transition array which is a K by K by T. Each row is the state at time t-1 and the columns are time t. The Gui plot shows the transition probability over time for each t-1 to t transition.

Usage

```
OQQ(nhmmobj, outfile = NULL)
```

Arguments

nhmmobj	an object created from the NHMM function
outfile	a directory to put the .png plot

Value

QQmean the transition probabilities for each time step. [K by K by T]

output: a plot where each row is the state at time t-1 and the columns are time t. The GUI plot shows the transition probability over time for each t-1 to t transition. If the columns are the same then the Markov property is probably weak.

Examples

```
#OQQ(my.nhmm)
```

OWcoef *Coefficients for W (emission inputs)*

Description

OWcoef calculates emission coefficient 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations. These coeffiecients are used to determine the mixing weights of the mixture components of the emission distributions. There are K intercept coefficients and A input coeffients for each J.

Usage

```
OWcoef(nhmmobj, plots = FALSE, outfile = NULL)
```

Arguments

`nhmmobj`	an object created from the NHMM function
`plots`	TRUE/FALSE- default is FALSE because the plot window can grow quite large depending on the number of sequences.
`outfile`	a directory to put the .png plot

Value

params [K+A by J] by 6. There are six values returned: 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations. (0.025, 0.975) are used to construct 95 likewise (0.05, 0.95) can be used to construct 90

output: plot window can grow quite large depending on the number of sequences. [K+A by J] panes.'

output: outputs statements of 90 and each X input coefficients (if any of the J coefficients for a variable are significant then that variable is deemed signficant.)

Examples

```
#thetas=OWcoef(my.nhmm, FALSE);
#thetas[,,,,3]  #mean values
```

`OXcoef` *Coefficients for X (transition inputs)*

Description

`OXcoef` calculates transition coefficient 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations. Each of K-1 states (K-1 for identifiability) has K Markov coefficients and B input coeffients.

Usage

```
OXcoef(nhmmobj, plots = FALSE, outfile = NULL)
```

Arguments

`nhmmobj`	an object created from the NHMM function
`plots`	TRUE/FALSE- default is FALSE because the plot window can grow quite large depending on the number of parameters.
`outfile`	a directory to put the .png plot

Value

params [K-1 by K+B] by 6. There are six values returned: 0.025, 0.05, mean, 0.50 (median), 0.95, 0.975 quantiles from the iterations. (0.025, 0.975) are used to construct 95 likewise (0.05, 0.95) can be used to construct 90

output: plot window can grow quite large depending on the number of states. [K-1 by K+B] panes.

output: outputs statements of 90 and each X input coefficients (if any of the K-1 coefficients for a variable are significant then that variable is deemed signficant.)

Examples

```
#thetas=OXcoef(my.nhmm, FALSE);
#thetas[,,,,3]  #mean values
```

`Oz` *Most probable state (similar to Viterbi)*

Description

`Oz` calculates the most probable state per time step (Viterbi like) with values from 1,...,K. The histogram of this sequence is displayed in the GUI output. If there are ties for a given day then the lowest number state is chosen.

Usage

```
Oz(nhmmobj, outfile = NULL)
```

Arguments

`nhmmobj`	an object created from the NHMM function
`outfile`	a directory to put the .png plot

Value

zbest the most probable sequence from all iterations

output: a plot of a histogram of the distribution of the most probable state sequence. If the number of states in the histogram is less than K, it probably means you should probably re-run the model with smaller K as some of the states have disappeared.

Examples

```
#Oz(my.nhmm)
```

Index