# Package 'hhsmm'

September 13, 2021

**Date** 2021-09-02

**Title** Hidden Hybrid Markov/Semi-Markov Model Fitting

**Version** 0.1.3

**Description** Develops algorithms for fitting, prediction, simulation and initialization of the hidden hybrid Markov/semi-Markov model, introduced by Guedon (2005) <doi:10.1016/j.csda.2004.05.033>.

**License** GPL-3

**Imports** Rcpp, Rdpack, MASS

**RdMacros** Rdpack

**Depends** R (>= 4.1.0), CMAPSS, mvtnorm

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**LinkingTo** Rcpp

**Suggests** testthat (>= 3.0.0)

**NeedsCompilation** yes

**Author** Morteza Amini [aut, cre, cph],
Afarin Bayat [aut]

**Maintainer** Morteza Amini <morteza.amini@ut.ac.ir>

**Repository** CRAN

**Date/Publication** 2021-09-13 05:50:02 UTC

## R topics documented:

---

`cov.mix.wt` *weighted covariance*

---

## Description

The weighted means and variances using the observation matrix and the estimated weight vectors

## Usage

```
cov.mix.wt(
  x,
  wt1 = rep(1/nrow(x), nrow(x)),
  wt2 = rep(1/nrow(x), nrow(x)),
  cor = FALSE,
  center = TRUE,
  method = c("unbiased", "ML")
)
```

## Arguments

| | |
|---|---|
| `x` | the observation matrix |
| `wt1` | the state probabilities matrix (number of observations times number of states) |
| `wt2` | the mixture components probabilities list (of length nstate) of matrices (number of observations times number of mixture components) |
| `cor` | logical. if TRUE the weighted correlation is also given |
| `center` | logical. if TRUE the weighted mean is also given |
| `method` | with two possible entries:<br>• `"unbiased"` the unbiased estimator is given<br>• `"ML"` the maximum likelihood estimator is given |

## Value

list of emission (mixture multivariate normal) parameters: (`mu`, `sigma` and `mix.p`)

### Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

### Examples

```
data(CMAPSS)
n = nrow(CMAPSS$train$x)
wt1 = matrix(runif(3*n),nrow=n,ncol=3)
wt2 = list()
for(j in 1:3) wt2[[j]] = matrix(runif(5*n),nrow=n,ncol=5)
emission = mixmvnorm_mstep(CMAPSS$train$x, wt1, wt2)
```

---

## dmixmvnorm *pdf of the mixture of multivariate normals for hhsmm*

---

### Description

The probability density function of a mixture multivariate normal for a specified observation vector, a specified state and a specified model's parameters

### Usage

```
dmixmvnorm(x, j, model)
```

### Arguments

| | |
|---|---|
| x | an observation vector or matrix |
| j | a specified state between 1 to nstate |
| model | a hhsmmspec model |

### Value

the probability density function value

### Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

### Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
```

```
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
p = dmixmvnorm(train$x,1,model)
```

---

`hhsmmdata` *convert to hhsmm data*

---

## Description

Converts a matrix of data and its associated vector of sequence lengths to a data list of class `"hhsmmdata"`

## Usage

```
hhsmmdata(x, N = NULL)
```

## Arguments

| | |
|---|---|
| `x` | a matrix of data |
| `N` | a vector of sequence lengths. If NULL then `N = nrow(x)` |

## Value

a data list of class `"hhsmmdata"` containing `x` and `N`

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
x = sapply(c(1,2), function(i) rnorm(100,i,i/2))
N = c(10, 15, 50, 25)
data = hhsmmdata(x,N)
```

hhsmmfit *hhsmm model fit*

## Description

Fits a hidden hybrid Markov-semi-Markov model to a data of class "hhsmmdata" and using an initial model created by hhsmmspec or initialize_model

## Usage

```
hhsmmfit(
  x,
  model,
  mstep = NULL,
  M = NA,
  maxit = 100,
  lock.transition = FALSE,
  lock.d = FALSE,
  lock.init = FALSE,
  graphical = FALSE,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | a data of class "hhsmmdata" |
| model | an initial model created by hhsmm.spec or initialize_model |
| mstep | the M step function for the EM algorithm, which also can be given in the model |
| M | the maximum duration in each state |
| maxit | the maximum number of iterations for the EM algorithm |
| lock.transition | logical. if TRUE the transition matrix will not be updated through the EM algorithm |
| lock.d | logical. if TRUE the sojourn probability matrix d will not be updated through the EM algorithm |
| lock.init | logical. if TRUE the initial probability vector will not be updated through the EM algorithm |
| graphical | logical. if TRUE a plot of the sojourn probabilities will be plotted through the EM algorithm |
| verbose | logical. if TRUE the outputs will be printed |
| ... | additional parameters for the dens.emission and mstep functions |

## Value

a list of class "hhsmm" containing the following items:

- loglike the log-likelihood of the fitted model
- AIC the Akaike information criterion of the fitted model
- BIC the Bayesian information criterion of the fitted model
- model the fitted model
- estep_variables the E step (forward-backward) probabilities of the final iteration of the EM algorithm
- M the maximum duration in each state
- J the number of states
- NN the vector of sequence lengths
- f the emission probability density function
- mstep the M step function of the EM algorithm
- yhat the estimated sequence of states

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## References

Guedon, Y. (2005). Hidden hybrid Markov/semi-Markov chains. *Computational statistics and Data analysis*, 49(3), 663-688.

OConnell, J., & Hojsgaard, S. (2011). Hidden semi Markov models for multiple observation sequences: The mhsmm package for R. *Journal of Statistical Software*, 39(4), 1-22.

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
semi <- c(FALSE,TRUE,FALSE)
initmodel1 = initialize_model(clus=clus,sojourn="gamma",M=max(train$N),semi = semi)
fit1 = hhsmmfit(x = train, model = initmodel1, M = max(train$N),
maxit = 100, lock.transition = FALSE, lock.d = FALSE, lock.init=FALSE,
graphical = FALSE)
```

hhsmmspec *hhsmm specification*

## Description

Specify a model of class "hhsmmspec" using the model parameters

## Usage

```
hhsmmspec(
  init,
  transition,
  parms.emission,
  sojourn = NULL,
  dens.emission,
  remission = NULL,
  mstep = NULL,
  semi = NULL
)
```

## Arguments

| | |
|---|---|
| init | vector of initial probabilities |
| transition | the transition matrix |
| parms.emission | the parameters of the emission distribution |
| sojourn | the sojourn distribution |
| dens.emission | the probability density function of the emission |
| remission | the random sample generation from the emission distribution |
| mstep | the M step function for the EM algorithm |
| semi | a logical vector of length nstate: the TRUE associated states are considered as semi-markov |

## Value

a model of class "hhsmmspec"

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
init = c(1,0)
transition = matrix(c(0,1,1,0),2,2)
parms.emission = list(mix.p=list(c(0.5,0.5),1),
mu=list(list(c(1,2),c(5,1)),c(2,7)),
              sigma=list(list(diag(2),2*diag(2)),0.5*diag(2)))
sojourn = list(lambda = 1, shift = 5, type = "poisson")
dens.emission = dmixmvnorm
remission = rmixmvnorm
mstep = mixmvnorm_mstep
semi = rep(TRUE,2)
model = hhsmmspec(init,transition,parms.emission,sojourn,dens.emission,remission,mstep,semi)
```

---

# homogeneity *Computing maximum homogeneity of two state sequences*

---

## Description

A function to compute the maximum homogeneity of two state sequences.

## Usage

```
homogeneity(state.seq1, state.seq2)
```

## Arguments

| | |
|---|---|
| state.seq1 | first state sequence |
| state.seq2 | second state sequence |

## Value

a vector of a length equal to the maximum number of states giving the maximum homogeneity ratios

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
state.seq1 = c(3,3,3,1,1,2,2,2,2)
state.seq2 = c(2,2,2,3,3,1,1,1,1)
homogeneity(state.seq1, state.seq2)
```

`initialize_model` *initialize the hhsmmspec model for a specified emission distribution*

## Description

Initialize the `hhsmmspec` model by using an initial clustering obtained by `initial_cluster` and the emission distribution characterized by mstep and dens.emission

## Usage

```
initialize_model(
  clus,
  mstep = mixmvnorm_mstep,
  dens.emission = dmixmvnorm,
  sojourn = NULL,
  semi = NULL,
  M,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `clus` | initial clustering obtained by `initial_cluster` |
| `mstep` | the mstep function of the EM algorithm with an style simillar to that of `mixmvnorm_mstep` |
| `dens.emission` | the density of the emission distribution with an style simillar to that of `dmixmvnorm` |
| `sojourn` | one of the following cases:<br>• `"nonparametric"` non-parametric sojourn distribution<br>• `"nbinom"` negative binomial sojourn distribution<br>• `"logarithmic"` logarithmic sojourn distribution<br>• `"poisson"` poisson sojourn distribution<br>• `"gamma"` gamma sojourn distribution<br>• `"weibull"` weibull sojourn distribution<br>• `"lnorm"` log-normal sojourn distribution<br>• `"auto"`automatic determination of the sojourn distribution using the chi-square test |
| `semi` | logical and of one of the following forms:<br>• a logical value: if TRUE all states are considered as semi-Markovian else Markovian<br>• a logical vector of length nstate: the TRUE associated states are considered as semi-Markovian and FALSE associated states are considered as Markovian<br>• `NULL` if `ltr`=TRUE then `semi = c(rep(TRUE,nstate-1),FALSE)`, else `semi = rep(TRUE,nstate)` |

| | |
|---|---|
| M | maximum number of waiting times in each state |
| verbose | logical. if TRUE the outputs will be printed the normal distributions will be estimated |
| ... | additional parameters of the mstep function |

## Value

a hhsmmspec model containing the following items:

- init initial probabilities of states
- transition transition matrix
- parms.emission parameters of the mixture normal emission (mu, sigma, mix.p)
- sojourn list of sojourn time distribution parameters and its type
- dens.emission the emission probability density function
- mstep the M step function of the EM algorithm
- semi a logical vector of length nstate with the TRUE associated states are considered as semi-Markovian

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
initmodel = initialize_model(clus=clus,sojourn="gamma",M=max(train$N))
```

`initial_cluster` *initial clustering of the data set*

## Description

Provides an initial clustering for a data of class "hhsmmdata" which determines the initial states and mixture components (if necessary) to be used for initial parameter and model estimation

## Usage

```
initial_cluster(
  train,
  nstate,
  nmix,
  ltr = FALSE,
  equispace = FALSE,
  final.absorb = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `train` | the train data set of class "hhsmmdata" |
| `nstate` | number of states |
| `nmix` | number of mixture components which is of one of the following forms:<br>• a vector of positive (non-zero) integers of length `nstate`<br>• a positive (non-zero) integer<br>• the text "auto": the number of mixture components will be determined automatically based on the within cluster sum of squares |
| `ltr` | logical. if TRUE a left to right hidden hybrid Markov/semi-Markov model is assumed |
| `equispace` | logical. if TRUE the left to right clustering will be performed simply with equal time spaces. This option is suitable for speach recognition applications. |
| `final.absorb` | logical. if TRUE the final state of the sequence is assumed to be the absorbance state |
| `verbose` | logical. if TRUE the outputs will be printed |

## Details

In reliability applications, the hhsmm models are often left-to-right and the modeling aims to predict the future states. In such cases, the `ltr`=TRUE and `final.absorb`=TRUE should be set.

## Value

a list containing the following items:

- `clust.X` a list of clustered observations for each sequence and state
- `mix.clus` a list of the clusters for the mixtures for each state
- `state.clus` the exact state clusters of each observation (available if `ltr`=FALSE)
- `nmix` the number of mixture components (a vector of positive (non-zero) integers of length `nstate`)
- `ltr` logical. if TRUE a left to right hidden hybrid Markov/semi-Markov model is assumed
- `final.absorb` logical. if TRUE the final state of the sequence is assumed to be the absorbance state

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
test <-  simulate(model, nsim = c(7,3,3,8), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
```

---

`initial_estimate` *initial estimation of the model parameters for a specified emission distribution*

---

## Description

Provides the initial estimates of the model parameters of a specified emission distribution characterized by the `mstep` function, for an initial clustering obtained by `initial_cluster`

## Usage

```
initial_estimate(clus, mstep = mixmvnorm_mstep, verbose = FALSE, ...)
```

## Arguments

| | |
|---|---|
| clus | an initial clustering obtained by initial_cluster |
| mstep | the mstep function of the EM algorithm with an style simillar to that of mixmvnorm_mstep |
| verbose | logical. if TRUE the outputs will be printed |
| ... | additional parameters of the mstep function |

## Value

a list containing the following items:

- emission list the estimated parameterers of the emission distribution
- leng list of waiting times in each state for each sequence
- clusters the exact clusters of each observation (available if ltr=FALSE)
- nmix the number of mixture components (a vector of positive (non-zero) integers of length nstate)
- ltr logical. if TRUE a left to right hidden hybrid Markovian/semi-Markovianmodel is assumed

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
par = initial_estimate(clus,verbose=TRUE)
```

## ltr_clus *left to right clustering*

### Description

A left to right initial clustering method using the mean differences and Hotelling's T-squared test

### Usage

```
ltr_clus(Dat, k)
```

### Arguments

| | |
|---|---|
| Dat | a data matrix |
| k | number of clusters |

### Value

a (left to right) clustering vector

### Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

### Examples

```
data(CMAPSS)
clus = ltr_clus(CMAPSS$train$x[1:CMAPSS$train$N[1],],3)
```

## make_model *make a hhsmmspec model for a specified emission distribution*

### Description

Provides a hhsmmspec model by using the parameters obtained by initial_estimate for the emission distribution characterized by mstep and dens.emission

### Usage

```
make_model(
  par,
  mstep = mixmvnorm_mstep,
  dens.emission = dmixmvnorm,
  semi = NULL,
  M,
  sojourn
)
```

## Arguments

| | |
|---|---|
| par | the parameters obtained by initial_estimate |
| mstep | the mstep function of the EM algorithm with an style simillar to that of mixmvnorm_mstep |
| dens.emission | the density of the emission distribution with an style simillar to that of dmixmvnorm |
| semi | logical and of one of the following forms:<br>• a logical value: if TRUE all states are considered as semi-Markovian else Markovian<br>• a logical vector of length nstate: the TRUE associated states are considered as semi-Markovian and FALSE associated states are considered as Markovian<br>• NULL if ltr=TRUE then semi = c(rep(TRUE,nstate-1),FALSE), else semi = rep(TRUE,nstate) |
| M | maximum number of waiting times in each state |
| sojourn | the sojourn time distribution which is one of the following cases:<br>• "nonparametric" non-parametric sojourn distribution<br>• "nbinom" negative binomial sojourn distribution<br>• "logarithmic" logarithmic sojourn distribution<br>• "poisson" poisson sojourn distribution<br>• "gamma" gamma sojourn distribution<br>• "weibull" weibull sojourn distribution<br>• "lnorm" log-normal sojourn distribution<br>• "auto" automatic determination of the sojourn distribution using the chi-square test |

## Value

a hhsmmspec model containing the following items:

- init initial probabilities of states
- transition transition matrix
- parms.emission parameters of the mixture normal emission (mu, sigma, mix.p)
- sojourn list of sojourn distribution parameters and its type
- dens.emission the emission probability density function
- mstep the M step function of the EM algorithm
- semi a logical vector of length nstate with the TRUE associated states are considered as semi-Markovian

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
par = initial_estimate(clus,verbose=TRUE)
model = make_model(par,semi=NULL,M=max(train$N),sojourn="gamma")
```

---

`mixdiagmvnorm_mstep` *the M step function of the EM algorithm*

---

## Description

The M step function of the EM algorithm for the mixture of multivariate normals with diagonal covariance matrix as the emission distribution using the observation matrix and the estimated weight vectors

## Usage

```
mixdiagmvnorm_mstep(x, wt1, wt2)
```

## Arguments

| | |
|---|---|
| `x` | the observation matrix |
| `wt1` | the state probabilities matrix (number of observations times number of states) |
| `wt2` | the mixture components probabilities list (of length nstate) of matrices (number of observations times number of mixture components) |

## Value

list of emission (mixture multivariate normal) parameters: (`mu`, `sigma` and `mix.p`), where `sigma` is a diagonal matrix

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
data(CMAPSS)
n = nrow(CMAPSS$train$x)
wt1 = matrix(runif(3*n),nrow=n,ncol=3)
wt2 = list()
for(j in 1:3) wt2[[j]] = matrix(runif(5*n),nrow=n,ncol=5)
emission = mixdiagmvnorm_mstep(CMAPSS$train$x, wt1, wt2)
```

---

# mixmvnorm_mstep *the M step function of the EM algorithm*

---

## Description

The M step function of the EM algorithm for the mixture of multivariate normals as the emission distribution using the observation matrix and the estimated weight vectors

## Usage

```
mixmvnorm_mstep(x, wt1, wt2)
```

## Arguments

| | |
|---|---|
| x | the observation matrix |
| wt1 | the state probabilities matrix (number of observations times number of states) |
| wt2 | the mixture components probabilities list (of length nstate) of matrices (number of observations times number of mixture components) |

## Value

list of emission (mixture multivariate normal) parameters: (mu, sigma and mix.p)

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
data(CMAPSS)
n = nrow(CMAPSS$train$x)
wt1 = matrix(runif(3*n),nrow=n,ncol=3)
wt2 = list()
for(j in 1:3) wt2[[j]] = matrix(runif(5*n),nrow=n,ncol=5)
emission = mixmvnorm_mstep(CMAPSS$train$x, wt1, wt2)
```

`predict.hhsmm` *prediction of state sequence for hhsmm*

## Description

Predicts the state sequence of a fitted hidden hybrid Markov/semi-Markov model estimated by `hhsmmfit` for a new (test) data of class `"hhsmmdata"` with an optional prediction of the residual useful lifetime (RUL) for a left to right model

## Usage

```
## S3 method for class 'hhsmm'
predict(
  object,
  newdata,
  future = 0,
  method = "viterbi",
  RUL.estimate = FALSE,
  confidence = "max",
  conf.level = 0.95,
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | a fitted model of class `"hhsmm"` estimated by `hhsmmfit` |
| `newdata` | a new (test) data of class `"hhsmmdata"` |
| `future` | number of future states to be predicted |
| `method` | the prediction method with two options:<br>• `"viterbi"` (default) uses the Viterbi algorithm for prediction<br>• `"smoothing"` uses the smoothing algorithm for prediction |
| `RUL.estimate` | logical. if TRUE the residual useful lifetime (RUL) of a left to right model, as well as the prediction interval will also be predicted (default is FALSE) |
| `confidence` | the method for obtaining the prediction interval of the RUL, with two cases:<br>• `"max"` (default) the maximum probability as the point predict and the high probability critical values as the lower and upper bounds<br>• `"mean"` the mean value as the point predict and the normal confidence lower and upper bounds as the prediction interval |
| `conf.level` | the confidence level of the prediction interval (default 0.95) |
| `...` | additional parameters for the dens.emission and mstep functions |

## Value

a list containing the following items:

- x the observation sequence
- s the predicted state sequence
- N the vector of sequence lengths
- p the state probabilities
- RUL the point predicts of the RUL
- RUL.low the lower bounds for the prediction intervals of the RUL
- RUL.up the upper bounds for the prediction intervals of the RUL

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## References

Guedon, Y. (2005). Hidden hybrid Markov/semi-Markov chains. *Computational statistics and Data analysis*, 49(3), 663-688.

OConnell, J., & Hojsgaard, S. (2011). Hidden semi Markov models for multiple observation sequences: The mhsmm package for R. *Journal of Statistical Software*, 39(4), 1-22.

## See Also

predict.hhsmmspec

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
test <-  simulate(model, nsim = c(7,3,3,8), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
semi <- c(FALSE,TRUE,FALSE)
initmodel1 = initialize_model(clus=clus,sojourn="gamma",M=max(train$N),semi = semi)
fit1 = hhsmmfit(x = train, model = initmodel1, M = max(train$N),
maxit = 100, lock.transition = FALSE, lock.d = FALSE, lock.init=FALSE,
graphical = FALSE)
yhat1 <- predict(fit1, test)
```

`predict.hhsmmspec` *prediction of state sequence for hhsmm*

## Description

Predicts the state sequence of a hidden hybrid Markov/semi-Markov model for a new (test) data of class `"hhsmmdata"` with an optional prediction of the residual useful lifetime (RUL) for a left to right model

## Usage

```
## S3 method for class 'hhsmmspec'
predict(object, newdata, method = "viterbi", M = NA, ...)
```

## Arguments

| | |
|---|---|
| `object` | a hidden hybrid Markov/semi-Markov model |
| `newdata` | a new (test) data of class `"hhsmmdata"` |
| `method` | the prediction method with two options:<br>• `"viterbi"` (default) uses the Viterbi algorithm for prediction<br>• `"smoothing"` uses the smoothing algorithm for prediction |
| `M` | maximum duration in states |
| `...` | additional parameters of the function `predict.hhsmm` |

## Value

a list containing the following items:

- `x` the observation sequence
- `s` the predicted state sequence
- `N` the vector of sequence lengths
- `p` the state probabilities
- `RUL` the point predicts of the RUL
- `RUL.low` the lower bounds for the prediction intervals of the RUL
- `RUL.up` the upper bounds for the prediction intervals of the RUL

## Author(s)

Morteza Amini, `<morteza.amini@ut.ac.ir>`, Afarin Bayat, `<aftbayat@gmail.com>`

## References

Guedon, Y. (2005). Hidden hybrid Markov/semi-Markov chains. *Computational statistics and Data analysis*, 49(3), 663-688.

OConnell, J., & Hojsgaard, S. (2011). Hidden semi Markov models for multiple observation sequences: The mhsmm package for R. *Journal of Statistical Software*, 39(4), 1-22.

## See Also

predict.hhsmm

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(10,8,8,18), seed = 1234, remission = rmixmvnorm)
test <-  simulate(model, nsim = c(5,3,3,8), seed = 1234, remission = rmixmvnorm)
clus = initial_cluster(train,nstate=3,nmix=c(2,2,2),ltr=FALSE,
final.absorb=FALSE,verbose=TRUE)
semi <- c(FALSE,TRUE,FALSE)
initmodel1 = initialize_model(clus=clus,sojourn="gamma",M=max(train$N),semi = semi)
yhat1 <- predict(initmodel1, test)
```

---

`rmixmvnorm` *Random data generation from the mixture of multivariate normals for hhsmm model*

---

## Description

Generates a vector of observations from mixture multivariate normal distribution in a specified state and using the parameters of a specified model

## Usage

```
rmixmvnorm(j, model)
```

## Arguments

| | |
|---|---|
| `j` | a specified state |
| `model` | a hhsmmspec model |

## Value

a random vector of observations from mixture of multivariate normal distributions

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,10,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
x = rmixmvnorm(1, model)
```

---

# simulate.hhsmmspec *Simulation of data from hhsmm model*

---

## Description

Simulates a data set of class "hhsmmdata" using a hhsmmspec model

## Usage

```
## S3 method for class 'hhsmmspec'
simulate(object, nsim, seed = NULL, remission = rmixmvnorm, ...)
```

## Arguments

| | |
|---|---|
| object | a hhsmmspec model |
| nsim | a vector of sequence lengths (might be of length 1) |
| seed | a random seed to be set |
| remission | a random emission generation function (default = rmixmvnorm) |
| ... | additional parameters of the remission function |

## Value

a list of class "hsmm.data" containing the following items:

- s the vector of states
- x observation matrix
- N vector of sequence lengths

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
J <- 3
initial <- c(1,0,0)
semi <- c(FALSE,TRUE,FALSE)
P <- matrix(c(0.8, 0.1, 0.1, 0.5, 0, 0.5, 0.1, 0.2, 0.7), nrow = J, byrow=TRUE)
par <- list(mu = list(list(7,8),list(10,9,11),list(12,14)),
sigma = list(list(3.8,4.9),list(4.3,4.2,5.4),list(4.5,6.1)),
mix.p = list(c(0.3,0.7),c(0.2,0.3,0.5),c(0.5,0.5)))
sojourn <- list(shape = c(0,3,0), scale = c(0,8,0), type = "gamma")
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixmvnorm, sojourn = sojourn, semi = semi)
train <- simulate(model, nsim = c(8,5,5,10), seed = 1234, remission = rmixmvnorm)
```

---

# train_test_split *Splitting the data sets to train and test*

---

## Description

A function to split the train data of class "hhsmmdata" to train and test subsets with an option to right trim the sequences

## Usage

```
train_test_split(train, train.ratio = 0.7, trim = FALSE)
```

## Arguments

| | |
|---|---|
| train | the train data of class "hhsmmdata" |
| train.ratio | a number in (0,1) which determines the ratio of the train subset |
| trim | logical. if TRUE the sequences will be right trimmed with random lengths |

## Details

In reliability applications, the hhsmm models are often left-to-right and the modeling aims to predict the future states. In such cases, the test sets are right trimmed and the prediction aims to predict the residual useful lifetime (RUL) of a new sequence.

## Value

a list containing:

- train the randomly selected subset of train data of class "hhsmmdata"
- test the randomly selected subset of test data of class "hhsmmdata". If trim=TRUE, the test list also contains the RUL (residual useful lifetime) values

## Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Afarin Bayat, <aftbayat@gmail.com>

## Examples

```
data(CMAPSS)
tt = train_test_split(CMAPSS$train,train.ratio = 0.7, trim = TRUE)
```

# Index