Package 'hhsmm'

January 7, 2023

Date 2023-01-07

Title Hidden Hybrid Markov/Semi-Markov Model Fitting

Version 0.3.4

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>, which also includes several tools for handling missing data, nonparametric mixture of B-splines emissions (Langrock et al., 2015 <doi:10.1111/biom.12282>), fitting regime switching regression (Kim et al., 2008 <doi:10.1016/j.jeconom.2007.10.002>) and auto-regressive hidden hybrid Markov/semi-Markov model, spline-based nonparametric estimation of additive state-switching models (Langrock et al., 2018 <doi:10.1111/stan.12133>) and many other useful tools (read for more description: Amini et al., 2022 <doi:10.1007/s00180-022-01248-x> and its arxiv version: <arXiv:2109.12489>).

License GPL-3

Imports Rcpp, Rdpack, MASS, mice, cpr, psych, progress, magic, splines2

RdMacros Rdpack

Depends R (>= 4.1.0), CMAPSS, mvtnorm

Encoding UTF-8

BugReports https://github.com/mortamini/hhsmm/issues

RoxygenNote 7.2.1

LinkingTo Rcpp

Suggests testthat (>= 3.0.0)

NeedsCompilation yes

Author Morteza Amini [aut, cre, cph], Afarin Bayat [aut], Reza Salehian [aut]

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

Repository CRAN

Date/Publication 2023-01-07 07:10:02 UTC

R topics documented:

additive_reg_mstep . . . 3
addreg_hhsmm_predict . . . 4
cov.miss.mix.wt . . . 5
cov.mix.wt . . . 7
dmixlm . . . 8
dmixmvnorm . . . 10
dnonpar . . . 11
dnorm_additive_reg . . . 12
hhsmmdata . . . 13
hhsmmfit . . . 14
hhsmmspec . . . 16
homogeneity . . . 18
initialize_model . . . 18
initial_cluster . . . 20
initial_estimate . . . 22
lagdata . . . 24
ltr_clus . . . 25
ltr_reg_clus . . . 25
make_model . . . 26
miss_mixmvnorm_mstep . . . 28
mixdiagmvnorm_mstep . . . 29
mixlm_mstep . . . 30
mixmvnorm_mstep . . . 31
nonpar_mstep . . . 32
predict.hhsmm . . . 33
predict.hhsmmspec . . . 35
raddreg . . . 37
rmixar . . . 38
rmixlm . . . 39
rmixmvnorm . . . 40
score . . . 41
simulate.hhsmmspec . . . 42
train_test_split . . . 43

Index **45**

additive_reg_mstep *the M step function of the EM algorithm*

Description

The M step function of the EM algorithm for the Gaussian linear (Markov-switching) regression as the emission distribution using the responses and covariates matrices and the estimated weight vectors

Usage

```
additive_reg_mstep(x, wt, control = list(K = 5, lambda0 = 0.01, resp.ind = 1))
```

Arguments

x	the observation matrix
wt	the state probabilities matrix (number of observations times number of states)
control	the parameters to control the M-step function. The simillar name is chosen with that of dnorm_additive_reg, to be used in ... argument of the hhsmmfit function. Here, it contains the following items:

- K the degrees of freedom for the B-spline, default is K=5
- lambda0 the initial value of the smoothing parameter, default is lambda0=0.01
- resp.ind a vector of the column numbers of x which contain response variables. The default is 1, which means that the first column of x is the univariate response variable

Value

list of emission (nonparametric mixture of splines) parameters:

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Reza Salehian, <reza.salehian@ut.ac.ir>

References

Langrock, R., Adam, T., Leos-Barajas, V., Mews, S., Miller, D. L., and Papastamatiou, Y. P. (2018). Spline-based nonparametric inference in general state-switching models. Statistica Neerlandica, 72(3), 179-200.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
```

```
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
clus = initial_cluster(train = train, nstate = 3, nmix = NULL,
ltr = FALSE, final.absorb = FALSE, verbose = TRUE, regress = TRUE)
initmodel = initialize_model(clus = clus ,mstep = additive_reg_mstep,
dens.emission = dnorm_additive_reg, sojourn = NULL, semi = rep(FALSE, 3),
M = max(train$N),verbose = TRUE)
fit1 = hhsmmfit(x = train, model = initmodel, mstep = additive_reg_mstep,
M = max(train$N))
plot(train$x[, 1] ~ train$x[, 2], col = train$s, pch = fit1$yhat,
xlab = "x", ylab = "y")
text(0,30, "colors are real states",col="red")
text(0,28, "characters are predicted states")
pred <- addreg_hhsmm_predict(fit1, train$x[, 2], 5)
yhat1 <- pred[[1]]
yhat2 <- pred[[2]]
yhat3 <- pred[[3]]

lines(yhat1[order(train$x[, 2])]~sort(train$x[, 2]),col = 2)
lines(yhat2[order(train$x[, 2])]~sort(train$x[, 2]),col = 1)
lines(yhat3[order(train$x[, 2])]~sort(train$x[, 2]),col = 3)
```

`addreg_hhsmm_predict` *predicting the response values for the regime switching model*

Description

This function computes the predictions of the response variable for the Gaussian linear (Markov-switching) regression model for different states for any observation matrix of the covariates

Usage

```
addreg_hhsmm_predict(object, x, K)
```

Arguments

`object`	a fitted model of class `"hhsmm"` estimated by `hhsmmfit`
`x`	the observation matrix of the covariates
`K`	the degrees of freedom for the B-spline

Value

list of predictions of the response variable

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
clus = initial_cluster(train = train, nstate = 3, nmix = NULL,
ltr = FALSE, final.absorb = FALSE, verbose = TRUE, regress = TRUE)
initmodel = initialize_model(clus = clus ,mstep = additive_reg_mstep,
dens.emission = dnorm_additive_reg, sojourn = NULL, semi = rep(FALSE, 3),
M = max(train$N),verbose = TRUE)
fit1 = hhsmmfit(x = train, model = initmodel, mstep = additive_reg_mstep,
M = max(train$N))
plot(train$x[, 1] ~ train$x[, 2], col = train$s, pch = fit1$yhat,
xlab = "x", ylab = "y")
text(0,30, "colors are real states",col="red")
text(0,28, "characters are predicted states")
pred <- addreg_hhsmm_predict(fit1, train$x[, 2], 5)
yhat1 <- pred[[1]]
yhat2 <- pred[[2]]
yhat3 <- pred[[3]]

lines(yhat1[order(train$x[, 2])]~sort(train$x[, 2]),col = 2)
lines(yhat2[order(train$x[, 2])]~sort(train$x[, 2]),col = 1)
lines(yhat3[order(train$x[, 2])]~sort(train$x[, 2]),col = 3)
```

---

cov.miss.mix.wt *weighted covariance for data with missing values*

Description

The weighted means and variances using the observation matrix and the estimated weight vectors for a data matrix containing missing values (NA or NaN)

Usage

```
cov.miss.mix.wt(
  x,
  means,
  secm,
  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, which can contain missing values (NA or NaN)
means	a list containing the means of the missing values given observed values
secm	a list containing the second moments of the missing values given observed values
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: • "unbiased" the unbiased estimator is given • "ML" the maximum likelihood estimator is given

Value

list containing the following items:

- center the weighted mean of x
- cov the weighted covariance of x
- n.obs the number of observations in x
- cor the weighted correlation of x, if the parameter cor is TRUE
- wt1 the state weighs wt1
- wt2 the mixture component weights wt2
- pmix the estimated mixture proportions

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

Examples

```
data(CMAPSS)
x0 = CMAPSS$train$x[1:CMAPSS$train$N[1], ]
n = nrow(x0)
wt1 = runif(n)
wt2 = runif(n)
p = ncol(x0)
sammissall = sample(1:n, trunc(n / 20))
means = secm = list()
for(ii in 1:n){
if(ii %in% sammissall){
   means[[ii]] = colMeans(x0[sammissall, ])
   secm[[ii]] = t(x0[sammissall, ]) %*% x0[sammissall, ]
 }else{
  means[[ii]] = secm[[ii]] = NA
 }
}
x0[sammissall,] = NA

cov.miss.mix.wt(x0, means, secm, wt1, wt2)
```

`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: • "unbiased" the unbiased estimator is given • "ML" the maximum likelihood estimator is given

Value

list containing the following items:

- center the weighted mean of x
- cov the weighted covariance of x
- n.obs the number of observations in x
- cor the weighted correlation of x, if the parameter cor is TRUE
- wt1 the state weighs wt1
- wt2 the mixture component weights wt2
- pmix the estimated mixture proportions

Author(s)

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

Examples

```
data(CMAPSS)
n = nrow(CMAPSS$train$x)
wt1 = runif(n)
wt2 = runif(n)
cov.mix.wt(CMAPSS$train$x, wt1, wt2)
```

---

dmixlm *pdf of the mixture of Gaussian linear (Markov-switching) models for hhsmm*

Description

The probability density function of a mixture Gaussian linear (Markov-switching) models for a specified observation vector, a specified state and a specified model's parameters

Usage

```
dmixlm(x, j, model, resp.ind = 1)
```

Arguments

x	the observation matrix including responses and covariates
j	a specified state between 1 to nstate
model	a hhsmmspec model
resp.ind	a vector of the column numbers of x which contain response variables. The default is 1, which means that the first column of x is the univariate response variable

Value

the probability density function value

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

References

Kim, C. J., Piger, J. and Startz, R. (2008). Estimation of Markov regime-switching regression models with endogenous switching. Journal of Econometrics, 143(2), 263-273.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5),
nrow = J, byrow = TRUE)
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
clus = initial_cluster(train = train, nstate = 3, nmix = c(1, 2, 1),
ltr = FALSE, final.absorb = FALSE, verbose = TRUE, regress = TRUE)
initmodel = initialize_model(clus = clus, mstep = mixlm_mstep,
dens.emission = dmixlm, sojourn = NULL, semi = rep(FALSE, 3),
M = max(train$N),verbose = TRUE)
fit1 = hhsmmfit(x = train, model = initmodel, mstep = mixlm_mstep,
M = max(train$N))
plot(train$x[,1] ~ train$x[, 2], col = train$s, pch = 16,
xlab = "x", ylab = "y")
abline(fit1$model$parms.emission$intercept[[1]],
fit1$model$parms.emission$coefficient[[1]], col = 1)
abline(fit1$model$parms.emission$intercept[[2]][[1]],
fit1$model$parms.emission$coefficient[[2]][[1]], col = 2)
abline(fit1$model$parms.emission$intercept[[2]][[2]],
```

```
fit1$model$parms.emission$coefficient[[2]][[2]], col = 2)
abline(fit1$model$parms.emission$intercept[[3]],
fit1$model$parms.emission$coefficient[[3]], col = 3)
```

---

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)
```

dnonpar *pdf of the mixture of B-splines for hhsmm*

Description

The probability density function of a mixture of B-splines for a specified observation vector, a specified state and a specified model's parameters

Usage

```
dnonpar(x, j, model, control = list(K = 5))
```

Arguments

x	an observation vector or matrix
j	a specified state between 1 to nstate
model	a hhsmmspec model
control	the parameters to control the density function. The simillar name is chosen with that of nonpar_mstep, to be used in ... argument of the hhsmmfit function. Here, it contains only the parameter K which is the degrees of freedom for the B-spline, default is K=5

Value

the probability density function value

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Reza Salehian, <reza.salehian@ut.ac.ir>

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 = NULL, ltr = FALSE,
final.absorb = FALSE, verbose = TRUE)
semi <- c(FALSE, TRUE, FALSE)
```

```
initmodel = initialize_model(clus = clus, mstep = nonpar_mstep,
sojourn = "gamma", M = max(train$N), semi = semi)
p = dnonpar(train$x, 1, initmodel)
```

---

dnorm_additive_reg *pdf of the Gaussian additive (Markov-switching) model for hhsmm*

Description

The probability density function of a Gaussian additive (Markov-switching) model for a specified observation vector, a specified state and a specified model's parameters

Usage

```
dnorm_additive_reg(x, j, model, control = list(K = 5, resp.ind = 1))
```

Arguments

x	the observation matrix including responses and covariates
j	a specified state between 1 to nstate
model	a hhsmmspec model
control	the parameters to control the density function. The simillar name is chosen with that of additive_reg_mstep, to be used in ... argument of the hhsmmfit function. Here, it contains the following items:

- K the degrees of freedom for the B-spline, default is K=5
- resp.ind a vector of the column numbers of x which contain response variables. The default is 1, which means that the first column of x is the univariate response variable

Value

the probability density function value

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Reza Salehian, <reza.salehian@ut.ac.ir>

References

Langrock, R., Adam, T., Leos-Barajas, V., Mews, S., Miller, D. L., and Papastamatiou, Y. P. (2018). Spline-based nonparametric inference in general state-switching models. Statistica Neerlandica, 72(3), 179-200.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
clus = initial_cluster(train = train, nstate = 3, nmix = NULL,
ltr = FALSE, final.absorb = FALSE, verbose = TRUE, regress = TRUE)
initmodel = initialize_model(clus = clus ,mstep = additive_reg_mstep,
dens.emission = dnorm_additive_reg, sojourn = NULL, semi = rep(FALSE, 3),
M = max(train$N),verbose = TRUE)
fit1 = hhsmmfit(x = train, model = initmodel, mstep = additive_reg_mstep,
M = max(train$N))
plot(train$x[, 1] ~ train$x[, 2], col = train$s, pch = fit1$yhat,
xlab = "x", ylab = "y")
text(0,30, "colors are real states",col="red")
text(0,28, "characters are predicted states")
pred <- addreg_hhsmm_predict(fit1, train$x[, 2], 5)
yhat1 <- pred[[1]]
yhat2 <- pred[[2]]
yhat3 <- pred[[3]]

lines(yhat1[order(train$x[, 2])]~sort(train$x[, 2]),col = 2)
lines(yhat2[order(train$x[, 2])]~sort(train$x[, 2]),col = 1)
lines(yhat3[order(train$x[, 2])]~sort(train$x[, 2]),col = 3)
```

`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>

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,
 par = list(maxit = 100, lock.transition = FALSE, lock.d = FALSE, lock.init = FALSE,
    graphical = FALSE, verbose = TRUE)
)
```

Arguments

x	a data of class "hhsmmdata", which can also contain missing values (NA or NaN)
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
...	additional parameters for the dens.emission and mstep functions
M	the maximum duration in each state
par	additional list of control parameters of the hhsmmfit function including the following items:

- 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

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)
initmodel1 = initialize_model(clus = clus, sojourn = "gamma",
M = max(train$N), semi = semi)
fit1 = hhsmmfit(x = train, model = initmodel1, M = max(train$N))
```

`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, which is one of the following cases: • a list containing `d`, which is a `nobs` (number of observations) times `nstates` (number of states) matrix of probabilities, and `type = "nonparametric"` for non-parametric sojourn distribution • a list containing the parameters `mu`, `shift` and `size` of a shifted negative binomial distribution, for each semi-Markovian state, and `type = "nbinom"` for negative binomial sojourn distribution

- a list containing the parameters shape and shift of a shifted logarithmic distribution, for each semi-Markovian state, and type = "logarithmic" for logarithmic sojourn distribution
- a list containing the parameters lambda and shift of the shifted poisson distribution, for each semi-Markovian state, and type = "poisson" for Poisson sojourn distribution
- a list containing the parameters shape and scale of the gamma distribution, for each semi-Markovian state, and type = "gamma" for gamma sojourn distribution
- a list containing the parameters shape and scale of the Weibull distribution, for each semi-Markovian state, and type = "weibull" for Weibull sojourn distribution
- a list containing the parameters meanlog and sdlog of the log-normal distribution, for each semi-Markovian state, and type = "lnorm" for log-normal 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>

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 = NULL,
  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. If NULL, the mixmvnorm_mstep is considered for the complete data set and miss_mixmvnorm_mstep is considered for the data with missing values (NA or NaN)
dens.emission	the density of the emission distribution with an style simillar to that of dmixmvnorm
sojourn	one of the following cases: • "nonparametric" non-parametric sojourn distribution • "nbinom" negative binomial sojourn distribution • "logarithmic" logarithmic sojourn distribution • "poisson" poisson sojourn distribution • "gamma" gamma sojourn distribution • "weibull" weibull sojourn distribution • "lnorm" log-normal sojourn distribution • "auto"automatic determination of the sojourn distribution using the chi-square test
semi	logical and of one of the following forms: • a logical value: if TRUE all states are considered as semi-Markovian else Markovian • a logical vector of length nstate: the TRUE associated states are considered as semi-Markovian and FALSE associated states are considered as Markovian • 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,
  regress = FALSE,
  resp.ind = 1
)
```

Arguments

train	the train data set of class "hhsmmdata", which can also contain missing data (NA or NaN)
nstate	number of states
nmix	number of mixture components which is of one of the following forms: • a vector of positive (non-zero) integers of length nstate • a positive (non-zero) integer • the text "auto": the number of mixture components will be determined automatically based on the within cluster sum of squares • NULL if no mixture distribution is not considered as the emission. This option is usefull for the nonparametric emission distribution (nonpar_mstep and dnonpar)
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 speech 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
regress	logical. if TRUE the linear regression clustering will be performed
resp.ind	the column indices of the response variables for the linear regression clustering approach. The default is 1, which means that the first column is the univariate response variable

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
- `miss` logical. if TRUE the `train$x` matrix contains missing data (NA or NaN)

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)
```

---

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)
```

lagdata *Create hhsmm data of lagged time series*

Description

Creates a data of class hhsmmdata containing lagged time series which can be used for fitting autoregressive hidden hybrid Makrov/semi-Markov model (AR-HHSMM)

Usage

```
lagdata(data, lags = 1)
```

Arguments

data	a data of class hhsmmdata containing a multivariate and multi-state time series
lags	a positive integer which is the number of lags to be calculated

Value

a data of class hhsmmdata containing lagged time series

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
par <- list(intercept = list(0.1, -0.1, 0.2),
coefficient = list(-0.6, 0.7, -0.5),
csigma = list(5.5, 4, 3.5), mix.p = list(1, 1, 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(50, 60, 84, 100), seed = 1234,
emission.control = list(autoregress = TRUE))
laggedtrain = lagdata(train)
```

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)
```

ltr_reg_clus *left to right linear regression clustering*

Description

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

Usage

```
ltr_reg_clus(Dat, k, resp.ind = 1)
```

Arguments

Dat	a data matrix
k	number of clusters
resp.ind	the column indices of the response variables for the linear regression clustering approach. The default is 1, which means that the first column is the univariate response variable

Value

a (left to right) clustering vector

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

---

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: • a logical value: if TRUE all states are considered as semi-Markovian else Markovian • a logical vector of length nstate: the TRUE associated states are considered as semi-Markovian and FALSE associated states are considered as Markovian

- 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:

- "nonparametric" non-parametric sojourn distribution
- "nbinom" negative binomial sojourn distribution
- "logarithmic" logarithmic sojourn distribution
- "poisson" poisson sojourn distribution
- "gamma" gamma sojourn distribution
- "weibull" weibull sojourn distribution
- "lnorm" log-normal sojourn distribution
- "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")
```

`miss_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 with missing values using the observation matrix and the estimated weight vectors

Usage

```
miss_mixmvnorm_mstep(x, wt1, wt2, par)
```

Arguments

`x`	the observation matrix which can contain missing values (NA or NaN)
`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)
`par`	the parameters of the model in the previous step of the EM algorithm. For initialization of the model when the data is initially imputed, `par` can be NULL

Value

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

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

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 = miss_mixmvnorm_mstep(CMAPSS$train$x, wt1, wt2, par=NULL)
```

`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>

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)
```

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

Description

The M step function of the EM algorithm for the mixture of Gaussian linear (Markov-switching) regressions as the emission distribution using the responses and covariates matrices and the estimated weight vectors

Usage

```
mixlm_mstep(x, wt1, wt2, resp.ind = 1)
```

Arguments

`x`	the observation matrix including responses and covariates
`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)
`resp.ind`	a vector of the column numbers of `x` which contain response variables. The default is 1, which means that the first column of `x` is the univariate response variable

Value

list of emission (mixture of Gaussian linear regression models) parameters: (`intercept`, `coefficients`, `csigma` (conditional covariance) and `mix.p`)

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

References

Kim, C. J., Piger, J. and Startz, R. (2008). Estimation of Markov regime-switching regression models with endogenous switching. Journal of Econometrics, 143(2), 263-273.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
```

```
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)
train <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
clus = initial_cluster(train = train, nstate = 3, nmix = c(1, 2, 1),
ltr = FALSE, final.absorb = FALSE, verbose = TRUE, regress = TRUE)
initmodel = initialize_model(clus = clus ,mstep = mixlm_mstep,
dens.emission = dmixlm, sojourn = NULL, semi = rep(FALSE, 3),
M = max(train$N),verbose = TRUE)
fit1 = hhsmmfit(x = train, model = initmodel, mstep = mixlm_mstep,
M = max(train$N))
plot(train$x[, 1] ~ train$x[, 2], col = train$s, pch = 16,
xlab = "x", ylab = "y")
abline(fit1$model$parms.emission$intercept[[1]],
fit1$model$parms.emission$coefficient[[1]], col = 1)
abline(fit1$model$parms.emission$intercept[[2]][[1]],
fit1$model$parms.emission$coefficient[[2]][[1]], col = 2)
abline(fit1$model$parms.emission$intercept[[2]][[2]],
fit1$model$parms.emission$coefficient[[2]][[2]], col = 2)
abline(fit1$model$parms.emission$intercept[[3]],
fit1$model$parms.emission$coefficient[[3]], col = 3)
```

`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)
```

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

Description

The M step function of the EM algorithm for the mixture of splines nonparametric density estimator

Usage

```
nonpar_mstep(x, wt, control = list(K = 5, lambda0 = 0.5))
```

Arguments

`x`	the observation matrix
`wt`	the state probabilities matrix (number of observations times number of states)
`control`	the parameters to control the M-step function. The simillar name is chosen with that of `dnonpar`, to be used in `...` argument of the `hhsmmfit` function. Here, it contains the following items: • `K` the degrees of freedom for the B-spline, default is `K=5` • `lambda0` the initial value of the smoothing parameter, default is `lambda0=0.5`

Value

list of emission (nonparametric mixture of splines) parameters: (`coef`)

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>, Reza Salehian, <reza.salehian@ut.ac.ir>

References

Langrock, R., Kneib, T., Sohn, A., & DeRuiter, S. L. (2015). Nonparametric inference in hidden Markov models using P-splines. *Biometrics*, 71(2), 520-528.

Examples

```
x <- rmvnorm(100, rep(0, 2), matrix(c(4, 2, 2, 3), 2, 2))
wt <- matrix(rep(1, 100), 100, 1)
emission = nonpar_mstep(x, wt)
coef <- emission$coef[[1]]
x_axis <- seq(min(x[, 1]), max(x[, 1]), length.out = 100)
y_axis <- seq(min(x[, 2]), max(x[, 2]), length.out = 100)
f1 <- function(x, y) {
  data = matrix(c(x, y), ncol = 2)
  tmpmodel = list(parms.emission = emission)
 dnonpar(data, 1, tmpmodel)
}
z1 <- outer(x_axis, y_axis, f1)
f2 <- function(x, y) {
  data = matrix(c(x, y), ncol = 2)
  dmvnorm(data, rep(0, 2), matrix(c(4, 2, 2, 3), 2, 2))
}
z2 <- outer(x_axis, y_axis, f2)
par(mfrow = c(1, 2))
persp(x_axis, y_axis, z1, theta = -60, phi = 45, col = rainbow(50))
persp(x_axis, y_axis, z2, theta = -60, phi = 45, col = rainbow(50))
```

`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", which also can contain missing values (NA or NaN)
future	number of future states to be predicted
method	the prediction method with two options: • "viterbi" (default) uses the Viterbi algorithm for prediction • "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: • "max" (default) the maximum probability as the point predict and the high probability critical values as the lower and upper bounds • "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))
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: • `"viterbi"` (default) uses the Viterbi algorithm for prediction • `"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)
```

raddreg *Random data generation from the Gaussian additive (Markov-switching) model for hhsmm model*

Description

Generates vectors of covariate and response observations from the Gaussian additive (Markov-switching) model, using B-Splines in a specified state and using the parameters of a specified model

Usage

```
raddreg(j, model, covar, ...)
```

Arguments

j	a specified state
model	a hhsmmspec model
covar	either a function which generates the covariate vector or a list containing the following items: • mean the mean vector of covariates (to be generated from multivariate normal distribution) • cov the variance-covariance matrix of covariates (to be generated from multivariate normal distribution)
...	additional arguments of the covar function

Value

a random matrix of observations from Gaussian additive (Markov-switching) model, in which the first columns are associated with the responses and the last columns are associated with the covariates

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

References

Langrock, R., Adam, T., Leos-Barajas, V., Mews, S., Miller, D. L., and Papastamatiou, Y. P. (2018). Spline-based nonparametric inference in general state-switching models. Statistica Neerlandica, 72(3), 179-200.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
par <- list(intercept = list(-21, -83, 33),
coef = list(array(c(1, 8, 52, 27, 38), dim = c(5, 1, 1)),
array(c(99, 87, 94, 77, 50), dim = c(5, 1, 1)),
array(c(-1, -8, -40, -22, -28), dim = c(5, 1, 1))),
sigma = list(0.2, 0.4, 0.1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dnorm_additive_reg, semi = semi)
train <- simulate(model, nsim = 70, seed = 1234,
remission = raddreg, covar = list(mean = 0, cov = 1))
plot(train$x[, 1] ~ train$x[, 2], col = train$s, pch = 16,
xlab = "x", ylab = "y")
```

`rmixar` *Random data generation from the mixture of Gaussian linear (Markov-switching) autoregressive models for hhsmm model*

Description

Generates vectors of observations from mixture of Gaussian linear (Markov-switching) autoregressive model in a specified state and using the parameters of a specified model

Usage

```
rmixar(j, model, x)
```

Arguments

`j`	a specified state
`model`	a `hhsmmspec` model
`x`	the previous x vector as the covariate of the autoregressive model

Value

a random matrix of observations from mixture of Gaussian linear (Markov-switching) autoregressive model

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

rmixlm *Random data generation from the mixture of Gaussian linear (Markov-switching) models for hhsmm model*

Description

Generates vectors of covariate and response observations from mixture of Gaussian linear (Markov-switching) models in a specified state and using the parameters of a specified model

Usage

```
rmixlm(j, model, covar, ...)
```

Arguments

j	a specified state
model	a hhsmmspec model
covar	either a function which generates the covariate vector or a list containing the following items: • mean the mean vector of covariates (to be generated from multivariate normal distribution) • cov the variance-covariance matrix of covariates (to be generated from multivariate normal distribution)
...	additional arguments of the covar function

Value

a random matrix of observations from mixture of Gaussian linear (Markov-switching) models, in which the first columns are associated with the responses and the last columns are associated with the covariates

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

References

Kim, C. J., Piger, J. and Startz, R. (2008). Estimation of Markov regime-switching regression models with endogenous switching. Journal of Econometrics, 143(2), 263-273.

Examples

```
J <- 3
initial <- c(1, 0, 0)
semi <- rep(FALSE, 3)
P <- matrix(c(0.5, 0.2, 0.3, 0.2, 0.5, 0.3, 0.1, 0.4, 0.5), nrow = J,
byrow = TRUE)
```

```
par <- list(intercept = list(3, list(-10, -1), 14),
coefficient = list(-1, list(1, 5), -7),
csigma = list(1.2, list(2.3, 3.4), 1.1),
mix.p = list(1, c(0.4, 0.6), 1))
model <- hhsmmspec(init = initial, transition = P, parms.emis = par,
dens.emis = dmixlm, semi = semi)

#use the covar as the list of mean and
#variance of the normal distribution

train1 <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = list(mean = 0, cov = 1))
plot(train1$x[,1] ~ train1$x[,2], col = train1$s, pch = 16,
xlab = "x", ylab = "y")

#use the covar as the runif function
#to generate one covariate from standard uniform distribution

train2 <- simulate(model, nsim = c(20, 30, 42, 50), seed = 1234,
remission = rmixlm, covar = runif, 1)
plot(train2$x[,1] ~ train2$x[,2], col = train2$s, pch = 16,
xlab = "x", ylab = "y")
```

`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)
```

`score` *the score of new observations*

Description

computes the score (log-likelihood) of new observations using a trained model

Usage

```
score(xnew, fit)
```

Arguments

`xnew`	a new single observation, observation matrix or a list of the class `hhsmmdata` containing $x and $N elements
`fit`	a fitted model using the `hhsmmfit` function

Value

the vector of scores (log-likelihood) of `xnew`

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

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, 4, 6, 7), 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))
score(test, fit1)
```

`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,
  ...,
 emission.control = list(autoregress = FALSE, lags = 1, start = list(mean = NULL, cov =
    NULL))
)
```

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
`emission.control`	a list of additional control parameters including the following items:

- autoregress logical. if TRUE the auto-regressive data generation will be considered with rmixar function
- lags a positive integer which is the number of lags to be considered for the auto-regressive sequence
- start a list containing the items mean which is the mean vector and cov which is the covarince matrix for starting value of the auto-regressive sequence (if autoregress == TRUE). If start is not specified the zero mean vector and the identity matrix will be considered as mean and cov, respectively.

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, trim.ratio = NULL)
```

Arguments

train	the train data of class "hhsmmdata"
train.ratio	a number in (0,1] which determines the ratio of the train subset. It can be equal to 1, if we need the test set to be equal to the train set and we only need to right trim the sequences
trim	logical. if TRUE the sequences will be right trimmed with random lengths
trim.ratio	a vector of trim ratios with a length equal to that of train$N, or a single trim ratio for all sequences. If it is NULL, then random trim ratios will be used

Details

This function splits the sample to train and test samples and trims the test sample from right, in order to provide a sample for examination of the prediction tools. 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"
- trimmed right trimmed test subset, if trim=TRUE, with trim ratios equal to trim.ratio
- trimmed.count the number of right trimmed individuals in each sequence of the test subset, if trim=TRUE

Author(s)

Morteza Amini, <morteza.amini@ut.ac.ir>

Examples

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

Index

additive_reg_mstep, 3, *12*
addreg_hhsmm_predict, 4

cov.miss.mix.wt, 5
cov.mix.wt, 7

dmixlm, 8
dmixmvnorm, 10, *19*, *26*
dnonpar, 11, *32*
dnorm_additive_reg, *3*, 12

hhsmmdata, 13, *41*
hhsmmfit, *3*, *11*, *12*, 14, *32*, *33*, *41*
hhsmmspec, *14*, 16, *18*, *20*, *27*, *37–40*, *42*
homogeneity, 18

initial_cluster, *18*, 20, *22*
initial_estimate, 22, *26*
initialize_model, *14*, 18

lagdata, 24
ltr_clus, 25
ltr_reg_clus, 25

make_model, 26
miss_mixmvnorm_mstep, *19*, 28
mixdiagmvnorm_mstep, 29
mixlm_mstep, 30
mixmvnorm_mstep, *19*, *23*, *26*, 31

nonpar_mstep, *11*, 32

predict.hhsmm, 33, *35*, *36*
predict.hhsmmspec, *34*, 35

raddreg, 37
rmixar, 38
rmixlm, 39
rmixmvnorm, 40, *42*

score, 41
simulate.hhsmmspec, 42

train_test_split, 43