

# Package ‘RHmm’

August 29, 2013

**Version** 2.0.3

**Date** 2013-04-13

**Title** Hidden Markov Models simulations and estimations

**Author** Ollivier TARAMASCO, Sebastian Bauer

**Maintainer** Ollivier TARAMASCO <ollivier.Taramasco@imag.fr>

**Depends** R (>= 2.10.0), MASS, nlme

**Description** Discrete, univariate or multivariate gaussian, mixture of  
univariate or multivariate gaussian HMM functions for simulation and estimation.

**License** GPL (>= 2)

**URL** <http://www.r-project.org>, <http://r-forge.r-project.org/projects/rhmm/>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2013-04-13 16:01:08

## R topics documented:

asymptoticCov . . . . .	2
asymptoticIterSimCovMat . . . . .	3
computeScoreAndInformation . . . . .	4
data_mixture . . . . .	5
distributionSet . . . . .	6
forwardBackward . . . . .	8
HMMFit . . . . .	10
HMMGraphicDiag . . . . .	13
HMMPlotSerie . . . . .	14
HMMSet . . . . .	15
HMMSim . . . . .	17
obs_n1d_3s . . . . .	18

obs_n3d_2s . . . . .	19
setAsymptoticCovMat . . . . .	20
viterbi . . . . .	21
weather . . . . .	22

<b>Index</b>	<b>23</b>
--------------	-----------

---

asymptoticCov	<i>Asymptotic covariance matrix of the HMM parameters</i>
---------------	---

---

**Description**

This function calculates the empirical asymptotic covariance matrix of the HMM parameters

**Usage**

```
asymptoticCov(HMM, obs)
```

**Arguments**

HMM	A HMMClass or a HMMFitClass object
obs	The vector, matrix, data frame, list of vectors or list of matrices of observations

**Value**

A matrix

**Numerical computations**

The Information matrix (of the independent parameters) is computed using the Lystig and Hugues’s algorithm. Then the covariance matrix is computed by inversion of this information matrix.

**References**

Lystig Theodore C. and Hugues James P. (2002) *Exact Computation of the Observed Information Matrix for Hidden Markov Models*, Journal of Computational and Graphical Statistics, Vol. 11, No 3, 678-689.

**See Also**

HMMFit

**Examples**

```
data(n1d_3s)
Res_n1d_3s<-HMMFit(obs_n1d_3s, nStates=3)
covMat <- asymptoticCov(Res_n1d_3s, obs_n1d_3s)
```

---

asymptoticIterSimCovMat

*Compute the asymptotic covariance matrix of a fitted HMM by simulation*

---

## Description

This ‘new’ function computes the empirical asymptotic covariance matrix of the fitted HMM.

## Usage

```
asymptoticIterSimCovMat(HMM, obs, nSimul, verbose=FALSE, oldCovMat=NULL)
```

## Arguments

HMM	a HMMClass or HMMFitClass object
obs	A vector, a matrix, a data frame, a list of vectors or a list of matrices of observations. <b>See HMMFit.</b>
nSimul	The number of simulation
verbose	A boolean. if true, displays some informations. Default false.
oldCovMat	An object containing <ul style="list-style-type: none"> <li>• esp2man: The current matrix of the empirical mean of <math>\theta\% * \%t(\theta)</math></li> <li>• mean: The current vector of the empirical mean of <math>\theta</math></li> <li>• cov: the current empirical covariance matrix of <math>\theta</math></li> <li>• nSimul: The current number of simulations</li> </ul> where $\theta$ is the vector of all parameters of the HMM.

## Value

An object with the same attributes than ‘oldCovMat’ parameter.

## Numerical computations

This is an “experimental” method. The HMM model is simulated nSimul times then fitted and the empirical covariance matrix is computed.

## See Also

[setAsymptoticCovMat](#), [asymptoticCov](#)

**Examples**

```
# Fit a 3 states 1D-gaussian model
data(n1d_3s)
Res <- HMMFit(obs_n1d_3s, nStates=3)
# First 10 computations of covariance matrix
Cov <- asymptoticIterSimCovMat(Res, obs_n1d_3s, 10)
# 10 more computations of covariance matrix
Cov <- asymptoticIterSimCovMat(Res, obs_n1d_3s, 10, verbose=TRUE, oldCovMat=Cov)
Res<-setAsymptoticCovMat(Res, Cov$cov)
summary(Res)
```

---

computeScoreAndInformation

*Score and Information matrix of the HMM parameters*

---

**Description**

This function calculates the score and the information matrix of the independent parameters of the HMM, using Lystig and Hugues's algorithm.

**Usage**

```
computeScoreAndInformation(HMM, obs)
```

**Arguments**

HMM	A HMMClass or a HMMFitClass object
obs	The vector, matrix, data frame, list of vectors or list of matrices of observations

**Value**

score	the score vector of the independent parameters
information	the information matrix of the independent parameters

**References**

Lystig Theodore C. and Hugues James P. (2002) *Exact Computation of the Observed Information Matrix for Hidden Markov Models*, Journal of Computational and Graphical Statistics, Vol. 11, No 3, 678-689.

**See Also**

HMMFit

**Examples**

```
data(n1d_3s)
Res_n1d_3s<-HMMFit(obs_n1d_3s, nStates=3)
ScoreAndInfor <- computeScoreAndInformation(Res_n1d_3s, obs_n1d_3s)
```

---

data_mixture	<i>Simulated univariate mixture of 3 gausssian distributions</i>
--------------	--

---

**Description**

This data set contains 10 samples with 100 observations of this HMM

**Usage**

```
data(data_mixture)
```

**Format**

R script

**Note**

Parameters for the simulation Model:

2 states HMM with gaussian mixture distribution

Initial probabilities:

Pi1	Pi2
0.4	0.6

Transition matrix:

	State 1	State 2
State 1	0.8	0.2
State 2	0.4	0.6

Conditionnal distribution parameters:

State 1

mean	var	prop
------	-----	------

mixt. 1	1	2	0.1
mixt. 2	2	3	0.2
mixt. 3	3	4	0.7

State 2

	mean	var	prop
mixt. 1	1 -1	1 4	1 0.4
mixt. 2	1 -2	1 2	1 0.3
mixt. 3	1 -3	1 1	1 0.3

---

distributionSet	<i>Set the parameters for the distributions of observations</i>
-----------------	---

---

**Description**

This function is used to create a distributionClass object which contains the parameters of the distribution of the observations for each hidden state. Since distributions can be univariate or multivariate, discrete or continuous, the different values of a distributionClass object depend of the nature of the distribution.

**Usage**

```
distributionSet(dis, ...)
```

**Arguments**

dis	Name of the distribution of observations. In 'NORMAL', 'DISCRETE', 'MIXTURE'.
...	Other parameters. See details.

**Details**

Typical usages are:

- distributionSet(dis="NORMAL", mean, var)
- distributionSet(dis="NORMAL", mean, cov)
- distributionSet(dis="MIXTURE", mean, var, proportion)
- distributionSet(dis="MIXTURE", mean, cov, proportion)
- distributionSet(dis="DISCRETE", proba, labels=NULL)

The parameters are:

**mean** - *Univariate normal distribution*: a vector of the means for each state of the model.  
 - *Multivariate normal distribution*: a list of the mean vectors for each state of the model.  
 - *Mixture of univariate normal distribution*: a list of vectors of the mixture means for each state of the model.  
 - *Mixture of multivariate normal distribution*: a list of lists of vectors of means for each state and each component of the mixture of the model.

**var** - *Univariate normal distribution*: a vector of the variances for each states of the model.  
 - *Mixture of univariate normal distribution*: a list of vectors of the mixture variances for each states of the model.

**cov** - *Multivariate normal distribution*: a list of covariance matrices of the multivariate normal distribution for each state of the model  
 - *Mixture of multivariate normal distribution*: a list of list of covariance matrices for each state and each component of the mixture.

**proportion** A list of vector of the mixture proportions for each state of the model.

**proba** A list of vector of discrete probabilities for each state of the model.

**labels** A vector of the labels of the discrete observations. Default NULL.

### Value

An 'distributionClass' class object with some of the following elements:

**dis** The name of the distribution.

**nStates** Number of hidden states.

**dimObs** Dimension of observations.

**nMixt** Number of mixtures for mixture of normal distributions.

**nLevels** Number of levels for discrete distributions.

**mean** The 'mean' argument for univariate normal, mixture of univariate normal and multivariate normal distributions.

**var** The 'var' argument for univariate normal and mixture of univariate normal distributions

**cov** The 'cov' argument for multivariate normal and mixture of multivariate normal distributions

**proba** The 'proba' argument for discrete distributions

### Examples

```
# 3 hidden states Markov Model with univariate normal distributions
# for the observations
# obs | hidden state = 1 are N(1, 1)
# obs | hidden state = 2 are N(-2, 2)
# obs | hidden state = 3 are N(5, 4)
n_1d_3s <- distributionSet("NORMAL", mean=c(1, -2, 5), var=c(1, 2, 4))
# 2 hidden states Markov Model with bivariate normal distributions
# for the observations
# obs | hidden state = 1 are N(m1, cov1)
# obs | hidden state = 2 are N(m2, cov2)
m1 <- c(1,1)
m2 <- c(-2, -2)
```

```

cov1 <- matrix(c(1, 1, 1, 4), nrow=2)
cov2 <- matrix(c(1, -1, -1, 9), nrow=2)
n_2d_2s <- distributionSet("NORMAL", mean=list(m1, m2),
                           cov=list(cov1, cov2))
# 3 hidden states Markov Model with a mixture of two normal
# distributions for the observations
# obs | hidden state = i are:
#   pi[1] * N(mmi[1], vari[1]) + pi[2] * N(mmi[2], vari[2])

mm1 <- c(1, -1)
mm2 <- c(-2, 2)
mm3 <- c(5, 5)
var1 <- c(1, 2)
var2 <- c(2, 3)
var3 <- c(1, 1)
p1 <- c(0.5, 0.5)
p2 <- c(0.8, 0.2)
p3 <- c(0.3, 0.7)
mn_2s <- distributionSet("MIXTURE", mean=list(mm1, mm2, mm3),
                        var=list(var1, var2, var3), proportion=list(p1, p2, p3))
# 2 hidden states Markov Model with discrete observations
dp1 <- c(0.2, 0.3, 0.3, 0.2)
dp2 <- c(0.1, 0.1, 0.1, 0.7)
labels <- c("I", "M", "A", "G")
d_2s <- distributionSet("DISCRETE", proba=list(dp1, dp2),
                      labels=labels)
# 2 hidden states Markov model with mixture of 3 2-d gaussian distribution
q1 <- rep(1/3, 3)
q2 <- runif(3)
q2 <- q2/sum(q2)
cov3 <- matrix(c(1,2,2,10), nrow=2)
cov4 <- matrix(c(1, 0, 0, 1), nrow=2)
cov5 <- matrix(c(2,4,4,50), nrow=2)
cov6 <- matrix(c(25,1, 1, 2), nrow=2)
mm4 <- c(100, 20)
mm5 <- c(20, -20)
mm6 <- c(0, 0)
m_2d_2s <- distributionSet("MIXTURE", mean=list(list(mm1,mm2,mm3), list(mm4,mm5,mm6)),
                        cov=list(list(cov1,cov2,cov3), list(cov4,cov5,cov6)), proportion=list(q1,q2))

```

---

forwardBackward

*forward-backward function*


---

## Description

The forward-backward function is used to compute quantities used in the Baum-Welch algorithm.

## Usage

```
forwardBackward(HMM, obs, logData=TRUE)
```



**Arguments**

HMM	a HMMClass or a HMMFitClass object
obs	a vector (matrix) of observations, or a list of vectors (or matrices) if there are more than one samples
logData	a boolean. If true, the function computes the logarithm of the Alpha, Beta and Rho quantities instead of the quantities themselves.

**Value**

If obs is one sample, a list of following elements, if obs is a list of samples, a list of list of following elements. See **note** for mathematical definitions.

Alpha	The matrix of (log) 'forward' probabilities (density) (size: number of obs. times number of hidden states)
Beta	The matrix of (log) 'backward' probabilities (density) (size: number of obs. times number of hidden states)
Delta	The matrix of 'conditional forward' probabilities (density) (size: number of obs. times number of hidden states)
Gamma	The matrix of probabilities (density) of being at time t in state i (size: number of obs. times number of hidden states)
Xsi	The matrix of probabilities (density) of being in state i at time t and being in state j at time t + 1 (size: number of obs. times number of hidden states)
Rho	The vector of (log) probabilities (density) of seeing the partial sequence obs[1] ... obs[t] (size number of obs.)
LLH	Log-likelihood

**Note**

Let  $o = (o(1), \dots, o(T))$  be the vector of observations, and  $O = (O(t), t = 1, \dots, T)$ , the corresponding random variables. Let  $Q = (Q(t), t = 1, \dots, T)$  be the hidden Markov chain whose values are in  $\{1, \dots, nStates\}$  We have the following definitions:

$\alpha_i(t) = P(O_1 = o(1), \dots, O(t) = o(t), Q(t) = i | HMM)$  which is the probability of seeing the partial sequence  $o(1), \dots, o(t)$  and ending up in state i at time t.

$\beta_i(t) = P(O_{t+1} = o(t+1), \dots, O(T) = o(T), Q(t) = i | HMM)$  which is the probability of the ending partial sequence  $o(t+1), \dots, o(T)$  given that we started at state i at time t.

$\delta_i(t) = P(Q(t) = i | O_1 = o(1), \dots, O(t) = o(t), HMM)$  which is the probability of being in state i at time t knowing the partial sequence  $o(1), \dots, o(t)$ .

$\Gamma_i(t) = P(Q(t) = i | O = o, HMM)$  which is the probability of being in state i at time t for the state sequence  $O = o$ .

$\xi_{ij}(t) = P(Q(t) = i, Q(t+1) = j | O = o, HMM)$  which is the probability of being in state i at

time  $t$  and being in state  $j$  at time  $t + 1$ .

$\rho(t) = P(O_1 = o(1), \dots, O_t = o(t) | HMM)$  witch is probabilities of seeing the partial sequence  $o(1), \dots, o(t)$ .

$$LLH = \ln \rho[T]$$

When the sequences of observations become larger, the probabilistic values in this algorithm get increasingly small and after enough iterations become almost zero. For that reason, the Alphas, Betas and Rhos are scaled during the iterations of the algorithm to avoid underflow problems. The logarithm of these probabilistic values are compute from the logarithm of the scaled quantities and should produce a more precise result.

If the conditional distribution is continuous, the probabilistic values are replaced by density values and can be greater than one.

## References

Jeff A. Bilmes (1997) *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models* <http://ssli.ee.washington.edu/people/bilmes/mypapers/em.ps.gz>

## Examples

```
data(n1d_3s)
#Fits an 2 states gaussian model for geyser duration
Res_n1d_3s <- HMMFit(obs_n1d_3s, nStates=3)
#Forward-backward procedure with log Alpha, Beta, Rho
fbLog <- forwardBackward(Res_n1d_3s, obs_n1d_3s)
#Forward-backward procedure with Alpha Beta and Rho
fb <- forwardBackward(Res_n1d_3s, obs_n1d_3s, FALSE)
```

---

HMMFit

*Fit an Hidden Markov Model*


---

## Description

This function returns an HMMFitClass object which contains the results of the Baum-Welch algorithm for the user's data

## Usage

```
HMMFit(obs, dis="NORMAL", nStates=2, asymptCov=FALSE, ...)
```

**Arguments**

<b>obs</b>	A vector, a matrix, a data frame, a list of vectors or a list of matrices of observations. See section <b>obs parameter</b> .
<b>dis</b>	Distribution name. In 'NORMAL', 'DISCRETE' or 'MIXTURE'. Default 'NORMAL'.
<b>nStates</b>	Number of hidden states. Default 2.
<b>asymptCov</b>	A boolean. <code>asymptCov=TRUE</code> if the asymptotic covariance matrix is computed. Default FALSE.
<b>...</b>	Other parameters. See <b>Details</b> .

**Details**

Typical usages are:

- `HMMFit(obs, dis="NORMAL", nStates=, ..., asymptCov=FALSE)`
- `HMMFit(obs, dis="DISCRETE", nStates=, levels=NULL, ..., asymptCov=FALSE)`
- `HMMFit(obs, dis="MIXTURE", nStates=, nMixt=, ..., asymptCov=FALSE)`

The different arguments are:

**obs** A vector, a matrix, a data frame, a list of vectors or a list of matrices of observations. See section **obs parameter**.

**dis** Distribution name. In 'NORMAL', 'DISCRETE' or 'MIXTURE'. Default 'NORMAL'.

**nStates** Number of hidden states. Default 2.

**nMixt** Number of mixtures of normal distributions if `dis = 'MIXTURE'`

**levels** A character vector of all different levels of 'obs'. By Default (`levels=NULL`), this vector is computed from 'obs'.

**asymptCov** A boolean. `asymptCov=TRUE` if the asymptotic covariance matrix is computed. Default FALSE.

**...** optional parameter:

**control:** A list of control parameters for the Baum-Welch algorithm. See **control parameter**

**Value**

a 'HMMFitClass' object with the following elements:

**HMM** A HMMClass object with the fitted values of the model. See **HMMSet**.

**LLH** Log-likelihood

**BIC** BIC criterium

**nIter** Number of iterations of the Baum-Welch algorithm

**relVariation** last relative variation of the LLH function

**asymptCov** Asymptotic covariance matrix of independent parameters. NULL if not computed.

**obs** the observations.

**call** The call object of the function call

**obs parameter**

If you fit the model with only one sample, `obs` is either a vector (for univariate distributions) or a matrix (for multivariate distributions) or a data frame. In the two last cases, the number of columns of `obs` defines the dimension of observations.

If you fit the model with more than one sample, `obs` is a list of samples. Each element of `obs` is then a vector (for univariate distributions) or a matrix (for multivariate distributions). The samples do not need to have the same length.

For discrete distributions, `obs` can be a vector (or a list of vectors) of any type of R factor objects.

**control parameter**

**init** Kind of initialisation = 'KMEANS' (for univariate or multivariate normal distributions), 'RANDOM' or 'USER'. Default 'RANDOM', see **Random Initialization**

**iter** Maximum number of iterations for the Baum-Welch algorithm. Default 500

**tol** Tolerance of the relative log-likelihood augmentation. Default 1e-6

**verbose** =0, no details, =1 iterations are displayed. Default 0

**nInit** Number of random initialisations. Default 5

**nIterInit** Number of maximum iterations of the Baum-Welch algorithm in the random initialisation phase. Default 5

**initPoint** An HMMClass object used to initialize the parameters of the Baum-Welch algorithm. Default NULL.  
if `initPoint != NULL`, `init` is set to "USER"

**random initialization**

'`initProb`' and '`transMat`' parameters are uniformly drawn.

For univariate normal distributions, empirical mean  $m$  and variance  $\sigma^2$  of all the samples are computed. Then for every states, an initial value of the '`mean`' parameter is uniformly drawn between  $m - 3\sigma$  and  $m + 3\sigma$  and an initial value of the '`var`' parameter is uniformly drawn between  $\frac{1}{2}\sigma^2$  and  $3\sigma^2$ .

For multivariate normal distributions, the same procedure is applied for each component of the mean vectors. The initial covariance matrix is diagonal, and each initial variance is computed as for univariate models.

For mixtures of univariate normal distributions, initial values for '`mean`' and '`var`' parameters are computed the same way than for normal distributions. The initial value of '`proportion`' parameter is uniformly drawn.

For mixtures of multivariate normal distributions, the same procedure is applied for each component of the mean vectors, all the covariance matrices are diagonal and each initial variance is computed as for univariate models. The initial value of '`proportion`' parameter is also uniformly drawn.

For discrete distributions, the initial values of '`proba`' parameters are uniformly drawn.

Of course, the initial values of the parameters '`initProba`', '`proba`', '`proportion`' and '`transMat`' are standardized to ensure that they can represent probabilities vectors or transition matrices.

**asymptotic covariance matrix**

The asymptotic covariance matrix of estimates is computed using the Lystig and Hugues's algorithm. See **asymptoticCov**.

The summary and print.summary methods display the results.

**References**

Bilmes Jeff A. (1997) *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models* <http://ssli.ee.washington.edu/people/bilmes/mypapers/em.ps.gz>

**Examples**

```
# Fit a 3 states 1D-gaussian model
data(n1d_3s)
HMMFit(obs_n1d_3s, nStates=3)

# Fit a 3 states gaussian HMM for obs_n1d_3s
# with iterations printing and kmeans initialization
Res_n1d_3s <- HMMFit(obs=obs_n1d_3s, nStates=3,
  control=list(verbose=1, init="KMEANS"),
  asymptCov=TRUE)
summary(Res_n1d_3s)

# Fit a 2 states 3D-gaussian model
data(n3d_2s)
summary(HMMFit(obs_n3d_2s, asymptCov=TRUE))

# Fit a 2 states mixture of 3 normal distributions HMM
# for data_mixture
data(data_mixture)
ResMixture <- HMMFit(data_mixture, nStates=2, nMixt=3,
  dis="MIXTURE")

# Fit a 3 states discrete HMM for weather data
data(weather)
ResWeather <- HMMFit(weather, dis='DISCRETE', nStates=3)
```

---

HMMGraphicDiag

*Graphic diagnostic of the HMM estimation*


---

**Description**

This function plots the kernel density of the observations and the normal (mixture of normal, discrete) density with estimated parameters for each hidden states.

**Usage**

```
HMMGraphicDiag(vit, HMM, obs, color="green")
```

**Arguments**

<code>vit</code>	A ViterbiClass object which gives the hidden states
<code>HMM</code>	A HMMClass or a HMMFitClass object which describes the model
<code>obs</code>	The vector, list of vectors of observations
<code>color</code>	Color for the kernel density plot

**Value**

None.

**Note**

HMMGraphicDiag is not implemented for multivariate distributions.

The kernel densities of observations for each hidden states of the model are plotting using:  
`plot(density(obs[vit$states=i]))` and `i` in `1..HMM$nStates` (or `HMM$HMM$nStates`)

**See Also**

HMMFit, viterbi

**Examples**

```
data(n1d_3s)
obs <- obs_n1d_3s
#Fits an 3 states gaussian model
ResFit <- HMMFit(obs, nStates=3)
VitPath <- viterbi(ResFit, obs)
# Graphic diagnostic
HMMGraphicDiag(VitPath, ResFit, obs)
```

---

HMMPlotSerie

---

*Plot univariates series in each estimated states*


---

**Description**

This function plots the time series in each hidden state.

**Usage**

```
HMMPlotSerie(obs, states, dates=NULL, dis="NORMAL", color="green")
```

**Arguments**

obs	The vector or the list of vectors of observations.
states	A ViterbiClass object which gives the hidden states or a vector or a list of vectors of integer from 1 to the number of hidden states.
dates	An R object representing dates that can be plot as axis labels (e.g. Date object)
dis	Distribution name = 'NORMAL', 'DISCRETE', 'MIXTURE'. Default 'NORMAL'.
color	Color for the kernel density plot

**Value**

None.

**Note**

HMMPlotSerie is not implemented for multivariate distributions.

The time series of observations for each hidden states of the model are plotted using:  
`plot(obs[states=i])` and `i` in `1..max(states)`.

**See Also**

`viterbi`, `plot`

**Examples**

```
data(n1d_3s)
#Fits an 3 states gaussian model
ResFit <- HMMFit(obs_n1d_3s, nStates=3)
VitPath <- viterbi(ResFit, obs_n1d_3s)
#plot the series
HMMPlotSerie(obs_n1d_3s, VitPath)
```

---

HMMSet

---

*Set the parameters for the hidden Markov models*


---

**Description**

This function is used to create a HMMClass object which contains the parameters of the HMM. An HMM is described by an initial state probability vector, transition matrices and a distributionClass object.

**Usage**

```
HMMSet(initProb, transMat, ...)
```

**Arguments**

<code>initProb</code>	The vector of probabilities of the initial state
<code>transMat</code>	The transition matrix of the hidden Markov chain. Since version 1.3.4 of RHmm, a list of transition matrices can be specified in order to define an inhomogeneous HMM.
<code>...</code>	Other parameters. See details.

**Details**

Typical usages are:

- `HMMSet(initProb, transMat, distribution)`
- `HMMSet(initProb, transMat, dis="NORMAL", mean, var)`
- `HMMSet(initProb, transMat, dis="NORMAL", mean, cov)`
- `HMMSet(initProb, transMat, dis="MIXTURE", mean, var, proportion)`
- `HMMSet(initProb, transMat, dis="DISCRETE", proba, labels=NULL)`

The different arguments are:

**distribution** The `distributionClass` object of the observations

**dis** `dis` parameter. See **distributionSet**

**mean** `mean` parameter. See **distributionSet**

**var** `var` parameter. See **distributionSet**

**cov** `cov` parameter. See **distributionSet**

**proportion** `proportion` parameter. See **distributionSet**

**proba** `proba` parameter. See **distributionSet**

**labels** `labels` parameter. See **distributionSet**

**Value**

An ‘HMMClass’ class object with the following elements:

**initProb** Initial state probabilities vector

**transMat** Transition matrix

**distribution** The `distributionClass` object which describes the conditional distribution. See **distributionSet**.

**See Also**

[distributionSet](#)



**Examples**

```
# 3 hidden states Markov Model with univariate normal distributions
# for the observations
# obs | hidden state = 1 are N(1, 1)
# obs | hidden state = 2 are N(-2, 2)
# obs | hidden state = 3 are N(5, 4)

n_1d_3s <- distributionSet("NORMAL", c(1, -2, 5), c(1, 2, 4))
initProb3 <- rep(1,3)/3
transMat3 <- rbind(c(0.5, 0.4, 0.1), c(0.3, 0.4, 0.3),
  c(0.2, 0.1, 0.7))
hmm1 <- HMMSet(initProb3, transMat3, n_1d_3s)
# or directly
hmm2 <- HMMSet(initProb3, transMat3, "NORMAL", mean=c(1, -2, 5),
  var=c(1, 2, 4))
```

HMMSim

*Simulation of an Hidden Markov Model***Description**

Simulation of an HMM for different classes of observations distributions.

**Usage**

```
HMMSim(nSim, HMM, lastState=NULL)
```

**Arguments**

nSim	Number of simulations.
HMM	An HMMClass object. See <b>HMMSet</b>
lastState	If not NULL, the state of the previous observation (usefull to complete a serie)

**Value**

a list with

obs	simulated observations (a vector for univariate distributions, a matrix for multi-variate distributions)
states	simulated hidden states

**See Also**

[HMMSet](#)

Examples

```
# simulate a 3 hidden states model with univariate normal distributions
n_1d_3s <- distributionSet("NORMAL", mean=c(1, -2, 5), var=c(1, 2, 4))
initProb3 <- rep(1,3)/3
transMat3 <- rbind(c(0.5, 0.4, 0.1), c(0.3, 0.4, 0.3), c(0.2, 0.1, 0.7))
hmm_1d_3s <- HMMSet(initProb3, transMat3, n_1d_3s)
simul <- HMMSim(1000, hmm_1d_3s)
#Complete the serie.
simul <- c(simul, HMMSim(1000, hmm_1d_3s, simul$states[1000]))
```

---

obs_n1d_3s	<i>A 3 states HMM with univariate gaussian distribution data set</i>
------------	--

---

Description

This data set contains 10 samples with 100 observations of this HMM.

Usage

```
data(n1d_3s)
```

Format

RData file

Details

File 'n1d\_3s.RData' contains the list of vectors 'obs\_n1d\_3s'.

Note

Parameters for the simulation  
Model:

3 states HMM with univariate gaussian distribution

Initial probabilities:

Pi1	Pi2	Pi3
0.2	0.4	0.4

Transition matrix:

State 1	State 2	State 3
---------	---------	---------

State 1	0.5	0.1	0.4
State 2	0.2	0.7	0.1
State 3	0.3	0.3	0.4

Conditionnal distribution parameters:

Distribution parameters:

	mean	var
State 1	10	4
State 2	-5	2
State 3	-1	1

---

obs\_n3d\_2s

*A 2 states HMM with 3D gaussian distribution data set*


---

## Description

This data set contains 1 sample with 1000 observations of this HMM.

## Usage

```
data(n3d_2s)
```

## Format

RData file

## Details

File 'n3d\_3s.RData' contains the matrix 'obs\_n3d\_2s'.

## Note

Parameters for the simulation

Model:

---

2 states HMM with 3-d gaussian distribution

Initial probabilities:

Pi1	Pi2
0.2	0.8

Transition matrix:

	State 1	State 2
State 1	0.1	0.9
State 2	0.8	0.2

Conditionnal distribution parameters:

Distribution parameters:  
State 1

mean	cov matrix		
2	1.0	1.6	-0.6
2	1.6	4.0	-3.6
2	-0.6	-3.6	9.0

State 2

mean	cov matrix		
-1	4.0	1.6	1.6
-2	1.6	1.0	1.0
-3	1.6	1.0	4.0

---

setAsymptoticCovMat	<i>Set the asymptotic covariance matrix of a fitted HMM</i>
---------------------	---

---

**Description**

This function sets the empirical asymptotic covariance matrix of the fitted HMM

**Usage**

setAsymptoticCovMat(HMMFit, asymptCovMat)

**Arguments**

- HMMFit            a HMMFitClass object
- asymptCovMat    The covariance matrix of the fitted model

**Value**

The HMMFit object

**See Also**

asymptoticCovMat

---

viterbi	<i>Viterbi algorithm</i>
---------	--------------------------

---

**Description**

This function calculates the optimal hidden states sequence using Viterbi's algorithm.

**Usage**

```
viterbi(HMM, obs)
```

**Arguments**

HMM	a HMMClass or a HMMFitClass object.
obs	The vector, matrix, data frame, list of vectors or list of matrices of observations.

**Value**

a viterbiClass object which is a list with:

States	Sequence of hidden states in 1...nStates
logViterbiScore	logarithm of the Viterbi's Score.
logProbSeq	logarithm of probability of having the sequence of states conditionally to having the observations.

**References**

Among hundreds of tutorials, you can have a look to use  
 Phil Blunsom (2004) *Hidden Markov Models*. <http://www.cs.mu.oz.au/460/2004/materials/hmm-tutorial.pdf>

**See Also**

[HMMSet](#), [HMMFit](#)

**Examples**

```
data(n1d_3s)
ResFit <- HMMFit(obs_n1d_3s, nStates=3)
VitPath <- viterbi(ResFit, obs_n1d_3s)
```

---

weather

*Simulated discrete HMM*

---

**Description**

This data set contains 5 samples of discrete HMM

**Usage**

```
data(weather)
```

**Format**

R script

**Source**

I took this example from Professor RD Boyle's tutorial, I used the same parameters and simulated the data with HMMSim.

**References**

<http://www.comp.leeds.ac.uk/roger/>

# Index

- \*Topic **datagen**
  - HMMSim, [17](#)
- \*Topic **datasets**
  - data\_mixture, [5](#)
  - obs\_n1d\_3s, [18](#)
  - obs\_n3d\_2s, [19](#)
  - weather, [22](#)
- \*Topic **distribution**
  - distributionSet, [6](#)
  - HMMSet, [15](#)
  - HMMSim, [17](#)
- \*Topic **hplot**
  - viterbi, [21](#)
- \*Topic **htest**
  - forwardBackward, [8](#)
  - HMMFit, [10](#)
  - HMMGraphicDiag, [13](#)
  - HMMPlotSerie, [14](#)
- \*Topic **models**
  - distributionSet, [6](#)
  - HMMSet, [15](#)

asymptoticCov, [2](#), [3](#)  
asymptoticIterSimCovMat, [3](#)

computeScoreAndInformation, [4](#)

data\_mixture, [5](#)  
distributionClass(distributionSet), [6](#)  
distributionSet, [6](#), [16](#)

forwardBackward, [8](#)  
forwardbackward(forwardBackward), [8](#)

HMMClass(HMMSet), [15](#)  
HMMFit, [10](#), [21](#)  
HMMFitClass(HMMFit), [10](#)  
HMMGraphicDiag, [13](#)  
HMMPlotSerie, [14](#)  
HMMSet, [15](#), [17](#), [21](#)  
HMMSim, [17](#)

n1d\_3s(obs\_n1d\_3s), [18](#)  
n3d\_2s(obs\_n3d\_2s), [19](#)

obs\_n1d\_3s, [18](#)  
obs\_n3d\_2s, [19](#)

print.summary.HMMFitClass(HMMFit), [10](#)

setAsymptoticCovMat, [3](#), [20](#)  
summary.HMMFitClass(HMMFit), [10](#)

viterbi, [21](#)

weather, [22](#)