

Package ‘mixAK’

August 13, 2009

Version 0.5

Date 2009-08-12

Title Mixture of methods including mixtures

Author Arnost Komarek <arnost.komarek@mff.cuni.cz>

Maintainer Arnost Komarek <arnost.komarek@mff.cuni.cz>

Depends R (>= 2.0.0), coda, colorspace, lme4

Description This package contains a mixture of statistical methods including the MCMC methods to analyze normal mixtures

License GPL (>= 2)

URL <http://www.karlin.mff.cuni.cz/~komarek>

Repository CRAN

Date/Publication 2009-08-13 08:58:49

R topics documented:

Acidity	2
autolayout	3
BLA	4
Dirichlet	5
Enzyme	6
Faithful	7
Galaxy	8
GLMM_longitClust	9
GLMM_MCMC	10
MatMPpinv	15
MatSqrt	16
MVN	18
MVNmixture	20
NMixChainsDerived	24

NMixClust	25
NMixMCMC	25
NMixPlugCondDensJoint2	34
NMixPlugCondDensMarg	36
NMixPlugDensJoint2	37
NMixPlugDensMarg	39
NMixPredCDFMarg	40
NMixPredCondDensJoint2	42
NMixPredCondDensMarg	44
NMixPredDensJoint2	46
NMixPredDensMarg	48
NMixSummComp	50
plot.NMixPlugCondDensJoint2	51
plot.NMixPlugCondDensMarg	52
plot.NMixPlugDensJoint2	53
plot.NMixPlugDensMarg	54
plot.NMixPredCDFMarg	55
plot.NMixPredCondDensJoint2	56
plot.NMixPredCondDensMarg	57
plot.NMixPredDensJoint2	58
plot.NMixPredDensMarg	59
rRotationMatrix	61
rSamplePair	62
SP2Rect	63
Tandmob	64
TandmobEmer	66
TMVN	68
TNorm	70
Wishart	73
Y2T	76
Index	77

Acidity

Acidity index of lakes in North-Central Wisconsin

Description

Acidity index measured in a sample of 155 lakes in North-Central Wisconsin.

Crawford et al. (1992) and Crawford (1994) analyzed these data as a mixture of normal distributions on the log-scale. Richardson and Green (1997) used a normal mixture estimated using reversible jump MCMC.

Usage

`data(Acidity)`

Format

A numeric vector with observed values.

Source

<http://www.stats.bris.ac.uk/~peter/mixdata>

References

Crawford, S. L. (1994). An application of the Laplace method to finite mixture distribution. *Journal of the American Statistical Association*, **89**, 259–267.

Crawford, S. L., DeGroot, M. H., Kadane, J. B., and Small, M. J. (1994). Modeling lake chemistry distributions: Approximate Bayesian methods for estimating a finite mixture model. *Technometrics*, **34**, 441–453.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with unknown number of components (with Discussion). *Journal of the Royal Statistical Society, Series B*, **59**, 731–792.

Examples

```
data(Acidity)
summary(Acidity)
```

autolayout

Automatic layout for several plots in one figure

Description

Returns a matrix which can be used in `layout` function as its `mat` argument.

Usage

```
autolayout(np)
```

Arguments

`np` number of plots that are to be produced on 1 figure

Value

A matrix.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

`par`.

Examples

```
autolayout(10)
```

 BLA

Best linear approximation with respect to the mean square error (theoretical linear regression).

Description

For a random vector $\mathbf{X} = (X_1, \dots, X_p)'$ for which a mean and a covariance matrix are given computes coefficients of the best linear approximations with respect to the mean square error of each component of \mathbf{X} given the remaining components of \mathbf{X} .

Usage

```
BLA(mean=c(0, 0), Sigma=diag(2))
```

Arguments

mean a numeric vector of means.
Sigma a covariance matrix.

Value

A list with the following components:

beta computed regression coefficients
sigmaR2 residual variances

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Anděl, J. (2007, odd. 2.5). Základy matematické statistiky. Praha: MATFYZPRESS.

Examples

```
##### X = (U(1), U(2), U(3))'
##### * U(1) <= U(2) <= U(3)
##### * ordered uniform Unif(0, 1) variates
EX <- (1:3)/4
varX <- matrix(c(3,2,1, 2,4,2, 1,2,3), ncol=3)/80
BLA(EX, Sigma=varX)

##### Uroda sena
##### * Y1 = uroda sena [cent/akr]
```

```
##### * Y2 = jarni srazky [palce]
##### * Y3 = kumulovana teplota nad 42 F
EY <- c(28.02, 4.91, 28.7)
varY <- matrix(c(19.54, 3.89, -3.76, 3.89, 1.21, -1.31, -3.76, -1.31, 4.52), ncol=3)
BLA(EY, Sigma=varY)

##### Z=(X, Y) ~ uniform distribution on a triangle
##### M = {(x,y): x>=0, y>=0, x+y<=3}
EZ <- c(1, 1)
varZ <- matrix(c(2, -1, -1, 2), nrow=2)/4
BLA(EZ, Sigma=varZ)

##### W=(X, Y) ~ uniform distribution on
##### M = {(x,y): x>=0, 0<=y<=1, y<=x<=y+1}
EW <- c(1, 1/2)
varW <- matrix(c(2, 1, 1, 1), nrow=2)/12
BLA(EW, Sigma=varW)
```

Dirichlet

Dirichlet distribution

Description

Random number generation for the Dirichlet distribution $D(\alpha_1, \dots, \alpha_K)$.

Usage

```
rDirichlet(n, alpha=c(1, 1))
```

Arguments

n number of observations to be sampled.
alpha parameters of the Dirichlet distribution ('prior sample sizes').

Value

Some objects.

Value for rDirichlet

A matrix with sampled values.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

- Devroye, L. (1986). *Non-Uniform Random Variate Generation*. New York: Springer-Verlag, Chap. XI.
- Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis. Second Edition*. Boca Raton: Chapman and Hall/CRC, pp. 576, 582.

See Also

[rbeta](#).

Examples

```
set.seed(1977)

alpha <- c(1, 2, 3)
Mean <- alpha/sum(alpha)
Var <- -(alpha %*% t(alpha))
diag(Var) <- diag(Var) + alpha*sum(alpha)
Var <- Var/(sum(alpha)^2*(1+sum(alpha)))
x <- rDirichlet(1000, alpha=alpha)
x[1:5,]

apply(x, 1, sum)[1:5]          ### should be all ones
rbind(Mean, apply(x, 2, mean))

var(x)
print(Var)
```

Enzyme

Enzymatic activity in the blood

Description

Enzymatic activity in the blood, for an enzyme involved in the metabolism of carcinogenic substances, among a group of 245 unrelated individuals.

Bechtel et al. (1993) identified a mixture of two skewed distributions by using maximum-likelihood estimation. Richardson and Green (1997) used a normal mixture estimated using reversible jump MCMC to estimate the distribution of the enzymatic activity.

Usage

```
data(Enzyme)
```

Format

A numeric vector with observed values.

Source

<http://www.stats.bris.ac.uk/~peter/mixdata>

References

Bechtel, Y. C., Bonaïti-Pellié, C., Poisson, N., Magnette, J., and Bechtel, P. R. (1993). A population and family study of N-acetyltransferase using caffeine urinary metabolites. *Clinical Pharmacology and Therapeutics*, **54**, 134–141.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with unknown number of components (with Discussion). *Journal of the Royal Statistical Society, Series B*, **59**, 731–792.

Examples

```
data(Enzyme)
summary(Enzyme)
```

Faithful

Old Faithful Geyser Data

Description

Waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA, version used in Härdle, W. (1991).

Usage

```
data(Faithful)
```

Format

A data frame with 272 observations on 2 variables.

eruptions eruption time in minutes

waiting waiting time to the next eruption in minutes

Details

There are many versions of this dataset around. Azzalini and Bowman (1990) use a more complete version.

Source

R package MASS

References

- Härdle, W. (1991). *Smoothing Techniques with Implementation in S*. New York: Springer.
- Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. *Applied Statistics*, **39**, 357-365.

See Also

[faithful](#), [geyser](#).

Examples

```
data(Faithful)
summary(Faithful)
```

Galaxy

Velocities of distant galaxies

Description

Velocities (in km/sec) of 82 distant galaxies, diverging from our own galaxy.

The dataset was first described by Roeder (1990) and subsequently analysed under different mixture models by several researchers including Escobar and West (1995) and Phillips and Smith (1996). Richardson and Green (1997) used a normal mixture estimated using reversible jump MCMC to estimate the distribution of the velocities.

REMARK: 78th observation is here 26.96 whereas it should be 26.69 (see [galaxies](#)). A value of 26.96 used in Richardson and Green (1997) is kept here.

Usage

```
data(Galaxy)
```

Format

A numeric vector with observed values.

Source

<http://www.stats.bris.ac.uk/~peter/mixdata>

References

- Escobar, M. D. and West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, **90**, 577–588.
- Phillips, D. B. and Smith, A. F. M. (1996). Bayesian model comparison via jump diffusions. In *Practical Markov Chain Monte Carlo*, eds: W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, ch. 13, pp. 215-239. London: Chapman and Hall.
- Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with unknown number of components (with Discussion). *Journal of the Royal Statistical Society, Series B*, **59**, 731–792.
- Roeder, K. (1990). Density estimation with confidence sets exemplified by superclusters and voids in the galaxies. *Journal of the American Statistical Association*, **85**, 617–624.

See Also

[galaxies](#).

Examples

```
data(Galaxy)
summary(Galaxy)
```

GLMM_longitClust *Clustering of longitudinal profiles based on fitted GLMM's*

Description

THIS FUNCTION IS BEING DEVELOPED AND ORDINARY USERS ARE NOT RECOMMENDED TO USE IT.

Usage

```
GLMM_longitClust(mod, w.prior, y, id, time, x, z, xz.common=TRUE, info)
```

Arguments

<code>mod</code>	a list containing models fitted with the GLMM_MCMC function. Each component of the list is the GLMM fitted in the training dataset of each cluster.
<code>w.prior</code>	a vector with prior cluster weights. The length of this argument must be the same as the length of argument <code>mod</code> . Can also be given relatively, e.g., as <code>c(1, 1)</code> which means that both prior weights are equal to 1/2.
<code>y</code>	vector, matrix or data frame (see argument <code>y</code> of GLMM_MCMC function) with responses of objects that are to be clustered.
<code>id</code>	vector which determines clustered observations (see also argument <code>y</code> of GLMM_MCMC function).
<code>time</code>	vector which gives indeces of observations within clusters. It appears (together with <code>id</code>) in the output as identifier of observations

<code>x</code>	see <code>xz.common</code> below.
<code>z</code>	see <code>xz.common</code> below.
<code>xz.common</code>	a logical value. If <code>TRUE</code> then it is assumed that the <code>X</code> and <code>Z</code> matrices are the same for GLMM in each cluster. In that case, arguments <code>x</code> and <code>z</code> have the same structure as arguments <code>x</code> and <code>z</code> of <code>GLMM_MCMC</code> function. If <code>FALSE</code> then <code>X</code> and <code>Z</code> matrices for the GLMM may differ across clusters. In that case, arguments <code>x</code> and <code>z</code> are both lists of length equal to the number of clusters and each component of lists <code>x</code> and <code>z</code> has the same structure as arguments <code>x</code> and <code>z</code> of <code>GLMM_MCMC</code> function.
<code>info</code>	interval in which the function prints the progress of computation

Details

I HOPE TO WRITE A PAPER DESCRIBING ALL THE DETAILS.

Value

A list with the following components:

<code>ident</code>	ADD DESCRIPTION
<code>marg</code>	ADD DESCRIPTION
<code>cond</code>	ADD DESCRIPTION
<code>ranef</code>	ADD DESCRIPTION

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. (20XX). PAPER TO BE WRITTEN. *JOURNAL*, **XX**, XXX-XXX.

See Also

[GLMM_MCMC](#).

Examples

WILL BE ADDED.

GLMM_MCMC

MCMC estimation of generalized linear mixed model with mixtures in the distributions.

Description

THIS FUNCTION IS BEING DEVELOPED AND ORDINARY USERS ARE NOT RECOMMENDED TO USE IT.

This function runs MCMC for a generalized linear mixed model with possibly several response variables and possibly normal mixtures in the distributions of random effects.

Usage

```
GLMM_MCMC(y, dist="gaussian", id, x, z, random.intercept,
          prior.beta, init.beta,
          scale.b, prior.b, init.b,
          prior.eps, init.eps,
          nMCMC=c(burn=10, keep=10, thin=1, info=10),
          store=c(b=FALSE), keep.chains=TRUE)
```

```
## S3 method for class 'GLMM_MCMC':
print(x, ...)
```

Arguments

<code>y</code>	vector, matrix or data frame with responses. If <code>y</code> is vector then there is only one response in the model. If <code>y</code> is matrix or data frame then each column gives values of one response. Missing values are allowed. If there are several responses specified then continuous responses must be put in the first columns and discrete responses in the subsequent columns.
<code>dist</code>	character (vector) which determines distribution (and a link function) for each response variable. Possible values are: “gaussian” for gaussian (normal) distribution (with identity link), “binomial(logit)” for binomial (0/1) distribution with a logit link. Single value is recycled if necessary.
<code>id</code>	vector which determines clustered observations. If not given then it is assumed that there are no clusters and all observations of one response are independent.
<code>x</code>	matrix or a list of matrices with covariates (intercept not included) for fixed effects. If there is more than one response, this must always be a list. Note that intercept is included in all models. Use a character value “empty” as a component of the list <code>x</code> if there are no covariates for a particular response.
<code>z</code>	matrix or a list of matrices with covariates (intercept not included) for random effects. If there is more than one response, this must always be a list. Note that random intercept is specified using the argument <code>random.intercept</code> .
<code>random.intercept</code>	logical (vector) which determines for which responses random intercept should be included.

<code>prior.beta</code>	<p>a list which specifies prior distribution for fixed effects (not the means of random effects). The prior distribution is normal and the user can specify the mean and variances. The list <code>prior.b</code> can have the components listed below.</p> <p>mean a vector with prior means, defaults to zeros.</p> <p>var a vector with prior variances, defaults to 10000 for all components.</p>
<code>init.beta</code>	<p>a numeric vector with initial values of fixed effects (not the means of random effects). A sensible value is determined using the maximum-likelihood fits (using <code>lmer</code> functions) and does not have to be given by the user.</p>
<code>scale.b</code>	<p>a list specifying how to scale the random effects during the MCMC. A sensible value is determined using the maximum-likelihood fits (using <code>lmer</code> functions) and does not have to be given by the user.</p> <p>If the user wishes to influence the shift and scale constants, these are given as components of the list <code>scale.b</code>. The components are named:</p> <p>shift ADD DESCRIPTION</p> <p>scale ADD DESCRIPTION</p>
<code>prior.b</code>	<p>a list which specifies prior distribution for (shifted and scaled) random effects. The prior is in principle a normal mixture (being a simple normal distribution if we restrict the number of mixture components to be equal to one).</p> <p>The list <code>prior.b</code> can have the components listed below. Their meaning is analogous to the components of the same name of the argument <code>prior</code> of function <code>NMixMCMC</code> (see therein for details).</p> <p>priorK a character string which specifies the type of the prior for K (the number of mixture components).</p> <p>priormuQ a character string which specifies the type of the prior for mixture means and mixture variances.</p> <p>Kmax maximal number of mixture components.</p> <p>lambda ADD DESCRIPTION</p> <p>delta ADD DESCRIPTION</p> <p>xi ADD DESCRIPTION</p> <p>ce ADD DESCRIPTION</p> <p>D ADD DESCRIPTION</p> <p>zeta ADD DESCRIPTION</p> <p>g ADD DESCRIPTION</p> <p>h ADD DESCRIPTION</p>
<code>init.b</code>	<p>a list with initial values for parameters related to the distribution of random effects and random effects themselves. Sensible initial values are determined by the function itself and do not have to be given by the user.</p>
<code>prior.eps</code>	<p>a list specifying prior distributions for error terms for continuous responses. The list <code>prior.eps</code> can have the components listed below. For all components, a sensible value leading to weakly informative prior distribution can be determined by the function.</p> <p>zeta ADD DESCRIPTION</p> <p>g ADD DESCRIPTION</p>

	h ADD DESCRIPTION
<code>init.eps</code>	<p>a list with initial values for parameters related to the distribution of error terms of continuous responses. The list <code>init.eps</code> can have the components listed below. For all components, a sensible value can be determined by the function.</p> <p>sigma a numeric vector with the initial values for residual standard deviations for each continuous response.</p> <p>gammaInv a numeric vector with the initial values for the inverted components of the hyperparameter gamma for each continuous response.</p>
<code>nMCMC</code>	<p>numeric vector of length 4 giving parameters of the MCMC simulation. Its components may be named (ordering is then unimportant) as:</p> <p>burn length of the burn-in (after discarding the thinned values), can be equal to zero as well.</p> <p>keep length of the kept chains (after discarding the thinned values), must be positive.</p> <p>thin thinning interval, must be positive.</p> <p>info interval in which the progress information is printed on the screen.</p> <p>In total $(M_{burn} + M_{keep}) \cdot M_{thin}$ MCMC scans are performed.</p>
<code>store</code>	<p>logical vector indicating whether the chains of parameters should be stored. Its components may be named (ordering is then unimportant) as:</p> <p>b if TRUE then the sampled values of random effects are stored. Defaults to FALSE.</p>
<code>keep.chains</code>	<p>logical. If FALSE, only summary statistics are returned in the resulting object. This might be useful in the model searching step to save some memory.</p>
<code>...</code>	<p>additional arguments passed to the default <code>print</code> method.</p>

Details

I HOPE TO WRITE A PAPER DESCRIBING ALL THE DETAILS.

Value

An object of class `GLMM_MCMC`. It can have the following components (some of them may be missing according to the context of the model):

<code>iter</code>	index of the last iteration performed.
<code>nMCMC</code>	used value of the argument <code>nMCMC</code> .
<code>R</code>	a two component vector giving the number of continuous responses and the number of discrete responses.
<code>dist</code>	a character vector of length <code>R</code> corresponding to the <code>dist</code> argument.
<code>p</code>	a numeric vector of length <code>R</code> giving the number of non-intercept beta parameters for each response.
<code>q</code>	a numeric vector of length <code>R</code> giving the number of non-intercept random effects for each response.

<code>fixed.intercept</code>	a logical vector of length R which indicates inclusion of fixed intercept for each response.
<code>random.intercept</code>	a logical vector of length R which indicates inclusion of random intercept for each response.
<code>lbeta</code>	length of the vector of fixed effects.
<code>dimb</code>	dimension of the distribution of random effects.
<code>prior.beta</code>	a list containing the used value of the argument <code>prior.beta</code> .
<code>prior.b</code>	a list containing the used value of the argument <code>prior.b</code> .
<code>prior.eps</code>	a list containing the used value of the argument <code>prior.eps</code> .
<code>init.beta</code>	a numeric vector with the used value of the argument <code>init.beta</code> .
<code>init.b</code>	a list containing the used value of the argument <code>init.b</code> .
<code>init.eps</code>	a list containing the used value of the argument <code>init.eps</code> .
<code>state.beta</code>	a numeric vector with the last sampled value of fixed effects β . It can be used as argument <code>init.beta</code> to restart MCMC.
<code>state.b</code>	a list with the last sampled values of parameters related to the distribution of random effects. It has components named <code>b</code> , <code>K</code> , <code>w</code> , <code>mu</code> , <code>Sigma</code> , <code>Li</code> , <code>Q</code> , <code>gammaInv</code> , <code>r</code> . It can be used as argument <code>init.b</code> to restart MCMC.
<code>state.eps</code>	a list with the last sampled values of parameters related to the distribution of residuals of continuous responses. It has components named <code>sigma</code> , <code>gammaInv</code> . It can be used as argument <code>init.eps</code> to restart MCMC.
<code>scale.b</code>	a list containing the used value of the argument <code>scale.b</code> .
<code>poster.mean.eta</code>	a <code>data.frame</code> with columns labeled <code>fixed</code> and <code>random</code> holding posterior means for fixed effect part of the linear predictor and the random effect part of the linear predictor. In each column, there are first all values for the first response, then all values for the second response etc.
<code>poster.mean.cluster</code>	a <code>data.frame</code> with columns labeled <code>b1</code> , ..., <code>bq</code> , <code>LogL</code> , <code>Logpb</code> with posterior means of random effects for each cluster and posterior means of $\log(L)$ (log-likelihood given random effects) and $\log\{p(\mathbf{b})\}$ for each cluster.
<code>poster.mean.w_b</code>	ADD DESCRIPTION
<code>poster.mean.mu_b</code>	ADD DESCRIPTION
<code>poster.mean.Q_b</code>	ADD DESCRIPTION
<code>poster.mean.Sigma_b</code>	ADD DESCRIPTION
<code>poster.mean.Li_b</code>	ADD DESCRIPTION
<code>summ.beta</code>	ADD DESCRIPTION
<code>summ.b.Mean</code>	ADD DESCRIPTION

```

summ.b.SDCorr          ADD DESCRIPTION
summ.sigma_eps        ADD DESCRIPTION
freqK_b               ADD DESCRIPTION
propK_b               ADD DESCRIPTION
K_b                   ADD DESCRIPTION
w_b                   ADD DESCRIPTION
mu_b                   ADD DESCRIPTION
Li_b                   ADD DESCRIPTION
Q_b                   ADD DESCRIPTION
Sigma_b               ADD DESCRIPTION
gammaInv_b            ADD DESCRIPTION
order_b               ADD DESCRIPTION
rank_b                ADD DESCRIPTION
mixture_b             ADD DESCRIPTION
b                     ADD DESCRIPTION
beta                  ADD DESCRIPTION
sigma_eps             ADD DESCRIPTION
gammaInv_eps          ADD DESCRIPTION

```

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. (20XX). PAPER TO BE WRITTEN. *JOURNAL*, **XX**, XXX-XXX.

See Also

[NMixMCMC](#).

Examples

WILL BE ADDED.

MatMPpinv

Moore-Penrose pseudoinverse of a squared matrix

Description

For a matrix A its Moore-Penrose pseudoinverse is such a matrix A^+ which satisfies

- (i) $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$,
- (ii) $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$,
- (iii) $(\mathbf{A}\mathbf{A}^+)' = \mathbf{A}\mathbf{A}^+$,
- (iv) $(\mathbf{A}^+\mathbf{A}) = \mathbf{A}^+\mathbf{A}$.

Computation is done using spectral decomposition. At this moment, it is implemented for symmetric matrices only.

Usage

```
MatMPpinv(A)
```

Arguments

A either a numeric vector in which case inverse of each element of A is returned or a squared matrix.

Value

Either a numeric vector or a matrix.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Golub, G. H. and Van Loan, C. F. (1996, Sec. 5.5). *Matrix Computations. Third Edition*. Baltimore: The Johns Hopkins University Press.

Examples

```
set.seed(770328)
A <- rWishart(1, 5, diag(4))
Ainv <- MatMPpinv(A)

### Check the conditions
prec <- 13
round(A - A %*% Ainv %*% A, prec)
round(Ainv - Ainv %*% A %*% Ainv, prec)
round(A %*% Ainv - t(A %*% Ainv), prec)
round(Ainv %*% A - t(Ainv %*% A), prec)
```

`MatSqrt`*Square root of a matrix*

Description

For a matrix A its square root is such a matrix B which satisfies $A = BB$.

Computation is done using spectral decomposition. When calculating the square roots of eigenvalues, always a root with positive real part and a sign of the imaginary part the same as the sign of the imaginary eigenvalue part is taken.

Usage

```
MatSqrt(A)
```

Arguments

`A` either a numeric vector in which case square roots of each element of `A` is returned or a squared matrix.

Value

Either a numeric vector or a matrix.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

Examples

```
MatSqrt(0:4)
MatSqrt((-4):0)
MatSqrt(c(-1, 1, -2, 2))

A <- (1:4) %*% t(1:4)
sqrtA <- MatSqrt(A)
sqrtA
round(sqrtA %*% sqrtA - A, 13)

B <- -A
sqrtB <- MatSqrt(B)
sqrtB
round(Re(sqrtB %*% sqrtB - B), 13)
round(Im(sqrtB %*% sqrtB - B), 13)

V <- eigen(A)$vectors
sqrtV <- MatSqrt(V)
sqrtV
round(sqrtV %*% sqrtV - V, 14)
```

```

Sigma <- matrix(c(1, 1, 1.5, 1, 4, 4.2, 1.5, 4.2, 9), nrow=3)
sqrtSigma <- MatSqrt(Sigma)
sqrtSigma
round(sqrtSigma %**% sqrtSigma - Sigma, 13)

D4 <- matrix(c(5, -4, 1, 0, 0,
              -4, 6, -4, 1, 0,
              1, -4, 6, -4, 1,
              0, 1, -4, 6, -4,
              0, 0, 1, -4, 5), nrow=5)
sqrtD4 <- MatSqrt(D4)
sqrtD4[abs(sqrtD4) < 1e-15] <- 0
sqrtD4
round(sqrtD4 %**% sqrtD4 - D4, 14)

X <- matrix(c(7, 15, 10, 22), nrow=2)
sqrtX <- MatSqrt(X)
sqrtX %**% sqrtX - X

```

 MVN

Multivariate normal distribution

Description

Density and random generation for the multivariate normal distribution with mean equal to mean, precision matrix equal to Q (or covariance matrix equal to Sigma).

Function `rcMVN` samples from the multivariate normal distribution with a canonical mean b , i.e., the mean is $\mu = Q^{-1} b$.

Usage

```
dMVN(x, mean=0, Q=1, Sigma, log=FALSE)
```

```
rMVN(n, mean=0, Q=1, Sigma)
```

```
rcMVN(n, b=0, Q=1, Sigma)
```

Arguments

mean	vector of mean.
b	vector of a canonical mean.
Q	precision matrix of the multivariate normal distribution. Ignored if Sigma is given.
Sigma	covariance matrix of the multivariate normal distribution. If Sigma is supplied, precision is computed from Σ as $Q = \Sigma^{-1}$.
n	number of observations to be sampled.
x	vector or matrix of the points where the density should be evaluated.
log	logical; if TRUE, log-density is computed

Value

Some objects.

Value for dMVN

A vector with evaluated values of the (log-)density

Value for rMVN

A list with the components:

x vector or matrix with sampled values

log.dens vector with the values of the log-density evaluated in the sampled values

Value for rcMVN

A list with the components:

x vector or matrix with sampled values

mean vector or the mean of the normal distribution

log.dens vector with the values of the log-density evaluated in the sampled values

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*. Boca Raton: Chapman and Hall/CRC.

See Also

[dnorm](#), [Mvnorm](#).

Examples

```
set.seed(1977)

### Univariate normal distribution
### =====
c(dMVN(0), dnorm(0))
c(dMVN(0, log=TRUE), dnorm(0, log=TRUE))

rbind(dMVN(c(-1, 0, 1)), dnorm(c(-1, 0, 1)))
rbind(dMVN(c(-1, 0, 1), log=TRUE), dnorm(c(-1, 0, 1), log=TRUE))

c(dMVN(1, mean=1.2, Q=0.5), dnorm(1, mean=1.2, sd=sqrt(2)))
c(dMVN(1, mean=1.2, Q=0.5, log=TRUE), dnorm(1, mean=1.2, sd=sqrt(2), log=TRUE))
```

```

rbind(dMVN(0:2, mean=1.2, Q=0.5), dnorm(0:2, mean=1.2, sd=sqrt(2)))
rbind(dMVN(0:2, mean=1.2, Q=0.5, log=TRUE), dnorm(0:2, mean=1.2, sd=sqrt(2), log=TRUE))

### Multivariate normal distribution
### =====
mu <- c(0, 6, 8)
L <- matrix(1:9, nrow=3)
L[upper.tri(L, diag=FALSE)] <- 0
Sigma <- L %*% t(L)
Q <- chol2inv(chol(Sigma))
b <- solve(Sigma, mu)

dMVN(mu, mean=mu, Q=Q)
dMVN(mu, mean=mu, Sigma=Sigma)
dMVN(mu, mean=mu, Q=Q, log=TRUE)
dMVN(mu, mean=mu, Sigma=Sigma, log=TRUE)

xx <- matrix(c(0,6,8, 1,5,7, -0.5,5.5,8.5, 0.5,6.5,7.5), ncol=3, byrow=TRUE)
dMVN(xx, mean=mu, Q=Q)
dMVN(xx, mean=mu, Sigma=Sigma)
dMVN(xx, mean=mu, Q=Q, log=TRUE)
dMVN(xx, mean=mu, Sigma=Sigma, log=TRUE)

zz <- rMVN(1000, mean=mu, Sigma=Sigma)
rbind(apply(zz$x, 2, mean), mu)
var(zz$x)
Sigma
cbind(dMVN(zz$x, mean=mu, Sigma=Sigma, log=TRUE), zz$log.dens)[1:10,]

zz <- rcMVN(1000, b=b, Sigma=Sigma)
rbind(apply(zz$x, 2, mean), mu)
var(zz$x)
Sigma
cbind(dMVN(zz$x, mean=mu, Sigma=Sigma, log=TRUE), zz$log.dens)[1:10,]

zz <- rMVN(1000, mean=rep(0, 3), Sigma=Sigma)
rbind(apply(zz$x, 2, mean), rep(0, 3))
var(zz$x)
Sigma
cbind(dMVN(zz$x, mean=rep(0, 3), Sigma=Sigma, log=TRUE), zz$log.dens)[1:10,]

### The same using the package mvtnorm
### =====
# require(mvtnorm)
# c(dMVN(mu, mean=mu, Sigma=Sigma), dmvnorm(mu, mean=mu, sigma=Sigma))
# c(dMVN(mu, mean=mu, Sigma=Sigma, log=TRUE), dmvnorm(mu, mean=mu, sigma=Sigma, log=TRUE))
#
# rbind(dMVN(xx, mean=mu, Sigma=Sigma), dmvnorm(xx, mean=mu, sigma=Sigma))
# rbind(dMVN(xx, mean=mu, Sigma=Sigma, log=TRUE), dmvnorm(xx, mean=mu, sigma=Sigma, log=TRUE))

```

Description

Density and random generation for the mixture of the p -variate normal distributions with means given by `mean`, precision matrix given by `Q` (or covariance matrices given by `Sigma`).

Usage

```
dMVNmixture(x, weight, mean, Q, Sigma, log=FALSE)
```

```
dMVNmixture2(x, weight, mean, Q, Sigma, log=FALSE)
```

```
rMVNmixture(n, weight, mean, Q, Sigma)
```

```
rMVNmixture2(n, weight, mean, Q, Sigma)
```

Arguments

<code>weight</code>	vector of length K with the mixture weights or values which are proportional to the weights.
<code>mean</code>	vector or matrix of mixture means. For $p = 1$ this should be a vector of length K , for $p > 1$ this should be a $K \times p$ matrix with mixture means in rows.
<code>Q</code>	precision matrices of the multivariate normal distribution. Ignored if <code>Sigma</code> is given. For $p = 1$ this should be a vector of length K , for $p > 1$ this should be a list of length K with the mixture precision matrices as components of the list.
<code>Sigma</code>	covariance matrix of the multivariate normal distribution. If <code>Sigma</code> is supplied, precisions are computed from Σ as $Q = \Sigma^{-1}$. For $p = 1$ this should be a vector of length K , for $p > 1$ this should be a list of length K with the mixture covariance matrices as components of the list.
<code>n</code>	number of observations to be sampled.
<code>x</code>	vector or matrix of the points where the density should be evaluated.
<code>log</code>	logical; if TRUE, log-density is computed

Details

Functions `dMVNmixture` and `dMVNmixture2` differ only internally in the way they compute the mixture density. In `dMVNmixture`, only multivariate normal densities are evaluated in compiled C++ code and mixing is done directly in R. In `dMVNmixture2`, everything is evaluated in compiled C++ code. Normally, both `dMVNmixture` and `dMVNmixture2` should return the same results.

Similarly for `rMVNmixture` and `rMVNmixture2`. Another difference is that `rMVNmixture` returns only random generated points and `rMVNmixture2` also values of the density evaluated in the generated points.

Value

Some objects.

Value for dMVNmixture

A vector with evaluated values of the (log-)density.

Value for dMVNmixture2

A vector with evaluated values of the (log-)density.

Value for rMVNmixture

A vector (for $n=1$ or for univariate mixture) or matrix with sampled values (in rows of the matrix).

Value for rMVNmixture2

A list with components named `x` which is a vector (for $n=1$ or for univariate mixture) or matrix with sampled values (in rows of the matrix) and `dens` which are the values of the density evaluated in `x`.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[dnorm](#), [MVN](#), [Mvnorm](#).

Examples

```
set.seed(1977)

##### Univariate normal mixture
##### =====
mu <- c(-1, 1)
Sigma <- c(0.25^2, 0.4^2)
Q <- 1/Sigma
w <- c(0.3, 0.7)

xx <- seq(-2, 2.5, length=100)
yyA <- dMVNmixture(xx, weight=w, mean=mu, Sigma=Sigma)
yyB <- dMVNmixture(xx, weight=w, mean=mu, Q=Q)
yyC <- dMVNmixture2(xx, weight=w, mean=mu, Sigma=Sigma)
yyD <- dMVNmixture2(xx, weight=w, mean=mu, Q=Q)

xxSample <- rMVNmixture(1000, weight=w, mean=mu, Sigma=Sigma)
xxSample2 <- rMVNmixture2(1000, weight=w, mean=mu, Sigma=Sigma)

sum(abs(xxSample2$dens - dMVNmixture(xxSample2$x, weight=w, mean=mu, Sigma=Sigma)) > 1e-15)
xxSample2 <- xxSample2$x

par(mfrow=c(2, 2), bty="n")
plot(xx, yyA, type="l", col="red", xlab="x", ylab="f(x)")
points(xx, yyB, col="darkblue")
```

```

hist(xxSample, col="lightblue", prob=TRUE, xlab="x", xlim=range(xx), ylim=c(0, max(yyA)), ma
lines(xx, yyA, col="red")
plot(xx, yyC, type="l", col="red", xlab="x", ylab="f(x)")
points(xx, yyD, col="darkblue")
hist(xxSample2, col="sandybrown", prob=TRUE, xlab="x", xlim=range(xx), ylim=c(0, max(yyA)),
lines(xx, yyC, col="red")

##### Bivariate normal mixture
##### =====
### Choice 1
sd11 <- sd12 <- 1.1
sd21 <- 0.5
sd22 <- 1.5
rho2 <- 0.7
Xlim <- c(-3, 4)
Ylim <- c(-6, 4)

### Choice 2
sd11 <- sd12 <- 0.3
sd21 <- 0.5
sd22 <- 0.3
rho2 <- 0.8
Xlim <- c(-3, 2.5)
Ylim <- c(-2.5, 2.5)

mu <- matrix(c(1,1, -1,-1), nrow=2, byrow=TRUE)
Sigma <- list(diag(c(sd11^2, sd12^2)), matrix(c(sd21^2, rho2*sd21*sd22, rho2*sd21*sd22, sd22
Q <- list(chol2inv(chol(Sigma[[1]])), chol2inv(chol(Sigma[[2]])))
w <- c(0.3, 0.7)

xx1 <- seq(mu[2,1]-3*sd21, mu[1,1]+3*sd11, length=100)
xx2 <- seq(mu[2,2]-3*sd22, mu[1,2]+3*sd12, length=90)
XX <- cbind(rep(xx1, length(xx2)), rep(xx2, each=length(xx1)))
yyA <- matrix(dMVNmixture(XX, weight=w, mean=mu, Sigma=Sigma), nrow=length(xx1), ncol=length
yyB <- matrix(dMVNmixture(XX, weight=w, mean=mu, Q=Q), nrow=length(xx1), ncol=length(xx2))
yyC <- matrix(dMVNmixture2(XX, weight=w, mean=mu, Sigma=Sigma), nrow=length(xx1), ncol=length
yyD <- matrix(dMVNmixture2(XX, weight=w, mean=mu, Q=Q), nrow=length(xx1), ncol=length(xx2))

xxSample <- rMVNmixture(1000, weight=w, mean=mu, Sigma=Sigma)
xxSample2 <- rMVNmixture2(1000, weight=w, mean=mu, Sigma=Sigma)

sum(abs(xxSample2$dens - dMVNmixture(xxSample2$x, weight=w, mean=mu, Sigma=Sigma)) > 1e-15)
xxSample2 <- xxSample2$x

par(mfrow=c(1, 2), bty="n")
plot(xxSample, col="darkblue", xlab="x1", ylab="x2", xlim=Xlim, ylim=Ylim)
contour(xx1, xx2, yyA, col="red", add=TRUE)
plot(xxSample2, col="darkblue", xlab="x1", ylab="x2", xlim=Xlim, ylim=Ylim)
contour(xx1, xx2, yyC, col="red", add=TRUE)

par(mfrow=c(2, 2), bty="n")
contour(xx1, xx2, yyA, col="red", xlab="x1", ylab="x2")
points(mu[,1], mu[,2], col="darkgreen")

```

```

persp(xx1, xx2, yyA, col="lightblue", xlab="x1", ylab="x2", zlab="f(x1, x2)")
contour(xx1, xx2, yyB, col="red", xlab="x1", ylab="x2")
points(mu[,1], mu[,2], col="darkgreen")
persp(xx1, xx2, yyB, col="lightblue", xlab="x1", ylab="x2", zlab="f(x1, x2)", phi=30, theta=)

par(mfrow=c(2, 2), bty="n")
contour(xx1, xx2, yyC, col="red", xlab="x1", ylab="x2")
points(mu[,1], mu[,2], col="darkgreen")
persp(xx1, xx2, yyC, col="lightblue", xlab="x1", ylab="x2", zlab="f(x1, x2)")
contour(xx1, xx2, yyD, col="red", xlab="x1", ylab="x2")
points(mu[,1], mu[,2], col="darkgreen")
persp(xx1, xx2, yyD, col="lightblue", xlab="x1", ylab="x2", zlab="f(x1, x2)", phi=30, theta=)

```

NMixChainsDerived *Create MCMC chains derived from previously sampled values*

Description

Currently, this function creates chains for marginal means of `exp(data)` from previously sampled values (see [NMixMCMC](#)). This is useful in survival context when a density of $Y = \log(T)$ is modelled using the function [NMixMCMC](#) and we are interested in inference on $ET = E \exp(Y)$.

Usage

```
NMixChainsDerived(object)
```

Arguments

`object` an object of class `NMixMCMC` or `NMixMCMClist`

Value

An object of the same class as argument `object`. When `object` was of class `NMixMCMC`, the resulting object contains additionally the following components:

```

chains.derived
  a data.frame with columns labeled expy.Mean.1, ..., expy.Mean.p
  containing the sampled values of  $E \exp(Y_1), \dots, E \exp(Y_p)$ .
summ.expy.Mean
  posterior summary statistics for  $E \exp(Y_1), \dots, E \exp(Y_p)$ .

```

When `object` was of the class `NMixMCMClist` then each of its components (`chains`) is augmented by new components `chains.derived` and `summ.expy.Mean`.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixMCMC](#).

`NMixClust`*Clustering based on a mixture model*

Description

It performs clustering based on posterior summary for (re-labeled) mixture components in a model with fixed number of components fitted with [NMixMCMC](#) function.

Usage

```
NMixClust(object, y)
```

Arguments

<code>object</code>	an object of class <code>NMixMCMC</code>
<code>y</code>	vector, matrix or data frame with observations to be clustered

Value

A `data.frame` with observations in the first columns and additionally columns labeled `prob1,...`, `probp` giving posterior probabilities of belonging to each component and a column labeled `component` giving the index of the component with the highest posterior probability.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixMCMC](#).

`NMixMCMC`*MCMC estimation of (multivariate) normal mixtures with possibly censored data.*

Description

This function runs MCMC for a model in which unknown density is specified as a normal mixture with either known or unknown number of components. With a prespecified number of components, MCMC is implemented through Gibbs sampling (see Diebolt and Robert, 1994) and dimension of the data can be arbitrary. With unknown number of components, currently only univariate case is implemented using the reversible jump MCMC (Richardson and Green, 1997). In a future, there are plans to implement the reversible jump MCMC in a higher dimension using the algorithm presented by Papageorgiou and Dellaportas (2006).

Further, the data are allowed to be censored in which case additional Gibbs step is used within the MCMC algorithm

Usage

```

NMixMCMC(y0, y1, censor, scale, prior,
         init, init2, RJMCMC,
         nMCMC=c(burn=10, keep=10, thin=1, info=10),
         PED, keep.chains=TRUE, onlyInit=FALSE, dens.zero=1e-300)

## S3 method for class 'NMixMCMC':
print(x, dic, ...)

## S3 method for class 'NMixMCMClist':
print(x, ped, dic, ...)

```

Arguments

y0	numeric vector of length n or $n \times p$ matrix with observed data. It contains exactly observed, right-censored, left-censored data and lower limits for interval-censored data.
y1	numeric vector of length n or $n \times p$ matrix with upper limits for interval-censored data. Elements corresponding to exactly observed, right-censored or left-censored data are ignored and can be filled arbitrarily (by NA's) as well. It does not have to be supplied if there are no interval-censored data.
censor	numeric vector of length n or $n \times p$ matrix with censoring indicators. The following values indicate: <ul style="list-style-type: none"> 0 right-censored observation, 1 exactly observed value, 2 left-censored observation, 3 interval-censored observation. <p>If it is not supplied then it is assumed that all values are exactly observed.</p>
scale	a list specifying how to scale the data before running MCMC. It should have two components: <ul style="list-style-type: none"> shift a vector of length 1 or p specifying shift vector m, scale a vector of length 1 or p specifying diagonal of the scaling matrix S. <p>If there is no censoring, and argument <code>scale</code> is missing then the data are scaled to have zero mean and unit variances, i.e., <code>scale(y0)</code> is used for MCMC. In the case there is censoring and <code>scale</code> is missing, <code>scale\$shift</code> is taken to be a sample mean of <code>init\$y</code> and <code>scale\$scale</code> are sample standard deviations of columns of <code>init\$y</code>.</p> <p>If you do not wish to scale the data before running MCMC, specify <code>scale=list(shift=0, scale=1)</code>.</p>
prior	a list with the parameters of the prior distribution. It should have the following components (for some of them, the program can assign default values and the user does not have to specify them if he/she wishes to use the defaults): <ul style="list-style-type: none"> priorK a character string which specifies the type of the prior for K (the number of mixture components). It should have one of the following values:

“fixed”

Number of mixture components is assumed to be fixed to K_{max} . This is a **default** value.

“uniform”

A priori $K \sim \text{Unif}\{1, \dots, K_{max}\}$.

“tpoisson”

A priori $K \sim \text{truncated-Pois}(\lambda, K_{max})$.

priormuQ a character string which specifies the type of the prior for $\mu_1, \dots, \mu_{K_{max}}$ (mixture means) and $\mathbf{Q}_1, \dots, \mathbf{Q}_{K_{max}}$ (inverted mixture covariance matrices). It should have one of the following values:

“independentC”

\equiv independent conjugate prior (this is a **default** value). That is, a priori

$$(\mu_j, \mathbf{Q}_j) \sim \text{N}(\xi_j, \mathbf{D}_j) \times \text{Wishart}(\zeta, \Xi)$$

independently for $j = 1, \dots, K$, where normal means ξ_1, \dots, ξ_K , normal variances $\mathbf{D}_1, \dots, \mathbf{D}_K$, and Wishart degrees of freedom ζ are specified further as xi, D, zeta components of the list `prior`.

“naturalC”

\equiv natural conjugate prior. That is, a priori

$$(\mu_j, \mathbf{Q}_j) \sim \text{N}(\xi_j, c_j^{-1} \mathbf{Q}_j^{-1}) \times \text{Wishart}(\zeta, \Xi)$$

independently for $j = 1, \dots, K$, where normal means ξ_1, \dots, ξ_K , precisions c_1, \dots, c_K , and Wishart degrees of freedom ζ are specified further as xi, ce, zeta components of the list `prior`.

For both, independent conjugate and natural conjugate prior, the Wishart scale matrix Ξ is assumed to be diagonal with $\gamma_1, \dots, \gamma_p$ on a diagonal. For γ_j^{-1} ($j = 1, \dots, K$) additional gamma hyperprior $G(g_j, h_j)$ is assumed. Values of g_1, \dots, g_p and h_1, \dots, h_p are further specified as g and h components of the `prior` list.

Kmax maximal number of mixture components K_{max} . It must **always be specified** by the user.

lambda parameter λ for the truncated Poisson prior on K . It must be positive and must **always be specified** if `priorK` is “tpoisson”.

delta parameter δ for the Dirichlet prior on the mixture weights w_1, \dots, w_K . It must be positive. Its **default** value is 1.

xi a numeric value, vector or matrix which specifies $\xi_1, \dots, \xi_{K_{max}}$ (prior means for the mixture means $\mu_1, \dots, \mu_{K_{max}}$). **Default** value is a matrix $K_{max} \times p$ with midpoints of columns of `init$y` in rows which follows Richardson and Green (1997).

If $p = 1$ and `xi` = ξ is a single value then $\xi_1 = \dots = \xi_{K_{max}} = \xi$.

If $p = 1$ and $\mathbf{x}i = \boldsymbol{\xi}$ is a vector of length K_{max} then the j -th element of $\mathbf{x}i$ gives ξ_j ($j = 1, \dots, K_{max}$).

If $p > 1$ and $\mathbf{x}i = \boldsymbol{\xi}$ is a vector of length p then $\boldsymbol{\xi}_1 = \dots = \boldsymbol{\xi}_{K_{max}} = \boldsymbol{\xi}$.

If $p > 1$ and $\mathbf{x}i$ is a $K_{max} \times p$ matrix then the j -th row of $\mathbf{x}i$ gives $\mathbf{x}i_j$ ($j = 1, \dots, K_{max}$).

ce a numeric value or vector which specifies prior precision parameters $c_1, \dots, c_{K_{max}}$ for the mixture means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{K_{max}}$ when `priormuQ` is “naturalC”. Its **default** value is a vector of ones which follows Cappe, Robert and Ryden (2003).

If `ce = c` is a single value then $c_1 = \dots = c_{K_{max}} = c$.

If `ce = c` is a vector of length K_{max} then the j -th element of `ce` gives c_j ($j = 1, \dots, K_{max}$).

D a numeric vector or matrix which specifies $D_1, \dots, D_{K_{max}}$ (prior variances or covariance matrices of the mixture means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{K_{max}}$ when `priormuQ` is “independentC”). Its **default** value is a diagonal matrix with squared ranges of each column of `init$y` on a diagonal.

If $p = 1$ and `D = d` is a single value then $d_1 = \dots = d_{K_{max}} = d$.

If $p = 1$ and `D = d` is a vector of length K_{max} then the j -th element of `D` gives d_j ($j = 1, \dots, K_{max}$).

If $p > 1$ and `D = D` is a $p \times p$ matrix then $D_1 = \dots = D_{K_{max}} = D$.

If $p > 1$ and `D` is a $(K_{max} \cdot p) \times p$ matrix then the first p rows of `D` give D_1 , rows $p + 1, \dots, 2p$ of `D` give D_2 etc.

zeta degrees of freedom ζ for the Wishart prior on the inverted mixture variances $Q_1, \dots, Q_{K_{max}}$. It must be higher than $p - 1$. Its **default** value is $p + 1$.

g a value or a vector of length p with the shape parameters g_1, \dots, g_p for the Gamma hyperpriors on $\gamma_1, \dots, \gamma_p$. It must be positive. Its **default** value is a vector $(0.2, \dots, 0.2)'$.

h a value or a vector of length p with the rate parameters h_1, \dots, h_p for the Gamma hyperpriors on $\gamma_1, \dots, \gamma_p$. It must be positive. Its **default** value is a vector containing $10/R_l^2$, where R_l is a range of the l -th column of `init$y`.

`init`

a list with the initial values for the MCMC. All initials can be determined by the program if they are not specified. The list may have the following components:

y a numeric vector or matrix with the initial values for the latent censored observations.

K a numeric value with the initial value for the number of mixture components.

w a numeric vector with the initial values for the mixture weights.

	mu	a numeric vector or matrix with the initial values for the mixture means.
	Sigma	a numeric vector or matrix with the initial values for the mixture variances.
	Li	a numeric vector with the initial values for the Colesky decomposition of the mixture inverse variances.
	gammaInv	a numeric vector with the initial values for the inverted components of the hyperparameter γ .
	r	a numeric vector with the initial values for the mixture allocations.
init2		a list with the initial values for the second chain needed to estimate the penalized expected deviance of Plummer (2008). The list <code>init2</code> has the same structure as the list <code>init</code> . All initials in <code>init2</code> can be determined by the program (differently than the values in <code>init</code>) if they are not specified.
		Ignored when <code>PED</code> is <code>FALSE</code> .
RJMCMC		a list with the parameters needed to run reversible jump MCMC for mixtures with varying number of components. It does not have to be specified if the number of components is fixed. Most of the parameters can be determined by the program if they are not specified. The list may have the following components: <ul style="list-style-type: none"> Paction probabilities (or proportional constants) which are used to choose an action of the sampler within each iteration of MCMC to update the mixture related parameters. Let <code>Paction</code> = $(p_1, p_2, p_3)'$. Then with probability p_1 only steps assuming fixed k (number of mixture components) are performed, with probability p_2 split-combine move is proposed and with probability p_3 birth-death move is proposed. If not specified (default) then in each iteration of MCMC, all sampler actions are performed. Psplit a numeric vector of length <code>prior\$Kmax</code> giving conditional probabilities of the split move given k as opposite to the combine move. Default value is $(1, 0.5, \dots, 0.5, 0)'$. Pbirth a numeric vector of length <code>prior\$Kmax</code> giving conditional probabilities of the birth move given k as opposite to the death move. Default value is $(1, 0.5, \dots, 0.5, 0)'$. par.u1 a two component vector with parameters of the beta distribution used to generate an auxiliary value u_1. A default value is <code>par.u1</code> = $(2, 2)'$, i.e., $u_1 \sim \text{Beta}(2, 2)$. par.u2 a two component vector (for $p = 1$) or a matrix (for $p > 1$) with two columns with parameters of the distributions of the auxiliary values $u_{2,1}, \dots, u_{2,p}$ in rows. A default value leads to $u_{2,d} \sim \text{Unif}(-1, 1)$ ($d = 1, \dots, p - 1$), $u_{2,p} \sim \text{Beta}(1, 2p)$. par.u3 a two component vector (for $p = 1$) or a matrix (for $p > 1$) with two columns with parameters of the distributions of the auxiliary values $u_{3,1}, \dots, u_{3,p}$ in rows. A default value leads to $u_{3,d} \sim \text{Unif}(0, 1)$ ($d = 1, \dots, p - 1$), $u_{3,p} \sim \text{Beta}(1, p)$.
nMCMC		numeric vector of length 4 giving parameters of the MCMC simulation. Its components may be named (ordering is then unimportant) as:

	burn length of the burn-in (after discarding the thinned values), can be equal to zero as well.
	keep length of the kept chains (after discarding the thinned values), must be positive.
	thin thinning interval, must be positive.
	info interval in which the progress information is printed on the screen.
	In total $(M_{burn} + M_{keep}) \cdot M_{thin}$ MCMC scans are performed.
PED	a logical value which indicates whether the penalized expected deviance (see Plummer, 2008 for more details) is to be computed (which requires two parallel chains). If not specified, PED is set to TRUE for models with fixed number of components and is set to FALSE for models with numbers of components estimated using RJ-MCMC.
keep.chains	logical. If FALSE, only summary statistics are returned in the resulting object. This might be useful in the model searching step to save some memory.
onlyInit	logical. If TRUE then the function only determines parameters of the prior distribution, initial values, values of <code>scale</code> and parameters for the reversible jump MCMC.
dens.zero	a small value used instead of zero when computing deviance related quantities.
x	an object of class <code>NMixMCMC</code> or <code>NMixMCMClist</code> to be printed.
dic	logical which indicates whether DIC should be printed. By default, DIC is printed only for models with a fixed number of mixture components.
ped	logical which indicates whether PED should be printed. By default, PED is printed only for models with a fixed number of mixture components.
...	additional arguments passed to the default <code>print</code> method.

Details

See accompanying paper (Komárek, 2009). In the rest of the helpfile, the same notation is used as in the paper, namely, n denotes the number of observations, p is dimension of the data, K is the number of mixture components, w_1, \dots, w_K are mixture weights, $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ are mixture means, $\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K$ are mixture variance-covariance matrices, $\boldsymbol{Q}_1, \dots, \boldsymbol{Q}_K$ are their inverses.

For the data $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ the following $g_y(\boldsymbol{y})$ density is assumed

$$g_y(\boldsymbol{y}) = |\boldsymbol{S}|^{-1} \sum_{j=1}^K w_j \varphi(\boldsymbol{S}^{-1}(\boldsymbol{y} - \boldsymbol{m} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)),$$

where $\varphi(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a density of the (multivariate) normal distribution with mean $\boldsymbol{\mu}$ and a-variance-covariance matrix $\boldsymbol{\Sigma}$. Finally, \boldsymbol{S} is a pre-specified diagonal scale matrix and \boldsymbol{m} is a pre-specified shift vector. Sometimes, by setting \boldsymbol{m} to sample means of components of \boldsymbol{y} and diagonal of \boldsymbol{S} to sample standard deviations of \boldsymbol{y} (considerable) improvement of the MCMC algorithm is achieved.

Value

An object of class `NMixMCMC` or class `NMixMCMClist`. Object of class `NMixMCMC` is returned if `PED` is `FALSE`. Object of class `NMixMCMClist` is returned if `PED` is `TRUE`.

Objects of class `NMixMCMC` have the following components:

<code>iter</code>	index of the last iteration performed.
<code>nMCMC</code>	used value of the argument <code>nMCMC</code> .
<code>dim</code>	dimension p of the distribution of data
<code>prior</code>	a list containing the used value of the argument <code>prior</code> .
<code>init</code>	a list containing the used value of the argument <code>init</code> .
<code>RJMCMC</code>	a list containing the used value of the argument <code>RJMCMC</code> .
<code>scale</code>	a list containing the used value of the argument <code>scale</code> .
<code>state</code>	a list having the components labeled <code>y</code> , <code>K</code> , <code>w</code> , <code>mu</code> , <code>Li</code> , <code>Q</code> , <code>Sigma</code> , <code>gammaInv</code> , <code>r</code> containing the last sampled values of generic parameters.
<code>freqK</code>	frequency table of K based on the sampled chain.
<code>propK</code>	posterior distribution of K based on the sampled chain.
<code>DIC</code>	a <code>data.frame</code> having columns labeled <code>DIC</code> , <code>pD</code> , <code>D.bar</code> , <code>D.in.bar</code> containing values used to compute deviance information criterion (DIC). Currently only DIC_3 of Celeux et al. (2006) is implemented.
<code>moves</code>	a <code>data.frame</code> which summarizes the acceptance probabilities of different move types of the sampler.
<code>K</code>	numeric vector with a chain for K (number of mixture components).
<code>w</code>	numeric vector or matrix with a chain for w (mixture weights). It is a matrix with K columns when K is fixed. Otherwise, it is a vector with weights put sequentially after each other.
<code>mu</code>	numeric vector or matrix with a chain for μ (mixture means). It is a matrix with $p \cdot K$ columns when K is fixed. Otherwise, it is a vector with means put sequentially after each other.
<code>Q</code>	numeric vector or matrix with a chain for lower triangles of Q (mixture inverse variances). It is a matrix with $\frac{p(p+1)}{2} \cdot K$ columns when K is fixed. Otherwise, it is a vector with lower triangles of Q matrices put sequentially after each other.
<code>Sigma</code>	numeric vector or matrix with a chain for lower triangles of Σ (mixture variances). It is a matrix with $\frac{p(p+1)}{2} \cdot K$ columns when K is fixed. Otherwise, it is a vector with lower triangles of Σ matrices put sequentially after each other.
<code>Li</code>	numeric vector or matrix with a chain for lower triangles of Cholesky decompositions of Q matrices. It is a matrix with $\frac{p(p+1)}{2} \cdot K$ columns when K is fixed. Otherwise, it is a vector with lower triangles put sequentially after each other.
<code>gammaInv</code>	matrix with p columns with a chain for inverses of the hyperparameter γ
<code>order</code>	numeric vector or matrix with order indices of mixture components. It is a matrix with K columns when K is fixed. Otherwise it is a vector with orders put sequentially after each other.

<code>rank</code>	numeric vector or matrix with rank indices of mixture components. It is a matrix with K columns when K is fixed. Otherwise it is a vector with ranks put sequentially after each other.
<code>mixture</code>	<code>data.frame</code> with columns labeled <code>y.Mean.*</code> , <code>y.SD.*</code> , <code>y.Corr.*.*</code> , <code>z.Mean.*</code> , <code>z.SD.*</code> , <code>z.Corr.*.*</code> containing the chains for the means, standard deviations and correlations of the distribution of the original (y) and scaled (z) data based on a normal mixture at each iteration.
<code>deviance</code>	<code>data.frame</code> with columns labeled <code>LogL0</code> , <code>LogL1</code> , <code>dev.complete</code> , <code>dev.observed</code> containing the chains of quantities needed to compute DIC.
<code>pm.y</code>	a <code>data.frame</code> with p columns with posterior means for (latent) values of observed data (useful when there is censoring).
<code>pm.z</code>	a <code>data.frame</code> with p columns with posterior means for (latent) values of scaled observed data (useful when there is censoring).
<code>pm.indDev</code>	a <code>data.frame</code> with columns labeled <code>LogL0</code> , <code>LogL1</code> , <code>dev.complete</code> , <code>dev.observed</code> , <code>pred.dens</code> containing posterior means of individual contributions to the deviance.
<code>pred.dens</code>	a numeric vector with the predictive density of the data based on the MCMC sample evaluated at data points. Note that when there is censoring, this is not exactly the predictive density as it is computed as the average of densities at each iteration evaluated at sampled values of latent observations at iterations.
<code>summ.y.Mean</code>	Posterior summary statistics based on chains stored in <code>y.Mean.*</code> columns of the <code>data.frame mixture</code> .
<code>summ.y.SDCorr</code>	Posterior summary statistics based on chains stored in <code>y.SD.*</code> and <code>y.Corr.*.*</code> columns of the <code>data.frame mixture</code> .
<code>summ.z.Mean</code>	Posterior summary statistics based on chains stored in <code>z.Mean.*</code> columns of the <code>data.frame mixture</code> .
<code>summ.z.SDCorr</code>	Posterior summary statistics based on chains stored in <code>z.SD.*</code> and <code>z.Corr.*.*</code> columns of the <code>data.frame mixture</code> .
<code>poster.mean.w</code>	a numeric vector with posterior means of mixture weights after re-labeling. It is computed only if K is fixed and even then I am not convinced that these are useful posterior summary statistics. In any case, they should be used with care.
<code>poster.mean.mu</code>	a matrix with posterior means of mixture means after re-labeling. It is computed only if K is fixed and even then I am not convinced that these are useful posterior summary statistics. In any case, they should be used with care.
<code>poster.mean.Q</code>	a list with posterior means of mixture inverse variances after re-labeling. It is computed only if K is fixed and even then I am not convinced that these are useful posterior summary statistics. In any case, they should be used with care.
<code>poster.mean.Sigma</code>	a list with posterior means of mixture variances after re-labeling. It is computed only if K is fixed and even then I am not convinced that these are useful posterior summary statistics. In any case, they should be used with care.

`poster.mean.Li`

a list with posterior means of Cholesky decompositions of mixture inverse variances after re-labeling. It is computed only if K is fixed and even then I am not convinced that these are useful posterior summary statistics. In any case, they should be used with care.

Object of class `NMixMCMClist`

Objects of class `NMixMCMClist` is the list having two components of class `NMixMCMC` representing two parallel chains and additionally the following components:

PED values of penalized expected deviance and related quantities. It is a vector with five components: `D.expect` = estimated expected deviance, where the estimate is based on two parallel chains; `popt` = estimated penalty, where the estimate is based on simple MCMC average based on two parallel chains; `PED` = estimated penalized expected deviance = `D.expect` + `popt`; `wpopt` = estimated penalty, where the estimate is based on weighted MCMC average (through importance sampling) based on two parallel chains; `wPED` = estimated penalized expected deviance = `D.expect` + `wpopt`.

popt contributions to the unweighted penalty from each observation.

wpopt contributions to the weighted penalty from each observation.

inv.D for each observation, number of iterations (in both chains), where the deviance was in fact equal to infinity (when the corresponding density was lower than `dens.zero`) and was not taken into account when computing `D.expect`.

inv.popt for each observation, number of iterations, where the penalty was in fact equal to infinity and was not taken into account when computing `popt`.

inv.wpopt for each observation, number of iterations, where the importance sampling weight was in fact equal to infinity and was not taken into account when computing `wpopt`.

sumISw for each observation, sum of importance sampling weights.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

- Celeux, G., Forbes, F., Robert, C. P., and Titterton, D. M. (2006). Deviance information criteria for missing data models. *Bayesian Analysis*, **1**, 651-674.
- Cappé, Robert and Rydén (2003). Reversible jump, birth-and-death and more general continuous time Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B*, **65**, 679-700.
- Dellaportas, P. and Papageorgiou, I. (2006). Multivariate mixtures of normals with unknown number of components. *Statistics and Computing*, **16**, 57-68.
- Diebolt, J. and Robert, C. P. (1994). Estimation of finite mixture distributions through Bayesian sampling. *Journal of the Royal Statistical Society, Series B*, **56**, 363-375.
- Komárek, A. (2009). A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

Plummer, M. (2008). Penalized loss functions for Bayesian model comparison. *Biostatistics*, **9**, 523-539.

Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with unknown number of components (with Discussion). *Journal of the Royal Statistical Society, Series B*, **59**, 731-792.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002). Bayesian measures of model complexity and fit (with Discussion). *Journal of the Royal Statistical Society, Series B*, **64**, 583-639.

See Also

[NMixPredDensMarg](#), [NMixPredDensJoint2](#).

Examples

```
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files Galaxy.pdf, Faithful.pdf, Tandmob.pdf
```

NMixPlugCondDensJoint2

Pairwise bivariate conditional densities: plug-in estimate

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimates of pairwise bivariate conditional densities (given one margin) obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances (plug-in estimate).

Usage

```
NMixPlugCondDensJoint2(x, ...)

## Default S3 method:
NMixPlugCondDensJoint2(x, icond, scale, w, mu, Sigma, ...)

## S3 method for class 'NMixMCMC':
NMixPlugCondDensJoint2(x, icond, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPlugCondDensJoint2(x, icond, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

<code>x</code>	<p>an object of class <code>NMixMCMC</code> for <code>NMixPlugCondDensJoint2.NMixMCMC</code> function.</p> <p>An object of class <code>GLMM_MCMC</code> for <code>NMixPlugCondDensJoint2.GLMM_MCMC</code> function.</p> <p>A list with the grid values (see below) for <code>NMixPlugCondDensJoint2.default</code> function.</p>
<code>icond</code>	index of the margin by which we want to condition
<code>scale</code>	a two component list giving the <code>shift</code> and the <code>scale</code> . If not given, <code>shift</code> is equal to zero and <code>scale</code> is equal to one.
<code>w</code>	a numeric vector with posterior summary statistics for the mixture weights. The length of this vector determines the number of mixture components.
<code>mu</code>	a matrix with posterior summary statistics for mixture means in rows. That is, <code>mu</code> has K rows and p columns, where K denotes the number of mixture components and p is dimension of the mixture distribution.
<code>Sigma</code>	a list with posterior summary statistics for mixture covariance matrices.
<code>grid</code>	<p>a list with the grid values for each margin in which the density should be evaluated. The value of <code>grid[[icond]]</code> determines the values by which we condition.</p> <p>If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code>.</p>
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The <code>shift</code> and <code>scale</code> vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPlugCondDensJoint2` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>icond</code>	index of the margin by which we condition.
<code>dens</code>	<p>a list with the computed conditional densities for each value of <code>x[[icond]]</code>. Each <code>dens[[j]]</code> is again a list with conditional densities for each pair of margins given margin <code>icond</code> equal to <code>x[[icond]][j]</code>. The value of <code>dens[[j]][["{i-k}"]]</code> gives values of conditional density of the (i,k)-th margins given margin <code>icond</code> equal to <code>x[[icond]][j]</code>.</p>

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[plot.NMixPlugCondDensJoint2](#), [NMixMCMC](#), [GLMM_MCMC](#), [NMixPredCondDensJoint2](#).

NMixPlugCondDensMarg

Univariate conditional densities: plug-in estimate

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimates of univariate conditional densities obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances (plug-in estimate).

Usage

```
NMixPlugCondDensMarg(x, ...)

## Default S3 method:
NMixPlugCondDensMarg(x, icond, scale, w, mu, Sigma, ...)

## S3 method for class 'NMixMCMC':
NMixPlugCondDensMarg(x, icond, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPlugCondDensMarg(x, icond, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

x	an object of class <code>NMixMCMC</code> for <code>NMixPlugCondDensMarg.NMixMCMC</code> function. An object of class <code>GLMM_MCMC</code> for <code>NMixPlugCondDensMarg.GLMM_MCMC</code> function. A list with the grid values (see below) for <code>NMixPlugCondDensMarg.default</code> function.
icond	index of the margin by which we want to condition
scale	a two component list giving the shift and the scale. If not given, shift is equal to zero and scale is equal to one.
w	a numeric vector with posterior summary statistics for the mixture weights. The length of this vector determines the number of mixture components.
mu	a matrix with posterior summary statistics for mixture means in rows. That is, <code>mu</code> has K rows and p columns, where K denotes the number of mixture components and p is dimension of the mixture distribution.
Sigma	a list with posterior summary statistics for mixture covariance matrices.

<code>grid</code>	a list with the grid values for each margin in which the density should be evaluated. The value of <code>grid[[icond]]</code> determines the values by which we condition. If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code> .
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPlugCondDensMarg` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>icond</code>	index of the margin by which we condition.
<code>dens</code>	a list with the computed conditional densities for each value of <code>x[[icond]]</code> . Each <code>dens[[j]]</code> is again a list with conditional densities for each margin given margin <code>icond</code> equal to <code>x[[icond]][j]</code> . The value of <code>dens[[j]][[imargin]]</code> gives a value of a marginal density of the <code>imargin</code> -th margin at <code>x[[icond]][j]</code> .

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[plot.NMixPlugCondDensMarg](#), [NMixMCMC](#), [GLMM_MCMC](#), [NMixPredCondDensMarg](#).

NMixPlugDensJoint2 *Pairwise bivariate densities: plug-in estimate*

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes marginal (pairwise bivariate) plug-in densities obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances.

Usage

```

NMixPlugDensJoint2(x, ...)

## Default S3 method:
NMixPlugDensJoint2(x, scale, w, mu, Sigma, ...)

## S3 method for class 'NMixMCMC':
NMixPlugDensJoint2(x, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPlugDensJoint2(x, grid, lgrid=50, scaled=FALSE, ...)

```

Arguments

<code>x</code>	<p>an object of class <code>NMixMCMC</code> for <code>NMixPlugDensJoint2.NMixMCMC</code> function.</p> <p>An object of class <code>GLMM_MCMC</code> for <code>NMixPlugDensJoint2.GLMM_MCMC</code> function.</p> <p>A list with the grid values (see below) for <code>NMixPlugDensJoint2.default</code> function.</p>
<code>scale</code>	a two component list giving the shift and the scale. If not given, shift is equal to zero and scale is equal to one.
<code>w</code>	a numeric vector with posterior summary statistics for the mixture weights. The length of this vector determines the number of mixture components.
<code>mu</code>	a matrix with posterior summary statistics for mixture means in rows. That is, <code>mu</code> has K rows and p columns, where K denotes the number of mixture components and p is dimension of the mixture distribution.
<code>Sigma</code>	a list with posterior summary statistics for for mixture covariance matrices.
<code>grid</code>	<p>a list with the grid values for each margin in which the density should be evaluated.</p> <p>If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code>.</p>
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPlugDensJoint2` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>dens</code>	a list with the computed densities for each pair of margins. The components of the list are named <code>1-2, 1-3, ...</code> , i.e., <code>dens[[1]] = dens[["1-2"]]</code> is

the pairwise predictive density for margins 1 and 2, etc. Each component of the `list` is a matrix in such a form that it can be directly passed together with the proper components of `x` to the plotting functions like `contour` or `image`.

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

`plot.NMixPlugDensJoint2`, `NMixMCMC`, `GLMM_MCMC`, `NMixPredDensJoint2`.

NMixPlugDensMarg *Marginal (univariate) densities: plug-in estimate*

Description

This function serves as an inference tool for the MCMC output obtained using the function `NMixMCMC`. It computes marginal (univariate) plug-in densities obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances.

Usage

```
NMixPlugDensMarg(x, ...)
```

```
## Default S3 method:
NMixPlugDensMarg(x, scale, w, mu, Sigma, ...)
```

```
## S3 method for class 'NMixMCMC':
NMixPlugDensMarg(x, grid, lgrid=50, scaled=FALSE, ...)
```

```
## S3 method for class 'GLMM_MCMC':
NMixPlugDensMarg(x, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

<code>x</code>	<p>an object of class <code>NMixMCMC</code> for <code>NMixPlugDensMarg.NMixMCMC</code> function.</p> <p>An object of class <code>GLMM_MCMC</code> for <code>NMixPlugDensMarg.GLMM_MCMC</code> function.</p> <p>A list with the grid values (see below) for <code>NMixPlugDensMarg.default</code> function.</p>
<code>scale</code>	<p>a two component list giving the <code>shift</code> and the <code>scale</code>. If not given, <code>shift</code> is equal to zero and <code>scale</code> is equal to one.</p>

<code>w</code>	a numeric vector with posterior summary statistics for the mixture weights. The length of this vector determines the number of mixture components.
<code>mu</code>	a matrix with posterior summary statistics for mixture means in rows. That is, <code>mu</code> has K rows and p columns, where K denotes the number of mixture components and p is dimension of the mixture distribution.
<code>Sigma</code>	a list with posterior summary statistics for for mixture covariance matrices.
<code>grid</code>	a list with the grid values for each margin in which the density should be evaluated. If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code> .
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPlugDensMarg` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>dens</code>	a list with the computed densities for each margin. The components of the list are named <code>1, ...</code> , i.e., <code>dens[[1]] = dens[["1"]]</code> is the predictive density for margin 1 etc.

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[plot.NMixPlugDensMarg](#), [NMixMCMC](#), [GLMM_MCMC](#), [NMixPredDensMarg](#).

`NMixPredCDFMarg` *Marginal (univariate) predictive cumulative distribution function*

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimated posterior predictive cumulative distribution function for each margin.

Usage

```

NMixPredCDFMarg(x, ...)

## Default S3 method:
NMixPredCDFMarg(x, scale, K, w, mu, Li, Krandom=TRUE, ...)

## S3 method for class 'NMixMCMC':
NMixPredCDFMarg(x, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPredCDFMarg(x, grid, lgrid=50, scaled=FALSE, ...)

```

Arguments

<code>x</code>	an object of class <code>NMixMCMC</code> for <code>NMixPredCDFMarg.NMixMCMC</code> function. An object of class <code>GLMM_MCMC</code> for <code>NMixPredCDFMarg.GLMM_MCMC</code> function. A list with the grid values (see below) for <code>NMixPredCDFMarg.default</code> function.
<code>scale</code>	a two component list giving the shift and the scale.
<code>K</code>	either a number (when <code>Krandom=FALSE</code>) or a numeric vector with the chain for the number of mixture components.
<code>w</code>	a numeric vector with the chain for the mixture weights.
<code>mu</code>	a numeric vector with the chain for the mixture means.
<code>Li</code>	a numeric vector with the chain for the mixture inverse variances (lower triangles only).
<code>Krandom</code>	a logical value which indicates whether the number of mixture components changes from one iteration to another.
<code>grid</code>	a numeric vector or a list with the grid values in which the predictive CDF should be evaluated. If <code>x\$dim</code> is 1 then <code>grid</code> may be a numeric vector. If <code>x\$dim</code> is higher than then <code>grid</code> must be a list with numeric vectors as components giving the grids for each margin. If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code> .
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the CDF of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPredCDFMarg` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
----------------	---

<code>freqK</code>	frequency table for the values of K (numbers of mixture components) in the MCMC chain.
<code>propK</code>	proportions derived from <code>freqK</code> .
<code>MCMC.length</code>	the length of the MCMC used to compute the predictive densities.
<code>cdf</code>	a list with the computed predictive CDF's for each margin. The components of the list are named 1, ..., i.e., <code>cdf[[1]] = cdf[["1"]]</code> is the predictive density for margin 1 etc.
<code>cdfK</code>	a list with the computed predictive CDF's for each margin, conditioned further by K . The components of the list are named 1, ... That is, <code>cdf[[1]][[1]] = cdf[["1"]][[1]]</code> is the predictive CDF for margin 1 conditioned by $K = 1$, <code>cdf[[1]][[2]] = cdf[["1"]][[2]]</code> is the predictive CDF for margin 1 conditioned by $K = 2$ etc. Note that <code>cdfK</code> provides some additional information only when <code>Krandom = TRUE</code> or when <code>x</code> results from the NMixMCMC call to the reversible jump MCMC.

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[plot.NMixPredCDFMarg](#), [NMixMCMC](#), [GLMM_MCMC](#).

NMixPredCondDensJoint2

Pairwise bivariate conditional predictive densities

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimates of pairwise bivariate conditional densities (given one margin) obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances (plug-in estimate).

Usage

```

NMixPredCondDensJoint2(x, ...)

## Default S3 method:
NMixPredCondDensJoint2(x, icond, scale, K, w, mu, Li, Krandom=FALSE, ...)

## S3 method for class 'NMixMCMC':
NMixPredCondDensJoint2(x, icond, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPredCondDensJoint2(x, icond, grid, lgrid=50, scaled=FALSE, ...)

```

Arguments

<code>x</code>	<p>an object of class <code>NMixMCMC</code> for <code>NMixPredCondDensJoint2.NMixMCMC</code> function.</p> <p>An object of class <code>GLMM_MCMC</code> for <code>NMixPredCondDensJoint2.GLMM_MCMC</code> function.</p> <p>A list with the grid values (see below) for <code>NMixPredCondDensJoint2.default</code> function.</p>
<code>icond</code>	index of the margin by which we want to condition
<code>scale</code>	a two component list giving the shift and the scale. If not given, shift is equal to zero and scale is equal to one.
<code>K</code>	either a number (when <code>Krandom=FALSE</code>) or a numeric vector with the chain for the number of mixture components.
<code>w</code>	a numeric vector with the chain for the mixture weights.
<code>mu</code>	a numeric vector with the chain for the mixture means.
<code>Li</code>	a numeric vector with the chain for the mixture inverse variances (lower triangles only).
<code>Krandom</code>	a logical value which indicates whether the number of mixture components changes from one iteration to another.
<code>grid</code>	<p>a list with the grid values for each margin in which the density should be evaluated. The value of <code>grid[[icond]]</code> determines the values by which we condition.</p> <p>If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code>.</p>
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPredCondDensJoint2` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>icond</code>	index of the margin by which we condition.
<code>dens</code>	a list with the computed conditional densities for each value of <code>x[[icond]]</code> . Each <code>dens[[j]]</code> is again a list with conditional densities for each pair of margins given margin <code>icond</code> equal to <code>x[[icond]][j]</code> . The value of <code>dens[[j]][["dQuote{i-k}"]]</code> gives values of conditional density of the (i,k)-th margins given margin <code>icond</code> equal to <code>x[[icond]][j]</code> .

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[plot.NMixPredCondDensJoint2](#), [NMixMCMC](#), [GLMM_MCMC](#).

NMixPredCondDensMarg

Univariate conditional predictive density

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimates of univariate conditional densities obtained by using posterior summary statistics (e.g., posterior means) of mixture weights, means and variances (plug-in estimate).

Usage

```
NMixPredCondDensMarg(x, ...)

## Default S3 method:
NMixPredCondDensMarg(x, icond, scale, K, w, mu, Li, Krandom=FALSE, ...)

## S3 method for class 'NMixMCMC':
NMixPredCondDensMarg(x, icond, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPredCondDensMarg(x, icond, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

<code>x</code>	<p>an object of class <code>NMixMCMC</code> for <code>NMixPredCondDensMarg.NMixMCMC</code> function.</p> <p>An object of class <code>GLMM_MCMC</code> for <code>NMixPredCondDensMarg.GLMM_MCMC</code> function.</p> <p>A list with the grid values (see below) for <code>NMixPredCondDensMarg.default</code> function.</p>
<code>icond</code>	index of the margin by which we want to condition
<code>scale</code>	a two component list giving the <code>shift</code> and the <code>scale</code> . If not given, <code>shift</code> is equal to zero and <code>scale</code> is equal to one.
<code>K</code>	either a number (when <code>Krandom=FALSE</code>) or a numeric vector with the chain for the number of mixture components.
<code>w</code>	a numeric vector with the chain for the mixture weights.
<code>mu</code>	a numeric vector with the chain for the mixture means.
<code>Li</code>	a numeric vector with the chain for the mixture inverse variances (lower triangles only).
<code>Krandom</code>	a logical value which indicates whether the number of mixture components changes from one iteration to another.
<code>grid</code>	<p>a list with the grid values for each margin in which the density should be evaluated. The value of <code>grid[[icond]]</code> determines the values by which we condition.</p> <p>If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code>.</p>
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The <code>shift</code> and <code>scale</code> vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPredCondDensMarg` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>icond</code>	index of the margin by which we condition.
<code>dens</code>	<p>a list with the computed conditional densities for each value of <code>x[[icond]]</code>. Each <code>dens[[j]]</code> is again a list with conditional densities for each margin given margin <code>icond</code> equal to <code>x[[icond]][j]</code>. The value of <code>dens[[j]][[imargin]]</code> gives a value of a marginal density of the <code>imargin</code>-th margin at <code>x[[icond]][j]</code>.</p>

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[plot.NMixPredCondDensMarg](#), [NMixMCMC](#), [GLMM_MCMC](#).

NMixPredDensJoint2 *Pairwise bivariate predictive density*

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimated posterior predictive densities for each pair of margins.

Usage

```
NMixPredDensJoint2(x, ...)

## Default S3 method:
NMixPredDensJoint2(x, scale, K, w, mu, Li, Krandom=TRUE, ...)

## S3 method for class 'NMixMCMC':
NMixPredDensJoint2(x, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'NMixMCMC':
NMixPredDensJoint2(x, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

x	an object of class <code>NMixMCMC</code> for <code>NMixPredDensJoint2.NMixMCMC</code> function. an object of class <code>NMixMCMC</code> for <code>NMixPlugDensJoint2.NMixMCMC</code> function. A list with the grid values (see below) for <code>NMixPredDensJoint2.default</code> function.
scale	a two component list giving the <code>shift</code> and the <code>scale</code> . If not given, <code>shift</code> is equal to zero and <code>scale</code> is equal to one.
K	either a number (when <code>Krandom=FALSE</code>) or a numeric vector with the chain for the number of mixture components.
w	a numeric vector with the chain for the mixture weights.
mu	a numeric vector with the chain for the mixture means.
Li	a numeric vector with the chain for the mixture inverse variances (lower triangles only).
Krandom	a logical value which indicates whether the number of mixture components changes from one iteration to another.

<code>grid</code>	a list with the grid values for each margin in which the predictive density should be evaluated. If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code> .
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPredDensJoint2` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>freqK</code>	frequency table for the values of K (numbers of mixture components) in the MCMC chain.
<code>propK</code>	proportions derived from <code>freqK</code> .
<code>MCMC.length</code>	the length of the MCMC used to compute the predictive densities.
<code>dens</code>	a list with the computed predictive densities for each pair of margins. The components of the list are named <code>1-2, 1-3, ...</code> , i.e., <code>dens[[1]] = dens[["1-2"]]</code> is the pairwise predictive density for margins 1 and 2, etc. Each component of the list is a matrix in such a form that it can be directly passed together with the proper components of <code>x</code> to the plotting functions like <code>contour</code> or <code>image</code> .
<code>densK</code>	a list with the computed predictive densities for each margin, conditioned further by K . The components of the list are named <code>1-2, 1-3, ...</code> . That is, <code>dens[[1]][[1]] = dens[["1-2"]][[1]]</code> is the pairwise predictive density for margins 1 and 2 conditioned by $K = 1$, <code>dens[[1]][[2]] = dens[["1-2"]][[2]]</code> is the pairwise predictive density for margins 1 and 2 conditioned by $K = 2$ etc. Note that <code>densK</code> provides some additional information only when <code>Krandom = TRUE</code> or when <code>x</code> results from the <code>NMixMCMC</code> call to the reversible jump MCMC.

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[plot.NMixPredDensJoint2](#), [NMixMCMC](#), [GLMM_MCMC](#), [NMixPredDensMarg](#).

Examples

```
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files Galaxy.pdf, Faithful.pdf, Tandmob.pdf
```

NMixPredDensMarg *Marginal (univariate) predictive density*

Description

This function serves as an inference tool for the MCMC output obtained using the function [NMixMCMC](#). It computes estimated posterior predictive densities for each margin.

Usage

```
NMixPredDensMarg(x, ...)

## Default S3 method:
NMixPredDensMarg(x, scale, K, w, mu, Li, Krandom=TRUE, ...)

## S3 method for class 'NMixMCMC':
NMixPredDensMarg(x, grid, lgrid=50, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC':
NMixPredDensMarg(x, grid, lgrid=50, scaled=FALSE, ...)
```

Arguments

x	an object of class <code>NMixMCMC</code> for <code>NMixPredDensMarg.NMixMCMC</code> function. An object of class <code>GLMM_MCMC</code> for <code>NMixPredDensMarg.GLMM_MCMC</code> function.
scale	a two component list giving the shift and the scale.
K	either a number (when <code>Krandom=FALSE</code>) or a numeric vector with the chain for the number of mixture components.
w	a numeric vector with the chain for the mixture weights.
mu	a numeric vector with the chain for the mixture means.
Li	a numeric vector with the chain for the mixture inverse variances (lower triangles only).

<code>Krandom</code>	a logical value which indicates whether the number of mixture components changes from one iteration to another.
<code>grid</code>	a numeric vector or a list with the grid values in which the predictive density should be evaluated. If <code>x\$dim</code> is 1 then <code>grid</code> may be a numeric vector. If <code>x\$dim</code> is higher than 1 then <code>grid</code> must be a list with numeric vectors as components giving the grids for each margin. If <code>grid</code> is not specified, it is created automatically using the information from the posterior summary statistics stored in <code>x</code> .
<code>lgrid</code>	a length of the grid used to create the <code>grid</code> if that is not specified.
<code>scaled</code>	if <code>TRUE</code> , the density of shifted and scaled data is summarized. The shift and scale vector are taken from the <code>scale</code> component of the object <code>x</code> .
<code>...</code>	optional additional arguments.

Value

An object of class `NMixPredDensMarg` which has the following components:

<code>x</code>	a list with the grid values for each margin. The components of the list are named <code>x1, ...</code> or take names from <code>grid</code> argument.
<code>freqK</code>	frequency table for the values of K (numbers of mixture components) in the MCMC chain.
<code>propK</code>	proportions derived from <code>freqK</code> .
<code>MCMC.length</code>	the length of the MCMC used to compute the predictive densities.
<code>dens</code>	a list with the computed predictive densities for each margin. The components of the list are named <code>1, ...</code> , i.e., <code>dens[[1]] = dens[["1"]]</code> is the predictive density for margin 1 etc.
<code>densK</code>	a list with the computed predictive densities for each margin, conditioned further by K . The components of the list are named <code>1, ...</code> . That is, <code>dens[[1]][[1]] = dens[["1"]][[1]]</code> is the predictive density for margin 1 conditioned by $K = 1$, <code>dens[[1]][[2]] = dens[["1"]][[2]]</code> is the predictive density for margin 1 conditioned by $K = 2$ etc. Note that <code>densK</code> provides some additional information only when <code>Krandom = TRUE</code> or when <code>x</code> results from the <code>NMixMCMC</code> call to the reversible jump MCMC.

There is also a `plot` method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[plot.NMixPredDensMarg](#), [NMixMCMC](#), [GLMM_MCMC](#), [NMixPredDensJoint2](#).

Examples

```
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files Galaxy.pdf, Faithful.pdf, Tandmob.pdf
```

NMixSummComp

Summary for the mixture components

Description

This function returns basic posterior summary for (re-labeled) mixture components in a model with fixed number of components fitted with [NMixMCMC](#) function. The summary also takes into account possible scaling and shifting of the data (see argument `scale` in [NMixMCMC](#) function).

Note that even though the mixture components are re-labeled before the summary is computed to achieve identifiability, posterior summaries of individual mixture means and variances are not always the quantity we would like to see. For density estimation, usually posterior predictive density ([NMixPredDensMarg](#), [NMixPredDensJoint2](#)) is usually the right stuff one should be interested in.

Usage

```
NMixSummComp(x)
```

Arguments

`x` an object of class `NMixMCMC`

Value

Invisible `x`. The rest is printed on output device.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixMCMC](#).

```
plot.NMixPlugCondDensJoint2
    Plot computed pairwise bivariate conditional densities (plug-in estimate)
```

Description

This is a basic plotting tool to visualize computed plug-in estimates of pairwise bivariate conditional densities using the [image](#) or [contour](#) plot. See also [NMixPlugCondDensJoint2](#).

Usage

```
## S3 method for class 'NMixPlugCondDensJoint2':
plot(x, ixcond, imargin,
     contour=FALSE, auto.layout=TRUE,
     col, lwd=1, main, xylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPlugCondDensJoint2</code> .
<code>ixcond</code>	if given then conditional densities of all pairs of margins given <code>x[[ixcond]][ixcond]</code> are plotted where <code>ixcond</code> is taken from <code>x</code> .
<code>imargin</code>	vector of length 2. if given then conditional densities of the (<code>imargin[1]</code> , <code>imargin[2]</code>) pair of margins given all values of <code>x[[ixcond]]</code> are plotted.
<code>contour</code>	logical. If <code>TRUE</code> then contours are drawn, otherwise image plot is created.
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>col</code>	color used to draw the contours or images.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot.
<code>xylab</code>	optional character vector of the length equal to the number of margins with labels used for x and y axes on the plots.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

```
invisible(x)
```

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPlugCondDensJoint2](#), [NMixMCMC](#).

```
plot.NMixPlugCondDensMarg
```

Plot computed univariate conditional densities (plug-in estimate)

Description

This is a basic plotting tool to visualize computed plug-in estimates of univariate conditional densities, see [NMixPlugCondDensMarg](#).

Usage

```
## S3 method for class 'NMixPlugCondDensMarg':
plot(x, ixcond, imargin, over=FALSE,
     auto.layout=TRUE, type="l", lwd=1, lty, col, main, xlab, ylab,
     annot=TRUE, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPlugCondDensMarg</code> .
<code>ixcond</code>	if given then conditional densities of all margins given <code>x[[ixcond]]</code> are plotted where <code>ixcond</code> is taken from <code>x</code> .
<code>imargin</code>	if given then conditional densities of the <code>imargin</code> -th margin given all values of <code>x[[ixcond]]</code> are plotted - either separately or all in one plot.
<code>over</code>	logical. If <code>TRUE</code> and <code>imargin</code> is given then all conditional densities are drawn in one plot.
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw the computed densities.
<code>type</code>	type of the plot.
<code>lwd</code>	line width.
<code>col</code>	color used to draw the lines. It can be a vector in which case different lines are drawn in different colors.
<code>lty</code>	type of the line. It can be a vector in which case different lines are drawn in different types.
<code>main</code>	main title of the plot. Either character which is replicated or a vector of characters.
<code>xlab</code>	label for the x-axis. Either character which is replicated or a vector of characters.
<code>ylab</code>	label for the y-axis. Either character which is replicated or a vector of characters.
<code>annot</code>	if <code>TRUE</code> and <code>imargin</code> is given and <code>over</code> is <code>TRUE</code> then a legend is added to the plot.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

`invisible(x)`

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPlugCondDensMarg](#), [NMixMCMC](#).

```
plot.NMixPlugDensJoint2
```

Plot computed marginal pairwise bivariate densities (plug-in estimate)

Description

This is a basic plotting tool to visualize computed marginal pairwise bivariate densities (plug-in version) using the `contour` plot. See also [NMixPlugDensJoint2](#).

Usage

```
## S3 method for class 'NMixPlugDensJoint2':
plot(x, contour=FALSE, auto.layout=TRUE,
     col, lwd=1, main, xylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPlugDensJoint2</code> .
<code>contour</code>	logical. If <code>TRUE</code> then contours are drawn, otherwise image plot is created.
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>col</code>	color used to draw the contours or images.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot.
<code>xylab</code>	optional character vector of the length equal to the number of margins with labels used for x and y axes on the plots.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

```
invisible(x)
```

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPlugDensJoint2](#), [NMixMCMC](#).

```
plot.NMixPlugDensMarg
```

Plot computed marginal predictive densities

Description

This is a basic plotting tool to visualize computed marginal plug-in estimates of densities, see [NMixPlugDensMarg](#).

Usage

```
## S3 method for class 'NMixPlugDensMarg':
plot(x, auto.layout=TRUE,
      type="l", col="darkblue", lty=1, lwd=1, main, xlab, ylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPlugDensMarg</code> .
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>type</code>	type of the plot.
<code>col</code>	color used to draw the lines.
<code>lty</code>	type of the line.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>xlab</code>	label for the x-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>ylab</code>	label for the y-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

```
invisible(x)
```

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPlugDensMarg](#), [NMixMCMC](#).

`plot.NMixPredCDFMarg`*Plot computed marginal predictive cumulative distribution functions*

Description

This is a basic plotting tool to visualize computed marginal cumulative distribution functions, see [NMixPredCDFMarg](#).

Usage

```
## S3 method for class 'NMixPredCDFMarg':  
plot(x, K=0, auto.layout=TRUE,  
     type="l", col="darkblue", lty=1, lwd=1, main, xlab, ylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPredCDFMarg</code> .
<code>K</code>	if equal to 0 then the overall predictive CDF's are plotted taken from the <code>dens</code> part of the object <code>x</code> . If higher than 0 then the predictive CDF conditioned by the value of <code>K</code> is plotted (taken from the <code>densK</code> part of the object <code>x</code>).
<code>auto.layout</code>	if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>type</code>	type of the plot.
<code>col</code>	color used to draw the lines.
<code>lty</code>	type of the line.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>xlab</code>	label for the x-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>ylab</code>	label for the y-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value`invisible(x)`**Author(s)**

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[NMixPredCDFMarg](#), [NMixMCMC](#).

```
plot.NMixPredCondDensJoint2
```

Plot computed predictive pairwise bivariate conditional densities

Description

This is a basic plotting tool to visualize computed predictive pairwise bivariate conditional densities using the [image](#) or [contour](#) plot. See also [NMixPredCondDensJoint2](#).

Usage

```
## S3 method for class 'NMixPredCondDensJoint2':
plot(x, ixcond, imargin,
     contour=FALSE, auto.layout=TRUE,
     col, lwd=1, main, xylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPredCondDensJoint2</code> .
<code>ixcond</code>	if given then conditional densities of all pairs of margins given <code>x[[ixcond]][ixcond]</code> are plotted where <code>ixcond</code> is taken from <code>x</code> .
<code>imargin</code>	vector of length 2. if given then conditional densities of the (<code>imargin[1]</code> , <code>imargin[2]</code>) pair of margins given all values of <code>x[[ixcond]]</code> are plotted.
<code>contour</code>	logical. If <code>TRUE</code> then contours are drawn, otherwise image plot is created.
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>col</code>	color used to draw the contours or images.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot.
<code>xylab</code>	optional character vector of the length equal to the number of margins with labels used for x and y axes on the plots.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

`invisible(x)`

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPredCondDensJoint2](#), [NMixMCMC](#).

```
plot.NMixPredCondDensMarg
```

Plot computed univariate conditional predictive densities

Description

This is a basic plotting tool to visualize computed plug-in estimates of univariate conditional densities, see [NMixPredCondDensMarg](#).

Usage

```
## S3 method for class 'NMixPredCondDensMarg':
plot(x, ixcond, imargin, over=FALSE,
     auto.layout=TRUE, type="l", lwd=1, lty, col, main, xlab, ylab,
     annot=TRUE, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPredCondDensMarg</code> .
<code>ixcond</code>	if given then conditional densities of all margins given <code>x[[ixcond]]</code> are plotted where <code>ixcond</code> is taken from <code>x</code> .
<code>imargin</code>	if given then conditional densities of the <code>imargin</code> -th margin given all values of <code>x[[ixcond]]</code> are plotted - either separately or all in one plot.
<code>over</code>	logical. If <code>TRUE</code> and <code>imargin</code> is given then all conditional densities are drawn in one plot.
<code>auto.layout</code>	if <code>TRUE</code> then the function determines itself how to divide the plotting region to draw the computed densities.
<code>type</code>	type of the plot.
<code>lwd</code>	line width.
<code>col</code>	color used to draw the lines. It can be a vector in which case different lines are drawn in different colors.
<code>lty</code>	type of the line. It can be a vector in which case different lines are drawn in different types.
<code>main</code>	main title of the plot. Either character which is replicated or a vector of characters.
<code>xlab</code>	label for the x-axis. Either character which is replicated or a vector of characters.

<code>ylab</code>	label for the y-axis. Either character which is replicated or a vector of characters.
<code>annot</code>	if TRUE and <code>imargin</code> is given and <code>over</code> is TRUE then a legend is added to the plot.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

`invisible(x)`

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPredCondDensMarg](#), [NMixMCMC](#).

`plot.NMixPredDensJoint2`

Plot computed marginal pairwise bivariate predictive densities

Description

This is a basic plotting tool to visualize computed marginal pairwise bivariate predictive densities using the [image](#) plot or [contour](#) plot. See also [NMixPredDensJoint2](#).

Usage

```
## S3 method for class 'NMixPredDensJoint2':
plot(x, K=0, contour=FALSE,
     auto.layout=TRUE,
     col, lwd=1, main, xylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPredDensJoint2</code> .
<code>K</code>	if equal to 0 then the overall predictive densities are plotted taken from the <code>dens</code> part of the object <code>x</code> . If higher than 0 then the predictive density conditioned by the value of <code>K</code> is plotted (taken from the <code>densK</code> part of the object <code>x</code>).
<code>contour</code>	logical. If TRUE then contours are drawn, otherwise image plot is created.
<code>auto.layout</code>	if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>col</code>	color used to draw the contours or images.
<code>lwd</code>	line width.

main	main title of the plot.
xylab	optional character vector of the length equal to the number of margins with labels used for x and y axes on the plots.
...	additional arguments passed to the plot function.

Value

invisible(x)

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[NMixPredDensJoint2](#), [NMixMCMC](#).

Examples

```
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files Galaxy.pdf, Faithful.pdf, Tandmob.pdf
```

```
plot.NMixPredDensMarg
```

Plot computed marginal predictive densities

Description

This is a basic plotting tool to visualize computed marginal predictive densities, see [NMixPredDensMarg](#).

Usage

```
## S3 method for class 'NMixPredDensMarg':
plot(x, K=0, auto.layout=TRUE,
     type="l", col="darkblue", lty=1, lwd=1, main, xlab, ylab, ...)
```

Arguments

<code>x</code>	an object of class <code>NMixPredDensMarg</code> .
<code>K</code>	if equal to 0 then the overall predictive densities are plotted taken from the <code>dens</code> part of the object <code>x</code> . If higher than 0 then the predictive density conditioned by the value of <code>K</code> is plotted (taken from the <code>densK</code> part of the object <code>x</code>).
<code>auto.layout</code>	if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.
<code>type</code>	type of the plot.
<code>col</code>	color used to draw the lines.
<code>lty</code>	type of the line.
<code>lwd</code>	line width.
<code>main</code>	main title of the plot. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>xlab</code>	label for the x-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>ylab</code>	label for the y-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.
<code>...</code>	additional arguments passed to the <code>plot</code> function.

Value

`invisible(x)`

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

See Also

[NMixPredDensMarg](#), [NMixMCMC](#).

Examples

```
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files Galaxy.pdf, Faithful.pdf, Tandmob.pdf
```

rRotationMatrix *Random rotation matrix*

Description

Generate a random rotation matrix, i.e., a matrix $\mathbf{P} = (p_{i,j})_{i=1,\dots,p,j=1,\dots,p}$, which satisfies

- a) $\mathbf{P}\mathbf{P}' = \mathbf{I}$,
- b) $\mathbf{P}'\mathbf{P} = \mathbf{I}$,
- c) $\det(\mathbf{P}) = 1$.

Usage

```
rRotationMatrix(n, dim)
```

Arguments

n number of matrices to generate.
dim dimension of a generated matrix/matrices.

Details

For $\dim = 2$, $p_{2,1}$ ($\sim (\theta)$) is generated from $\text{Unif}(0, 1)$ and the rest computed as follows: $p_{1,1} = p_{2,2} = \sqrt{1 - p_{2,1}^2}$ ($\cos(\theta)$) and $p_{1,2} = -p_{2,1}$ ($-\sin(\theta)$).

For $\dim > 2$, the matrix \mathbf{P} is generated in the following steps:

- 1) Generate a $p \times p$ matrix \mathbf{A} with independent $\text{Unif}(0, 1)$ elements and check whether \mathbf{A} is of full rank p .
- 2) Computes a QR decomposition of \mathbf{A} , i.e., $\mathbf{A} = \mathbf{Q}\mathbf{R}$ where \mathbf{Q} satisfies $\mathbf{Q}\mathbf{Q}' = \mathbf{I}$, $\mathbf{Q}'\mathbf{Q} = \mathbf{I}$, $\det(\mathbf{Q}) = (-1)^{p+1}$, and columns of \mathbf{Q} spans the linear space generated by the columns of \mathbf{A} .
- 3) For odd \dim , return matrix \mathbf{Q} . For even \dim , return corrected matrix \mathbf{Q} to satisfy the determinant condition.

Value

For $n=1$, a matrix is returned.

For $n>1$, a list of matrices is returned.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Golub, G. H. and Van Loan, C. F. (1996, Sec. 5.1). *Matrix Computations. Third Edition*. Baltimore: The Johns Hopkins University Press.

Examples

```

P <- rRotationMatrix(n=1, dim=5)
print(P)
round(P %*% t(P), 10)
round(t(P) %*% P, 10)
det(P)

n <- 10
P <- rRotationMatrix(n=n, dim=5)
for (i in 1:3){
  cat(paste("*** i=", i, "\n", sep=""))
  print(P[[i]])
  print(round(P[[i]] %*% t(P[[i]]), 10))
  print(round(t(P[[i]]) %*% P[[i]], 10))
  print(det(P[[i]]))
}

```

rSamplePair

Sample a pair (with replacement)

Description

For given K , the function samples with replacement from a uniform distribution on a set of pairs $(1, 2), (1, 3), \dots, (1, K), (2, 3), \dots, (2, K), \dots, (K - 1, K)$.

Usage

```
rSamplePair(n, K)
```

Arguments

n number of pairs to sample.
 K a numeric value which determines K (see above).

Value

A two-component numeric vector for $n = 2$ or a matrix with 2 columns with sampled pairs in rows for $n > 2$.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

Examples

```
rSamplePair(n=1, K=2)
rSamplePair(n=10, K=2)

rSamplePair(n=1, K=3)
rSamplePair(n=10, K=3)

rSamplePair(n=1, K=4)
rSamplePair(n=10, K=4)
```

SP2Rect	<i>Creation of a rectangular symmetric matrix from its lower triangle (symmetric packed format)</i>
---------	---

Description

It creates a symmetric matrix from its lower triangle.

Usage

```
SP2Rect(LT, dim)
```

Arguments

LT	a numeric vector with the lower triangle (stored columnwise) of the matrix we want to reconstruct.
dim	number of rows and columns of a resulting matrix.

Value

A matrix.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

Examples

```
SP2Rect(3, dim=1)
SP2Rect(c(1, 0.5, 2), dim=2)
SP2Rect(c(1, 0.5, 0.25, 2, -0.5, 3), dim=3)
```

Tandmob

*Signal Tandmobiel data***Description**

This is the dataset resulting from a longitudinal prospective dental study performed in Flanders (North of Belgium) in 1996 – 2001. The cohort of 4 468 randomly sampled children who attended the first year of the basic school at the beginning of the study was annually dental examined by one of 16 trained dentists. The original dataset consists thus of at most 6 dental observations for each child.

The dataset presented here contains mainly the information on the emergence and caries times summarized in the interval-censored observations. Some baseline covariates are also included here. This is a copy of `tandmob2` data in the package `bayesSurv`.

For more detail on the design of the study see Vanobbergen et al. (2000).

This data set was used in the analyses presented in Komárek et al. (2005), in Lesaffre, Komárek, and Declerck (2005) and in Komárek and Lesaffre (2007).

IMPORTANT NOTICE: It is possible to use these data for your research work under the condition that each manuscript is first approved by

Prof. Emmanuel Lesaffre

Biostatistical Centre Katholieke Universiteit Leuven

Kapucijnenvoer 35

B-3000 Leuven

Belgium

`<emmanuel.lesaffre@med.kuleuven.be>`

Usage

```
data(Tandmob)
```

Format

a~data frame with 4 430 rows (38 sampled children did not come to any of the designed dental examinations) and the following variables

IDNR identification number of a child

GENDER character *boy* or *girl*

GENDERNum numeric, 0 = *boy*, 1 = *girl*

DOB character, date of birth in the format DDmmmYY

PROVINCE factor, code of the province with

0 = Antwerpen

1 = Vlaams Brabant

2 = Limburg

3 = Oost Vlaanderen

4 = West Vlaanderen

EDUC factor, code of the educational system with

- 0 = Free
- 1 = Community school
- 2 = Province/council school

STARTBR factor, code indicating the starting age of brushing the teeth (as reported by parents) with

- 1 = [0, 1] years
- 2 = (1, 2] years
- 3 = (2, 3] years
- 4 = (3, 4] years
- 5 = (4, 5] years
- 6 = later than at the age of 5

FLUOR binary covariate, 0 = no, 1 = yes. This is the covariate *fluorosis* used in the paper Komárek et al. (2005).

BAD.xx binary, indicator whether a deciduous tooth xx was removed because of orthodontical reasons or not.

xx takes values 53, 63, 73, 83 (deciduous lateral canines), 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

EBEG.xx lower limit of the emergence (in years of age) of the permanent tooth xx. NA if the emergence was left-censored.

xx takes values 11, 21, 31, 41 (permanent incisors), 12, 22, 32, 42 (permanent central canines), 13, 23, 33, 43 (permanent lateral canines), 14, 24, 34, 44 (permanent first premolars), 15, 25, 35, 45 (permanent second premolars), 16, 26, 36, 46 (permanent first molars), 17, 27, 37, 47 (permanent second molars).

EEND.xx upper limit of the emergence (in years of age) of the permanent tooth xx. NA if the emergence was right-censored.

xx takes values as for the variable EBEG . xx.

FBEG.xx lower limit for the caries time (in years of age, 'F' stands for 'failure') of the permanent tooth xx. NA if the caries time was left-censored.

xx takes values as for the variable EBEG . xx.

FEND.xx upper limit for the caries time (in years of age, 'F' stands for 'failure') of the permanent tooth xx. NA if the caries time was right-censored.

xx takes values as for the variable EBEG . xx.

Unfortunately, for all teeth except 16, 26, 36 and 46 almost all the caries times are right-censored. For teeth 16, 26, 36, 46, the amount of right-censoring is only about 25%.

Txx.DMF indicator whether a deciduous tooth xx was *decayed* or *missing due to caries* or *filled* on at most the last examination before the first examination when the emergence of the permanent successor was recorded.

xx takes values 53, 63, 73, 83 (deciduous lateral incisors), 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

Txx.CAR indicator whether a~deciduous tooth xx was removed due to the orthodontical reasons or decayed on at most the last examination before the first examination when the emergence of the permanent successor was recorded.

Source

Biostatistical Centre, Katholieke Universiteit Leuven, Kapucijnenvoer 35, 3000 Leuven, Belgium
 URL: <http://med.kuleuven.be/biostat/>

Data collection was supported by Unilever, Belgium. The Signal Tandmobiel project comprises the following partners: D. Declerck (Dental School, Catholic University Leuven), L. Martens (Dental School, University Ghent), J. Vanobbergen (Oral Health Promotion and Prevention, Flemish Dental Association), P. Bottenberg (Dental School, University Brussels), E. Lesaffre (Biostatistical Centre, Catholic University Leuven), K. Hoppenbrouwers (Youth Health Department, Catholic University Leuven; Flemish Association for Youth Health Care).

References

- Komárek, A., Lesaffre, E., Härkänen, T., Declerck, D., and Virtanen, J. I. (2005). A Bayesian analysis of multivariate doubly-interval-censored dental data. *Biostatistics*, **6**, 145–155.
- Komárek, A. and Lesaffre, E. (2007). Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution. *Statistica Sinica*, **17**, 549–569.
- Lesaffre, E., Komárek, A., and Declerck, D. (2005). An overview of methods for interval-censored data with an emphasis on applications in dentistry. *Statistical Methods in Medical Research*, **14**, 539–552.
- Vanobbergen, J., Martens, L., Lesaffre, E., and Declerck, D. (2000). The Signal-Tandmobiel project – a longitudinal intervention health promotion study in Flanders (Belgium): baseline and first year results. *European Journal of Paediatric Dentistry*, **2**, 87–96.

See Also

[tandmob2](#)

Examples

```
data(Tandmob)
summary(Tandmob)
```

TandmobEmer

Signal Tandmobiel data - emergence times

Description

This is a part of the [Tandmob](#) data containing only emergence times and some baseline covariates. Here, all left-censored emergence times have been changed into interval-censored with the lower limit of the intervals equal to 5 years of age (clinically minimal time before which the permanent teeth hardly emerge). Also censoring indicators are added to be able to use the data directly with the [NMixMCMC](#) function.

IMPORTANT NOTICE: It is possible to use these data for your research work under the condition that each manuscript is first approved by Prof. Emmanuel Lesaffre

Biostatistical Centre Katholieke Universiteit Leuven
 Kapucijnenvoer 35
 B-3000 Leuven
 Belgium
 <emmanuel.lesaffre@med.kuleuven.be>

Usage

```
data(TandmobEmer)
```

Format

a~data frame with 4 430 rows and the following variables

IDNR identification number of a child

GENDER character *boy* or *girl*

GENDERNUM numeric, 0 = *boy*, 1 = *girl*

DOB character, date of birth in the format DDmmmYY

PROVINCE factor, code of the province with

0 = Antwerpen

1 = Vlaams Brabant

2 = Limburg

3 = Oost Vlaanderen

4 = West Vlaanderen

EDUC factor, code of the educational system with

0 = Free

1 = Community school

2 = Province/council school

STARTBR factor, code indicating the starting age of brushing the teeth (as reported by parents) with

1 = [0, 1] years

2 = (1, 2] years

3 = (2, 3] years

4 = (3, 4] years

5 = (4, 5] years

6 = later than at the age of 5

EBEG.xx lower limit of the emergence (in years of age) of the permanent tooth xx. It is equal to 5 if the emergence was originally left-censored.

xx takes values 11, 21, 31, 41 (permanent incisors), 12, 22, 32, 42 (permanent central canines), 13, 23, 33, 43 (permanent lateral canines), 14, 24, 34, 44 (permanent first premolars), 15, 25, 35, 45 (permanent second premolars), 16, 26, 36, 46 (permanent first molars), 17, 27, 37, 47 (permanent second molars).

EEND.xx upper limit of the emergence (in years of age) of the permanent tooth xx. NA if the emergence was right-censored.

xx takes values as for the variable EBEG . xx.

CENSOR.xx censoring indicator for the emergence. It is equal to 3 for interval-censored times and equal to 0 for right-censored times.

xx takes values as for the variable EBEG .xx.

Source

Biostatistical Centre, Katholieke Universiteit Leuven, Kapucijnenvoer 35, 3000 Leuven, Belgium

URL: <http://med.kuleuven.be/biostat/>

Data collection was supported by Unilever, Belgium. The Signal Tandmobiel project comprises the following partners: D. Declerck (Dental School, Catholic University Leuven), L. Martens (Dental School, University Ghent), J. Vanobbergen (Oral Health Promotion and Prevention, Flemish Dental Association), P. Bottenberg (Dental School, University Brussels), E. Lesaffre (Biostatistical Centre, Catholic University Leuven), K. Hoppenbrouwers (Youth Health Department, Catholic University Leuven; Flemish Association for Youth Health Care).

References

Komárek, A. A new R package for Bayesian estimation of multivariate normal mixtures allowing for selection of the number of components and interval-censored data. *Computational Statistics and Data Analysis*. To appear.

Vanobbergen, J., Martens, L., Lesaffre, E., and Declerck, D. (2000). The Signal-Tandmobiel project – a longitudinal intervention health promotion study in Flanders (Belgium): baseline and first year results. *European Journal of Paediatric Dentistry*, **2**, 87–96.

See Also

[Tandmob](#)

Examples

```
data(TandmobEmer)
summary(TandmobEmer)
```

TMVN

Truncated multivariate normal distribution

Description

Random generation for the truncated multivariate normal distribution. The mean and covariance matrix of the original multivariate normal distribution are `mean` and `Sigma`. Truncation limits are given by `a`, `b`, type of truncation is given by `trunc`.

This function uses a Gibbs algorithm to produce a Markov chain whose stationary distribution is the targeted truncated multivariate normal distribution, see Geweke (1991) for more details. Be aware that the sampled values are not i.i.d.!

Usage

```
rTMVN(n, mean=c(0, 0), Sigma=diag(2), a, b, trunc, xinit)
```

Arguments

<code>mean</code>	a numeric vector of the mean of the original multivariate normal distribution.
<code>Sigma</code>	covariance matrix of the original multivariate normal distribution.
<code>a</code>	a numeric vector of the same length as <code>mean</code> of truncation limits 1.
<code>b</code>	a numeric vector of the same length as <code>mean</code> of truncation limits 2.
<code>trunc</code>	a numeric vector of the same length as <code>mean</code> describing the type of truncation in each margin. <code>trunc=0</code> normal distribution is truncated on the interval (a, ∞) . Value of b is ignored. <code>trunc=1</code> degenerated normal distribution, all values are with probability 1 equal to a , b is ignored. <code>trunc=2</code> normal distribution is truncated on the interval $(-\infty, a)$. Value of b is ignored. <code>trunc=3</code> normal distribution is truncated on the interval (a, b) . <code>trunc=4</code> there is no truncation, values of a and b are ignored. If <code>trunc</code> is not given, it is assumed that it is equal to 4. Note that a , b and <code>trunc</code> must have the same length, with exception that b does not have to be supplied if all <code>trunc</code> values 0, 1, 2 or 4.
<code>xinit</code>	a numeric vector of the same length as <code>mean</code> with the initial value for the Gibbs sampler. If it is not supplied, the function determines itself the initial value.
<code>n</code>	number of observations to be sampled.

Value

A matrix with the sampled values (Markov chain) in rows.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Geweke, J. (1991). Efficient simulation from the multivariate normal and Student-t distributions subject to linear constraints and the evaluation of constraint probabilities. *Computer Sciences and Statistics*, **23**, 571–578.

See Also

[rTNorm](#).

Examples

```

set.seed(1977)

exam2 <- function(n, mu, sigma, rho, a, b, trunc)
{
  Sigma <- matrix(c(sigma[1]^2, rho*sigma[1]*sigma[2], rho*sigma[1]*sigma[2], sigma[2]^2), nrow=2, byrow=TRUE)
  x <- rTMVN(n, mean=mu, Sigma=Sigma, a=a, b=b, trunc=trunc)
  x1.gr <- seq(mu[1]-3.5*sigma[1], mu[1]+3.5*sigma[1], length=100)
  x2.gr <- seq(mu[2]-3.5*sigma[2], mu[2]+3.5*sigma[2], length=100)
  z <- cbind(rep(x1.gr, 100), rep(x2.gr, each=100))
  dens.z <- matrix(dMVN(z, mean=mu, Sigma=Sigma), ncol=100)

  MEAN <- round(apply(x, 2, mean), 3)
  SIGMA <- var(x)
  SD <- sqrt(diag(SIGMA))
  RHO <- round(SIGMA[1,2]/(SD[1]*SD[2]), 3)
  SD <- round(SD, 3)

  layout(matrix(c(0,1,1,0, 2,2,3,3), nrow=2, byrow=TRUE))
  contour(x1.gr, x2.gr, dens.z, col="darkblue", xlab="x[1]", ylab="x[2]")
  points(x[,1], x[,2], col="red")
  title(sub=paste("Sample mean = ", MEAN[1], ", ", MEAN[2], ", Sample SD = ", SD[1], ", ", SD[2], ", ", RHO))
  plot(1:n, x[,1], type="l", xlab="Iteration", ylab="x[1]", col="darkgreen")
  plot(1:n, x[,2], type="l", xlab="Iteration", ylab="x[2]", col="darkgreen")

  return(x)
}

x1 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0, a=c(-6, -9), b=c(4, 11), trunc=c(0, 2))
x2 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-6, -9), b=c(4, 11), trunc=c(0, 2))
x3 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-100, -100), b=c(100, 100), trunc=c(0, 2))
x4 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=-0.7, a=c(-6, -9), b=c(4, 11), trunc=c(0, 2))
x5 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=-0.9, a=c(-6, -9), b=c(4, 11), trunc=c(0, 2))

x6 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(0, 0), trunc=c(0, 2))
x7 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-1, 1), trunc=c(0, 2))
x8 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-1, 1), trunc=c(1, 2))
x9 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-1.5, 0.5), b=c(-0.5, 1.5), trunc=c(0, 2))
x10 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(-1.5, 0.5), b=c(-0.5, 1.5), trunc=c(0, 2))

```

Description

Random generation for the truncated normal distribution. The mean and standard deviation of the original normal distribution are mean and sd. Truncation limits are given by a, b, type of truncation is given by trunc.

Usage

```
rTNorm(n, mean=0, sd=1, a, b, trunc)
```

Arguments

mean	mean (if common for all observations) or a vector of length n of means.
sd	standard deviation (if common for all observations) or a vector of length n of standard deviations. Note that <code>mean</code> and <code>sd</code> must have the same length, either 1 or n .
a	truncation limit 1 (if common for all observations) or a vector of length n of truncation limits 1.
b	truncation limit 2 (if common for all observations) or a vector of length n of truncation limits 2.
trunc	type of truncation (if common for all observations) or a vector of length n of types of truncation trunc=0 normal distribution is truncated on the interval (a, ∞) .. Value of b is ignored. trunc=1 degenerated normal distribution, all values are with probability 1 equal to a , b is ignored. trunc=2 normal distribution is truncated on the interval $(-\infty, a)$. Value of b is ignored. trunc=3 normal distribution is truncated on the interval (a, b) . trunc=4 there is no truncation, values of a and b are ignored. If <code>trunc</code> is not given, it is assumed that it is equal to 4. Note that a , b and <code>trunc</code> must have the same length, either 1 or n with exception that b does not have to be supplied if <code>trunc</code> is 0, 1, 2 or 4.
n	number of observations to be sampled.

Value

A numeric vector with sampled values.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

Geweke, J. (1991). Efficient simulation from the multivariate normal and Student-t distributions subject to linear constraints and the evaluation of constraint probabilities. *Computer Sciences and Statistics*, **23**, 571–578.

See Also

[rnorm](#), [rTMVN](#).

Examples

```

set.seed(1977)

### Not truncated normal distribution
x1 <- rTNorm(1000, mean=10, sd=3)
c(mean(x1), sd(x1), range(x1))

### Truncation from left only
x2 <- rTNorm(1000, mean=10, sd=3, a=7, trunc=0)
c(mean(x2), sd(x2), range(x2))

### Degenerated normal distribution
x6 <- rTNorm(1000, mean=10, sd=3, a=13, trunc=1)
c(mean(x6), sd(x6), range(x6))

### Truncation from right only
x3 <- rTNorm(1000, mean=10, sd=3, a=13, trunc=2)
c(mean(x3), sd(x3), range(x3))

### Truncation from both sides
x4 <- rTNorm(1000, mean=10, sd=3, a=7, b=13, trunc=3)
c(mean(x4), sd(x4), range(x4))

x5 <- rTNorm(1000, mean=10, sd=3, a=5.5, b=14.5, trunc=3)
c(mean(x5), sd(x5), range(x5))

oldPar <- par(mfrow=c(2, 3))
hist(x1, main="N(10, 3^2)")
hist(x2, main="TN(10, 3^2, 7, Infy)")
hist(x6, main="TN(10, 3^2, 13, 13)")
hist(x3, main="TN(10, 3^2, -Infy, 13)")
hist(x4, main="TN(10, 3^2, 7, 13)")
hist(x5, main="TN(10, 3^2, 5.5, 14.5)")
par(oldPar)

### Different truncation limits
n <- 1000
a <- rnorm(n, -2, 1)
b <- a + rgamma(n, 1, 1)
trunc <- rep(c(0, 1, 2, 3, 4), each=n/5)
x7 <- rTNorm(n, mean=1, sd=2, a=a, b=b, trunc=trunc)
cbind(trunc, a, x7)[1:10,]
sum(x7[1:(n/5)] > a[1:(n/5)])      ## must be equal to n/5

cbind(trunc, a, x7)[201:210,]
sum(x7[(n/5+1):(2*n/5)] == a[(n/5+1):(2*n/5)])      ## must be equal to n/5

cbind(trunc, x7, a)[401:410,]
sum(x7[(2*n/5+1):(3*n/5)] < a[(2*n/5+1):(3*n/5)])      ## must be equal to n/5

cbind(trunc, a, x7, b)[601:610,]
sum(x7[(3*n/5+1):(4*n/5)] > a[(3*n/5+1):(4*n/5)])      ## must be equal to n/5

```

```

sum(x7[(3*n/5+1):(4*n/5)] < b[(3*n/5+1):(4*n/5)])      ## must be equal to n/5

cbind(trunc, x7)[801:810,]

### Different moments and truncation limits
n <- 1000
mu <- rnorm(n, 1, 0.2)
sigma <- 0.5 + rgamma(n, 1, 1)
a <- rnorm(n, -2, 1)
b <- a + rgamma(n, 1, 1)
trunc <- rep(c(0, 1, 2, 3, 4), each=n/5)
x8 <- rTNorm(n, mean=1, sd=2, a=a, b=b, trunc=trunc)

### Truncation from left only
### (extreme cases when we truncate to the area
### where the original normal distribution has
### almost zero probability)
x2b <- rTNorm(1000, mean=0, sd=1, a=7.9, trunc=0)
c(mean(x2b), sd(x2b), range(x2b))

x2c <- rTNorm(1000, mean=1, sd=2, a=16, trunc=0)
c(mean(x2c), sd(x2c), range(x2c))

### Truncation from right only (extreme cases)
x3b <- rTNorm(1000, mean=0, sd=1, a=-7.9, trunc=2)
c(mean(x3b), sd(x3b), range(x3b))

x3c <- rTNorm(1000, mean=1, sd=2, a=-13, trunc=2)
c(mean(x3c), sd(x3c), range(x3c))

### Truncation from both sides (extreme cases)
x4b <- rTNorm(1000, mean=0, sd=1, a=-9, b=-7.9, trunc=3)
c(mean(x4b), sd(x4b), range(x4b))

x4c <- rTNorm(1000, mean=0, sd=1, a=7.9, b=9, trunc=3)
c(mean(x4c), sd(x4c), range(x4c))

```

Wishart

Wishart distribution

Description

Wishart distribution

$\text{Wishart}(\nu, \mathbf{S}),$

where ν are degrees of freedom of the Wishart distribution and \mathbf{S} is its scale matrix. The same parametrization as in Gelman (2004) is assumed, that is, if $\mathbf{W} \sim \text{Wishart}(\nu, \mathbf{S})$ then

$$E(\mathbf{W}) = \nu \mathbf{S}.$$

Usage

```
dWishart(W, df, S, log=FALSE)
```

```
rWishart(n, df, S)
```

Arguments

<code>W</code>	Either a matrix with the same number of rows and columns as <code>S</code> (1 point sampled from the Wishart distribution) or a matrix with <code>ncol</code> equal to <code>ncol*(ncol+1)/2</code> and <code>n</code> rows (<code>n</code> points sampled from the Wishart distribution for which only lower triangles are given in rows of the matrix <code>W</code>).
<code>n</code>	number of observations to be sampled.
<code>df</code>	degrees of freedom of the Wishart distribution.
<code>S</code>	scale matrix of the Wishart distribution.
<code>log</code>	logical; if <code>TRUE</code> , log-density is computed

Details

The density of the Wishart distribution is the following

$$f(\mathbf{W}) = \left(2^{\nu p/2} \pi^{p(p-1)/4} \prod_{i=1}^p \Gamma\left(\frac{\nu+1-i}{2}\right) \right)^{-1} |\mathbf{S}|^{-\nu/2} |\mathbf{W}|^{(\nu-p-1)/2} \exp\left(-\frac{1}{2}\text{tr}(\mathbf{S}^{-1}\mathbf{W})\right),$$

where p is number of rows and columns of the matrix \mathbf{W} .

In the univariate case, $\text{Wishart}(\nu, S)$ is the same as $\text{Gamma}(\nu/2, 1/(2S))$.

Generation of random numbers is performed by the algorithm described in Ripley (1987, pp. 99).

Value

Some objects.

Value for dWishart

A numeric vector with evaluated (log-)density.

Value for rWishart

If `n` equals 1 then a sampled symmetric matrix `W` is returned.

If `n` > 1 then a matrix with sampled points (lower triangles of \mathbf{W}) in rows is returned.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

References

- Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis, Second edition*. Boca Raton: Chapman and Hall/CRC.
- Ripley, B. D. (1987). *Stochastic Simulation*. New York: John Wiley and Sons.

Examples

```

set.seed(1977)
### The same as gamma(shape=df/2, rate=1/(2*S))
df <- 1
S <- 3

w <- rWishart(n=1000, df=df, S=S)
mean(w)      ## should be close to df*S
var(w)       ## should be close to 2*df*S^2

dWishart(w[1], df=df, S=S)
dWishart(w[1], df=df, S=S, log=TRUE)

dens.w <- dWishart(w, df=df, S=S)
dens.wG <- dgamma(w, shape=df/2, rate=1/(2*S))
rbind(dens.w[1:10], dens.wG[1:10])

ldens.w <- dWishart(w, df=df, S=S, log=TRUE)
ldens.wG <- dgamma(w, shape=df/2, rate=1/(2*S), log=TRUE)
rbind(ldens.w[1:10], ldens.wG[1:10])

### Bivariate Wishart
df <- 2
S <- matrix(c(1,3,3,13), nrow=2)

print(w2a <- rWishart(n=1, df=df, S=S))
dWishart(w2a, df=df, S=S)

w2 <- rWishart(n=1000, df=df, S=S)
print(w2[1:10,])
apply(w2, 2, mean)      ## should be close to df*S
(df*S)[lower.tri(S, diag=TRUE)]

dens.w2 <- dWishart(w2, df=df, S=S)
ldens.w2 <- dWishart(w2, df=df, S=S, log=TRUE)
cbind(w2[1:10,], data.frame(Density=dens.w2[1:10], Log.Density=ldens.w2[1:10]))

### Trivariate Wishart
df <- 3.5
S <- matrix(c(1,2,3,2,20,26,3,26,70), nrow=3)

print(w3a <- rWishart(n=1, df=df, S=S))
dWishart(w3a, df=df, S=S)

w3 <- rWishart(n=1000, df=df, S=S)
print(w3[1:10,])

```

```
apply(w3, 2, mean)          ## should be close to df*S
(df*S)[lower.tri(S, diag=TRUE)]

dens.w3 <- dWishart(w3, df=df, S=S)
ldens.w3 <- dWishart(w3, df=df, S=S, log=TRUE)
cbind(w3[1:10,], data.frame(Density=dens.w3[1:10], Log.Density=ldens.w3[1:10]))
```

Y2T

Transform fitted distribution of $Y=\log(T)$ into distribution of T

Description

This method transforms fitted distribution of $Y = \log(T)$ into distribution of T .

Usage

```
Y2T(x, ...)
```

```
## S3 method for class 'NMixPredDensMarg':
Y2T(x, ...)
```

```
## S3 method for class 'NMixPredCDFMarg':
Y2T(x, ...)
```

Arguments

`x` an object of appropriate class.
`...` optional additional arguments.

Value

An object of the same class as argument `x`.

Author(s)

Arnošt Komárek <arnost.komarek[AT]mff.cuni.cz>

See Also

[NMixPredDensMarg](#), [NMixPredCDFMarg](#).

Index

*Topic **algebra**

MatMPpinv, 14
MatSqrt, 16

*Topic **array**

MatMPpinv, 14
MatSqrt, 16
rRotationMatrix, 60
SP2Rect, 62

*Topic **datasets**

Acidity, 1
Enzyme, 5
Faithful, 6
Galaxy, 7
Tandmob, 63
TandmobEmer, 65

*Topic **distribution**

MVN, 17
MVNmixture, 20
rRotationMatrix, 60
rSamplePair, 61
TMVN, 67
TNorm, 69
Wishart, 72

*Topic **dplot**

autolayout, 2
NMixPlugCondDensJoint2, 33
NMixPlugCondDensMarg, 35
NMixPlugDensJoint2, 36
NMixPlugDensMarg, 38
NMixPredCDFMarg, 39
NMixPredCondDensJoint2, 41
NMixPredCondDensMarg, 43
NMixPredDensJoint2, 45
NMixPredDensMarg, 47
plot.NMixPlugCondDensJoint2,
50
plot.NMixPlugCondDensMarg, 51
plot.NMixPlugDensJoint2, 52
plot.NMixPlugDensMarg, 53

plot.NMixPredCDFMarg, 54
plot.NMixPredCondDensJoint2,
55
plot.NMixPredCondDensMarg, 56
plot.NMixPredDensJoint2, 57
plot.NMixPredDensMarg, 58

*Topic **models**

GLMM_longitClust, 8
GLMM_MCMC, 10

*Topic **multivariate**

BLA, 3
GLMM_longitClust, 8
GLMM_MCMC, 10
MVN, 17
MVNmixture, 20
NMixChainsDerived, 23
NMixClust, 24
NMixMCMC, 24
NMixPlugCondDensJoint2, 33
NMixPlugCondDensMarg, 35
NMixPlugDensJoint2, 36
NMixPlugDensMarg, 38
NMixPredCDFMarg, 39
NMixPredCondDensJoint2, 41
NMixPredCondDensMarg, 43
NMixPredDensJoint2, 45
NMixPredDensMarg, 47
NMixSummComp, 49
rRotationMatrix, 60
TMVN, 67
Wishart, 72

*Topic **smooth**

NMixChainsDerived, 23
NMixClust, 24
NMixPlugCondDensJoint2, 33
NMixPlugCondDensMarg, 35
NMixPlugDensJoint2, 36
NMixPlugDensMarg, 38
NMixPredCDFMarg, 39

- NMixturePredCondDensJoint2, 41
- NMixturePredCondDensMarg, 43
- NMixturePredDensJoint2, 45
- NMixturePredDensMarg, 47
- NMixtureSummComp, 49
- Y2T, 75
- *Topic survival**
 - NMixtureMCMC, 24
- Acidity, 1
- autolayout, 2
- BLA, 3
- contour, 38, 46, 50, 52, 55, 57
- Dirichlet, 4
- dMVN (MVN), 17
- dMVNmixture (MVNmixture), 20
- dMVNmixture2 (MVNmixture), 20
- dnorm, 18, 21
- dWishart (Wishart), 72
- Enzyme, 5
- Faithful, 6
- faithful, 7
- galaxies, 7, 8
- Galaxy, 7
- geyser, 7
- GLMM_longitClust, 8
- GLMM_MCMC, 8, 9, 10, 35, 36, 38, 39, 41, 43, 45, 47, 49
- image, 38, 46, 50, 55, 57
- layout, 2
- lmer, 11
- MatMPpinv, 14
- MatSqrt, 16
- MVN, 17, 21
- MVNmixture, 19
- Mvnorm, 18, 21
- NMixtureChainsDerived, 23
- NMixtureClust, 24
- NMixtureMCMC, 11, 14, 23, 24, 24, 33, 35, 36, 38, 39, 41, 43, 45–50, 52, 53, 55–59, 65
- NMixturePlugCondDensJoint2, 33, 50
- NMixturePlugCondDensMarg, 35, 51, 52
- NMixturePlugDensJoint2, 36, 52
- NMixturePlugDensMarg, 38, 53
- NMixturePredCDFMarg, 39, 54, 55, 75
- NMixturePredCondDensJoint2, 35, 41, 55, 56
- NMixturePredCondDensMarg, 36, 43, 56, 57
- NMixturePredDensJoint2, 33, 38, 45, 49, 57, 58
- NMixturePredDensMarg, 33, 39, 47, 47, 49, 58, 59, 75
- NMixtureSummComp, 49
- par, 3
- plot.NMixturePlugCondDensJoint2, 35, 50
- plot.NMixturePlugCondDensMarg, 36, 51
- plot.NMixturePlugDensJoint2, 38, 52
- plot.NMixturePlugDensMarg, 39, 53
- plot.NMixturePredCDFMarg, 41, 54
- plot.NMixturePredCondDensJoint2, 43, 55
- plot.NMixturePredCondDensMarg, 45, 56
- plot.NMixturePredDensJoint2, 47, 57
- plot.NMixturePredDensMarg, 49, 58
- print.GLMM_MCMC (GLMM_MCMC), 10
- print.NMixtureMCMC (NMixtureMCMC), 24
- print.NMixtureMCMClist (NMixtureMCMC), 24
- rbeta, 5
- rcMVN (MVN), 17
- rDirichlet (Dirichlet), 4
- rMVN (MVN), 17
- rMVNmixture (MVNmixture), 20
- rMVNmixture2 (MVNmixture), 20
- rnorm, 70
- rRotationMatrix, 60
- rSamplePair, 61
- rTMVN, 70
- rTMVN (TMVN), 67
- rTNorm, 68
- rTNorm (TNorm), 69
- rWishart (Wishart), 72
- SP2Rect, 62
- Tandmob, 63, 65, 67
- tandmob2, 63, 65
- TandmobEmer, 65

TMVN, [67](#)

TNorm, [69](#)

Wishart, [72](#)

Y2T, [75](#)