Package ‘mixAK’

September 25, 2023

Version 5.6
Date 2023-09-25
Title Multivariate Normal Mixture Models and Mixtures of Generalized Linear Mixed Models Including Model Based Clustering
Depends R (>= 3.0.0), colorspace, lme4 (>= 1.0)
Imports graphics, stats, methods, splines, fastGHQuad, mnormt, parallel, coda
Suggests mvtnorm
Description Contains a mixture of statistical methods including the MCMC methods to analyze normal mixtures. Additionally, model based clustering methods are implemented to perform classification based on (multivariate) longitudinal (or otherwise correlated) data. The basis for such clustering is a mixture of multivariate generalized linear mixed models.
Encoding UTF-8
License GPL (>= 3)
URL http://msekce.karlin.mff.cuni.cz/~komarek/
NeedsCompilation yes
Author Arnošt Komárek [aut, cre] (<https://orcid.org/0000-0001-8778-3762>)
Maintainer Arnošt Komárek <arnost.komarek@mff.cuni.cz>
Repository CRAN
Date/Publication 2023-09-25 11:00:02 UTC

R topics documented:

  Acidity ................................................................. 3
  autolayout .......................................................... 4
  BLA ................................................................. 4
  BsBasis ............................................................... 6
  cbplot ................................................................. 7
  Dirichlet ............................................................ 9
  Enzyme ............................................................. 10
  Faithful .............................................................. 11
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fitted.GLMM_MCMC</td>
</tr>
<tr>
<td>Galaxy</td>
</tr>
<tr>
<td>generatePermutations</td>
</tr>
<tr>
<td>getProfiles</td>
</tr>
<tr>
<td>GLMM_longitDA</td>
</tr>
<tr>
<td>GLMM_longitDA2</td>
</tr>
<tr>
<td>GLMM_MCMC</td>
</tr>
<tr>
<td>MatMPpinv</td>
</tr>
<tr>
<td>MatSqrt</td>
</tr>
<tr>
<td>MVN</td>
</tr>
<tr>
<td>MVNmixture</td>
</tr>
<tr>
<td>MVT</td>
</tr>
<tr>
<td>NMixChainComp</td>
</tr>
<tr>
<td>NMixChainsDerived</td>
</tr>
<tr>
<td>NMixCluster</td>
</tr>
<tr>
<td>NMixEM</td>
</tr>
<tr>
<td>NMixMCMC</td>
</tr>
<tr>
<td>NMixPlugCondDensJoint2</td>
</tr>
<tr>
<td>NMixPlugCondDensMarg</td>
</tr>
<tr>
<td>NMixPlugDA</td>
</tr>
<tr>
<td>NMixPlugDensJoint2</td>
</tr>
<tr>
<td>NMixPlugDensMarg</td>
</tr>
<tr>
<td>NMixPredCDFMarg</td>
</tr>
<tr>
<td>NMixPredCondCDFMarg</td>
</tr>
<tr>
<td>NMixPredCondDensJoint2</td>
</tr>
<tr>
<td>NMixPredCondDensMarg</td>
</tr>
<tr>
<td>NMixPredDA</td>
</tr>
<tr>
<td>NMixPredDensJoint2</td>
</tr>
<tr>
<td>NMixPredDensMarg</td>
</tr>
<tr>
<td>NMixPseudoGOF</td>
</tr>
<tr>
<td>NMixRelabel</td>
</tr>
<tr>
<td>NMixSummComp</td>
</tr>
<tr>
<td>PBC910</td>
</tr>
<tr>
<td>PBCseq</td>
</tr>
<tr>
<td>plot.NMixPlugCondDensJoint2</td>
</tr>
<tr>
<td>plot.NMixPlugCondDensMarg</td>
</tr>
<tr>
<td>plot.NMixPlugDensJoint2</td>
</tr>
<tr>
<td>plot.NMixPlugDensMarg</td>
</tr>
<tr>
<td>plot.NMixPredCDFMarg</td>
</tr>
<tr>
<td>plot.NMixPredCondCDFMarg</td>
</tr>
<tr>
<td>plot.NMixPredCondDensJoint2</td>
</tr>
<tr>
<td>plot.NMixPredCondDensMarg</td>
</tr>
<tr>
<td>plot.NMixPredDensJoint2</td>
</tr>
<tr>
<td>plot.NMixPredDensMarg</td>
</tr>
<tr>
<td>plotProfiles</td>
</tr>
<tr>
<td>rRotationMatrix</td>
</tr>
<tr>
<td>rSamplePair</td>
</tr>
<tr>
<td>SimData</td>
</tr>
</tbody>
</table>
Acidity

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

Originally from http://www.stats.bris.ac.uk/~peter/mixdata/

References


Examples

data(Acidity)
supply(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@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 $X = (X_1, \ldots, X_p)'$ 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 $X$ given the remaining components of $X$.

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@mff.cuni.cz>

References


Examples

```r
##### X = (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)
```
BsBasis  B-spline basis

Description

It creates a B-spline basis based on a specific dataset. B-splines are assumed to have common boundary knots and equidistant inner knots.

Usage

BsBasis(degree, ninner, knotsBound, knots, intercept=FALSE, x, tgrid, Bname="B", plot=FALSE, lwd=1, col="blue", xlab="Time", ylab="B(t)", pch=16, cex.pch=1, knotcol="red")

Arguments

degree degree of the B-spline.
ninner number of inner knots.
knotsBound 2-component vector with boundary knots.
knots knots of the B-spline (including boundary ones). If knots is given ninner and knotsBound are ignored.
intercept logical, if FALSE, the first basis B-spline is removed from the B-spline basis and it is assumed that intercept is added to the statistical models used afterwards.
x a numeric vector to be used to create the B-spline basis.
tgrid if given then it is used to plot the basis.
Bname label for the created columns with the B-spline basis.
plot logical, if TRUE the B-spline basis is plotted.
lwd, col, xlab, ylab arguments passed to the plotting function.
pch, cex.pch plotting character and cex used to plot knots
knotcol color for knots on the plot with the B-spline basis.

Value

A matrix with the B-spline basis. Number of rows is equal to the length of x.
Additionally, the resulting matrix has attributes:
degree B-spline degree
intercept logical indicating the presence of the intercept B-spline
knots a numeric vector of knots
knotsInner a numeric vector of inner knots
knotsBound a numeric vector of boundary knots
cbplot

Description

This routine typically plots a function together with its confidence or credible bands. The credible band can be indicated either by additional lines or by a shaded region or by both.

Usage

cbplot(x, y, low, upp, type=c("l", "s"), band.type=c("ls", "s", "l"), add=FALSE, col="darkblue", lty=1, lwd=2,
          cbcol=col, cblty=4, cbld=wb, cscol=rainbow_hcl(1, start=180, end=180), slw=5,
          xlim, ylim, xlab, ylab, main="", sub="", cex.lab=1, cex.axis=1, ...)
cbplot

Arguments

x
a numeric vector with x coordinates corresponding to y, low, upp.

y
a numeric vector with y coordinates of the function to plot.

low
a numeric vector with y coordinates of the lower limit of the credible band.

upp
a numeric vector with y coordinates of the upper limit of the credible band.

type
argument with the same meaning as type in plot.default.

band.type
a character which specifies the graphical way to show the credible band, “ls” stands for line and shaded region, “s” stands for shaded region only and “l” stands for line only.

add
if TRUE then everything is added to the current plot.

col, lty, lwd
graphical parameters to draw the x-y line.

cbcoll, cblty, cblwd
graphical parameters to draw the x-low and x-upp lines.

scol, slwd
graphical parameters for the shaded region between the credible/confidence bounds.

xlim, ylim, xlab, ylab, main, sub, cex.lab, cex.axis
other graphical parameters.

...
additional arguments passed to the plot function.

Value

invisible(x)

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

Examples

### Artificial credible bands around the CDF's of N(100, 15*15)
### and N(80, 10*10)

iq <- seq(55, 145, length=100)
Fiq <- pnorm(iq, 100, 15)
low <- Fiq - 0.1
upp <- Fiq + 0.1

iq2 <- seq(35, 125, length=100)
Fiq2 <- pnorm(iq, 80, 10)
low2 <- Fiq2 - 0.1
upp2 <- Fiq2 + 0.1

cbplot(iq, Fiq, low, upp, xlim=c(35, 145))

cbplot(iq2, Fiq2, low2, upp2, add=TRUE, col="red4",
  scol=rainbow_hcl(1, start=20, end=20))
Dirichlet distribution

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

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@mff.cuni.cz>

References

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

Originally from http://www.stats.bris.ac.uk/~peter/mixdata/

References


Examples

data(Enzyme)
summary(Enzyme)
Faithful

Old Faithful Geyser Data

Description


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


See Also

geyser.

Examples

data(Faithful)
summary(Faithful)
fitted.GLMM_MCMC

Fitted profiles in the GLMM model

Description

It calculates fitted profiles in the (multivariate) GLMM with a normal mixture in the random effects distribution based on selected posterior summary statistic of the model parameters.

Usage

```r
## S3 method for class 'GLMM_MCMC'
fitted(object, x, z,
statistic=c("median", "mean", "Q1", "Q3", "2.5\%", "97.5\%"),
overall=FALSE, glmer=TRUE, nAGQ=100, ...)
```

Arguments

- `object` object of class `GLMM_MCMC`.
- `x` matrix or list of matrices (in the case of multiple responses) for “fixed effects” part of the model used in the calculation of fitted values.
- `z` matrix or list of matrices (in the case of multiple responses) for “random effects” part of the model used in the calculation of fitted values.
- `statistic` character which specifies the posterior summary statistic to be used to calculate fitted profiles. Default is the posterior median. It applies only to the overall fit.
- `overall` logical. If TRUE, fitted profiles based on posterior mean/median/Q1/Q3/2.5%/97.5% of the model parameters are computed. If FALSE, fitted profiles based on posterior means given mixture component are calculated. Note that this depends on used re-labelling of the mixture components and hence might be misleading if re-labelling is not successful!
- `glmer` a logical value. If TRUE, the real marginal means are calculated using Gaussian quadrature.
- `nAGQ` number of quadrature points used when glmer is TRUE.
- `...` possibly extra arguments. Nothing useful at this moment.

Value

A list (one component for each of multivariate responses from the model) with fitted values calculated using the x and z matrices. If overall is FALSE, these are then matrices with one column for each mixture component.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>
Galaxy

See Also

GLMM_MCMC.

Examples

### WILL BE ADDED.

<table>
<thead>
<tr>
<th>Galaxy</th>
<th>Velocities of distant galaxies</th>
</tr>
</thead>
</table>

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

Originally from http://www.stats.bris.ac.uk/~peter/mixdata/

References


See Also

galaxies.
getProfiles

**Examples**

data(Galaxy)
summary(Galaxy)

generatePermutations  *Generate all permutations of* \((1, \ldots, K)\)

**Description**

It generates a matrix containing all permutations of \((1, \ldots, K)\).

**Usage**

generatePermutations(K)

**Arguments**

\(K\)  integer value of \(K\).

**Value**

A matrix of dimension \(K! \times K\) with generated permutations in rows.

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**Examples**

generatePermutations(1)
generatePermutations(2)
generatePermutations(3)
generatePermutations(4)

generatePermutations  *Generate all permutations of* \((1, \ldots, K)\)

**getProfiles**  *Individual longitudinal profiles of a given variable*

**Description**

It creates a list with individual longitudinal profiles of a given variable.

**Usage**

getProfiles(t, y, id, data)
Arguments

t | a character string giving the name of the variable with “time”.
y | a character string giving the names of the responses variables to keep in the resulting object.
id | a character string giving the name of the variable which identifies subjects.
data | a data.frame with all the variables.

Value

A list of data.frames, one for each subject identified by id in the original data.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also

plotProfiles.

Examples

data(PBCseq, package="mixAK")
ip <- getProfiles(t="day", y=c("age", "lbili", "platelet", "spiders"),
id="id", data=PBCseq)
print(ip[[2]])
print(ip[[34]])

XLIM <- c(0, 910)
lcol1 <- rainbow_hcl(1, start=40, end=40)
oldPar <- par(mfrow=c(1, 3), bty="n")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="lbili", tvar="day",
xlab="Time (days)", col=lcol1, auto.layout=FALSE, main="Log(bilirubin)")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="platelet", tvar="day",
xlab="Time (days)", col=lcol1, auto.layout=FALSE, main="Platelet count")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="spiders", tvar="day",
xlab="Time (days)", col=lcol1, auto.layout=FALSE)
par(oldPar)

---

Description

The idea is that we fit (possibly different) GLMM’s for data in training groups using the function GLMM_MCMC and then use the fitted models for discrimination of new observations. For more details we refer to Komárek et al. (2010).

Currently, only continuous responses for which linear mixed models are assumed are allowed.
Usage

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

Arguments

mod       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.
w.prior   a vector with prior cluster weights. The length of this argument must be the same as the length of argument mod. Can also be given relatively, e.g., as c(1, 1) which means that both prior weights are equal to 1/2.
y         vector, matrix or data frame (see argument y of GLMM_MCMC function) with responses of objects that are to be clustered.
id        vector which determines clustered observations (see also argument y of GLMM_MCMC function).
time      vector which gives indices of observations within clusters. It appears (together with id) in the output as identifier of observations
x         see xz.common below.
z         see xz.common below.
xz.common a logical value.
          If TRUE then it is assumed that the X and Z matrices are the same for GLMM in each cluster. In that case, arguments x and z have the same structure as arguments x and z of GLMM_MCMC function.
          If FALSE then X and Z matrices for the GLMM may differ across clusters. In that case, arguments x and z are both lists of length equal to the number of clusters and each component of lists x and z has the same structure as arguments x and z of GLMM_MCMC function.
info      interval in which the function prints the progress of computation

Details

This function complements a paper Komárek et al. (2010).

Value

A list with the following components:

ident      ADD DESCRIPTION
marg       ADD DESCRIPTION
cond       ADD DESCRIPTION
ranef      ADD DESCRIPTION

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>
References

See Also
GLMM_MCMC, GLMM_longitDA2.

Examples
### WILL BE ADDED.

---

**GLMM_longitDA2**

*Discriminant analysis for longitudinal profiles based on fitted GLMM's*

---

**Description**

WILL BE ADDED.

**Usage**

```r
GLMM_longitDA2(mod, w.prior, y, id, x, z, xz.common = TRUE,
keep.comp.prob = FALSE, level = 0.95,
info, silent = FALSE)
```

**Arguments**

- **mod**
  - 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.

- **w.prior**
  - a vector with prior cluster weights. The length of this argument must be the same as the length of argument `mod`. Can also be given relatively, e.g., as `c(1, 1)` which means that both prior weights are equal to 1/2.

- **y**
  - vector, matrix or data frame (see argument `y` of `GLMM_MCMC` function) with responses of objects that are to be clustered.

- **id**
  - vector which determines clustered observations (see also argument `y` of `GLMM_MCMC` function).

- **x**
  - see `xz.common` below.

- **z**
  - see `xz.common` below.

- **xz.common**
  - a logical value.
  - If TRUE then it is assumed that the X and Z matrices are the same for GLMM in each cluster. In that case, arguments `x` and `z` have the same structure as arguments `x` and `z` of `GLMM_MCMC` function.
If FALSE then X and Z matrices for the GLMM may differ across clusters. In that case, arguments x and z are both lists of length equal to the number of clusters and each component of lists x and z has the same structure as arguments x and z of \texttt{GLMM_MCMC} function.

\begin{itemize}
  \item \texttt{keep.comp.prob} a logical value indicating whether the allocation probabilities should be kept for all MCMC iterations. This may ask for quite some memory but it is necessary if credible intervals etc. should be calculated for the component probabilities.
  \item \texttt{level} level of HPD credible intervals that are calculated for the component probabilities if \texttt{keep.comp.prob} is \texttt{TRUE}.
  \item \texttt{info} interval in which the function prints the progress of computation (unless \texttt{silent} is \texttt{TRUE}).
  \item \texttt{silent} logical value indicating whether to switch-off printing the information during calculations.
\end{itemize}

\section*{Details}

This function complements a paper being currently in preparation. \texttt{GLMM_longitDA2} differs in many aspects from \texttt{GLMM_longitDA2}!

\section*{Value}

A list with the following components:

\begin{itemize}
  \item \texttt{ADD} \texttt{ADD DESCRIPTION}
\end{itemize}

\section*{Author(s)}

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

\section*{See Also}

\texttt{GLMM_MCMC}, \texttt{GLMM_longitDA}.

\section*{Examples}

\texttt{######## WILL BE ADDED.}

\begin{table}
\centering
\begin{tabular}{ll}
\texttt{GLMM_MCMC} & \emph{MCMC estimation of a (multivariate) generalized linear mixed model with a normal mixture in the distribution of random effects} \\
\end{tabular}
\end{table}

\section*{Description}

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.
GLMM_MCMC

Usage

GLMM_MCMC(y, dist = "gaussian", id, x, z, random.intercept,
prior.alpha, init.alpha, init2.alpha,
scale.b, prior.b, init.b, init2.b,
prior.eps, init.eps, init2.eps,
nMCMC = c(burn = 10, keep = 10, thin = 1, info = 10),
tuneMCMC = list(alpha = 1, b = 1),
store = c(b = FALSE), PED = TRUE, keep.chains = TRUE,
dens.zero = 1e-300, parallel = FALSE, cltype, silent = FALSE)

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

## S3 method for class 'GLMM_MCMCList'
print(x, ...)

Arguments

y vector, matrix or data frame with responses. If y is vector then there is only
one response in the model. If y 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.
dist character (vector) which determines distribution (and a link function) for each
response variable. Possible values are: "gaussian" for gaussian (normal) distri-
bution (with identity link), "binomial(logit)" for binomial (0/1) distribution with
a logit link, "poisson(log)" for Poisson distribution with a log link. Single value
is recycled if necessary.
id vector which determines longitudinally or otherwise dependent observations. If
not given then it is assumed that there are no clusters and all observations of one
response are independent.
x 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 in included in all models. Use a character value "empty" as a
component of the list x if there are no covariates for a particular response.
z 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 random.intercept.
REMARK: For a particular response, matrices x and z may not have the same
columns. That is, matrix x includes covariates which are not involved among
random effects and matrix z includes covariates which are involved among ran-
dom effects (and implicitly among fixed effects as well).
random.intercept logical (vector) which determines for which responses random intercept should
be included.
prior.alpha 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 prior.alpha can have the components listed below.

mean a vector with prior means, defaults to zeros.
var a vector with prior variances, defaults to 10000 for all components.

init.alpha a numeric vector with initial values of fixed effects (not the means of random effects) for the first chain. A sensible value is determined using the maximum-likelihood fits (using lmer functions) and does not have to be given by the user.

init2.alpha a numeric vector with initial values of fixed effects for the second chain.

scale.b a list specifying how to scale the random effects during the MCMC. A sensible value is determined using the maximum-likelihood fits (using lmer functions) and does not have to be given by the user.

If the user wishes to influence the shift and scale constants, these are given as components of the list scale.b. The components are named:

shift see vignette PBCseq.pdf for details
scale see vignette PBCseq.pdf for details

prior.b 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).

The list prior.b can have the components listed below. Their meaning is analogous to the components of the same name of the argument prior of function NMixMCMC (see therein for details).

distribution a character string which specifies the assumed prior distribution for random effects. It can be either “normal” (multivaruate normal - default) or “MVT” (multivariate Student t distribution).
priorK a character string which specifies the type of the prior for $K$ (the number of mixture components).
priormuQ a character string which specifies the type of the prior for mixture means and mixture variances.

Kmax maximal number of mixture components.
lambda see vignette PBCseq.pdf for details
delta see vignette PBCseq.pdf for details
xi see vignette PBCseq.pdf for details
ce see vignette PBCseq.pdf for details
D see vignette PBCseq.pdf for details
zeta see vignette PBCseq.pdf for details
gD see vignette PBCseq.pdf for details
hD see vignette PBCseq.pdf for details
gdf shape parameter of the prior distribution for the degrees of freedom if the random effects are assumed to follow the MVT distribution
hdf rate parameter of the prior distribution for the degrees of freedom if the random effects are assumed to follow the MVT distribution

init.b a list with initial values of the first chain 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.
b  
K  
w  
mu  
Sigma  
Li  
gammaInv  
df  
r

init2.b  a list with initial values of the second chain for parameters related to the distribution of random effects and random effects themselves.

prior.eps  a list specifying prior distributions for error terms for continuous responses. The list prior.eps can have the components listed below. For all components, a sensible value leading to weakly informative prior distribution can be determined by the function.

zeta  see vignette PBCseq.pdf for details

g  see vignette PBCseq.pdf for details

h  see vignette PBCseq.pdf for details

init.eps  a list with initial values of the first chain for parameters related to the distribution of error terms of continuous responses. The list init.eps can have the components listed below. For all components, a sensible value can be determined by the function.

sigma  a numeric vector with the initial values for residual standard deviations for each continuous response.

gammaInv  a numeric vector with the initial values for the inverted components of the hyperparameter gamma for each continuous response.

init2.eps  a list with initial values of the second chain for parameters related to the distribution of error terms of continuous responses.

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.

tuneMCMC  a list with tuning scale parameters for proposal distribution of fixed and random effects. It is used only when there are some discrete response profiles. The components of the list have the following meaning:

alpha  scale parameters by which we multiply the proposal covariance matrix when updating the fixed effects pertaining to the discrete response profiles. There is one scale parameter for each DISCRETE profile. A single value is recycled if necessary.
b a scale parameter by which we multiply the proposal covariance matrix when updating the random effects. It is used only when there are some discrete response profiles in the model.

store logical vector indicating whether the chains of parameters should be stored. Its components may be named (ordering is then unimportant) as:

b if TRUE then the sampled values of random effects are stored. Defaults to FALSE.

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).

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.

dens.zero a small value used instead of zero when computing deviance related quantities.

parallel a logical value which indicates whether parallel computation (based on a package parallel) should be used when running two chains for the purpose of PED calculation.

cltype optional argument applicable if parallel is TRUE. If cltype is given, it is passed as the type argument into the call to makeCluster.

silent a logical value indicating whether the information on the MCMC progress is to be suppressed.

... additional arguments passed to the default print method.

Details

See accompanying papers (Komárek et al., 2010, Komárek and Komárková, 2013).

Value

An object of class GLMM_MCMClist (if PED argument is TRUE) or GLMM_MCMC (if PED argument is FALSE).

Object of class GLMM_MCMC

Object of class GLMM_MCMC can have the following components (some of them may be missing according to the context of the model):

iter index of the last iteration performed.

nMCMC used value of the argument nMCMC.

dist a character vector of length R corresponding to the dist argument.

R a two component vector giving the number of continuous responses and the number of discrete responses.

p a numeric vector of length R giving the number of non-intercept alpha parameters for each response.

q a numeric vector of length R giving the number of non-intercept random effects for each response.
**fixed.intercept** a logical vector of length R which indicates inclusion of fixed intercept for each response.

**random.intercept** a logical vector of length R which indicates inclusion of random intercept for each response.

**lalpha** length of the vector of fixed effects.

**dimb** dimension of the distribution of random effects.

**prior.alpha** a list containing the used value of the argument `prior.alpha`.

**prior.b** a list containing the used value of the argument `prior.b`.

**prior.eps** a list containing the used value of the argument `prior.eps`.

**init.alpha** a numeric vector with the used value of the argument `init.alpha`.

**init.b** a list containing the used value of the argument `init.b`.

**init.eps** a list containing the used value of the argument `init.eps`.

**state.first.alpha** a numeric vector with the first stored (after burn-in) value of fixed effects \( \alpha \).

**state.last.alpha** a numeric vector with the last sampled value of fixed effects \( \alpha \). It can be used as argument `init.alpha` to restart MCMC.

**state.first.b** a list with the first stored (after burn-in) values of parameters related to the distribution of random effects. It has components named \( b, K, w, mu, Sigma, Li, Q, gammaInv, r \).

**state.last.b** a list with the last sampled values of parameters related to the distribution of random effects. It has components named \( b, K, w, mu, Sigma, Li, Q, gammaInv, r \). It can be used as argument `init.b` to restart MCMC.

**state.first.eps** a list with the first stored (after burn-in) values of parameters related to the distribution of residuals of continuous responses. It has components named `sigma, gammaInv`.

**state.last.eps** a list with the last sampled values of parameters related to the distribution of residuals of continuous responses. It has components named `sigma, gammaInv`. It can be used as argument `init.eps` to restart MCMC.

**prop.accept.alpha** acceptance proportion from the Metropolis-Hastings algorithm for fixed effects (separately for each response type). Note that the acceptance proportion is equal to one for continuous responses since the Gibbs algorithm is used there.

**prop.accept.b** acceptance proportion from the Metropolis-Hastings algorithm for random effects (separately for each cluster). Note that the acceptance proportion is equal to one for models with continuous responses only since the Gibbs algorithm is used there.

**scale.b** a list containing the used value of the argument `scale.b`.

**summ.Deviance** a data.frame with posterior summary statistics for the deviance (approximated using the Laplacian approximation) and conditional (given random effects) deviance.

**summ.alpha** a data.frame with posterior summary statistics for fixed effects.

**summ.b.Mean** a matrix with posterior summary statistics for means of random effects.

**summ.b.SDCorr** a matrix with posterior summary statistics for standard deviations of random effects and correlations of each pair of random effects.

**summ.sigma_eps** a matrix with posterior summary statistics for standard deviations of the error terms in the (mixed) models of continuous responses.
poster.comp.prob_u a matrix which is present in the output object if the number of mixture components in the distribution of random effects is fixed and equal to $K$. In that case, poster.comp.prob_u is a matrix with $K$ columns and $I$ rows ($I$ is the number of subjects defining the longitudinal profiles or correlated observations) with estimated posterior component probabilities – posterior means of the components of the underlying 0/1 allocation vector.

**WARNING:** By default, the labels of components are based on artificial identifiability constraints based on ordering of the mixture means in the first margin. Very often, such identifiability constraint is not satisfactory!

poster.comp.prob_b a matrix which is present in the output object if the number of mixture components in the distribution of random effects is fixed and equal to $K$. In that case, poster.comp.prob_b is a matrix with $K$ columns and $I$ rows ($I$ is the number of subjects defining the longitudinal profiles or correlated observations) with estimated posterior component probabilities – posterior mean over model parameters including random effects.

**WARNING:** By default, the labels of components are based on artificial identifiability constraints based on ordering of the mixture means in the first margin. Very often, such identifiability constraint is not satisfactory!

freqK_b frequency table for the MCMC sample of the number of mixture components in the distribution of the random effects.

propK_b posterior probabilities for the numbers of mixture components in the distribution of random effects.

poster.mean.y a list with data.frames, one data.frame per response profile. Each data.frame with columns labeled id, observed, fitted, stres, eta.fixed and eta.random holding identifier for clusters of grouped observations, observed values and posterior means for fitted values (response expectation given fixed and random effects), standardized residuals (derived from fitted values), 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.

poster.mean.profile a data.frame with columns labeled b1, ..., bq, Logpb, Cond.Deviance, Deviance with posterior means of random effects for each cluster, posterior means of $\log\{p(b)\}$, conditional deviances, i.e., minus twice the conditional (given random effects) log-likelihood for each cluster and GLMM deviances, i.e., minus twice the marginal (random effects integrated out) log-likelihoods for each cluster. The value of the marginal (random effects integrated out) log-likelihood at each MCMC iteration is obtained using the Laplacian approximation.

poster.mean.w_b a numeric vector with posterior means of mixture weights after re-labeling. It is computed only if $K_b$ is fixed and even then I am not convinced that these are useful posterior summary statistics (see label switching problem mentioned above). In any case, they should be used with care.

poster.mean.mu_b a matrix with posterior means of mixture means after re-labeling. It is computed only if $K_b$ is fixed and even then I am not convinced that these are useful posterior summary statistics (see label switching problem mentioned above). In any case, they should be used with care.

poster.mean.Q_b a list with posterior means of mixture inverse variances after re-labeling. It is computed only if $K_b$ is fixed and even then I am not convinced that these are useful posterior summary statistics (see label switching problem mentioned above). In any case, they should be used with care.
poster.mean.Sigma_b a list with posterior means of mixture variances after re-labeling. It is computed only if $K_b$ is fixed and even then I am not convinced that these are useful posterior summary statistics (see label switching problem mentioned above). In any case, they should be used with care.

poster.mean.Li_b a list with posterior means of Cholesky decompositions of mixture inverse variances after re-labeling. It is computed only if $K_b$ is fixed and even then I am not convinced that these are useful posterior summary statistics (see label switching problem mentioned above). In any case, they should be used with care.

Deviance numeric vector with a chain for the GLMM deviances, i.e., twice the marginal (random effects integrated out) log-likelihoods of the GLMM. The marginal log-likelihood is obtained using the Laplacian approximation at each iteration of MCMC.

Cond.Deviance numeric vector with a chain for the conditional deviances, i.e., twice the conditional (given random effects) log-likelihoods.

$K_b$ numeric vector with a chain for $K_b$ (number of mixture components in the distribution of random effects).

$w_b$ numeric vector or matrix with a chain for $w_b$ (mixture weights for the distribution of random effects). It is a matrix with $K_b$ columns when $K_b$ is fixed. Otherwise, it is a vector with weights put sequentially after each other.

$mu_b$ numeric vector or matrix with a chain for $\mu_b$ (mixture means for the distribution of random effects). It is a matrix with $dimb \cdot K_b$ columns when $K_b$ is fixed. Otherwise, it is a vector with means put sequentially after each other.

$Q_b$ numeric vector or matrix with a chain for lower triangles of $Q_b$ (mixture inverse variances for the distribution of random effects). It is a matrix with $\frac{dimb(dimb+1)}{2} \cdot K_b$ columns when $K_b$ is fixed. Otherwise, it is a vector with lower triangles of $Q_b$ matrices put sequentially after each other.

Sigma_b numeric vector or matrix with a chain for lower triangles of $\Sigma_b$ (mixture variances for the distribution of random effects). It is a matrix with $\frac{dimb(dimb+1)}{2} \cdot K_b$ columns when $K_b$ is fixed. Otherwise, it is a vector with lower triangles of $\Sigma_b$ matrices put sequentially after each other.

Li_b numeric vector or matrix with a chain for lower triangles of Cholesky decompositions of $Q_b$ matrices. It is a matrix with $\frac{dimb(dimb+1)}{2} \cdot K_b$ columns when $K_b$ is fixed. Otherwise, it is a vector with lower triangles put sequentially after each other.

gammaInv_b matrix with $dimb$ columns with a chain for inverses of the hyperparameter $\gamma_b$.

order_b numeric vector or matrix with order indeces of mixture components in the distribution of random effects related to artificial identifiability constraint defined by ordering of the first component of the mixture means.

It is a matrix with $K_b$ columns when $K_b$ is fixed. Otherwise it is a vector with orders put sequentially after each other.

rank_b numeric vector or matrix with rank indeces of mixture components in the distribution of random effects related to artificial identifiability constraint defined by ordering of the first component of the mixture means.

It is a matrix with $K_b$ columns when $K_b$ is fixed. Otherwise it is a vector with ranks put sequentially after each other.
mixture_b data.frame with columns labeled b.Mean.*, b.SD.*, b.Corr.* containing the chains for the means, standard deviations and correlations of the distribution of the random effects based on a normal mixture at each iteration.

b a matrix with the MCMC chains for random effects. It is included only if store[b] is TRUE.

alpha numeric vector or matrix with the MCMC chain(s) for fixed effects.

sigma_eps numeric vector or matrix with the MCMC chain(s) for standard deviations of the error terms in the (mixed) models for continuous responses.

gammaInv_eps matrix with dimb columns with MCMC chain(s) for inverses of the hyperparameter $\gamma_b$.

relabel_b a list which specifies the algorithm used to re-label the MCMC output to compute order_b, rank_b, poster.comp.prob_u, poster.comp.prob_b, poster.mean.w_b, poster.mean.mu_b, poster.mean.Q_b, poster.mean.Sigma_b, poster.mean.Li_b.

Cpar a list with components useful to call underlying C++ functions (not interesting for ordinary users).

Object of class GLMM_MCMClist

Object 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.

D posterior mean of the deviance for each subject.

popt contributions to the unweighted penalty from each subject.

wpopt contributions to the weighted penalty from each subject.

inv.D for each subject, 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 subject, 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 subject, 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 subject, sum of importance sampling weights.

Deviance1 sampled value of the observed data deviance from chain 1

Deviance2 sampled values of the observed data deviance from chain 2

Deviance_repl1_ch1 sampled values of the deviance of data replicated according to the chain 1 evaluated under the parameters from chain 1

Deviance_repl1_ch2 sampled values of the deviance of data replicated according to the chain 1 evaluated under the parameters from chain 2
**MatMPpinv**

**Deviance_repl2_ch1** sampled values of the deviance of data replicated according to the chain 2 evaluated under the parameters from chain 1

**Deviance_repl2_ch2** sampled values of the deviance of data replicated according to the chain 2 evaluated under the parameters from chain 2

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**References**


**See Also**

NMixMCMC.

**Examples**

```r
## See also additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - files http://www.karlin.mff.cuni.cz/~komarek/software/mixAK/PBCseq.pdf,
##   PBCseq.R
## ==============================================================
```

---

**MatMPpinv**

*Moore-Penrose pseudoinverse of a squared matrix*

**Description**

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

1. $AA^+A = A$,
2. $A^+AA^+ = A^+$,
3. $(AA^+)' = AA^+$,
4. $(A^+A) = A^+A$.

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

Usage

\[
\text{MatMPpinv}(A)
\]

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@mff.cuni.cz>

References


Examples

```r
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@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)

### The following example crashes on r-devel Windows x64 x86_64,
### on r-patched Linux x86_64
### due to failure of LAPACK zgesv routine
###
### Commented on 16/01/2010
###
# 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),
              nrow=5, byrow=TRUE)
D4
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 $\Sigma$ 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@mff.cuni.cz>

References

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

\begin{verbatim}
 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)
\end{verbatim}

Arguments

- **weight**: vector of length $K$ with the mixture weights or values which are proportional to the weights.
- **mean**: 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.
- **Q**: precision matrices of the multivariate normal distribution. Ignored if $\Sigma$ 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.
- **Sigma**: covariance matrix of the multivariate normal distribution. If $\Sigma$ is supplied, precisions are computed from $\Sigma$ 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.
- **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

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 \texttt{dMVNmixture}

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

Value for \texttt{dMVNmixture2}

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

Value for \texttt{rMVNmixture}

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

Value for \texttt{rMVNmixture2}

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

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also

\texttt{dnorm}, \texttt{MVN}, \texttt{Mvnorm}.

Examples

\begin{verbatim}
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)),

\end{verbatim}
main="Sampled values")
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)),
     main="Sampled values")
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, sd22^2), nrow=2))
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(xx2))
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(xx2))
yyD <- matrix(dMVNmixture2(XX, weight=w, mean=mu, Q=Q), nrow=length(xx1), ncol=length(xx2))

### Starting from version 3.6, the above command led to SegFault
### on CRAN r-patched-solaris-sparc check.
### Commented here on 20140806 (version 3.6-1).
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)
MVT
Multivariate Student t distribution

Description
Density and random generation for the multivariate Student t distribution with location equal to mu, precision matrix equal to Q (or scale matrix equal to Sigma).

Mentioned functions implement the multivariate Student t distribution with a density given by

\[ p(z) = \frac{\Gamma\left(\frac{\nu+p}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right) \nu^{\frac{p}{2}}} |\Sigma|^{-\frac{1}{2}} \left\{ 1 + \frac{(z - \mu)' \Sigma^{-1} (z - \mu)}{\nu} \right\}^{-\frac{\nu+p}{2}}, \]

where \( p \) is the dimension, \( \nu > 0 \) degrees of freedom, \( \mu \) the location parameter and \( \Sigma \) the scale matrix.

For \( \nu > 1 \), the mean in equal to \( \mu \), for \( \nu > 2 \), the covariance matrix is equal to \( \frac{\nu}{\nu - 2} \Sigma \).

Usage

dMVT(x, df, mu=0, Q=1, Sigma, log=FALSE)
rMVT(n, df, mu=0, Q=1, Sigma)

Arguments

df
  degrees of freedom of the multivariate Student t distribution.
mu
  vector of the location parameter.
Q
  precision (inverted scale) matrix of the multivariate Student t distribution. Ignored if Sigma is given.
Sigma    scale matrix of the multivariate Student t distribution. If Sigma is supplied, precision is computed from \( \Sigma \) 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 dMVT

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

Value for rMVT

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

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also
dt, Mvt.

Examples

set.seed(1977)

### Univariate central t distribution
z <- rMVT(10, df=1, mu=0, Q=1)
ldz <- dMVT(z$x, df=1, log=TRUE)
boxplot(as.numeric(z$x))
cbind(z$log.dens, ldz, dt(as.numeric(z$x), df=1, log=TRUE))

### Multivariate t distribution
mu <- c(1, 2, 3)
Sigma <- matrix(c(1, 1, -1.5, 1, 4, 1.8, -1.5, 1.8, 9), nrow=3)
Q <- chol2inv(chol(Sigma))
nu <- 3
z <- rMVT(1000, df=nu, mu=mu, Sigma=Sigma)
apply(z$x, 2, mean)      ## should be close to mu
((nu - 2) / nu) * var(z$x)  ## should be close to Sigma
dz <- dMVT(z$x, df=nu, mu=mu, Sigma=Sigma)
ldz <- dMVT(z$x, df=nu, mu=mu, Sigma=Sigma, log=TRUE)
### Compare with mvtnorm package
#require(mvtnorm)
#ldz2 <- dmvt(z$x, sigma=Sigma, df=nu, delta=mu, log=TRUE)
#plot(z$log.dens, ldz2, pch=21, col="red3", bg="orange", xlab="mixAK", ylab="mvtnorm")
#plot(ldz, ldz2, pch=21, col="red3", bg="orange", xlab="mixAK", ylab="mvtnorm")

---

NMixChainComp  

**Chains for mixture parameters**

**Description**

This function returns chains for parameters derived from the (re-labeled) mixture weights, means, covariance matrices.

First, mixture means and shifted-scaled to the original (data) scale, mixture covariance matrices are scaled to the original (data) scale (see argument scale in NMixMCMC function or argument scale.b in GLMM_MCMC). Possible derived parameters are standard deviations and correlation coefficients.

**Usage**

NMixChainComp(x, relabel = TRUE, param)

```r
# Default S3 method:  
NMixChainComp(x, relabel = TRUE, param)
```

```r
## S3 method for class 'NMixMCMC' 
NMixChainComp(x, relabel = TRUE, 
  param = c("w", "mu", "var", "sd", "cor", "Sigma", "Q", "Li"))
```

```r
## S3 method for class 'GLMM_MCMC' 
NMixChainComp(x, relabel = TRUE, 
  param = c("w_b", "mu_b", "var_b", "sd_b", "cor_b", "Sigma_b", "Q_b", "Li_b"))
```

**Arguments**

- **x**  
an object of class NMixMCMC or GLMM_MCMC.
- **relabel**  
a logical argument indicating whether the chains are to be returned with components being re-labeled (see NMixRelabel) or whether the chains are to be returned as originally sampled.
- **param**  
a character string indicating which sample is to be returned:
  - **w, w_b** mixture weights;
  - **mu, mu_b** mixture means;
  - **var, var_b** mixture variances;
  - **sd, sd_b** mixture standard deviations;
  - **cor, cor_b** correlations derived from the mixture covariance matrices;
  - **Sigma, Sigma_b** mixture covariance matrices (their lower triangles);
\( Q, Q_b \) mixture inverted covariance matrices (their lower triangles); 
\( Li, Li_b \) Cholesky factors (their lower triangles) of the mixture inverted covariance matrices.

**Value**

A matrix with sampled values in rows, parameters in columns.

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**See Also**

NMixMCMC, GLMM_MCMC.

---

**NMixChainsDerived**

Create MCMC chains derived from previously sampled values

**Description**

Currently, this function creates chains for marginal means of \( \exp(\text{data}) \) 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 \( E \exp(Y_1), \ldots, E \exp(Y_p) \).

summ.expy.Mean posterior summary statistics for \( E \exp(Y_1), \ldots, E \exp(Y_p) \).

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@mff.cuni.cz>
See Also

NMixMCMC.

NMixCluster

Clustering based on the MCMC output of the mixture model

Description

TO BE ADDED.

This function only works for models with a fixed number of mixture components.

Usage

NMixCluster(object, ...)

## Default S3 method:
NMixCluster(object, ...)

## S3 method for class 'GLMM_MCMC'
NMixCluster(object,

prob = c("poster.comp.prob", "quant.comp.prob", "poster.comp.prob_b",
          "quant.comp.prob_b", "poster.comp.prob_u"),
pquant = 0.5, HPD = FALSE, pHPD = 0.95, pthresh = -1, unclass.na = FALSE, ...)

Arguments

object an object of apropriate class.

prob character string which identifies estimates of the component probabilities to be used for clustering.

pquant when prob is either "quant.comp.prob" or "quant.comp.prob_b", argument pquant is the probability of the quantile of the component probabilities to be used for clustering.

HPD logical value. If TRUE then only those subjects are classified for which the lower limit of the pHPD*100% HPD credible interval of the component probability exceeds the value of pthresh.

pHPD credible level of the HPD credible interval, see argument HPD.

pthresh an optional threshold for the estimated component probability (when HPD is FALSE) or for the lower limit of the HPD credible interval (when HPD is TRUE) to classify a subject. No effect when pthresh is negative.

unclass.na logical value taken into account when pthresh is positive. If unclass.na is TRUE, unclassified subjects get classification NA. If unclass.na is FALSE, unclassified subjects create a separate (last) group.

... optional additional arguments.
**Value**

A data.frame with three (when HPD is FALSE) or five (when HPD is TRUE) columns.

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**See Also**

NMixMCMC, GLMM_MCMC.

**Examples**

```r
## TO BE ADDED.
```

---

**NMixEM**

*EM algorithm for a homoscedastic normal mixture*

**Description**

This function computes ML estimates of the parameters of the $p$-dimensional $K$-component normal mixture using the EM algorithm

**Usage**

```r
NMixEM(y, K, weight, mean, Sigma, toler=1e-5, maxiter=500)
```

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

**Arguments**

- `y` vector (if $p = 1$) matrix or data frame (if $p > 1$) with data. Rows correspond to observations, columns correspond to margins.
- `K` required number of mixture components.
- `weight` a numeric vector with initial mixture weights. If not given, initial weights are all equal to $1/K$.
- `mean` vector or matrix of initial 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.
- `Sigma` number or $p \times p$ matrix giving the initial variance/covariance matrix.
- `toler` tolerance to determine convergence.
- `maxiter` maximum number of iterations of the EM algorithm.
- `x` an object of class NMixEM.
- `...` additional arguments passed to the default print method.
Value

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

- **K**: number of mixture components
- **weight**: estimated mixture weights
- **mean**: estimated mixture means
- **Sigma**: estimated covariance matrix
- **loglik**: log-likelihood value at fitted values
- **aic**: Akaike information criterion \((-2\hat{\ell} + 2\nu)\), where \(\hat{\ell}\) stands for the log-likelihood value at fitted values and \(\nu\) for the number of free model parameters
- **bic**: Bayesian (Schwarz) information criterion \((-2\hat{\ell} + \log(n)\nu)\), where \(\hat{\ell}\) stands for the log-likelihood value at fitted values and \(\nu\) for the number of free model parameters, and \(n\) for the sample size
- **iter**: number of iterations of the EM algorithm used to get the solution
- **iter.loglik**: values of the log-likelihood at iterations of the EM algorithm
- **iter.Qfun**: values of the EM objective function at iterations of the EM algorithm
- **dim**: dimension \(p\)
- **nobs**: number of observations \(n\)

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References


Examples

```r
## Not run:
## Estimates for 3-component mixture in Anderson's iris data
## ==========================================================
data(iris, package="datasets")
summary(iris)
VARS <- names(iris)[1:4]
fit <- NMixEM(iris[, VARS], K = 3)
print(fit)
apply(subset(iris, Species == "versicolor"), [, VARS], 2, mean)
apply(subset(iris, Species == "setosa"), [, VARS], 2, mean)
apply(subset(iris, Species == "virginica"), [, VARS], 2, mean)

## Estimates of 6-component mixture in Galaxy data
## ==============================================================
data(Galaxy, package="mixAK")
```
summary(Galaxy)

fit2 <- NMixEM(Galaxy, K = 6)
y <- seq(5, 40, length=300)
fy <- dMVNmixture(y, weight=fit2$weight, mean=fit2$mean,
Sigma=rep(fit2$Sigma, fit2$K))
hist(Galaxy, prob=TRUE, breaks=seq(5, 40, by=0.5),
main="", xlab="Velocity (km/sec)", col="sandybrown")
lines(y, fy, col="darkblue", lwd=2)

## End(Not run)

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).

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, x_w, 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,
    parallel = FALSE, cltype)

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

## S3 method for class 'NMixMCMClis1'
print(x, ped, dic, ...)

Arguments

y0 numeric vector of length n or n x 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 x 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:

0 right-censored observation,
1 exactly observed value,
2 left-censored observation,
3 interval-censored observation.

If it is not supplied then it is assumed that all values are exactly observed.

\( x_w \) optional vector providing a categorical covariate that may influence the mixture weights. Internally, it is converted into a factor.

Added in version 4.0 (03/2015).

scale a list specifying how to scale the data before running MCMC. It should have two components:

- **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 \( S \).

If there is no censoring, and argument scale is missing then the data are scaled to have zero mean and unit variances, i.e., \( \text{scale}(y0) \) is used for MCMC. In the case there is censoring and scale is missing, \( \text{scale}\$\text{shift} \) is taken to be a sample mean of \( \text{init}\$y \) and \( \text{scale}\$\text{scale} \) are sample standard deviations of columns of \( \text{init}\$y \).

If you do not wish to scale the data before running MCMC, specify scale=list(shift=0, scale=1).

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):

- **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 \( K \sim \text{Unif}\{1, \ldots, K_{max}\} \).

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

- **priormuQ** a character string which specifies the type of the prior for \( \mu_1, \ldots, \mu_{K_{max}} \) (mixture means) and \( Q_1, \ldots, Q_{K_{max}} \) (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

    \[
    (\mu_j, Q_j) \sim N(\xi_j, D_j) \times \text{Wishart}(\zeta, \Xi)
    \]
independently for \( j = 1, \ldots, K \), where normal means \( \xi_1, \ldots, \xi_K \), normal variances \( D_1, \ldots, D_K \), and Wishart degrees of freedom \( \zeta \) are specified further as \( x_i, D, \zeta \) components of the list prior.

“naturalC”

\[ \equiv \text{natural conjugate prior. That is, a priori} \]

\[ (\mu_j, Q_j) \sim N(\xi_j, c_j^{-1} Q_j^{-1}) \times \text{Wishart}(\zeta, \Xi) \]

independently for \( j = 1, \ldots, K \), where normal means \( \xi_1, \ldots, \xi_K \), precisions \( c_1, \ldots, c_K \), and Wishart degrees of freedom \( \zeta \) are specified further as \( x_i, c, \zeta \) components of the list prior.

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

\( K_{max} \) maximal number of mixture components \( K_{max} \). It must always be specified by the user.

\( \lambda \) parameter \( \lambda \) for the truncated Poisson prior on \( K \). It must be positive and must always be specified if \( \text{priorK} \) is “tpoisson”.

\( \delta \) parameter \( \delta \) for the Dirichlet prior on the mixture weights \( w_1, \ldots, w_K \). It must be positive. Its default value is 1.

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

If \( p = 1 \) and \( x_i = \xi \) is a single value then \( \xi_1 = \cdots = \xi_{K_{max}} = \xi \).

If \( p = 1 \) and \( x_i = \xi \) is a vector of length \( K_{max} \) then the \( j \)-th element of \( x_i \) gives \( \xi_j \) (\( j = 1, \ldots, K_{max} \)).

If \( p > 1 \) and \( x_i = \xi \) is a vector of length \( p \) then \( \xi_1 = \cdots = \xi_{K_{max}} = \xi \).

If \( p > 1 \) and \( x_i \) is a \( K_{max} \times p \) matrix then the \( j \)-th row of \( x_i \) gives \( \xi_j \) (\( j = 1, \ldots, K_{max} \)).

\( c \) a numeric value or vector which specifies prior precision parameters \( c_1, \ldots, c_{K_{max}} \) for the mixture means \( \mu_1, \ldots, \mu_{K_{max}} \) when \( \text{priormuQ} \) is “naturalC”. Its default value is a vector of ones which follows Cappe, Robert and Ryden (2003).

If \( c = c \) is a single value then \( c_1 = \cdots = c_{K_{max}} = c \).

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

\( D \) a numeric vector or matrix which specifies \( D_1, \ldots, D_{K_{max}} \) (prior variances or covariance matrices of the mixture means \( \mu_1, \ldots, \mu_{K_{max}} \) when \( \text{priormuQ} \)
NMixMCMC

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 $d_1 = \cdots = d_{K_{\text{max}}} = d$.

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

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

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

zeta degrees of freedom $\zeta$ for the Wishart prior on the inverted mixture variances $Q_1, \ldots, Q_{K_{\text{max}}}$.

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 $g_1, \ldots, g_p$ for the Gamma hyperpriors on $\gamma_1, \ldots, \gamma_p$. It must be positive. Its default value is a vector $(0.2, \ldots, 0.2)'$.

h a value or a vector of length $p$ with the rate parameters $h_1, \ldots, h_p$ for the Gamma hyperpriors on $\gamma_1, \ldots, \gamma_p$. 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 $\gamma$.

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 init2 has the same structure as the list init. All initials in init2 can be determined by the program (differently than the values in init) if they are not specified.

Ignored when PED is FALSE.

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:
**Paction** probabilities (or proportionalit constants) which are used to choose an action of the sampler within each iteration of MCMC to update the mixture related parameters. Let $P_{\text{action}} = (p_1, p_2, p_3)'$. 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.

**Pspl**t a numeric vector of length prior$K_{\text{max}}$ giving conditional probabilities of the split move given $k$ as opposite to the combine move. Default value is $(1, 0.5, \ldots, 0.5, 0)'$.

**Pbirth** a numeric vector of length prior$K_{\text{max}}$ giving conditional probabilities of the birth move given $k$ as opposite to the death move. Default value is $(1, 0.5, \ldots, 0.5, 0)'$.

**par.u1** a two component vector with parameters of the beta distribution used to generate an auxiliary value $u_1$. A default value is $\text{par.u1} = (2, 2)'$, 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 $u_{2,1}, \ldots, u_{2,p}$ in rows. A default value leads to $u_{2,d} \sim \text{Unif}(-1, 1)$ ($d = 1, \ldots, p - 1$), $u_{2,p} \sim \text{Beta}(1, 2p)$.

**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 $u_{3,1}, \ldots, u_{3,p}$ in rows. A default value leads to $u_{3,d} \sim \text{Unif}(0, 1)$ ($d = 1, \ldots, p - 1$), $u_{3,p} \sim \text{Beta}(1, p)$.

**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 $(M_{\text{burn}} + M_{\text{keep}}) \cdot M_{\text{thin}}$ 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 scale 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 NMixMCMC or NMixMCMClist 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.
parallel a logical value which indicates whether parallel computation (based on a package parallel) should be used when running two chains for the purpose of PED calculation
cltre optional argument applicable if parallel is TRUE. If cltype is given, it is passed as the type argument into the call to makeCluster.
... additional arguments passed to the default print 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, \( w_1, \ldots, w_K \) are mixture weights, \( \mu_1, \ldots, \mu_K \) are mixture means, \( \Sigma_1, \ldots, \Sigma_K \) are mixture variance-covariance matrices, \( Q_1, \ldots, Q_K \) are their inverses.

For the data \( y_1, \ldots, y_n \) the following \( g_y(y) \) density is assumed

\[
g_y(y) = |S|^{-1} \sum_{j=1}^{K} w_j \varphi(S^{-1}(y - m_j | \mu_j, \Sigma_j)),
\]

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

Object of class NMixMCMC
Objects of class NMixMCMC have the following components:

iter index of the last iteration performed.
nMCMC used value of the argument nMCMC.
dim dimension \( p \) of the distribution of data
nx_w number of levels of a factor covariate on mixture weights (equal to 1 if there were no covariates on mixture weights)
prior a list containing the used value of the argument prior.
init a list containing the used initial values for the MCMC (the first iteration of the burn-in).
state.first a list having the components labeled y, K, w, mu, Li, Q, Sigma, gammaInv, r containing the values of generic parameters at the first stored (after burn-in) iteration of the MCMC.

state.last a list having the components labeled y, K, w, mu, Li, Q, Sigma, gammaInv, r containing the last sampled values of generic parameters.

RJMCMC a list containing the used value of the argument RJMCMC.

scale a list containing the used value of the argument scale.

data.frame having columns labeled DIC, pD, D.bar, D.in.bar containing values used to compute deviance information criterion (DIC). Currently only DIC3 of Celeux et al. (2006) is implemented.

moves a data.frame which summarizes the acceptance probabilities of different move types of the sampler.

K numeric vector with a chain for K (number of mixture components).

w 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.

mu 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.

Q 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.

Sigma numeric vector or matrix with a chain for lower triangles of \( \Sigma \) (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 \( \Sigma \) matrices put sequentially after each other.

Li 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.

gammaInv matrix with p columns with a chain for inverses of the hyperparameter \( \gamma \).

order numeric vector or matrix with order indeces of mixture components related to artificial identifiability constraint defined by a suitable re-labeling algorithm (by default, simple ordering of the first component of the mixture means is used).

It is a matrix with K columns when K is fixed. Otherwise it is a vector with orders put sequentially after each other.

rank numeric vector or matrix with rank indeces of mixture components related to artificial identifiability constraint defined by a suitable re-labeling algorithm (by default, simple ordering of the first component of the mixture means is used).

It is a matrix with K columns when K is fixed. Otherwise it is a vector with ranks put sequentially after each other.

mixture data.frame with columns labeled y.Mean.*, y.SD.*, y.Corr.*, z.Mean.*, z.SD.*, z.Corr.* 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.

deviance data.frame with columns labeled LogL0, LogL1, dev.complete, dev.observed containing the chains of quantities needed to compute DIC.
pm.y a data.frame with $p$ columns with posterior means for (latent) values of observed data (useful when there is censoring).

pm.z a data.frame with $p$ columns with posterior means for (latent) values of scaled observed data (useful when there is censoring).

pm.indDev a data.frame with columns labeled LogL0, LogL1, dev.complete, dev.observed, pred.dens containing posterior means of individual contributions to the deviance.

pred.dens 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.

poster.comp.prob_u a matrix which is present in the output object if the number of mixture components in the distribution of random effects is fixed and equal to $K$. In that case, poster.comp.prob_u is a matrix with $K$ columns and $n$ rows with estimated posterior component probabilities – posterior means of the components of the underlying 0/1 allocation vector.

WARNING: By default, the labels of components are based on artificial identifiability constraints based on ordering of the mixture means in the first margin. Very often, such identifiability constraint is not satisfactory!

poster.comp.prob_b a matrix which is present in the output object if the number of mixture components in the distribution of random effects is fixed and equal to $K$. In that case, poster.comp.prob_b is a matrix with $K$ columns and $n$ rows with estimated posterior component probabilities – posterior mean over model parameters.

WARNING: By default, the labels of components are based on artificial identifiability constraints based on ordering of the mixture means in the first margin. Very often, such identifiability constraint is not satisfactory!

summ.y.Mean Posterior summary statistics based on chains stored in y.Mean.* columns of the data.frame mixture.

summ.y.SDCorr Posterior summary statistics based on chains stored in y.SD.* and y.Corr.*.* columns of the data.frame mixture.

summ.z.Mean Posterior summary statistics based on chains stored in z.Mean.* columns of the data.frame mixture.

summ.z.SDCorr Posterior summary statistics based on chains stored in z.SD.* and z.Corr.*.* columns of the data.frame mixture.

poster.mean.w 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 (see label switching problem mentioned above). In any case, they should be used with care.

poster.mean.mu 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 (see label switching problem mentioned above). In any case, they should be used with care.

poster.mean.Q 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 (see label switching problem mentioned above). In any case, they should be used with care.
**poster.mean.Sigma** 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 (see label switching problem mentioned above). 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 (see label switching problem mentioned above). In any case, they should be used with care.

**relabel** a list which specifies the algorithm used to re-label the MCMC output to compute order, rank, poster.comp.prob_u, poster.comp.prob_b, poster.mean.w, poster.mean.mu, poster.mean.Q, poster.mean.Sigma, poster.mean.Li.

**Cpar** a list with components useful to call underlying C++ functions (not interesting for ordinary users).

### Object of class NMixMCMClist

Object 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 $dens.zer0$) 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@mf.cuni.cz>

### References


See Also

NMixPredDensMarg, NMixPredDensJoint2.

Examples

```r
## Not run:
## See also additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
##   https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Galaxy.pdf
##   https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Faithful.pdf
##   https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Tandmob.pdf
##
## # Simple analysis of Anderson's iris data
## library("colorspace")

data(iris, package="datasets")
summary(iris)
VARS <- names(iris)[1:4]
#COLS <- rainbow_hcl(3, start = 60, end = 240)
COLS <- c("red", "darkblue", "darkgreen")
names(COLS) <- levels(iris[, "Species"])

## Prior distribution and the length of MCMC
Prior <- list(priorK = "fixed", Kmax = 3)
```
nMCMC <- c(burn=5000, keep=10000, thin=5, info=1000)

### Run MCMC
set.seed(20091230)
fit <- NMixMCMC(y0 = iris[, VARS], prior = Prior, nMCMC = nMCMC)

### Basic posterior summary
print(fit)

### Univariate marginal posterior predictive densities
### based on chain #1
pdens1 <- NMixPredDensMarg(fit[[1]], lgrid=150)
plot(pdens1)
plot(pdens1, main=VARS, xlab=VARS)

### Bivariate (for each pair of margins) predictive densities
### based on chain #1
pdens2a <- NMixPredDensJoint2(fit[[1]])
plot(pdens2a)
plot(pdens2a, xlab=VARS)
plot(pdens2a, xlab=VARS, contour=TRUE)

### Determine the grid to compute bivariate densities
grid <- list(Sepal.Length=seq(3.5, 8.5, length=75),
             Sepal.Width=seq(1.8, 4.5, length=75),
             Petal.Length=seq(0, 7, length=75),
             Petal.Width=seq(-0.2, 3, length=75))
pdens2b <- NMixPredDensJoint2(fit[[1]], grid=grid)
plot(pdens2b, xlab=VARS)

### Plot with contours
ICOL <- rev(heat_hcl(20, c=c(80, 30), l=c(30, 90), power=c(1/5, 2)))
oldPar <- par(mfrow=c(2, 3), bty="n")
for (i in 1:3){
  for (j in (i+1):4){
    NAME <- paste(i, "-", j, sep="")
    MAIN <- paste(VARS[i], "x", VARS[j])
    image(pdens2b$x[[i]], pdens2b$x[[j]], pdens2b$dens[[NAME]], col=ICOL,
          xlab=VARS[i], ylab=VARS[j], main=MAIN)
    contour(pdens2b$x[[i]], pdens2b$x[[j]], pdens2b$dens[[NAME]], add=TRUE, col="brown4")
  }
}

### Plot with data
for (i in 1:3){
  for (j in (i+1):4){
    NAME <- paste(i, "-", j, sep="")
    MAIN <- paste(VARS[i], "x", VARS[j])
    image(pdens2b$x[[i]], pdens2b$x[[j]], pdens2b$dens[[NAME]], col=ICOL,
          xlab=VARS[i], ylab=VARS[j], main=MAIN)
    for (spec in levels(iris[, "Species"])){
      Data <- subset(iris, Species==spec)
    }
  }
}
### Clustering based on posterior summary statistics of component allocations
### or on the posterior distribution of component allocations
### (these are two equivalent estimators of probabilities of belonging
### to each mixture components for each observation)

```r
p1 <- fit[[1]]$poster.comp.prob_u
p2 <- fit[[1]]$poster.comp.prob_b
```

### Clustering based on posterior summary statistics of mixture weight, means, variances

```r
p3 <- NMixPlugDA(fit[[1]], iris[, VARS])
p3 <- p3[, paste("prob", 1:3, sep="")]
```

### Observations from "setosa" species (all would be allocated in component 1)

```r
apply(p1[1:50,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p2[1:50,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p3[1:50,], 2, quantile, prob=seq(0, 1, by=0.1))
```

### Observations from "versicolor" species (almost all would be allocated in component 2)

```r
apply(p1[51:100,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p2[51:100,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p3[51:100,], 2, quantile, prob=seq(0, 1, by=0.1))
```

### Observations from "virginica" species (all would be allocated in component 3)

```r
apply(p1[101:150,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p2[101:150,], 2, quantile, prob=seq(0, 1, by=0.1))
apply(p3[101:150,], 2, quantile, prob=seq(0, 1, by=0.1))
```

## End(Not run)

---

**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**

```r
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

- **x**: an object of class NMixMCMC for NMixPlugCondDensJoint2.NMixMCMC function. An object of class GLMM_MCMC for NMixPlugCondDensJoint2.GLMM_MCMC function. A list with the grid values (see below) for NMixPlugCondDensJoint2.default 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, mu 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.

- **grid**: a list with the grid values for each margin in which the density should be evaluated. The value of grid[[icond]] determines the values by which we condition.

  If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

- **lgrid**: a length of the grid used to create the grid if that is not specified.

- **scaled**: if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.

- **...**: optional additional arguments.

### Value

An object of class NMixPlugCondDensJoint2 which has the following components:

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

- **icond**: index of the margin by which we condition.
A list with the computed conditional densities for each value of $x[[\text{icond}]]$. Each $\text{dens}[[j]]$ is again a list with conditional densities for each pair of margins given margin $\text{icond}$ equal to $x[[\text{icond}]][j]$. The value of $\text{dens}[[j]][[i-k]]$ gives values of conditional density of the $(i,k)$-th margins given margin $\text{icond}$ equal to $x[[\text{icond}]][j]$.

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

**Author(s)**

Arnošt Komárek <arnost.komarek@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**

```r
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 NMixMCMC for NMixPlugCondDensMarg.NMixMCMC function. An object of class GLMM_MCMC for NMixPlugCondDensMarg.GLMM_MCMC function. A list with the grid values (see below) for NMixPlugCondDensMarg.default 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, mu 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.

grid a list with the grid values for each margin in which the density should be evaluated. The value of grid[[icond]] determines the values by which we condition.

If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in \( x \).

lgrid a length of the grid used to create the grid if that is not specified.

scaled if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object \( x \).

... optional additional arguments.

Value

An object of class \texttt{NMixPlugCondDensMarg} which has the following components:

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

icond index of the margin by which we condition.

dens a list with the computed conditional densities for each value of \( x[[icond]] \). Each dens[[j]] is again a list with conditional densities for each margin given margin icond equal to \( x[[icond]][j] \). The value of dens[[j]][[imargin]] gives a value of a marginal density of the imargin-th margin at \( x[[icond]][j] \).

There is also a \texttt{plot} method implemented for the resulting object.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also

\texttt{plot.NMixPlugCondDensMarg, NMixMCMC, GLMM_MCMC, NMixPredCondDensMarg}. 
**NMixPlugDA**

*Discriminant analysis based on plug-in estimates from the mixture model*

**Description**

It performs discriminant analysis based on posterior summary for (re-labeled) mixture components in a model with fixed number of components fitted with **NMixMCMC** function.

**Usage**

```
NMixPlugDA(object, y)
```

**Arguments**

- `object`: an object of class **NMixMCMC**
- `y`: vector, matrix or data frame with observations to be clustered

**Value**

A `data.frame` with columns labeled `prob1`,..., `probp` giving plug-in estimates of probabilities of belonging to each component and a column labeled `component` giving the index of the component with the highest component probability.

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**See Also**

**NMixMCMC, NMixPredDA.**

---

**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**

- **x**
  - An object of class NMixMCMC for NMixPlugDensJoint2.NMixMCMC function.
  - An object of class GLMM_MCMC for NMixPlugDensJoint2.GLMM_MCMC function.
  - A list with the grid values (see below) for NMixPlugDensJoint2.default function.
- **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, mu 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.
- **grid**
  - A list with the grid values for each margin in which the density should be evaluated.
  - If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.
- **lgrid**
  - A length of the grid used to create the grid if that is not specified.
- **scaled**
  - If TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.
- **...**
  - Optional additional arguments.

**Value**

An object of class NMixPlugDensJoint2 which has the following components:

- **x**
  - A list with the grid values for each margin. The components of the list are named x1, ..., or take names from grid argument.
- **dens**
  - A list with the computed densities for each pair of margins. The components of the list are named 1-2, 1-3, ..., i.e., dens[[1]]=dens[['1-2']] 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@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=500, scaled=FALSE, 

## S3 method for class 'GLMM_MCMC' 
NMixPlugDensMarg(x, grid, lgrid=500, scaled=FALSE, 

Arguments

x  a list with the grid values (see below) for NMixPlugDensMarg.default function. 
An object of class NMixMCMC for NMixPlugDensMarg.NMixMCMC function. 
An object of class GLMM_MCMC for NMixPlugDensMarg.GLMM_MCMC function. 
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, mu 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.
NMixPredCDFMarg

grid a list with the grid values for each margin in which the density should be evaluated. If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

lgrid a length of the grid used to create the grid if that is not specified.
scaled if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.

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

x a list with the grid values for each margin. The components of the list are named x1,... or take names from grid argument.
dens a list with the computed densities for each margin. The components of the list are named 1,..., i.e., dens[[1]]=dens["1"] 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@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=500, scaled=FALSE, ...)

## S3 method for class 'GLMM_MCMC'
NMixPredCDFMarg(x, grid, lgrid=500, scaled=FALSE, ...)
Arguments

x

an object of class NMixMCMC for NMixPredCDFMarg. NMixMCMC function.
An object of class GLMM_MCMC for NMixPredCDFMarg. GLMM_MCMC function.
A list with the grid values (see below) for NMixPredCDFMarg. default function.
scale

da two component list giving the shift and the scale.
K

either a number (when Krandom=FALSE) or a numeric vector with the chain for
the number of mixture components.
w

da numeric vector with the chain for the mixture weights.
mu

da numeric vector with the chain for the mixture means.
Li

da numeric vector with the chain for the mixture inverse variances (lower triangles
only).
Krandom

da logical value which indicates whether the number of mixture components
changes from one iteration to another.
grid

da numeric vector or a list with the grid values in which the predictive CDF should
be evaluated.
    If x$dim is 1 then grid may be a numeric vector. If x$dim is higher than then
grid must be a list with numeric vectors as components giving the grids for each
margin.
    If grid is not specified, it is created automatically using the information from
the posterior summary statistics stored in x.
lgrid

da length of the grid used to create the grid if that is not specified.
scaled

if TRUE, the CDF of shifted and scaled data is summarized. The shift and scale
vector are taken from the scale component of the object x.
...

optional additional arguments.

Value

An object of class NMixPredCDFMarg which has the following components:

x

a list with the grid values for each margin. The components of the list are named
x1,... or take names from grid argument.

freqK

frequency table for the values of K (numbers of mixture components) in the
MCMC chain.

propK

proportions derived from freqK.

MCMC.length

the length of the MCMC used to compute the predictive cdf’s.

cdf

a list with the computed predictive CDF’s for each margin. The components of
the list are named 1,..., i.e., cdf[[1]]=cdf["1"] is the predictive cdf for
margin 1 etc.

cdfK

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, cdf[[1]][[1]]
= cdf["1"][[1]] is the predictive CDF for margin 1 conditioned by K = 1, cdf[[1]][[2]] = cdf["1"][[2]] is the predictive CDF for margin 1 conditioned by K = 2 etc.

Note that cdfK provides some additional information only when Krandom = TRUE or when x 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@mff.cuni.cz>

**References**


**See Also**

`plot.NMixPredCDFMarg`, `NMixMCMC`, `GLMM_MCMC`.

---

**NMixPredCondCDFMarg**  
*Univariate conditional predictive cumulative distribution function*

---

**Description**

This function serves as an inference tool for the MCMC output obtained using the function `NMixMCMC`. It computes (posterior predictive) estimates of univariate conditional cumulative distribution functions.

**Usage**

```r
NMixPredCondCDFMarg(x, ...)  
```

**Arguments**

- `x`  
  an object of class `NMixMCMC` for `NMixPredCondCDFMarg.NMixMCMC` function. An object of class `GLMM_MCMC` for `NMixPredCondCDFMarg.GLMM_MCMC` function. A list with the grid values (see below) for `NMixPredCondCDFMarg.default` function.

- `icond`  
  index of the margin by which we want to condition
prob a numeric vector. If given then also the posterior pointwise quantiles of the conditional cdf’s are computed for probabilities given by prob. These can be used to draw pointwise credible intervals.

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.

K either a number (when Krandom=FALSE) 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.

grid a list with the grid values for each margin in which the cdf should be evaluated. The value of grid[[icond]] determines the values by which we condition. If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

lgrid a length of the grid used to create the grid if that is not specified.

scaled if TRUE, the cdf of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.

... optional additional arguments.

Value
An object of class NMixPredCondCDFMarg which has the following components:

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

icond index of the margin by which we condition.

cdf a list with the computed conditional cdf’s for each value of x[[icond]]. Each cdf[[j]] is again a list with conditional cdf’s for each margin given margin icond equal to x[[icond]][j]. The value of cdf[[j]][[imargin]] gives a value of a marginal cdf of the imargin-th margin at x[[icond]][j].

prob a value of the argument prob.

qXX% if prob is given then there is one additional component named qXX%, e.g., q50% for each value of prob which has the same structure as the component cdf and keeps computed posterior pointwise quantiles.

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

Author(s)
Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also
plot.NMixPredCondCDFMarg, 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 (posterior predictive) estimates of pairwise bivariate conditional densities (given one margin).

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
- x: an object of class NMixMCMC for NMixPredCondDensJoint2.NMixMCMC function. An object of class GLMM_MCMC for NMixPredCondDensJoint2.GLMM_MCMC function. A list with the grid values (see below) for NMixPredCondDensJoint2.default 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.
- K: either a number (when Krandom=FALSE) 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.
- grid: a list with the grid values for each margin in which the density should be evaluated. The value of grid[[icond]] determines the values by which we condition.
If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

- **lgrid**: a length of the grid used to create the grid if that is not specified.
- **scaled**: if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.
- **...**: optional additional arguments.

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

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

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

### Author(s)
Arnošt Komárek <arnost.komarek@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 (posterior predictive) estimates of univariate conditional densities.

### Usage

```r
NMixPredCondDensMarg(x, ...)  
## Default S3 method:  
NMixPredCondDensMarg(x, icond, prob, scale, K, w, mu, Li, Krandom=FALSE, ...)  
## S3 method for class 'NMixMCMC'  
NMixPredCondDensMarg(x, icond, prob, grid, lgrid=50, scaled=FALSE, ...)
```
## S3 method for class 'GLMM_MCMC'
NMixPredCondDensMarg(x, icond, prob, grid, lgrid=50, scaled=FALSE, ...)

**Arguments**

- **x**: an object of class NMixMCMC for NMixPredCondDensMarg.NMixMCMC function. An object of class GLMM_MCMC for NMixPredCondDensMarg.GLMM_MCMC function.
- **icond**: a list with the grid values (see below) for NMixPredCondDensMarg.default function.
- **prob**: 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.
- **grid**: a list with the grid values for each margin in which the density should be evaluated. The value of grid[[icond]] determines the values by which we condition.
- **lgrid**: either a number (when Krandom=FALSE) 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.
- **scaled**: if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.
- **...**: optional additional arguments.

**Value**

An object of class NMixPredCondDensMarg which has the following components:

- **x**: a list with the grid values for each margin. The components of the list are named x1,... or take names from grid argument.
- **icond**: index of the margin by which we condition.
- **dens**: a list with the computed conditional densities for each value of x[[icond]]. Each dens[[j]] is again a list with conditional densities for each margin given margin icond equal to x[[icond]][j]. The value of dens[[j]][[imargin]] gives a value of a marginal density of the imargin-th margin at x[[icond]][j].
prob  a value of the argument prob.
qXX%  if prob is given then there is one additional component named qXX%, e.g., q50% for each value of prob which has the same structure as the component dens and keeps computed posterior pointwise quantiles.

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

Author(s)
Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also
plot.NMixPredCondDensMarg, NMixMCMC, GLMM_MCMC.

NMixPredDA  Discriminant analysis based on MCMC output from the mixture model

Description
It performs discriminant analysis based on sampled (re-labeled) MCMC chains from the mixture model fitted with NMixMCMC function. Observations to be discriminated may be censored.

Discrimination is based on posterior predictive probabilities of belonging to (re-labeled) mixture components.

Usage
NMixPredDA(object, y0, y1, censor, inity, info)

Arguments
object  an object of class NMixMCMC
y0     vector, matrix or data frame with observations (or limits of censored-observations) to be clustered. See NMixMCMC for details.
       If y0 is not given then the function discriminates original observations used to generate MCMC sample stored in object.
y1     vector, matrix or data frame with upper limits of interval-censored observations (if there are any). See NMixMCMC for details.
censor vector, matrix or data frame with censoring indicators (if there are any censored observations). See NMixMCMC for details.
inity  optional vector, matrix or data frame with initial values of censored observations (if there are any censored observations)
info   number which specifies frequency used to re-display the iteration counter during the computation.
Value
A data.frame with columns labeled prob1,..., probp giving posterior predictive probabilities of belonging to each component and a column labeled component giving the index of the component with the highest component probability.

Author(s)
Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also
NMixMCMC, NMixPlugDA.

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 'GLMM_MCMC'
NMixPredDensJoint2(x, grid, lgrid=50, scaled=FALSE, ...)

Arguments

- **x**: an object of class NMixMCMC for NMixPredDensJoint2.NMixMCMC function. An object of class GLMM_MCMC for NMixPredDensJoint2.GLMM_MCMC function. A list with the grid values (see below) for NMixPredDensJoint2.default function.
- **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.
- **K**: either a number (when Krandom=FALSE) or a numeric vector with the chain for the number of mixture components.
- **w**: a numeric vector with the chain for the mixture weights.
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.

grid a list with the grid values for each margin in which the predictive density should be evaluated.

If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

lgrid a length of the grid used to create the grid if that is not specified.

scaled if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.

... optional additional arguments.

Value

An object of class NMixPredDensJoint2 which has the following components:

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

freqK frequency table for the values of K (numbers of mixture components) in the MCMC chain.

propK proportions derived from freqK.

MCMC.length the length of the MCMC used to compute the predictive densities.

dens a list with the computed predictive densities for each pair of margins. The components of the list are named 1-2, 1-3, ..., i.e., dens[[1]] = dens["1-2"] 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.

densK a list with the computed predictive densities for each margin, conditioned further by K. The components of the list are named 1-2, 1-3, ..., i.e., dens[[1]][[1]] = dens["1-2"][[1]] is the pairwise predictive density for margins 1 and 2 conditioned by K = 1, dens[[1]][[2]] = dens["1-2"][[2]] is the pairwise predictive density for margins 1 and 2 conditioned by K = 2 etc.

Note that densK provides some additional information only when Krandom = TRUE or when x 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@mff.cuni.cz>

References

See Also

plot.NMixPredDensJoint2, NMixMCMC, GLMM_MCMC, NMixPredDensMarg.

Examples

```r
# See additional material available in
# YOUR_R_DIR/library/mixAK/doc/
# or YOUR_R_DIR/site-library/mixAK/doc/
# https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Galaxy.pdf
# https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Faithful.pdf
# https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/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**

```r
NMixPredDensMarg(x, ...)
```

#### Default S3 method:

```r
NMixPredDensMarg(x, scale, K, w, mu, Li, Krandom=TRUE, ...)
```

#### S3 method for class 'NMixMCMC'

```r
NMixPredDensMarg(x, grid, lgrid=500, scaled=FALSE, ...)
```

#### S3 method for class 'GLMM_MCMC'

```r
NMixPredDensMarg(x, grid, lgrid=500, scaled=FALSE, ...)
```

**Arguments**

- `x` an object of class `NMixMCMC` for `NMixPredDensMarg`. `NMixMCMC` function. An object of class `GLMM_MCMC` for `NMixPredDensMarg`. `GLMM_MCMC` function. A list with the grid values (see below) for `NMixPredDensMarg`. `default` function.
- `scale` a two component list giving the shift and the scale.
- `K` either a number (when `Krandom=FALSE`) 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.

grid
a numeric vector or a list with the grid values in which the predictive density should be evaluated.
If x$dim is 1 then grid may be a numeric vector. If x$dim is higher than then grid must be a list with numeric vectors as components giving the grids for each margin.
If grid is not specified, it is created automatically using the information from the posterior summary statistics stored in x.

lgrid
a length of the grid used to create the grid if that is not specified.
scaled
if TRUE, the density of shifted and scaled data is summarized. The shift and scale vector are taken from the scale component of the object x.

... optional additional arguments.

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

x
a list with the grid values for each margin. The components of the list are named x1, . . . or take names from grid argument.
freqK
frequency table for the values of K (numbers of mixture components) in the MCMC chain.
propK
proportions derived from freqK.
MCMC.length
the length of the MCMC used to compute the predictive densities.
dens
a list with the computed predictive densities for each margin. The components of the list are named 1, . . . , i.e., dens[[1]] = dens["1"] is the predictive density for margin 1 etc.
densK
a list with the computed predictive densities for each margin, conditioned further by K. The components of the list are named 1, . . . . That is, dens[[1]][[1]] = dens["1"] is the predictive density for margin 1 conditioned by K = 1, dens[[1]][[2]] = dens["1"] is the predictive density for margin 1 conditioned by K = 2 etc.

Note that densK provides some additional information only when Krandom = TRUE or when x 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@mff.cuni.cz>

References
NMixPseudoGOF

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/
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Galaxy.pdf
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Faithful.pdf
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Tandmob.pdf
##

NMixPseudoGOF

Pseudo goodness-of-fit test for a normal mixture model

Description

It takes a (fitted) normal mixture, creates hyperrectangles according to a specified grid, computes probability masses in each hyperrectangle derived from the (fitted) normal mixture. From computed probability masses expected frequencies (using the sample size of supplied data) are computed and compared to frequencies observed in supplied data. From expected and observed frequencies, a Pearson chi-squared like statistic is computed and returned together with residuals derived from that statistic.

Also pseudo degrees of freedom are returned which are equal to a number of hyperrectangles minus number of free parameters of the normal mixture. For a $K$-component mixture of dimension $p$, the number of free parameters is computed as

$$q = K - 1 + K \cdot p + K \cdot p(p + 1)/2$$

Note that computation of $q$ does not take into account the positive (semi-)definiteness restriction on covariance matrices.

WARNING: There is no statistical theory developed that would guarantee that computed chi-squared like statistics follows a chi-squared distribution with computed pseudo degrees of freedom under the null hypothesis that the distribution that generated the data is a normal mixture. This function serves purely for descriptive purposes!

Usage

NMixPseudoGOF(x, ...)

## Default S3 method:
NMixPseudoGOF(x, scale, w, mu, Sigma, breaks, nbreaks=10, digits=3, ...)

## S3 method for class 'NMixMCMC'
NMixPseudoGOF(x, y, breaks, nbreaks=10, digits=3, ...)
Arguments

x  data object (see argument y below) for `NMixPseudoGOF.default` function. An object of class `NMixMCMC` for `NMixPseudoGOF.NMixMCMC` function.

y  a numeric vector, matrix or data frame with the data. It is a numeric vector if \( p \) is one. It is a matrix or data frame with \( p \) columns if \( p > 1 \).

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 mixture weights. The length of this vector determines the number of mixture components.

mu  a matrix with mixture means in rows. That is, \( \mu \) 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 mixture covariance matrices.

breaks  a numeric vector or a list with the breaks defining the hyperrectangles. It is a numeric vector if \( p \) is equal to one. It is a list of length \( p \) of numeric vectors. Each component of the list determines the breaks for each margin.

nb breaks  a number or a numeric vector with the number of breaks for each margin. It is only used if the argument `breaks` is not given to determine sensible break values.

digits  a number or a numeric vector with the number of digits to which the breaks should be rounded in the case they are created by the function. If it is a vector then different rounding may be used for each margin.

...  optional additional arguments.

Value

ADD DESCRIPTION

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also

`NMixMCMC`.

---

`NMixRelabel`  Re-labeling the MCMC output of the mixture model
Description

This function takes an object generated by the \texttt{NMixMCMC} or \texttt{GLMM_MCMC} function and internally re-labels the mixture components using selected re-labeling algorithm. It also computes posterior summary statistics for mixture means, weights, variances which correspond to newly labeled MCMC sample. Further, posterior component probabilities (\texttt{poster.comp.prob_u} and \texttt{poster.comp.prob_b} components of the object \texttt{object}) are updated according to the newly labeled MCMC sample.

This function only works for models with a fixed number of mixture components.

Usage

\begin{verbatim}
NMixRelabel(object, type=c("mean", "weight", "stephens"), par, ...)

## Default S3 method:
NMixRelabel(object, type = c("mean", "weight", "stephens"), par, ...)

## S3 method for class 'NMixMCMC'
NMixRelabel(object, type = c("mean", "weight", "stephens"), par,
            prob=c(0.025, 0.5, 0.975), keep.comp.prob = FALSE, info, ...)

## S3 method for class 'NMixMCMClist'
NMixRelabel(object, type = c("mean", "weight", "stephens"), par,
            prob=c(0.025, 0.5, 0.975), keep.comp.prob = FALSE, info,
            silent = FALSE, parallel = FALSE, ...)

## S3 method for class 'GLMM_MCMC'
NMixRelabel(object, type = c("mean", "weight", "stephens"), par,
            prob = c(0.025, 0.5, 0.975), keep.comp.prob = FALSE, info,
            silent = FALSE, ...)

## S3 method for class 'GLMM_MCMClist'
NMixRelabel(object, type = c("mean", "weight", "stephens"), par,
            prob = c(0.025, 0.5, 0.975), keep.comp.prob = FALSE, jointly = FALSE,
            info, silent = FALSE, parallel = FALSE, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{object} an object of appropriate class.
  \item \texttt{type} character string which specifies the type of the re-labeling algorithm.
  \item \texttt{par} additional parameters for particular re-labeling algorithms.
  \begin{itemize}
    \item \texttt{mean} \texttt{par} specifies margin which is used to order the components. It is set to 1 if not given.
    \item \texttt{weight} \texttt{par} is empty.
    \item \texttt{stephens} \texttt{par} is a list with components \texttt{type.init}, \texttt{par}, \texttt{maxiter}.
      Component \texttt{type.init} is a character string being equal to either of “identity”, “mean”, “weight”. It determines the way which is used to obtain initial re-labeling.
  \end{itemize}
\end{itemize}
Component `par` determines the margin in the case that `type.init` is equal to “mean”.
Component `maxiter` determines maximum number of iterations of the re-labeling algorithm.

`prob` probabilities for which the posterior quantiles of component allocation probabilities are computed.

`keep.comp.prob` logical. If TRUE, posterior sample of component allocation probabilities (for each subject) is kept in the resulting object.

`jointly` a logical value. If it is TRUE then both chains are processed together. In the output, all posterior summary statistics are then also related to both chains as if it is one long chain. If it is FALSE then both chains are processed independently.

`info` number which specifies frequency used to re-display the iteration counter during the computation.

`silent` a logical value indicating whether the information on the MCMC progress is to be suppressed.

`parallel` a logical value indicating whether parallel computation (based on a package `parallel`) should be used (if possible) for re-labelling of the two chains.

`...` optional additional arguments.

**Value**

An object being equal to the value of the `object` argument in which the following components are updated according to new labeling of the mixture components.

**Value for NMixMCMC object**

When the argument `object` is of class `NMixMCMC`, the resulting object is equal to `object` with the following components being modified:

- `relabel` see `NMixMCMC`
- `order` see `NMixMCMC`
- `rank` see `NMixMCMC`
- `poster.mean.w` see `NMixMCMC`
- `poster.mean.mu` see `NMixMCMC`
- `poster.mean.Q` see `NMixMCMC`
- `poster.mean.Sigma` see `NMixMCMC`
- `poster.mean.Li` see `NMixMCMC`
- `poster.comp.prob_u` see `NMixMCMC`
- `poster.comp.prob_b` see `NMixMCMC`

Additionally, new components are added, namely

```
quant.comp.prob_b` a list with the posterior quantiles of component probabilities. One list component for each quantile specified by `prob` argument.
```
comp.prob_b  posterior sample of individual component probabilities (also given random effects). It is an $M \times n \cdot K$ matrix where $M$ is the length of the posterior sample, $n$ is the number of subjects, and $K$ is the number of mixture components. Component labels correspond to the re-labelled sample. It is included in the resulting object only if keep.comp.prob argument is TRUE.

Value for GLMM_MCMC object

When the argument object is of class GLMM_MCMC, the resulting object is equal to object with the following components being modified:

- relabel_b see GLMM_MCMC
- order_b see GLMM_MCMC
- rank_b see GLMM_MCMC
- poster.mean.w_b see GLMM_MCMC
- poster.mean.mu_b see GLMM_MCMC
- poster.mean.Q_b see GLMM_MCMC
- poster.mean.Sigma_b see GLMM_MCMC
- poster.mean.Li_b see GLMM_MCMC
- poster.comp.prob_u see GLMM_MCMC
- poster.comp.prob_b see GLMM_MCMC

Additionally, new components are added, namely

- quant.comp.prob_b  a list with the posterior quantiles of component probabilities. One list component for each quantile specified by prob argument.
- comp.prob_b  posterior sample of individual component probabilities (also given random effects). It is an $M \times I \cdot K$ matrix where $M$ is the length of the posterior sample, $I$ is the number of subjects, and $K$ is the number of mixture components. Component labels correspond to the re-labelled sample. It is included in the resulting object only if keep.comp.prob argument is TRUE.
- poster.comp.prob a matrix with the posterior means of component probabilities which are calculated with random effects integrated out.
- quant.comp.prob a list with the posterior quantiles of component probabilities. One list component for each quantile specified by prob argument.
- comp.prob  posterior sample of individual component probabilities (with random effects integrated out). It is an $M \times I \cdot K$ matrix where $M$ is the length of the posterior sample, $I$ is the number of subjects, and $K$ is the number of mixture components. Component labels correspond to the re-labelled sample. It is included in the resulting object only if keep.comp.prob argument is TRUE.

Remark. These are the component probabilities which should normally be used for clustering purposes.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>
References


See Also

NMixMCMC, GLMM_MCMC.

Examples

## See also additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## - file PBCseq.R and
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/PBCseq.pdf
##
## ==NMixSummComp==

Description

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

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

Usage

NMixSummComp(x)

## Default S3 method:
NMixSummComp(x)
## S3 method for class 'NMixMCMC'
NMixSummComp(x)

## S3 method for class 'GLMM_MCMC'
NMixSummComp(x)

### Arguments

- **x**: an object of class `NMixMCMC` or `GLMM_MCMC`

### Value

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

### Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

### See Also

`NMixMCMC`, `GLMM_MCMC`.

---

**PBC910**

*Subset of Mayo Clinic Primary Biliary Cholangitis (Cirrhosis) data*

### Description

This is a subset of `PBCseq` data which contains only data from 260 patients known to be alive and without liver transplantation at 910 days of follow-up. Furthermore, only a selection of longitudinal measurements is included and only those measurements that were obtained by 910 days. The `PBC910` dataset was used in papers Komárek and Komárková (2013, 2014).

### Usage

```r
data(PBCseq)
```

### Format

a data frame with 918 rows and the following variables

- **id**: identification number of a patient
- **day**: number of days between enrollment and this visit date (all measurements below refer to this date)
- **month**: number of months between enrollment and this visit date
- **fu.days**: total number of follow up days
- **delta.ltx.death**: 0/1 censoring indicator for event = death or liver transplantation related to `fu.days`
- **lbili**: natural logarithm of above
platelet  platelet count
spiders  0/1 presence of blood vessel malformations in the skin
jspiders  jittered version of a variable spiders

Source

URL: http://lib.stat.cmu.edu/datasets/

References


See Also

PBC910, pbc, pbcseq

Examples

data(PBC910)
summary(PBC910)

---

**PBCseq**  
*Mayo Clinic Primary Biliary Cholangitis (Cirrhosis), sequential data*

Description

This data is a continuation of the PBC data set (*pbc*), and contains the follow-up laboratory data for each study patient. An analysis based on the data can be found in Murtagh et al. (1994).

The primary PBC data set contains only baseline measurements of the laboratory parameters. This data set contains multiple laboratory results, but only on the 312 randomized patients. Some baseline data values in this file differ from the original PBC file, for instance, the data errors in prothrombin time and age which were discovered after the original analysis (see Fleming and Harrington, 1991, figure 4.6.7).

Usage

data(PBCseq)
Format

a data frame with 1 945 rows and the following variables

- **id**: identification number of a patient
- **sex**: 0/1 for male and female
- **fsex**: factor of above
- **drug**: 0/1 for placebo and D-penicillamine
- **fdrug**: factor of above
- **age**: age at entry in years
- **fu.days**: total number of follow up days
- **alive**: number of days when the patient is known to be alive and without liver transplantation
- **status**: status at endpoint, 0/1/2 for censored, liver transplant, dead
- **fstatus**: factor of above
- **delta.death**: 0/1 censoring indicator for event = death (i.e., liver transplantation means censoring)
- **delta.ltx.death**: 0/1 censoring indicator for event = death or liver transplantation
- **day**: number of days between enrollment and this visit date (all measurements below refer to this date)
- **month**: number of months between enrollment and this visit date
- **ascites**: 0/1 presence of ascites
- **fascites**: factor of above
- **hepatom**: 0/1 presence of hepatomegaly or enlarged liver
- **fhepatom**: factor of above
- **spiders**: 0/1 presence of blood vessel malformations in the skin
- **fspiders**: factor of above
- **edema**: presence and status of edema, 0 for no edema, 0.5 for untreated or successfully treated edema, 1 for edema despite diuretic therapy
- **fedema**: factor of above
- **stage**: histologic stage of disease (needs biopsy)
- **fstage**: factor of above
- **bili**: serum bilirubin (mg/dl)
- **lbili**: natural logarithm of above
- **albumin**: serum albumin (mg/dl)
- **lalbumin**: natural logarithm of above
- **alk.phos**: alkaline phosphotase (U/liter)
- **lalk.phos**: natural logarithm of above
- **chol**: serum cholesterol (mg/dl)
- **lchol**: natural logarithm of above
- **sgot**: serum glutamic-oxaloacetic transaminase (the enzyme name has subsequently changed to “ALT” in the medical literature) (U/ml)
lsgot  natural logarithm of above
platelet  platelet count
lplatelet  natural logarithm of above
protime  standardised blood clotting time
lprotime  natural logarithm of above

Source

URL: http://lib.stat.cmu.edu/datasets/

References


Murtaugh, P. A., Dickson, E. R., Van Dam, G. M., Malinchoc, M., Grambsch, P. M., Langworthy,
repeated patient visits. Hepatology, 20, 126-134.

New York: Springer-Verlag.

See Also

pb, pbcseq

Examples

data(PBCseq)
summary(PBCseq)

plot.NMixPlugCondDensJoint2

Plot computed pairwise bivariate conditional densities (plug-in esti-
mate)

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,
     add.contour=TRUE, col.add.contour="brown",
     auto.layout=TRUE,
     col, lwd=1, main, xylab, ...)

Arguments

x       an object of class NMixPlugCondDensJoint2.
ixcond  if given then conditional densities of all pairs of margins given x[[icond]][[ixcond]]
         are plotted where icond is taken from x.
imargin vector of length 2. if given then conditional densities of the (imargin[1],
         imargin[2]) pair of margins given all values of x[[icond]] are plotted.
contour logical. If TRUE then contours are drawn, otherwise image plot is created.
add.contour logical. If TRUE and contour is FALSE (i.e., image plot is drawn) then contours
         are added to the image plots.
col.add.contour color of contours which are added to the image plot.
auto.layout if TRUE then the function determines itself how to divide the plotting region to
             draw densities for all margins.
col       color used to draw the contours or images.
lwd       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@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, ylim,
     annot=TRUE, ...)

Arguments

x

an object of class NMixPlugCondDensMarg.

ixcond

if given then conditional densities of all margins given x[[icond]][ixcond] are plotted where icond is taken from x.

imargin

if given then conditional densities of the imargin-th margin given all values of x[[icond]] are plotted - either separately or all in one plot.

over

logical. If TRUE and imargin is given then all conditional densities are drawn in one plot.

auto.layout

if TRUE then the function determines itself how to divide the plotting region to draw the computed densities.

type

type of the plot.

lwd

line width.

col

color used to draw the lines. It can be a vector in which case different lines are drawn in different colors.

lty

type of the line. It can be a vector in which case different lines are drawn in different types.

main

main title of the plot. Either character which is replicated or a vector of characters.

xlab

label for the x-axis. Either character which is replicated or a vector of characters.

ylab

label for the y-axis. Either character which is replicated or a vector of characters.

ylim

limits for the y-axis.

annot

if TRUE and imargin is given and over is TRUE then a legend is added to the plot.

...

additional arguments passed to the plot function.
Value
invisible(x)

Author(s)
Arnošt Komárek <arnost.komarek@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
```r
## S3 method for class 'NMixPlugDensJoint2'
plot(x, contour=FALSE, 
     add.contour=TRUE, col.add.contour="brown", 
     auto.layout=TRUE, 
     col, lwd=1, main, xylab, ...)
```

Arguments
- **x**: an object of class `NMixPlugDensJoint2`.
- **contour**: logical. If TRUE then contours are drawn, otherwise image plot is created.
- **add.contour**: logical. If TRUE and `contour` is FALSE (i.e., image plot is drawn) then contours are added to the image plots.
- **col.add.contour**: color of contours which are added to the image plot.
- **auto.layout**: if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.
- **col**: color used to draw the contours or images.
- **lwd**: 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.
plot.NMixPlugDensMarg

Value

invisible(x)

Author(s)

Arnošt Komárek <arnost.komarek@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

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

Value

invisible(x)
### plot.NMixPredCDFMarg

**Author(s)**

Arnošt Komárek <arnost.komarek@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**

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

**Arguments**

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

**Value**

`invisible(x)`
plot.NMixPredCondCDFMarg

Author(s)
Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References

See Also
NMixPredCDFMarg, NMixMCMC.

plot.NMixPredCondCDFMarg

Plot computed univariate conditional predictive cumulative distribution functions

Description
This is a basic plotting tool to visualize computed posterior predictive estimates of univariate conditional cdf’s, see NMixPredCondCDFMarg.

Usage
```r
## S3 method for class 'NMixPredCondCDFMarg'
plot(x, ixcond, imargin, prob, over=FALSE,
     auto.layout=TRUE, type="l", lwd=1, lty, col, qlwd=1, qlty, qcol,
     main, xlab, ylab, ylim,
     annot=TRUE, ...)
```

Arguments
- `x` an object of class NMixPredCondCDFMarg.
- `ixcond` if given then conditional cdf’s of all margins given `x[[icond]][ixcond]` are plotted where icond is taken from x.
- `imargin` if given then conditional cdf’s of the imargin-th margin given all values of `x[[icond]]` are plotted - either separately or all in one plot.
- `prob` probabilities of pointwise posterior quantiles which should be added to the plot. Computed values of requested posterior quantiles must be present in the object x (see argument prob of NMixPredCondCDFMarg).
- `over` logical. If TRUE and imargin is given then all conditional cdf’s are drawn in one plot.
- `auto.layout` if TRUE then the function determines itself how to divide the plotting region to draw the computed cdf’s.
plot.NMixPredCondDensJoint2

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

Usage

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

Arguments

x an object of class NMixPredCondDensJoint2.

ixcond if given then conditional densities of all pairs of margins given x[[i-cond]][ixcond] are plotted where i-cond is taken from x.

imargin vector of length 2. if given then conditional densities of the (imargin[1], imargin[2]) pair of margins given all values of x[i-cond] are plotted.

contour logical. If TRUE then contours are drawn, otherwise image plot is created.

add.contour logical. If TRUE and contour is FALSE (i.e., image plot is drawn) then contours are added to the image plots.

col.add.contour color of contours which are added to the image plot.

auto.layout if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.

col color used to draw the contours or images.

lwd 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@mff.cuni.cz>

See Also

NMixPredCondDensJoint2, NMixMCMC.

Description

This is a basic plotting tool to visualize computed posterior predictive estimates of univariate conditional densities, see NMixPredCondDensMarg.
Usage

## S3 method for class 'NMixPredCondDensMarg'

plot(x, ixcond, imargin, prob, over=FALSE, 
     auto.layout=TRUE, type="l", lwd=1, lty, col, qlwd=1, qlty, qcol, 
     main, xlab, ylab, ylim, 
     annot=TRUE, ...)  

Arguments

- **x**
  - an object of class NMixPredCondDensMarg.
- **ixcond**
  - if given then conditional densities of all margins given \(x[[\text{icond}]]\) are plotted where \(\text{icond}\) is taken from \(x\).
- **imargin**
  - if given then conditional densities of the \(\text{imargin}\)-th margin given all values of \(x[[\text{icond}]]\) are plotted - either separately or all in one plot.
- **prob**
  - probabilities of pointwise posterior quantiles which should be added to the plot. Computed values of requested posterior quantiles must be present in the object \(x\) (see argument \(\text{prob}\) of \(\text{NMixPredCondDensMarg}\)).
- **over**
  - logical. If \(\text{TRUE}\) and \(\text{imargin}\) is given then all conditional densities are drawn in one plot.
- **auto.layout**
  - if \(\text{TRUE}\) then the function determines itself how to divide the plotting region to draw the computed densities.
- **type**
  - type of the plot.
- **lwd**
  - line width.
- **lty**
  - type of the line. It can be a vector in which case different lines are drawn in different types.
- **col**
  - color used to draw the lines. It can be a vector in which case different lines are drawn in different colors.
- **qlwd**
  - line width for pointwise posterior quantiles.
- **qlty**
  - type of the line for pointwise posterior quantiles.
- **qcol**
  - color used to draw pointwise posterior quantiles.
- **main**
  - main title of the plot. Either character which is replicated or a vector of characters.
- **xlab**
  - label for the x-axis. Either character which is replicated or a vector of characters.
- **ylab**
  - label for the y-axis. Either character which is replicated or a vector of characters.
- **ylim**
  - limits for the y-axis.
- **annot**
  - if \(\text{TRUE}\) and \(\text{imargin}\) is given and \(\text{over}\) is \(\text{TRUE}\) then a legend is added to the plot.
- **...**
  - additional arguments passed to the \text{plot} function.

Value

invisible(x)
plot.NMixPredDensJoint2

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

Arguments
- `x`: an object of class NMixPredDensJoint2.
- `K`: if equal to 0 then the overall predictive densities are plotted taken from the dens part of the object `x`. If higher than 0 then the predictive density conditioned by the value of `K` is plotted (taken from the densK part of the object `x`).
- `contour`: logical. If TRUE then contours are drawn, otherwise image plot is created.
- `add.contour`: logical. If TRUE and contour is FALSE (i.e., image plot is drawn) then contours are added to the image plots.
- `col.add.contour`: color of contours which are added to the image plot.
- `auto.layout`: if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.
- `col`: color used to draw the contours or images.
- `lwd`: 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@mff.cuni.cz>

References


See Also

NMixPredDensJoint2, NMixMCMC.

Examples

## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/

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

x  an object of class NMixPredDensMarg.

K  if equal to 0 then the overall predictive densities are plotted taken from the dens part of the object x.
   If higher than 0 then the predictive density conditioned by the value of K is plotted (taken from the densK part of the object x).
auto.layout if TRUE then the function determines itself how to divide the plotting region to draw densities for all margins.

type type of the plot.

col color used to draw the lines.

lty type of the line.

lwd line width.

main main title of the plot. Either character which is replicated or a vector of characters of the length equal to the number of margins.

xlab label for the x-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.

ylab label for the y-axis. Either character which is replicated or a vector of characters of the length equal to the number of margins.

... additional arguments passed to the plot function.

Value

invisible(x)

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References


See Also

NMixPredDensMarg, NMixMCMC.

Examples

```r
## See additional material available in
## YOUR_R_DIR/library/mixAK/doc/
## or YOUR_R_DIR/site-library/mixAK/doc/
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Galaxy.pdf
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Faithful.pdf
## https://www2.karlin.mff.cuni.cz/~komarek/software/mixAK/Tandmob.pdf
##
```
plotProfiles  

*Plot individual longitudinal profiles*

**Description**

It creates a plot of individual longitudinal profiles. It is based on the output from `getProfiles` function.

**Usage**

```r
plotProfiles(ip, data, var, trans, tvar, gvar, 
auto.layout=TRUE, lines=TRUE, points=FALSE, add=FALSE, 
xlab="Time", ylab, xaxt="s", yaxt="s", xlim, ylim, main, 
lcol, col, bg, lty=1, lwd=1, pch=21, cex.points=1, 
highlight, lines.highlight=TRUE, points.highlight=TRUE, 
lcol.highlight="red3", col.highlight="red3", bg.highlight="orange", 
lty.highlight=1, lwd.highlight=2, 
pch.highlight=23, cex.highlight=1)
```

**Arguments**

- `ip`: output from `getProfiles` function containing extracted individual longitudinal profiles of each subject.
- `data`: data.frame used to produce `ip`. It is used to detect ranges for some variables.
- `var`: character string identifying the response variable to plot.
- `trans`: possible transformation of the response variable.
- `tvar`: character string identifying the time variable.
- `gvar`: character string identifying the group variable for which different colors are used.
- `auto.layout`: logical. If `TRUE`, the layout of the plotting region is determined automatically.
- `lines`: logical. If `TRUE`, lines are drawn in the plot connecting observations within individuals.
- `points`: logical. If `TRUE`, points are added to the plot.
- `add`: logical. If `TRUE`, the new plot overlays the old one.
- `lcol`: color for lines.
- `col`: color for points.
- `xlab`, `ylab`, `xaxt`, `yaxt`, `xlim`, `ylim`, `main`, `bg`, `lty`, `lwd`, `pch`: arguments passed to standard plotting functions. `col` might also be a vector in which case different colors are used for profiles from different groups identified by the `gvar` variable.
- `cex.points`: passed as a cex argument to `points` function used when `points = TRUE`. 
highlight  an optional numeric vector giving the indices of ip for which the longitudinal profiles should be highlighted.

lines.highlight logical. If TRUE, highlighting is done using lines.

points.highlight logical. If TRUE, highlighting is done using points.

lcol.highlight, col.highlight, bg.highlight, lty.highlight, lwd.highlight, pch.highlight, cex.highlight arguments col, bg, lty, lwd, pch, cex passed to lines and/or points functions which provide highlighting of selected profiles.

Value

Invisible ip.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

See Also

getProfiles.

Examples

data(PBCseq, package="mixAK")
ip <- getProfiles(t="day", y=c("age", "fdrug", "bili", "platelet", "spiders"),
                   id="id", data=PBCseq)

XLIM <- c(0, 910)
lcol2 <- c("darkgreen", "red")

oldPar <- par(mfrow=c(1, 3), bty="n")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="bili", trans=log, tvar="day", gvar="fdrug",
             xlab="Time (days)", col=lcol2, auto.layout=FALSE, main="Log(bilirubin)",
             highlight=c(2, 4), col.highlight="darkblue")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="platelet", tvar="day", gvar="fdrug",
             xlab="Time (days)", col=lcol2, auto.layout=FALSE, main="Platelet count",
             highlight=c(2, 4), col.highlight="darkblue")
plotProfiles(ip=ip, data=PBCseq, xlim=XLIM, var="spiders", tvar="day", gvar="fdrug",
             xlab="Time (days)", col=lcol2, auto.layout=FALSE,
             lines=FALSE, points=TRUE,
             highlight=c(2, 4), col.highlight="darkblue", bg.highlight="skyblue")
par(oldPar)
Random rotation matrix

Description

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

a) $PP' = I$,

b) $P'P = I$,

c) $\det(P) = 1$.

Usage

rRotationMatrix(n, dim)

Arguments

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

Details

For $dim = 2$, $p_{2,1} (\sin(\theta))$ is generated from 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 $P$ is generated in the following steps:

1) Generate a $p \times p$ matrix $A$ with independent Unif(0, 1) elements and check whether $A$ is of full rank $p$.

2) Computes a QR decomposition of $A$, i.e., $A = QR$ where $Q$ satisfies $QQ' = I$, $Q'Q = I$, $\det(Q) = (-1)^{p+1}$, and columns of $Q$ spans the linear space generated by the columns of $A$.

3) For odd $dim$, return matrix $Q$. For even $dim$, return corrected matrix $Q$ 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@mff.cuni.cz>

References

Examples

```r
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 
\((1, 2), (1, 3), \ldots, (1, K), (2, 3), \ldots, (2, K), \ldots, (K - 1, K)\).

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@mff.cuni.cz>
SimData

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)

SimData

Simulated dataset

Description

A simulated dataset used as an example dataset in Komárek and Komárková (2014).

Usage

data(SimData)

Format

a data frame with 1 157 rows and the following variables

id  identification number of a subject.
tday visit time in days.
tmonth visit time in months.
yN response variable generated according to a linear mixed model with normal errors. It intentionally contains 50 NA’s.
yP response variable generated according to a Poisson generalized linear mixed model. It intentionally contains 50 NA’s.
yB response variable generated according to a Bernoulli generalized linear mixed model. It intentionally contains 50 NA’s.
yBjit a jittered version of yB.

References


Examples

data(SimData)
summary(SimData)
SP2Rect  

Conversion of a symmetric matrix stored in a packed format (lower triangle only) into a matrix

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@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)

summaryDiff  

Posterior summary statistics for a difference of two quantities

Description

It calculates (posterior) summary statistics for a difference of two quantities supplied as (MCMC) samples. Within the mixAK package, it is primarily used to calculate posterior summary for the difference of the deviances of two competing models.

Usage

summaryDiff(x, y, prob=c(0.025, 0.5, 0.975), cut=c(-2*log(9), 0), na.rm=TRUE)
Arguments

- **x**: a numeric vector with the sample of the first quantity.
- **y**: a numeric vector with the sample of the second quantity to be subtracted from x.
- **prob**: a numeric vector of probabilities for quantiles to be calculated from the sample of differences.
- **cut**: numeric value(s) which specify the cutoff(s) we are interested in estimating $P(x - y < \text{cut})$ from the sample. The default values are motivated by the arguments given in Section 4 of Aitkin, Liu and Chadwick (2009) and in Section 7.5 of Aitkin (2010).
- **na.rm**: logical indicating on how to handle NA’s.

Value

A list with the components

- **summary**: a named vector with the (posterior) summary statistics based on the differences.
- **Pcut**: estimated (posterior) probabilities that the difference lies below the cut values.

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References


Examples

```r
set.seed(16336886)
x <- runif(100, 0, 100)
y <- runif(100, 0, 100)
sdiff <- summaryDiff(x, y)
```

---

`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
Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat)
Katholieke Universiteit Leuven
Kapucijnenvoer 35
B-3000 Leuven
Belgium
<emmanuel.lesaffre@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 DDmmYY
    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\n
**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.
TandmobEmer

Source

Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat), Katholieke Universiteit Leuven, Kapucijnenvoer 35, 3000 Leuven, Belgium

URL: https://gbiomed.kuleuven.be/english/research/5000687/5000696/

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


Examples

```r
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
Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat)
Katholieke Universiteit Leuven
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 DDmmYY

**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

Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat), Katholieke Universiteit Leuven, Kapucijnenvoer 35, 3000 Leuven, Belgium

URL: https://gbiomed.kuleuven.be/english/research/50000687/50000696/

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


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

mean  a numeric vector of the mean of the original multivariate normal distribution.
Sigma  covariance matrix of the original multivariate normal distribution.
a  a numeric vector of the same length as mean of truncation limits 1.
b  a numeric vector of the same length as mean of truncation limits 2.
trunc  a numeric vector of the same length as mean describing the type of truncation in each margin.
  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 (−∞, 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 trunc is not given, it is assumed that it is equal to 4. Note that a, b and trunc must have the same length, with exception that b does not have to be supplied if all trunc values 0, 1, 2 or 4.

xinit  a numeric vector of the same length as mean with the initial value for the Gibbs sampler. If it is not supplied, the function determines itself the initial value.
n  number of observations to be sampled.

Value

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

Author(s)

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References


See Also

rTNorm.
## Not run:
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)
  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], ", Sample rho = ", RHO, sep=""))
  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(3, 3))
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(3, 3))
x3 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=(-100, -100), b=c(100, 100),
  trunc=c(3, 3))
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(3, 3))
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(3, 3))
x6 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=c(0, 0), b=c(0, 2),
  trunc=c(0, 2))
x7 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=-1, b=1, trunc=c(0, 2))
x8 <- exam2(1000, mu=c(1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=-1, b=1, trunc=c(1, 2))
x9 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=1.5, b=0.5, trunc=c(0.5, 1.5))
x10 <- exam2(1000, mu=c(-1, 1), sigma=c(1, sqrt(2)), rho=0.7, a=-1.5, b=0.5, trunc=c(4, 3))

## End(Not run)
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

\[ r \text{TNorm}(n, \text{mean}=0, \text{sd}=1, a, b, \text{trunc}) \]

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 mean and sd 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 \((a, \infty)\). 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 trunc is not given, it is assumed that it is equal to 4. Note that \( a \), \( b \) and trunc must have the same length, either 1 or \( n \) with exception that \( b \) does not have to be supplied if trunc 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@mff.cuni.cz>

References

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, Infty)")
hist(x6, main="TN(10, 3^2, 13, 13)")
hist(x3, main="TN(10, 3^2, -Infty, 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,]
s <- sum(x7[1:(n/5)] > a[1:(n/5)])  ## must be equal to n/5

cbind(trunc, a, x7)[201:210,]
s <- 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,]
```r
tracePlots

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)[601:810,]

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))
```

### TracePlots

Traceplots for selected parameters

#### Description

This function draws traceplots of selected parameters from the MCMC simulations ran using `NMixMCMC` or `GLMM_MCMC` functions.
Usage

tracePlots(x, ...)

## Default S3 method:
tracePlots(x, ...)

## S3 method for class 'NMixMCMC'
tracePlots(x, param=c("Emix", "SDmix", "Cormix", "K", "w", "mu", "sd", "gammaInv"),
            relabel=FALSE, order,
            auto.layout=TRUE, xlab="Iteration", ylab, col="slateblue", main="", ...)

## S3 method for class 'NMixMCMClst'
tracePlots(x, param=c("Emix", "SDmix", "Cormix", "K", "w", "mu", "sd", "gammaInv"),
            relabel=FALSE,
            auto.layout=TRUE, xlab="Iteration", ylab, col=c("blue3", "red3"), main="", ...)

## S3 method for class 'GLMM_MCMC'
tracePlots(x, param=c("Deviance", "Cond.Deviance",
                        "alpha", "E_b", "SD_b", "Cor_b", "sigma_eps",
                        "w_b", "mu_b", "sd_b", "gammaInv_b", "gammaInv_eps"),
            relabel=FALSE, order,
            auto.layout=TRUE, xlab="Iteration", ylab, col="slateblue", main="", ...)

## S3 method for class 'GLMM_MCMClst'
tracePlots(x, param=c("Deviance", "Cond.Deviance",
                        "alpha", "E_b", "SD_b", "Cor_b", "sigma_eps",
                        "w_b", "mu_b", "sd_b", "gammaInv_b", "gammaInv_eps"),
            relabel=FALSE,
            auto.layout=TRUE, xlab="Iteration", ylab, col=c("blue3", "red3"), main="", ...)

Arguments

- **x**
  - a character string which specifies which sort of parameters is to be plotted.

- **param**
  - an object of appropriate class.

  - **Emix**
    - overall means (for each margin) of the normal mixture;

  - **SDmix**
    - overall standard deviations (for each margin) of the normal mixture;

  - **Cormix**
    - overall correlations (each pair) of the normal mixture;

  - **K**
    - number of mixture components;

  - **w, w_b**
    - weights of each of mixture components. If relabel is FALSE, weights are not re-labeled before plotting;

  - **mu, mu_b**
    - component means (each margin, each mixture component) of the normal mixture. The mixture means are shifted and scaled using x$scale$shift and x$scale$scale before plotting. If relabel is FALSE, means are not re-labeled before plotting;

  - **sd, sd_b**
    - component standard deviations (each margin, each mixture component) of the normal mixture. The mixture standard deviations are scaled
using \( x \cdot \text{scale} \cdot \text{scale} \) before plotting. If \( \text{relabel} \) is \text{FALSE}, standard deviations are not re-labeled before plotting;

\text{gammaInv, gammaInv_b, gammaInv_eps} \text{ variance hyperparameters;}

\text{Deviance} \text{ deviance (marginal, with random effects integrated out) of the GLMM;}

\text{Cond.Deviance} \text{ conditional deviance (given random effects) of the GLMM;}

\text{alpha} \text{ fixed effects of the fitted GLMM;}

\text{Eb} \text{ overall means (for each margin) of the random effects of the fitted GLMM;}

\text{SDb} \text{ overall standard deviations (for each margin) of the random effects of the fitted GLMM;}

\text{Corb} \text{ overall correlations (each pair) of the distribution of the random effects of the fitted GLMM.}

\text{sigma_eps} \text{ standard deviations of the error terms in the (mixed) models for continuous responses.}

\text{relabel} \text{ logical value. It indicates whether the chains with \text{param} being \( w, \mu, sd, w_b, \mu_b, sd_b \) should be re-labeled before plotting. Re-labelling is given by argument \text{order}. If \text{order} is missing then \( x \cdot \text{order} \) or \( x \cdot \text{order}_b \) determines re-labelling.}

\text{order} \text{ a matrix with } K \text{ columns and } M \text{ rows where } M \text{ is the length of MCMC. Each row of } \text{order} \text{ must be a permutation of } (1, \ldots, K).}

\text{auto.layout} \text{ logical value. If \text{TRUE}, the plotting region is automatically divided to produce traceplots of all parameters. Note that layout must be set up automatically if there are more than 28 parameters to be plotted (often the case for correlations with \text{param} being \text{Corb} or for mixture means with \text{param} being \text{mu}_b).}

\text{xlab, ylab, col, main} \text{ arguments passed to \text{plot} function. They all can be of length one (the value is used on all plots) or of length equal to the number of parameters to be plotted.}

\text{...} \text{ other arguments passed to \text{plot} function.}

\text{Value}

\text{invisible(x)}

\text{Author(s)}

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

\text{See Also}

\text{NMixMCMC, GLMM_MCMC, NMixRelabel, traceplot.}
Description

Wishart distribution

\[ \text{Wishart}(\nu, S), \]

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

\[ \mathbb{E}(W) = \nu S. \]

Prior to version 3.4-1 of this package, functions \texttt{dWISHART} and \texttt{rWISHART} were called as \texttt{dWishart} and \texttt{rWishart}, respectively. The names were changed in order to avoid conflicts with \texttt{rWishart} from a standard package \texttt{stats}.

Usage

\[ \texttt{dWISHART}(W, df, S, \log=\text{FALSE}) \]

\[ \texttt{rWISHART}(n, df, S) \]

Arguments

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

Details

The density of the Wishart distribution is the following

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

where \( p \) is number of rows and columns of the matrix \( W \).

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 \( W \)) in rows is returned.

Author(s)
Arnošt Komárek <arnost.komarek@mff.cuni.cz>

References


See Also
rWishart.

Examples
```r
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(1,3,3,3,13), 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* \( Y = \text{trans}(T) \) *into distribution of* \( T \)*

**Description**

This method transforms fitted distribution of \( Y = \text{trans}(T) \) into distribution of \( T \). Default transformation is a logarithmic transformation where \( \text{trans}(t) = \log(t) \), \( \text{itrans}(y) = \exp(y) \), \( \text{dtrans}(t) = 1/t \).

**Usage**

Y2T(x, ...)

## S3 method for class 'NMixPredDensMarg'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPlugDensMarg'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)
## S3 method for class 'NMixPredCDFMarg'
Y2T(x, itrans=exp, ...)

## S3 method for class 'NMixPredDensJoint2'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPlugDensJoint2'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPredCondDensMarg'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPlugCondDensMarg'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPredCondCDFMarg'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPredCondDensJoint2'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

## S3 method for class 'NMixPlugCondDensJoint2'
Y2T(x, itrans=exp, dtrans=function(x){return(1 / x)}, ...)

**Arguments**

- **x**
  - an object of appropriate class.

- **itrans**
  - either an object of class function or a list of objects of class function giving inverse transformations for each margin. If itrans is a single function then it is assumed that all margins were transformed in the same way.

- **dtrans**
  - either an object of class function or a list of objects of class function giving derivatives of transformations for each margin. If dtrans is a single function then it is assumed that all margins were transformed in the same way.

- **...**
  - optional additional arguments.

**Value**

An object of the same class as argument x.

**Author(s)**

Arnošt Komárek <arnost.komarek@mff.cuni.cz>

**See Also**

NMixPredDensMarg, NMixPlugDensMarg, NMixPredCDFMarg, NMixPredDensJoint2, NMixPlugDensJoint2, NMixPredCondDensMarg, NMixPlugCondDensMarg, NMixPredCondCDFMarg, NMixPredCondDensJoint2, NMixPlugCondDensJoint2.
Index

* **algebra**
  MatMPpinv, 27
  MatSqrt, 28
* **arith**
  generatePermutations, 14
* **array**
  MatMPpinv, 27
  MatSqrt, 28
  rRotationMatrix, 97
  SP2Rect, 100
* **cluster**
  GLMM_longitDA, 15
  GLMM_longitDA2, 17
  NMixCluster, 40
  NMixPlugDA, 58
  NMixPredDA, 68
  NMixRelabel, 74
* **datasets**
  Acidity, 3
  Enzyme, 10
  Faithful, 11
  Galaxy, 13
  PBC910, 79
  PBCseq, 80
  SimData, 99
  Tandmob, 101
  TandmobEmer, 104
* **distribution**
  MVN, 30
  MVNmixture, 32
  MVT, 36
  rRotationMatrix, 97
  rSamplePair, 98
  TMVN, 106
  TNorm, 108
  Wishart, 114
* **dplot**
  autolayout, 4
  BsBasis, 6
  cbplot, 7
  getProfiles, 14
  NMixPlugCondDensJoint2, 54
  NMixPlugCondDensMarg, 56
  NMixPlugDensJoint2, 58
  NMixPlugDensMarg, 60
  NMixPredCDFMarg, 61
  NMixPredCondCDFMarg, 63
  NMixPredCondDensJoint2, 65
  NMixPredCondDensMarg, 66
  NMixPredDensJoint2, 69
  NMixPredDensMarg, 71
  plot.NMixPlugCondDensJoint2, 82
  plot.NMixPlugCondDensMarg, 84
  plot.NMixPlugDensJoint2, 85
  plot.NMixPlugDensMarg, 86
  plot.NMixPredCDFMarg, 87
  plot.NMixPredCondCDFMarg, 88
  plot.NMixPredCondDensJoint2, 89
  plot.NMixPredCondDensMarg, 90
  plot.NMixPredDensJoint2, 92
  plot.NMixPredDensMarg, 93
  plotProfiles, 95
  tracePlots, 111
* **htest**
  summaryDiff, 100
* **models**
  GLMM_longitDA, 15
  GLMM_longitDA2, 17
  GLMM_MCMC, 18
* **multivariate**
  BLA, 4
  GLMM_longitDA, 15
  GLMM_longitDA2, 17
  GLMM_MCMC, 18
  MVN, 30
  MVNmixture, 32
  MVT, 36
  NMixChainComp, 38
NMixChainsDerived, 39
NMixCluster, 40
NMixEM, 41
NMixMCMC, 43
NMixPlugCondDensJoint2, 54
NMixPlugCondDensMarg, 56
NMixPlugDA, 58
NMixPlugDensJoint2, 58
NMixPlugDensMarg, 60
NMixPredCDFMarg, 61
NMixPredCondCDFMarg, 63
NMixPredCondDensJoint2, 65
NMixPredCondDensMarg, 66
NMixPredDA, 68
NMixPredDensJoint2, 69
NMixPredDensMarg, 71
NMixPseudoGOF, 73
NMixRelabel, 74
NMixSummComp, 78
rRotationMatrix, 97
TMVN, 106
Wishart, 114

* smooth
BsBasis, 6
fitted.GLM_Model, 12
NMixChainComp, 38
NMixChainsDerived, 39
NMixPlugCondDensJoint2, 54
NMixPlugCondDensMarg, 56
NMixPlugDA, 58
NMixPlugDensJoint2, 58
NMixPlugDensMarg, 60
NMixPredCDFMarg, 61
NMixPredCondCDFMarg, 63
NMixPredCondDensJoint2, 65
NMixPredCondDensMarg, 66
NMixPredDA, 68
NMixPredDensJoint2, 69
NMixPredDensMarg, 71
NMixSummComp, 78
Y2T, 116

* survival
NMixMCMC, 43

* univar
summaryDiff, 100

* utilities
generatePermutations, 14
autolayout, 4

BLA, 4
bs, 7
BsBasis, 6

C_BLA (BLA), 4
C_deriv_1dMVT_x (MVT), 36
C_dmixMVN_R (MVNmixture), 32
C_dmixNorm_R (MVNmixture), 32
C_dMVN1_R (MVN), 30
C_dMVT1_R (MVT), 36
C_generatePermutations (generatePermutations), 14
C_GLMM_longitDA (GLMM_longitDA), 15
C_GLMM_longitDA2 (GLMM_longitDA2), 17
C_GLMM_MCMC (GLMM_MCMC), 18
C_GLMM_NMIXRelabel (NMixRelabel), 74
C_GLMM_PED (GLMM_PED), 18
C_ldWishart_R (Wishart), 114
Cระหวนrs (MatMPspinv), 27
C_NMIX_ChainsDerived (NMixChainsDerived), 39
C_NMIX_MCMC (NMixMCMC), 43
C_NMIX_NMIXRelabel (NMixRelabel), 74
C_NMIX_PED (NMixMCMC), 43
C_NMIX_PredCDFMarg (NMixPredCDFMarg), 61
C_NMIX_PredCondDensCDFMarg (NMixPredCondDensMarg), 66
C_NMIX_PredCondDensJoint2 (NMixPredCondDensJoint2), 65
C_NMIX_PredDA (NMixPredDA), 68
C_NMIX_PredDensJoint2 (NMixPredDensJoint2), 69
C_NMIX_PredDensMarg (NMixPredDensMarg), 71
C_rDirichlet_R (Dirichlet), 9
C_rmixMVN_R (MVNmixture), 32
C_rmixNorm_R (MVNmixture), 32
C_rMVN1_R (MVN), 30
C_rMVN2_R (MVN), 30
C_rMVT1_R (MVT), 36
C_rotationMatrix_R (rRotationMatrix), 97
C_rTMVN1_R (TMVN), 106
C_rTNorm1_R (TNorm), 108
C_rWishart_R (Wishart), 114
C_SamplePair_R (rSamplePair), 98
C_sqrtGE (MatSqrt), 28
cbplot, 7
contour, 59, 70, 82, 85, 89, 92
Dirichlet, 9
dMVN (MVN), 30
dMVNmixture (MVNmixture), 32
dMVNmixture2 (MVNmixture), 32
dMVT (MVT), 36
dnorm, 31, 34
dt, 37
dWISHART (Wishart), 114

Enzyme, 10
Faithful, 11
fitted.GLMM_MCMC, 12

geyser, 11
GLMM_longitDA, 15, 18
GLMM_longitDA2, 17, 17, 18
GLMM_MCMC, 12, 13, 15–18, 18, 38, 39, 41, 56,
57, 60, 61, 63, 64, 66, 68, 71, 73, 75,
77–79, 111, 113

image, 59, 70, 82, 89, 92

layout, 4
lines, 96
lmer, 20

makeCluster, 22, 48
MatMPpinv, 27
MatSqrt, 28
MVN, 30, 34
MVNmixture, 32
Mvnorm, 31, 34
MVT, 36
Mvt, 37
NMixChainComp, 38
NMixChainsDerived, 39
NMixCluster, 40
NMixEM, 41
NMixMCMC, 20, 27, 38–41, 43, 54, 56–58,
60–66, 68–76, 78, 79, 83, 85–90,
92–94, 104, 111, 113
NMixPlugCondDensJoint2, 54, 82, 83, 117

NMixPlugCondDensMarg, 56, 84, 85, 117
NMixPlugDA, 58, 69
NMixPlugDensJoint, 58, 85, 86, 117
NMixPlugDensMarg, 60, 86, 87, 117
NMixPredCDFMarg, 61, 87, 88, 117
NMixPredCondCDFMarg, 63, 88, 89, 117
NMixPredCondDensJoint2, 56, 65, 89, 90,
117
NMixPredCondDensMarg, 57, 66, 90–92, 117
NMixPredDA, 58, 68
NMixPredDensJoint2, 52, 60, 69, 73, 78, 92,
93, 117
NMixPredDensMarg, 52, 61, 71, 71, 78, 93, 94,
117
NMixPseudoGOF, 73
NMixRelabel, 38, 74, 113
NMixSummComp, 78

par, 4
pbc, 80, 82
PBC910, 79, 80
PBCseq, 79, 80
pbcseq, 80, 82
plot.default, 8
plot.NMixPlugCondDensJoint2, 56, 82
plot.NMixPlugCondDensMarg, 57, 84
plot.NMixPlugDensJoint2, 60, 85
plot.NMixPlugDensMarg, 61, 86
plot.NMixPredCDFMarg, 63, 87
plot.NMixPredCondCDFMarg, 64, 88
plot.NMixPredCondDensJoint2, 66, 89
plot.NMixPredCondDensMarg, 68, 90
plot.NMixPredDensJoint2, 71, 92
plot.NMixPredDensMarg, 73, 93
plotProfiles, 15, 95
points, 95, 96
print.GLMM_MCMC (GLMM_MCMC), 18
print.GLMM_MCMClst (GLMM_MCMC), 18
print.NMixEM (NMixEM), 41
print.NMixMCMC (NMixMCMC), 43
print.NMixMCMClst (NMixMCMC), 43
rbeta, 9
rcMVN (MVN), 30
rDirichlet (Dirichlet), 9
rMVN (MVN), 30
rMVNmixture (MVNmixture), 32
rMVNmixture2 (MVNmixture), 32
rMVT (MVT), 36
INDEX

rnorm, 110
rRotationMatrix, 97
rSamplePair, 98
rTMVN, 110
rTMVN (TMVN), 106
rTNorm, 107
rTNorm (TNorm), 108
rWISHART (Wishart), 114
rWishart, 114, 115

SimData, 99
SP2Rect, 100
summaryDiff, 100

Tandmob, 101, 104, 106
TandmobEmer, 104
TMVN, 106
TNorm, 108
traceplot, 113
tracePlots, 111

Wishart, 114

Y2T, 116