Package 'glmmAK'

December 27, 2012

Version 1.7

Date 2012-12-27

Title Generalized Linear Mixed Models

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

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

Depends R (>= 2.10.0), smoothSurv, coda, mvtnorm

Description Later

Encoding UTF-8

License GPL (>= 2)

URL http://msekce.karlin.mff.cuni.cz/~komarek

ZipData no

R topics documented:

Index **[51](#)**

AKmiscel   *Miscellaneous smaller functions*

Description

Functions `mfLand` and `mfPort` try to split a graphical device in a "nice" way to produce several plots on 1 page in landscape or portrait format.

Usage

```
mfLand(np)

mfPort(np)
```

Arguments

np   number of plots that are to be produced on 1 page

Value

A 2-component vector giving the number of rows and columns into which the graphical device could be splitted.

Author(s)

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

See Also

[par](#).

Examples

```
mfLand(6)
mfPort(6)
```

| BPvalue | *Univariate Bayesian P-value (pseudo-contour probability)* |

Description

For a sample (from the posterior distribution) of θ this function computes

$$P = 2\min\big\{P(\theta < 0),\, P(\theta > 0)\big\},$$

which can be viewed as a counterpart of a classical two-sided P-value.

Note that this is the same as a univariate pseudo-contour probability as described in Besag et al. (1995, p. 30) and in Held (2004).

Usage

```
BPvalue(sample)
```

Arguments

sample   vector, matrix or data frame with sampled values. If it is a matrix or data frame then it is assumed that each column corresponds to a separate parameter and the P-value is computed separately for each column.

Value

Vector of computed P-values.

Author(s)

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

References

Besag, J., Green, P., Higdon, D. and Mengersen, K. (1995). Bayesian computation and stochastic systems (with Discussion). *Statistical Science,* **10**, 3 - 66.

Held, L. (2004). Simultaneous posterior probability statements from Monte Carlo output. *Journal of Computational and Graphical Statistics,* **13**, 20 - 35.

Examples

```
m <- 1000
sample <- rnorm(m, mean=1)
BPvalue(sample)
## compare with
2*pnorm(0, mean=1, lower.tail=TRUE)

sample <- data.frame(x1=rnorm(m), x2=rnorm(m, mean=-1), x3=rnorm(m, mean=2))
BPvalue(sample)
```

```
## compare with
2*pnorm(0, mean=0)
2*pnorm(0, mean=-1, lower.tail=FALSE)
2*pnorm(0, mean=2, lower.tail=TRUE)
```

---

copula	*Copulas*

Description

Functions to compute the cumulative distribution functions and densities for several bivariate copulas.

These functions do not have anything to do with the GLMM's in this package. They are here simply because of an interest of the author to play with copulas a little bit.

Usage

```
Cplackett(u, v, theta=1)
Cgauss(u, v, theta=0)
Cclayton(u, v, theta=0)
cplackett(u, v, theta=1)
cgauss(u, v, theta=0)
cclayton(u, v, theta=0)
```

Arguments

u   Unif(0, 1) quantiles for the first margin. It can be a vector or a matrix

v   Unif(0, 1) quantiles for the second margin. It can be a vector or a matrix

theta   value of the association parameter

Value

A vector or a matrix with the values of the corresponding cdf or density.

Author(s)

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

References

Nelsen, R. B. (2006). *An Introduction to Copulas,* Second Edition New York: Springer.

Examples

```
### Margin1 = normal mixture
### Margin2 = normal
intcpt <- c(1, 5)
sds <- c(1, 3)
x <- seq(intcpt[1]-2.5*sds[1], intcpt[1]+1+2.5*sds[1], length=40)
y <- seq(intcpt[2]-2.5*sds[2], intcpt[2]+2.5*sds[2], length=41)
Fx <- 0.6*pnorm(x, mean=intcpt[1], sd=sds[1]) +
   0.4*pnorm(x, mean=intcpt[1]+2, sd=0.5*sds[1])
Fy <- pnorm(y, mean=intcpt[2], sd=sds[2])
fx <- 0.6*dnorm(x, mean=intcpt[1], sd=sds[1]) +
   0.4*dnorm(x, mean=intcpt[1]+2, sd=0.5*sds[1])
fy <- dnorm(y, mean=intcpt[2], sd=sds[2])
u <- matrix(rep(Fx, length(y)), ncol=length(y))
v <- matrix(rep(Fy, length(x)), nrow=length(x), byrow=TRUE)
du <- matrix(rep(fx, length(y)), ncol=length(y))
dv <- matrix(rep(fy, length(x)), nrow=length(x), byrow=TRUE)


### Copula distribution functions
theta <- c(3, 0.3, 1)
CC <- list()
CC$plackett <- Cplackett(u, v, theta=theta[1])
CC$gauss <- Cgauss(u, v, theta=theta[2])
CC$clayton <- Cclayton(u, v, theta=theta[3])

### Copula densities
cc <- list()
cc$plackett <- cplackett(u, v, theta=theta[1]) * du * dv
cc$gauss <- cgauss(u, v, theta=theta[2]) * du * dv
cc$clayton <- cclayton(u, v, theta=theta[3]) * du * dv

### Figures
lcol <- "red"
pcol <- "seagreen2"
mains <- paste(c("Plackett ", "Gauss ", "Clayton "), "copula, theta=", theta, sep="")
zlab <- "F(x,y)"
zlab2 <- "f(x,y)"
tangle <- -25
ptangle <- 40

oldpar <- par(bty="n", mfcol=c(2, 3))
contour(x, y, cc$plackett, main=mains[1], col=lcol)
persp(x, y, cc$plackett, zlab=zlab2, main=mains[1], col=pcol, theta=tangle, phi=ptangle)
contour(x, y, cc$gauss, main=mains[2], col=lcol)
persp(x, y, cc$gauss, zlab=zlab2, main=mains[2], col=pcol, theta=tangle, phi=ptangle)
contour(x, y, cc$clayton, main=mains[3], col=lcol)
persp(x, y, cc$clayton, zlab=zlab2, main=mains[3], col=pcol, theta=tangle, phi=ptangle)

par(bty="n", mfcol=c(2, 3))
contour(x, y, CC$plackett, main=mains[1], col=lcol)
persp(x, y, CC$plackett, zlab=zlab, main=mains[1], col=pcol, theta=tangle, phi=ptangle)
```

```
contour(x, y, CC$gauss, main=mains[2], col=lcol)
persp(x, y, CC$gauss, zlab=zlab, main=mains[2], col=pcol, theta=tangle, phi=ptangle)
contour(x, y, CC$clayton, main=mains[3], col=lcol)
persp(x, y, CC$clayton, zlab=zlab, main=mains[3], col=pcol, theta=tangle, phi=ptangle)

par(bty="n")
layout(matrix(c(0,1,1,0,  2,2,3,3), nrow=2, byrow=TRUE))
contour(x, y, cc$plackett, main=mains[1], col=lcol)
contour(x, y, cc$gauss, main=mains[2], col=lcol)
contour(x, y, cc$clayton, main=mains[3], col=lcol)

par(bty="n")
layout(matrix(c(0,1,1,0,  2,2,3,3), nrow=2, byrow=TRUE))
persp(x, y, cc$plackett, zlab=zlab2, main=mains[1], col=pcol, theta=tangle, phi=ptangle)
persp(x, y, cc$gauss, zlab=zlab2, main=mains[2], col=pcol, theta=tangle, phi=ptangle)
persp(x, y, cc$clayton, zlab=zlab2, main=mains[3], col=pcol, theta=tangle, phi=ptangle)

par(bty="n")
layout(matrix(c(0,1,1,0,  2,2,3,3), nrow=2, byrow=TRUE))
contour(x, y, CC$plackett, main=mains[1], col=lcol)
contour(x, y, CC$gauss, main=mains[2], col=lcol)
contour(x, y, CC$clayton, main=mains[3], col=lcol)

par(bty="n")
layout(matrix(c(0,1,1,0,  2,2,3,3), nrow=2, byrow=TRUE))
persp(x, y, CC$plackett, zlab=zlab, main=mains[1], col=pcol, theta=tangle, phi=ptangle)
persp(x, y, CC$gauss, zlab=zlab, main=mains[2], col=pcol, theta=tangle, phi=ptangle)
persp(x, y, CC$clayton, zlab=zlab, main=mains[3], col=pcol, theta=tangle, phi=ptangle)

par(oldpar)
```

---

cumlogit   *Cumulative logit model for ordinal responses*

Description

Fits the cumulative logit model using the maximum-likelihood. The log-likelihood is maximized using the Newton-Raphson algorithm. The function returns the inverse of both observed and expected information matrix. Summary of the model produced by the summary function uses by default the inverse of the observed information matrix for the inference. This can be changed by the user such that the expected information is used instead.

Usage

```
cumlogit(y, v, x, C=1, logit.order=c("decreasing", "increasing"),
    epsilon=1e-08, maxit=25, trace=FALSE)

## S3 method for class 'cumlogit'
print(x, vcov=c("observed", "expected"), ...)
```

```
## S3 method for class 'cumlogit'
summary(object, vcov=c("observed", "expected"), ...)
```

Arguments

y	response vector taking values $0, 1, \ldots, C$.
v	matrix or data.frame with covarites whose effect does not necessarily satisfy proportional odds assumption.
	Intercept is included by default in the model and should be included neither in x, nor in v.
x	vector, matrix or data.frame with covarites whose effect is assumed to satisfy proportional odds assumption.
C	number of response categories minus 1.
logit.order	either "decreasing" or "increasing" indicating in which direction the logits are formed.

For `logit.order="decreasing"`:

$$
\begin{aligned}
\log\left\{\frac{P(Y \geq 1)}{P(Y=0)}\right\} &= \beta'x + \gamma_1'v \\
\log\left\{\frac{P(Y \geq 2)}{P(Y \leq 1)}\right\} &= \beta'x + \gamma_2'v \\
&\vdots \\
\log\left\{\frac{P(Y=C)}{P(Y \leq C-1)}\right\} &= \beta'x + \gamma_C'v
\end{aligned}
$$

For `logit.order="increasing"`:

$$
\begin{aligned}
\log\left\{\frac{P(Y=0)}{P(Y \geq 1)}\right\} &= \beta'x + \gamma_1'v \\
\log\left\{\frac{P(Y \leq 1)}{P(Y \geq 2)}\right\} &= \beta'x + \gamma_2'v \\
&\vdots \\
\log\left\{\frac{P(Y <= C-1)}{P(Y=C)}\right\} &= \beta'x + \gamma_C'v
\end{aligned}
$$

vcov	character indicating which type of the information matrix should be used for the inference
epsilon	positive convergence tolerance ε. The iterations converge when

$$
\left|\frac{\ell_{new} - \ell_{old}}{\ell_{new}}\right| \leq \varepsilon,
$$

where ℓ denotes the value of the log-likelihood.

maxit	integer giving the maximal number of iterations.
trace	logical indicating if output should be produced for each iteration.
object	an object of class "cumlogit".
...	other arguments passed to print or summary.

Value

An object of class "cumlogit". This has components

coefficients	the coefficients of the linear predictor.
loglik	the value of the log-likelihood.
score	the score vector.
vcov	the inverse of the observed information matrix.
expect.vcov	the inverse of the expected information matrix.
logit.order	character indicating the way in which the logits are formed.
linear.predictors	
	the values of the linear predictor for each observation and each logit.
fitted.values	the values of fitted category probabilities for each observation.
converged	logical indicating whether the optimization routine converged.
iter	number of iterations performed
C	see the function argument
y	see the function argument
v	see the function argument
x	see the function argument

Author(s)

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

References

Agresti, A. (2002). *Categorical Data Analysis. Second edition*. Hoboken: John Wiley \& Sons. Section 7.2.

See Also

[glm](), [polr]().

Examples

```
## Simulate the data for C = 3
## ===========================
set.seed(775988621)
N3 <- 1000

  ## covariates:
x3 <- data.frame(x1=rnorm(N3), x2=runif(N3, 0, 1))
v3 <- data.frame(v1=rnorm(N3), v2=runif(N3, 0, 1))

  ## regression coefficients
  ## (theta = c(0.1, -0.2,    1, 0.1, -0.2,    0, 0.05, -0.25,    -0.5, 0.05, -0.25)):
alpha3 <- c(1, 0, -0.5)
```

```
beta3 <- c(0.1, -0.2)
gamma3 <- list(c(0.1, -0.2), c(0.05, -0.25), c(0.05, -0.25))

 ## linear predictors and inverse logits:
eta3 <- data.frame(eta1=alpha3[1] + as.matrix(x3) %*% beta3 +
                         as.matrix(v3) %*% gamma3[[1]],
                   eta2=alpha3[2] + as.matrix(x3) %*% beta3 +
                         as.matrix(v3) %*% gamma3[[2]],
                   eta3=alpha3[3] + as.matrix(x3) %*% beta3 +
                         as.matrix(v3) %*% gamma3[[3]])
ilogit3 <- data.frame(ilogit1 = exp(eta3[,1])/(1 + exp(eta3[,1])),
                      ilogit2=exp(eta3[,2])/(1 + exp(eta3[,2])),
                      ilogit3=exp(eta3[,3])/(1 + exp(eta3[,3])))

 ## category probabilities:
pis3 <- data.frame(pi0=1-ilogit3[,1],
                   pi1=ilogit3[,1]-ilogit3[,2],
                   pi2=ilogit3[,2]-ilogit3[,3],
                   pi3=ilogit3[,3])

 ## response:
rtemp <- function(prob, C){
  return(sample(0:C, size=1, prob=prob))
}
y3 <- apply(pis3, 1, rtemp, C=3)

## Fit the model
## ===============
fit <- cumlogit(y=y3, x=x3, v=v3, C=3)
print(fit)

fit2 <- cumlogit(y=y3, x=x3, v=v3, C=3, logit.order="increasing")
print(fit2)
```

---

cumlogitRE	*Logit and cumulative logit model with random effects*

Description

This function implements MCMC sampling for the logit model with binary response and the cumulative logit model for multinomial ordinal response. Details are given in Komárek and Lesaffre (2008). On as many places as possible, the same notation as in this paper is used also in this manual page.

In general, the following (cumulative) logit model for response Y is assumed:

$$
\begin{aligned}
\log\left\{\frac{P(Y\geq 1)}{P(Y=0)}\right\} &= \eta^{(1)} \\
\log\left\{\frac{P(Y\geq 2)}{P(Y\leq 1)}\right\} &= \eta^{(2)} \\
&\vdots \\
\log\left\{\frac{P(Y=C)}{P(Y\leq C-1)}\right\} &= \eta^{(C)},
\end{aligned}
$$

where the form of the linear predictors $\eta_1, \ldots, \eta_C$ depends on whether a hierarchical centering is used or not. In the following, β denotes fixed effects and b random effects.

No hierarchical centering (DEFAULT)

The linear predictor for the cth logit has the following form

$$\eta^{(c)} = \beta^{(c)'}(v', v_b') + \beta^{*'}(x', x_b') + b'(v_b', x_b') \qquad (c = 1, \ldots, C),$$

where $\beta = (\beta^{(1)'}, \ldots, \beta^{(C)'}, \beta^{*'})'$ is the vector or the fixed-effects and b is a vector of random effects with zero location.

Hierarchical centering

The linear predictor for the cth logit has the following form

$$\eta^{(c)} = \beta^{(c)'}v + \beta^{*'}x + b^{(c)'}v_b + b^{*'}x_b \qquad (c = 1, \ldots, C),$$

where $\beta = (\beta^{(1)'}, \ldots, \beta^{(C)'}, \beta^{*'})'$ is the vector or the fixed-effects and $b = (b^{(1)'}, \ldots, b^{(C)'}, b^{*'})'$ is a vector of random effects with location $\alpha = (\alpha^{(1)'}, \ldots, \alpha^{(C)'}, \alpha^{*'})'$.

Normal random effects (drandom="normal")

A vector of random effects is assumed to follow a (multivariate) normal distribution.

That is, if there is **no hierarchical centering** we assume for b:

$$b \sim N(\mathbf{0}, \mathbf{D}_b),$$

where $\mathbf{D}_b$ is their variance-covariance matrix.

If the random effects are **hierarchically centered** then we assume for $b = (b^{(1)'}, \ldots, b^{(C)'}, b^{*'})'$:

$$b \sim N(\alpha, \mathbf{D}_b),$$

where $\alpha = (\alpha^{(1)'}, \ldots, \alpha^{(C)'}, \alpha^{*'})'$ is a vector of random effect locations (means) and $\mathbf{D}_b$ is their variance-covariance matrix.

Further, in a Bayesian model, it is assumed that $\mathbf{D}_b^{-1}$ has a Wishart $W(\nu_b, S_b)$ prior with ν_b degrees of freedom and scale matrix S_b. That is, a priori

$$\mathbf{D}_b^{-1} \sim W(\nu_b, S_b),$$

$$E(\mathbf{D}_b^{-1}) = \nu_b S_b.$$

Note that ν_b must be higher than the number of random effects minus 1.

Alternatively, when there is only a univariate random effect with the variance d_b^2, it is possible to specify a uniform prior for the standard deviation of the random effect. That is, a priori

$$d_b \sim \text{Unif}(0, S).$$

G-spline distributed random effects (drandom="gspline")

See `gspline1` and `gspline2` for description of the G-spline (penalized Gaussian mixture) distribution. Further, see Komárek and Lesaffre (2008) for description of the prior distribution on the G-spline. Brief description follows here as well.

Univariate G-spline

If there is only a univariate random effect in the model and its distribution is specified as a G-spline that it is assumed that

$$b \sim \alpha + \sum_{j=-K}^{K} w_j N\Big(\tau\mu_j, (\tau\sigma)^2\Big),$$

where α is a location parameter, τ is a scale parameter and $w = (w_{-K}, \ldots, w_K)'$ is a vector of G-spline weights. For **hierarchically centered** random effects, the location parameter α is fixed to zero.

Further, $M = (\mu_{-K}, \ldots, \mu_K)'$ is a vector of fixed equidistant knots (component means), where

$$\mu_j = j\delta \qquad (j = -K, \ldots, K)$$

and σ is a fixed basis standard deviation.

The constraints $0 < w_j < 1$ $(j = -K, \ldots, K)$ and $\sum_{j=-K}^{K} w_j = 1$ are sufficient for G-spline to be a density. To avoid constraint estimation we will estimate transformed weights $a = (a_{-K}, \ldots, a_K)'$ instead which relates to the original weights by

$$w_j = \frac{\exp(a_j)}{\sum_{k=-K}^{K} \exp(a_k)} \qquad (j = -K, \ldots, K),$$

$$a_j = \log \frac{w_j}{w_0} \qquad (j = -K, \ldots, K),$$

In the estimation procedure a penalty on the a coefficients in the form of a Gaussian Markov random field prior is imposed.

Bivariate G-spline

If there is a bivariate random effect $b = (b_1, b_2)'$ in the model and its distribution is specified as a G-spline that it is assumed that

$$b \sim (\alpha_1, \alpha_2)' + \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} \mathrm{N}\Big((\tau_1\mu_{1,j_1}, \tau_2\mu_{2,j_2})', \mathrm{diag}\big((\tau_1\sigma_1)^2, (\tau_2\sigma_2)^2\big)\Big),$$

where α_1, α_2 are location parameters, τ_1, τ_2 are scale parameters and $W = (w_{-K_1,-K_2}, \ldots, w_{K_1,K_2})'$ is a matrix of G-spline weights. For **hierarchically centered** random effects, the location parameters α_1, α_2 are fixed to zero.

Further, $M_1 = (\mu_{1,-K_1}, \ldots, \mu_{1,K_1})'$ is a vector of fixed equidistant knots (component means) in the first margin, where

$$\mu_{1,j_1} = j_1\delta_1 \qquad (j_1 = -K_1, \ldots, K_1)$$

and σ_1 is a fixed basis standard deviation in the first margin. Similarly for the second margin.

Similar reparametrization of the G-spline weights as in the univariate case is used to avoid constrained estimation.

Usage

```
cumlogitRE(y, v, x, vb, xb, cluster,
     intcpt.random=FALSE,
     hierar.center=FALSE,
     drandom=c("normal", "gspline"),
     C=1,
     logit.order=c("decreasing", "increasing"),
     prior.fixed,
     prior.random,
     prior.gspline,
```

```
        init.fixed,
        init.random,
        init.gspline,
        nsimul = list(niter=10, nthin=1, nburn=0, nwrite=10),
        store = list(prob=FALSE, b=FALSE, alloc=FALSE, acoef=FALSE),
        dir=getwd(),
        precision=8)
```

Arguments

y	response vector taking values $0, 1, \dots, C$.
v	vector, matrix or data.frame with covarites for **fixed** effects whose effect does not necessarily satisfy proportional odds assumption.
	If the argument intcpt.random is set to FALSE then the fixed intercept is included by default in the model. The intercept column should be included neither in x, nor in v.
x	vector, matrix or data.frame with covarites for **fixed** effects whose effect is assumed to satisfy proportional odds assumption.
vb	vector, matrix or data.frame with covarites for **random** effects whose effect does not necessarily satisfy proportional odds assumption.
	If you want to include **random intercept**, do it by setting the argument intcpt.random to TRUE. The intercept column should be included neither in xb, nor in vb.
xb	vector, matrix or data.frame with covarites for **random** effects whose effect is assumed to satisfy proportional odds assumption.
cluster	vector which determines clusters. Needed only when there are any random effects in the model.
intcpt.random	logical indicating whether a **random** intercept should be included in the model.
hierar.center	logical indicating whether a hierarchical centering of random effects should be used or not.
drandom	character indicating assumed distribution of random effects (if there are any).
C	number of response categories minus 1.
logit.order	either "decreasing" or "increasing" indicating in which direction the logits are formed. See the same argument in [cumlogit](#) for more details.
	Currently, only "decreasing" is implemented for cumlogitRE.
prior.fixed	list specifying the prior distribution for the fixed effects (regression coefficients).
	mean vector giving the prior mean for each regression coefficient. It can be a single number only in which case it is recycled.
	var vector giving the prior variance for each regression coefficient. It can be a single number only in which case it is recycled.
	In the case that prior mean and/or variances are different for different regression coefficients then they should be given in the following order: intercept for the first logit, $\beta(\text{v})$ for the first logit, $\dots$, intercept for the last logit, $\beta(\text{v})$ for the last logit, $\beta(\text{x})$,

prior.random   list specifying the prior distribution for the parameters of the distribution of random effects. Composition of this list depends on the chosen distribution of random effects (**normal** or **Gspline**).

Mdistrib character specifying the prior distribution of the location of random effects. It is ignored if hierar.center=FALSE, in which case the location of the random effects is fixed to zero. It can be one of the following.

"fixed"
Locations of random effects are assumed to be fixed and are not updated.

"normal"
Locations of random effects are assumed to be apriori normally distributed. Parameters of the normal distribution are specified further by items Mmean and Mvar.
This is also a default choice when Mdistrib is not specified.

Mmean vector giving the prior means for the locations of random effects. It can be a single number only in which case it is recycled.
It is ignored when hierar.center is FALSE in which case all random effects have zero location.

Mvar vector giving the prior variances for the locations of random effects. It can be a single number only in which case it is recycled.
In the case that prior means and/or variances are different for different locations of random effects then they should be given in the similar order as specified above for argument prior.fixed.
It is ignored when hierar.center is FALSE in which case all random effects have zero location.

Ddistrib character specifying the prior distribution of the covariance matrix of random effects. It can be one of the following.

"fixed"
covariance matrix of random effects is assumed to be fixed and is not updated.

"wishart"
inverse of the covariance matrix of **normally** distributed random effects is assumed to have a priori Wishart distribution. Parameters of the Wishart distribution are specified further by items Ddf and DinvScale.
This is also a default choice when Ddistrib is not specified and random effects are normally distributed.
This option is not allowed when the random effects distribution is modelled using **G-splines**.

"sduniform"
when random effects are **normally** distributed then this option may be used when there is only a univariate random effect in the model. Its standard deviation is then assumed to follow a uniform distribution on the interval $(0, S)$. Value of S is specified further by an item Dupper.
When random effects distribution is modelled using **G-splines** then sduniform

can be used also for multivariate random effects. It is then assumed that the overall standard deviation τ_m in the mth margin follows a uniform distribution on the interval $(0, S_m)$.

"gamma"
when random effects are **normally** distributed then this option may be used when there is only a univariate random effect in the model. Its inverse variance is then assumed to have a Gamma prior with the shape specified further by the item Dshape and the rate (inverse scale) specified by the item DinvScale.
When random effects distribution is modelled using **G-splines** then gamma can be used also for multivariate random effects. It is then assumed that the overall inverse variance τ_m^{-2} in the mth margin follows a Gamma prior with the shapes specified further by the item Dshape and the rates (inverse scales) specified by the item DinvScale.

Ddf degrees of freedom ν_b for the Wishart prior distribution of the inverse covariance matrix of **normally** distributed random effects.

Dshape number or vector with shape parameters for the gamma priors of the variance components of the random effects.
If it is a single number and random effects are multivariate it may be recycled.

DinvScale number or matrix determining the inverse scale matrix S_b^{-1} for the Wishart prior distribution of the inverse covariance matrix of **normally** distributed random effects.
Or number or vector giving the rate (inverse scale) parameter(s) for the gamma priors of the variance components.
If it is a single number and Ddistrib is **wishart** then it is assumed that S_b^{-1} is diagonal with that single number on a diagonal.

Dupper upper limit for the uniform distribution of the standard deviation of the random effect(s) when Ddistrib is equal to **sduniform**.
If it is a single number and random effects are multivariate it may be recycled.

prior.gspline list specifying the G-spline distribution of random effects and prior distribution of the G-spline parameters. This argument is required only when drandom is equal to **gspline**. In the following let q denote the number (dimension) of random effects.

The list prior.gspline can have the following components.

K vector of length q or a number (it is recycled) which specifies, for each marginal G-spline, the number of knots on each side of the zero knot. That is, the m-th marginal G-spline has $2K_m + 1$ knots.
It is set to 15 if not explicitly specified.

delta vector of length q or a number (it is recycled) which specifies the distance between two consecutive knots for each marginal G-spline. That is, the m-th marginal G-spline has the following knots

$$\mu_{m,j} = j\, \delta_m, \ j = -K_m, \ldots, K_m.$$

It is set to 0.3 if not explicitly specified.

sigma vector of length q or a number (it is recycled) which specifies the basis standard deviation of each marginal G-spline.

σ_m is set to $(2/3)\delta_m$ if not explicitly specified.

CARorder vector of length q or a number (it is recycled) giving the order of the intrinsic conditional autoregression in the Gaussian Markov random field prior for the transformed G-spline weights in each margin.

It does not need to be specified when `neighbor.system` is different from `uniCAR`.

It is set to 3 if not explicitly specified.

neighbor.system character specifying the type of the Gaussian Markov random field in the prior for the transformed G-spline weights a of a **bivariate** G-spline. It does not have to be specified for univariate G-splines. It can be one of the following.

"`uniCAR`"
univariate (in each margin) conditional autoregression as described in Komárek and Lesaffre (2008). That is a priori

$$p(a \mid \lambda) \propto \exp\left[-\left\{\frac{\lambda_1}{2}\sum_{j_1}\cdots\sum_{j_q}\left(\Delta_1^d a_{j_1,\ldots,j_q}\right)^2 + \cdots + \frac{\lambda_q}{2}\sum_{j_1}\cdots\sum_{j_q}\left(\Delta_q^d a_{j_1,\ldots,j_q}\right)^2\right\}\right],$$

where Δ_m^d is a difference operator of order d in the mth margin, e.g.,

$$\Delta_1^3 a_{j_1,j_2,\ldots,j_q} = a_{j_1,j_2,\ldots,j_q} - 3a_{j_1-1,j_2,\ldots,j_q} + 3a_{j_1-2,j_2,\ldots,j_q} - a_{j_1-3,j_2,\ldots,j_q},$$

and $\lambda = (\lambda_1,\ldots,\lambda_q)'$ are smoothing hyperparameters.

This is also a default choice when `neighbor.system` is not specified.

"`eight.neighbors`"
this prior is applicable for **bivariate** G-splines only and is based on eight nearest neighbors in a spatial meaning. That is, except on edges, each full conditional of a depends only on eight nearest neighbors and local quadratic smoothing. The prior is then defined as

$$p(a \mid \lambda) \propto \exp\left\{-\frac{\lambda}{2}\sum_{j_1=-K_1}^{K_1-1}\sum_{j_2=-K_2}^{K_2-1}\left(\Delta a_{j_1,j_2}\right)^2\right\},$$

where

$$\Delta a_{j_1,j_2} = a_{j_1,j_2} - a_{j_1+1,j_2} - a_{j_1,j_2+1} + a_{j_1+1,j_2+1}.$$

Parameter λ is a common smoothing hyperparameter.

"`twelve.neighbors`"
not (yet) implemented

Ldistrib character specifying the prior distribution of the smoothing hyperparameters $\lambda_m, m = 1,\ldots,q$ (precision parameters of the Markov random fields in each margin) or of a common hyperparameter λ It can be one of the following.

"fixed"
smoothing hyperparameters λ are fixed to their initial values and are not up-
dated.

"gamma"
each of the smoothing hyperparameters $\lambda_1, \ldots, \lambda_q$ is assumed to follow a
Gamma prior with the shapes specified further by the item Lshape and the
rates (inverse scales) specified further by the item LinvScale.
This is also a default choice when Ldistrib is not specified.

"sduniform"
square root of the inversion of each smoothing hyperparameter, i.e., $\sqrt{\lambda_1^{-1}}, \ldots, \sqrt{\lambda_q^{-1}}$
is assumed to follow apriori a uniform distribution on the intervals $(0, \ S_m^\lambda)$.
Values of $S_1^\lambda, \ldots, S_q^\lambda$ are specified further by the item Lupper.

Lequal logical indicating whether all smoothing hyperparameters should be
kept equal.
It is set to FALSE if not explicitly specified and neighbor.system is uniCAR.
It is always TRUE when neighbor.system is different from uniCAR.

Lshape number or vector with shape parameters for the **gamma** priors of the
smoothing hyperparameters λ.
If it is a single number and there is more than one smoothing hyperparam-
eter λ in the model it may be recycled.

LinvScale number or vector with rate (inverse scale) parameters for the **gamma**
priors of the smoothing hyperparameters λ.
If it is a single number and there is more than one smoothing hyperparam-
eter λ in the model it may be recycled.

Lupper number or vector with upper limits for the uniform distribution on
$\sqrt{\lambda^{-1}}$ parameters when Ldistrib is **sduniform**.
If it is a single number and there is more than one smoothing hyperparam-
eter λ in the model it may be recycled.

Aident character specifying in which way the transformed G-spline weights (a
coefficients) are identified. It can be one of the following.

"mean"
with this option, the a coefficients are forced to sum up to zero and have a
zero mean.
Note that this option usually causes problems during MCMC, especially
with bivariate G-splines. The reason is that if there are many almost zero
weights, they lead to many negative a coefficients and to satisfy the zero
mean constrain, there must be some a's which are highly positive. When
exponentiating them to get weights w, an overflow occur.

"reference"
with this option, one of the a coefficients in each margin is chosen as the
reference one and is always equal to zero. Index of the reference coefficient
is specified by the item Areference (see below).
This is also a default choice when Aident is not specified.

Areference vector or number (it is recycled) which specifies the index of the reference a coefficient in each margin in the case `Aident` is equal to **reference**. For the m-th margin, it must be an integer between $-K_m, \ldots, K_m$. To avoid numerical problems, the index of the reference a coefficient may change during the MCMC.

AtypeUpdate character specifying in which way the transformed G-spline weights (a coefficients) are updated. It can be one of the following.

"slice"
slice sampler of Neal (2003).

"ars.quantile"
adaptive rejection sampling of Gilks and Wild (1992) with starting abscissae being quantiles of the envelop at the previous iteration.

"ars.mode"
adaptive rejection sampling of Gilks and Wild (1992) with starting abscissae being the mode plus/minus 3 times estimated standard deviation of the full conditional distribution.

"block"
all a coefficients are updated in 1 block using the Metropolis-Hastings algorithm. This is only available for univariate G-splines.

Default is `slice`.

init.fixed
optional vector with initial values for fixed effects, supplied in the same order as described above in the argument `prior.fixed`.

If not given, initials are determined from the maximum-likelihood fit to the model where random effects are replaced by corresponding fixed effects.

init.random
optional list with initial values for the random effects and parameters determining their distribution. It can have the following components.

b vector or matrix with initial values of cluster specific random effects. Number of rows of the matrix or the length of the vector must be equal to `length(unique(cluster))`. Columns of the matrix correspond to the random effects in the same order as described above for argument `prior.random`. If not given, initials are taken to be equal over clusters and equal to the corresponding fixed effects from the maximum-likelihood fit to the model with fixed effects only.

mean vector giving the initial values of the means of random effects.
If not given, initials are taken to be equal to the corresponding fixed effects from the maximum-likelihood fit to the model with fixed effects only.

var matrix giving the initial value for the covariance matrix of the random effects when `drandom` is **normal**.
Vector giving the initial values of marginal overall variances of the random effects when `drandom` is **gspline**.

init.gspline
optional list with initial values related to the G-spline distribution of random effects (if `drandom` is equal to **gspline**). It can have the following components.

> **lambda** vector with initial values of the smoothing hyperparameters (precision parameters of the Markov random field). If not fixed and not given, the initials are sampled from the prior distribution.
>
> **weights** for **univariate** G-splines: vector with initial weights.
> For **bivariate** G-splines: matrix with initial weights.
> If the initial weights do not sum up to 1 they are re-scaled.
>
> **alloc** vector or matrix with initial component allocations for the individual random effects. For **univariate** random effects this should be a vector with numbers from $\{-K, \ldots, K\}$. For **bivariate** random effects this should be a matrix with two columns where in the first column numbers from $\{-K_1, \ldots, K_1\}$ appear and in the second column numbers from $\{-K_2, \ldots, K_2\}$ appear.

nsimul   list indicating the length of the MCMC. It should have the following components.

> **niter** total number of the MCMC iterations after discarding the thinned values
>
> **nthin** thinning of the sample
>
> **nburn** length of the burn-in period
>
> **nwrite** frequency with which the iteration count changes. Further, during the burn-in, only every nwriteth sampled value is stored on the disk

store   list indicating which chains (out of these not stored by default) should be compulsory stored. The list has the logical components with the following names.

> **prob** if TRUE values of individual predictive probabilities are stored.
>
> **b** if TRUE values of cluster specific random effects are stored.
>
> **alloc** if TRUE values of allocation indicators are stored.
>
> **acoef** if TRUE and distribution of random effects is given as a **bivariate** G-spline values of log-G-spline weights (a coefficients) are stored for all components.

dir   character string specifying the directory in which the sampled values are stored.

precision   precision with which the sampled values are written in files.

Value

The function returns a complete list of parameters of the prior distribution and initial values.

The main task of this function is to sample from the posterior distribution using MCMC. Sampled values are stored in various files which are described below.

Files created

iteration.sim one column labeled iteration with indeces of MCMC iterations to which the stored sampled values correspond.

betaF.sim sampled values of the fixed effects $\beta = (\beta^{(1)'}, \ldots, \beta^{(C)'}, \beta^{*'})'$.

> **Note** that in models with **G-spline** distributed random effects which are not hierarchically centered, the average effect of the covariates involved in the random effects (needed for inference) is obtained as a sum of the corresponding β coefficient and a scaled mean of the G-spline. β coefficients adjusted in this way are stored in the file 'betaRadj.sim' (see below).

betaR.sim sampled values of the location parameters $\alpha = (\alpha^{(1)'}, \ldots, \alpha^{(C)'}, \alpha^{*'})'$ of the random effects when the **hierarchical centering** was used.

> **Note** that in models with **G-spline** distributed random effects which are hierarchically centered, the average effect of the covariates involved in the random effects (needed for inference) is obtained as a sum of the corresponding α coefficient and a mean of the G-spline. α coefficients adjusted in this way are stored in the file 'betaRadj.sim' (see below).

varR.sim variance components of the random effects. Format of the file depends on the assumed distribution of the random effects.

> **Normal random effects** Let q be the dimension of the random effects. The first $0.5q(q+1)$ columns of 'varR.sim' contain a lower triangle (in column major order) of the matrix $\mathbf{D}_b$, the second $0.5q(q+1)$ columns of 'varR.sim' contain a lower triangle of the matrix $\mathbf{D}_b^{-1}$.

> **Univariate G-spline random effects** The first column of 'varR.sim' contains the G-spline variance parameter τ^2, the second column its inverse.

> **Bivariate G-spline random effects** The first two columns of 'varR.sim' contain the G-spline variance parameters τ_1^2, τ_2^2, the second two columns their inverse.

loglik.sim sampled values of the log-likelihood (conditioned by the values of random effects).

probability.sim sampled values of category probabilities for each observations.

> Created only if store$prob is TRUE.

b.sim sampled values of individual random effects.

> Stores complete chains only if store$b is TRUE.

Files created for models with G-spline distributed random effects

gspline.sim information concerning the fixed parameters of the G-spline which includes: dimension q of the G-spline, numbers of knots on each side of the reference knot for each margin $(K_1, \ldots, K_q)$, basis standard deviations $\sigma_1, \ldots, \sigma_q$ for each margin and knots $\mu_{1,-K_1}, \ldots \mu_{1,K_1}, \ldots, \mu_{q,-K_q}, \ldots \mu_{q,K_q}$ for each margin.

weight.sim this file is created only for **bivariate** G-splines and stores the weights w of the G-spline which are higher than a certain threshold value. That is, the weights that are numerically equal to zero are not recorded here. The link between the weights and G-spline components is provided by the file 'knotInd.sim'.

knotInd.sim this file is created only for **bivariate** G-splines. In its first column, it stores the number of G-spline components for which the weights are recorded on a corresponding row of the file 'weight.sim'. Subsequently, it stores indeces of the G-spline components for which the weights are given in the file 'weight.sim'. The indeces are stored as single indeces on the scale $0, \ldots, (2K_1 + 1)(2K_2 + 1) - 1$ such that index 0 corresponds to the component $(-K_1, -K_2)$, index 1 corresponds to the component $(-K_1 + 1, -K_2)$, ..., index $K_1 - 1$ corresponds to the component $(K_1, -K_2)$, etc.

logweight.sim sampled values of the transformed G-spline weights a.

> For **univariate** G-spline, this file stores always complete chains, irrespective of the value of store$acoef. For **bivariate** G-splines, this file stores complete chains only if store$acoef is TRUE.

gmoment.sim first two moments of the unshifted and unscaled G-spline at each iteration.

For **univariate** G-spline the column labeled 'gmean' is equal to

$$\text{gmean} = \sum_{j=-K}^{K} w_j \mu_j$$

and the column labeled 'gvar' is equal to

$$\text{gvar} = \sum_{j=-K}^{K} w_j (\mu_j - \text{gmean})^2 + \sigma^2.$$

See Komárek and Lesaffre, E. (2008) for formulas that apply in the **bivariate** case.

Values stored here are the values of $\beta_1^*, \ldots, \beta_q^*$ and $d_{1,1}^*, d_{2,1}^*, \ldots, d_{q,q}^*$ as defined in Komárek and Lesaffre (2008).

betaRadj.sim sampled values of the average (overall) effects of the random effects. See notes under the files 'betaF.sim' and 'betaR.sim' above.

Values stored here are the values of $\gamma_1, \ldots, \gamma_q$ as defined in Komárek and Lesaffre (2008).

varRadj.sim sampled components of the variance-covariance matrix of the random-effects.

Values stored here are the values of $d_{1,1}, d_{2,1}, \ldots, d_{q,q}$ as defined in Komárek and Lesaffre (2008).

alloc.sim sampled values of the component allocations (in Komárek and Lesaffre (2008) denoted by r_i) for individual random effects.

For **univariate** G-spline, the allocations are stored on the scale $-K, \ldots, K$.

For **bivariate** G-spline, the allocation are stored as single indeces on the scale $0, \ldots, (2K_1 + 1)(2K_2 + 1) - 1$ where the link between the single and double indeces is the same as in the file 'knotInd.sim'.

Stores complete chains only if store$alloc is TRUE.

lambda.sim sampled values of smoothing hyperparameter(s) λ.

Author(s)

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

References

Agresti, A. (2002). *Categorical Data Analysis. Second edition*. Hoboken: John Wiley \& Sons.

Gelfand, A. E., Sahu, S. K., and Carlin, B. P. (1995). Efficient parametrisations for normal linear mixed models. *Biometrika*, **82**, 479–488.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics,* **41**, 337–348.

Neal, R. M. (2003). Slice sampling (with Discussion). *The Annals of Statistics,* **31**, 705–767.

Komárek, A. and Lesaffre, E. (2008). Generalized linear mixed model with a penalized Gaussian mixture as a random-effects distribution. *Computational Statistics and Data Analysis*, **52**, 3441–3458.

Molenberghs, G. and Verbeke, G. (2005). *Models for Discrete Longitudinal Data*. New York: Springer Science+Business Media.

See Also

cumlogit, logpoissonRE, glm, polr.

Examples

```
### See ex-Toenail.pdf and ex-Toenail.R
### available in the documentation
### to the package
```

---

cumlogitRE.predict   *Prediction for logit and cumulative logit model with random effects*

Description

This function compute predictive probabilities of the response categories for specified combinations of covariates. It is based on the MCMC output obtained using cumlogitRE.

Usage

```
cumlogitRE.predict(nobs, v, x, vb, xb, cluster,
   intcpt.random=FALSE, hierar.center=FALSE,
   drandom=c("normal", "gspline"),
   C=1, logit.order=c("decreasing", "increasing"),
   betaF, betaR, varR, is.varR=TRUE,
   prior.gspline,
   probs, values=FALSE,
   dir=getwd(), wfile, indfile, header=TRUE, logw, is.indfile,
   skip=0, nwrite)
```

Arguments

nobs	number of covariate combinations for which we want to perform a prediction
v	covariate combinations for which we want to perform a prediction.
	It should have the same structure as in cumlogitRE used to obtain the MCMC output
x	covariate combinations for which we want to perform a prediction.
	It should have the same structure as in cumlogitRE used to obtain the MCMC output
vb	covariate combinations for which we want to perform a prediction.
	It should have the same structure as in cumlogitRE used to obtain the MCMC output
xb	covariate combinations for which we want to perform a prediction.
	It should have the same structure as in cumlogitRE used to obtain the MCMC output

cluster   vector defining pertinence of the single observations to clusters. It is useful
                 when we want to predict longitudinal profiles.

                 See also the same argument in cumlogitRE.

intcpt.random   see the same argument in cumlogitRE

hierar.center   see the same argument in cumlogitRE

drandom   see the same argument in cumlogitRE

C   see the same argument in cumlogitRE

logit.order   see the same argument in cumlogitRE

betaF   sampled values of the fixed effects. This should be a (sub)sample from the
                 MCMC output stored in the file 'betaF.sim'

betaR   sampled values of the mean of random effects. This should be a (sub)sample
                 from the MCMC output stored in the file 'betaR.sim'

                 It is only needed if hierar.center is TRUE.

varR   sampled values of either (co)variance matrices or precision (matrices) for ran-
                 dom effects if there are any. This should be a (sub)sample of either the first or
                 second half of the columns stored in the file 'varR.sim'

is.varR   logical indicating whether varR gives (co)variance (is.varR TRUE) or precisions
                 (inverse variances) (is.varR FALSE)

prior.gspline   if drandom is **gspline** this is a list specifying the G-splines. It should have the
                 same structure as the same argument in cumlogitRE used to obtain the MCMC
                 output. However, it is satisfactory if the items **K**, **delta** and **sigma** are given.

probs   probabilities for which the (pointwise) sample quantiles of the predictive prob-
                 abilities should be computed.

                 If not given only average (and values) of the predictive probabilities are com-
                 puted

values   if TRUE also values of the predictive probabilities at each (MCMC) iteration are
                 returned.

                 If FALSE only sample mean (and quantiles) of the predictive probabilities are
                 returned

dir   character giving the directory where the file with (sampled) G-spline (log-)weights
                 is stored.

                 Needed only if drandom is **gspline**.

wfile   character giving the name of the file with (sampled) G-spline (log-)weights.

                 Needed only if drandom is **gspline**. In most cases, for **univariate** G-spline this
                 argument will be equal to "logweight.sim" and for **bivariate** G-spline equal
                 to "weight.sim".

indfile   character giving the name of the file where we stored indeces of these G-spline
                 components for which the weights are stored in the file given by wfile. The
                 corresponding file should have the same structure as 'knotInd.sim' created by
                 cumlogitRE.

                 Needed only if is.indfile is TRUE. In most cases, for **univariate** G-spline
                 it does not have to be specified and for **bivariate** G-spline it will be equal to
                 "knotInd.sim".

header	logical indicating whether the files wfile, indfile contain a header.
	Needed only if drandom is **gspline**.
logw	logical indicating whether the file wfile contains logarithms of the weights.
	Needed only if drandom is **gspline**. In most cases, for **univariate** G-spline it will be TRUE and for **bivariate** G-spline it will be FALSE.
is.indfile	logical.
	If TRUE then wfile contains only the non-zero weights and the G-spline is reconstructed using indfile.
	If FALSE then wfile must contain on each row weights of all components and indfile is ignored.
	Needed only if drandom is **gspline** and random effects are bivariate.
skip	number of data rows that should be skipped at the beginning of the files wfile, indfile.
nwrite	frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change)

Value

A list with the following components (description below applies for the case with prob=0.5)

Mean	a matrix with $C + 1$ columns giving in each row posterior predictive mean of category probabilities $\mathrm{P}(Y = 0), \ldots, \mathrm{P}(Y = C)$ for a given covariate combination.
50%	a matrix with $C + 1$ columns giving in each row posterior predictive quantile (here 50% quantile) of category probabilities for a given covariate combination. There is one component of this type in the resulting list for each value of probs.
values	a matrix with $(C + 1)n$ columns, where n denotes the number of covariate combinations for which we perform the prediction, and number of rows equal to the length of the MCMC. The first $C + 1$ columns give sampled category probabilities for the first covariate combination, the second $C + 1$ columns give sampled category probabilities for the second covariate combination etc.
	It is returned only if values is TRUE.

Author(s)

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

References

Komárek, A. and Lesaffre, E. (2008). Generalized linear mixed model with a penalized Gaussian mixture as a random-effects distribution. *Computational Statistics and Data Analysis*, **52**, 3441–3458.

See Also

cumlogitRE, cumlogit, glm, polr.

---

densplotAK   *Probability density function estimate from MCMC output*

Description

Displays a plot of the density estimate for each variable in x, calculated by the density function.

This is slightly modified version of [densplot](#) function of a coda package to conform to my personal preferences.

Usage

```
densplotAK(x, plot=TRUE, show.obs=FALSE, bwf, bty="n", main ="",
   xlim, ylim, xlab, ylab, ...)
```

Arguments

x	an [mcmc](#) or [mcmc.list](#) object.
plot	if TRUE this function works more or less in the same way as coda function [densplot](#) function. If FALSE this function returns one data frame for each chain with computed density which can be used for future plotting.
show.obs	show observations along the x-axis?
bwf	function for calculating the bandwidth. If omitted, the bandwidth is calculate by 1.06 times the minimum of the standard deviation and the interquartile range divided by 1.34 times the sample size to the negative one fifth power.
xlim, ylim, xlab, ylab	
	further arguments passed to the [plot.default](#) function.
bty, main, ...	further arguments passed to the [plot.default](#) function.

Author(s)

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

See Also

[densplot](#).

---

epileptic	*Epileptic seizures*

Description

This data set considers information from a clinical trial of 59 epileptics, reported by Thall and Vail (1990). For each patient, the number of epileptic seizures was recorded during a baseline period of eight weeks. Patients were then randomized to treatment with the anti-epileptic drug progabide, or to placebo in addition to standard chemotherapy. The number of seizures was then recorded in four consecutive two-weeks intervals.

Usage

```
data(epileptic)
```

Format

A data frame with 295 observations on the following 5 variables.

id a unique identifier for the subject in the study.

seizure a numeric vector giving the number of epileptic seizures.

visit a numeric vector giving the number of the visit, 0=baseline, and 1,2,3, and 4 for the four consecutive two-weeks intervals.

trt a numeric vector giving the treatment group.

age a numeric vector giving the age at the entry.

Source

Thall, P. F., and Vail, S. C. (1990) Some covariance models for longitudinal count data with overdispersion. *Biometrics*, **46**, 657–671.

References

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9–25.

Kleinman, K. P. and Ibrahim, J. G. (1998). A semi-parametric Bayesian approach to generalized linear mixed models. *Statistics in Medicine*, **17**, 2579–2596.

See Also

[epilepticBC](#)

Examples

```
data(epileptic)
## maybe str(epileptic); plot(epileptic) ...
```

---

epilepticBC   *Epileptic seizures, Breslow and Clayton transformed*

Description

This data set considers information from a clinical trial of 59 epileptics, reported by Thall and Vail (1990). For each patient, the number of epileptic seizures was recorded during a baseline period of eight weeks. Patients were then randomized to treatment with the anti-epileptic drug progabide, or to placebo in addition to standard chemotherapy. The number of seizures was then recorded in four consecutive two-weeks intervals.

This is a transformed version of the original data (`epileptic`) which allows directly to fit models described in Breslow and Clayton (1993) or in Kleinman and Ibrahim (1998).

There are 4 rows in the dataset for each of 59 patients.

Usage

```
data(epilepticBC)
```

Format

A data frame with 236 observations on the following 10 variables.

id a unique identifier for the subject in the study.

visit a numeric vector giving the number of the visit, 1,2,3, and 4 for the four two-weeks intervals following the baseline measurement.

seizure0 a numeric vector giving the number of epileptic seizures in the eight-week period prior the randomization.

age a numeric vector giving the age (in years) at the entry.

Seizure a numeric vector giving the number of epileptic seizures in a given two-week interval.

Base a numeric vector giving the transformed seizure0. Base=log(seizure0/4).

Trt a numeric vector giving the treatment group.

Base.Trt a numeric vector giving the interaction covariate Base*Trt.

Age a numeric vector giving the transformed age. Age=log(age).

Visit a numeric vector giving the centered visit. Visit=(2*visit-5)/10.

Source

Thall, P. F., and Vail, S. C. (1990) Some covariance models for longitudinal count data with overdispersion. *Biometrics*, **46**, 657–671.

References

Breslow, N. E. and Clayton, D. G. (1993). Approximate inference in generalized linear mixed models. *Journal of the American Statistical Association*, **88**, 9–25.

Kleinman, K. P. and Ibrahim, J. G. (1998). A semi-parametric Bayesian approach to generalized linear mixed models. *Statistics in Medicine*, **17**, 2579–2596.

See Also

[epileptic](#)

Examples

```
data(epilepticBC)
## maybe str(epilepticBC); plot(epilepticBC) ...
```

---

glmmAK.files2coda   *Conversion of sampled values into coda mcmc objects*

Description

It takes the values sampled by [cumlogitRE](#) or [logpoissonRE](#) and stored in *.sim files and converts them into coda [mcmc](#) objects.

Usage

```
glmmAK.files2coda(dir, drandom=c("none", "normal", "gspline"),
   quiet=FALSE, skip=0,
   params=list(prob=FALSE, ecount=FALSE, b=FALSE, alloc=FALSE))
```

Arguments

dir	character specifying a directory with sampled values
drandom	string specifying the distribution of random effects used in the original [cumlogitRE](#) or [logpoissonRE](#) call
quiet	logical, passed to [scan](#) function
skip	number of MCMC iterations that should be skipped at the beginning of the chain
params	a list of logical values specifying which of the optional sampled parameters should be read

Value

A list of coda [mcmc](#) objects.

Author(s)

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

See Also

[mcmc](#).

GMRF   *Gaussian Markov random fields*

Description

Moments, density and random generation for the Gaussian Markov random field with mean equal to 'mean', precision matrix equal to 'Q' (or covariance matrix equal to 'Sigma') and possibly constrained by a linear constraint 'Ax=b'.

Generation of random numbers is performed by Algorithm 2.6 in Rue and Held (2005, pp. 38).

Usage

```
momentsGMRF(mean=0, Q=1, Sigma, A, b=0)

rGMRF(n, mean=0, Q=1, Sigma, A, b=0)

dGMRF(x, mean=0, Q=1, Sigma, A, b=0, log=FALSE)

dGMRF2(x, mean=0, Q=1, Sigma, A, b=0, log=FALSE)
```

Arguments

mean	vector of mean. If length(mean) is equal to 1, it is recycled and all components have the same mean.
Q	precision matrix of the GMRF.
Sigma	covariance matrix of the GMRF. Only one of Q and Sigma must be given. If Sigma is supplied, precision is computed from Σ as $Q = \Sigma^{-1}$.
A	optional matrix defining the constraint $Ax = b$ for sampled vectors x.
	If not supplied, the GMRF is assumed to be unconstrained.
	Currently at most 1 constraint is allowed, that is A must be a vector and b a number.
b	vector or the right-hand side of the constraint. If length(b) is equal to 1, it is recycled and all constraint right-hand sides are the same.
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 momentsGMRF

A list with the components:

mean mean of the (constrained) GMRF

Sigma covariance matrix of the (constrained) GMRF

and the following attributes:

mean.unconstr mean of the GMRF before imposing the constraints

Sigma.unconstr covariance matrix Σ_u of the GMRF before imposing the constraints

Q.unconstr.cholesky Cholesky decomposition of the matrix $Q_u = \Sigma_u^{-1}$

nconstraint number of constraints

A left-hand side of the constrains

b right-hand side of the constraints

Value for rGMRF

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 dGMRF, dGMRF2

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

Author(s)

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

References

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

See Also

dnorm, Mvnorm.

Examples

```
set.seed(1977)

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))
```

```
A <- rep(1, nrow(Sigma))
b <- 0

##### Unconstrained GMRF
##### ==================
## Moments
momentsGMRF(mean=mu, Sigma=Sigma)
momentsGMRF(mean=mu, Q=Q)

## Random numbers
z <- rGMRF(1000, mean=mu, Sigma=Sigma)
apply(z$x, 2, mean)
var(z$x)

## Random numbers, again
z <- rGMRF(10, mean=mu, Sigma=Sigma)
print(z)

## Values of the log-density
dGMRF(z$x, mean=mu, Sigma=Sigma, log=TRUE)
dGMRF(z$x, mean=mu, Q=Q, log=TRUE)
dGMRF2(z$x, mean=mu, Sigma=Sigma, log=TRUE)
dGMRF2(z$x, mean=mu, Q=Q, log=TRUE)

## Values of the density
dGMRF(z$x, mean=mu, Sigma=Sigma)
dGMRF(z$x, mean=mu, Q=Q)
dGMRF2(z$x, mean=mu, Sigma=Sigma)
dGMRF2(z$x, mean=mu, Q=Q)


##### Constrained GMRF
##### ================
## Moments
momentsGMRF(mean=mu, Sigma=Sigma, A=A, b=b)
momentsGMRF(mean=mu, Q=Q, A=A, b=b)

## Random numbers
z <- rGMRF(1000, mean=mu, Sigma=Sigma, A=A, b=b)
apply(z$x, 2, mean)
var(z$x)

## Random numbers, again
z <- rGMRF(10, mean=mu, Sigma=Sigma, A=A, b=b)
print(z)
A %*% t(z$x)

## Values of the log-density
dGMRF(z$x, mean=mu, Sigma=Sigma, A=A, b=b, log=TRUE)
dGMRF(z$x, mean=mu, Q=Q, A=A, b=b, log=TRUE)
dGMRF2(z$x, mean=mu, Sigma=Sigma, A=A, b=b, log=TRUE)
dGMRF2(z$x, mean=mu, Q=Q, A=A, b=b, log=TRUE)
```

```
## Values of the log-density
dGMRF(z$x, mean=mu, Sigma=Sigma, A=A, b=b)
dGMRF(z$x, mean=mu, Q=Q, A=A, b=b)
dGMRF2(z$x, mean=mu, Sigma=Sigma, A=A, b=b)
dGMRF2(z$x, mean=mu, Q=Q, A=A, b=b)
```

---

gspline1	*Density and random number generation from a univariate G-spline (penalized Gaussian mixture)*

Description

Univariate G-spline (penalized Gaussian mixture) is distributed as

$$\alpha + \sum_{j=-K}^{K} w_j \mathrm{N}(\tau\mu_j,\ \tau^2\sigma_j^2)$$

Usage

```
rgspline1(n, mu, sigma, weight, intcpt=0, scale=1, logw=TRUE)

dgspline1(x, mu, sigma, weight, intcpt=0, scale=1, logw=TRUE)
```

Arguments

n	number of observations to be generated
x	grid of values at which we evaluate the G-spline values
mu	a vector with G-spline knots μ_j $(j = -K, \ldots, K)$(means of basis G-splines)
sigma	basis standard deviation(s) σ_j $(j = -K, \ldots, K)$. If a single number is supplied then it is assumed that all basis G-splines have the same standard deviation. Alternatively a vector of the same length as mu can be given in which case the basis G-splines do not necessarily have the same standard deviations
weight	a vector with G-spline (log-)weights. It should have the same length as mu
intcpt	G-spline intercept value α)
scale	G-spline scale value τ)
logw	logical indicating whether logarithmic weights are supplied in weight

Value

Values of the density or generated random numbers.

Author(s)

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

Examples

```
knots <- c(-2, 0, 2)
sigma <- 1
weight <- c(0.3, 0.1, 0.6)
intcpt <- 3
scale <- 0.2
xgrid <- seq(1.8, 4.2, length=300)

dx <- dgspline1(xgrid, mu=knots, sigma=sigma, weight=weight,
   intcpt=intcpt, scale=scale, logw=FALSE)
x <- rgspline1(100, mu=knots, sigma=sigma, weight=weight,
   intcpt=intcpt, scale=scale, logw=FALSE)
hist(x, col="seagreen2", prob=TRUE, xlim=range(xgrid), xlab="x", ylab="g(x)")
lines(xgrid, dx, col="red", lwd=2)
```

---

gspline2	*Density and random number generation from a bivariate G-spline (penalized Gaussian mixture)*

Description

Density has not been implemented yet.

Bivariate G-spline (penalized Gaussian mixture) is distributed as

$$(\alpha_1, \; \alpha_2)' + \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1, j_2} \mathrm{N}_2((\tau_1 \mu_{1, j_1}, \; \tau_2 \mu_{2, j_2})', \; \mathrm{diag}(\tau_1^2 \sigma_{1, j_1}^2, \; \tau_2^2 \sigma_{2, j_2}^2))$$

Usage

```
rgspline2(n, mu1, mu2, sigma1, sigma2, weight, knotInd,
   intcpt=0, scale=1, logw=TRUE)
```

Arguments

n	number of observations to be generated
mu1	a vector with G-spline knots μ_{1, j_1} $(j_1 = -K_1, \dots, K_1)$ (means of basis G-splines) in the 1st margin
mu2	a vector with G-spline knots μ_{2, j_2} $(j_2 = -K_2, \dots, K_2)$ (means of basis G-splines) in the 2nd margin
sigma1	basis standard deviation(s) σ_{1, j_1} $(j_1 = -K_1, \dots, K_1)$ in the 1st margin.
	If a single number is supplied then it is assumed that all basis G-splines have the same standard deviation. Alternatively a vector of the same length as mu1 can be given in which case the basis G-splines do not necessarily have the same standard deviations

sigma2	basis standard deviation(s) σ_{2,j_2} $(j_2 = -K_2, \ldots, K_2)$ in the 2nd margin.
	If a single number is supplied then it is assumed that all basis G-splines have the same standard deviation. Alternatively a vector of the same length as mu2 can be given in which case the basis G-splines do not necessarily have the same standard deviations
weight	a vector or matrix with G-spline (log-)weights.
	If missing(knotInd) then it should have the same length as length(mu1)*length(mu2).
	If knotInd is supplied then weight must be a vector of the same length as knotInd which contains (log-)weights of G-spline components having non-zero weights.
knotInd	If supplied then it contains indeces of G-spline components which correspond to non-zero weights. Indeces should be on the scale from 0 to length(mu1)*length(mu2)-1 (similarly like in the file knotInd.sim created, e.g., by the function cumlogitRE)
intcpt	G-spline intercept value(s) $(\alpha_1, \alpha_2)'$
	If a single value is supplied then it is assumed that intercept values in both margins are the same.
scale	G-spline scale value(s) $(\tau_1, \tau_2)'$
	If a single value is supplied then it is assumed that scale values in both margins are the same.
logw	logical indicating whether logarithmic weights are supplied in weight

Value

Values of the density or generated random numbers.

Author(s)

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

Examples

```
knots1 <- c(-2, 0, 2)
knots2 <- c(-2, -1, 0, 1, 2)
sigma1 <- 0.5
sigma2 <- 0.5
intcpt <- c(3, -1)
scale <- c(0.2, 0.5)

weight <- matrix(c(1,2,1,  2,3,2,  3,4,3,  2,3,2,  1,2,1), ncol=5)
xA <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
   sigma2=sigma2, weight=weight, intcpt=intcpt, scale=scale, logw=FALSE)
xB <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
   sigma2=sigma2, weight=log(weight), intcpt=intcpt, scale=scale, logw=TRUE)

oldpar <- par(mfrow=c(1, 2), bty="n")
plot(xA[,2], xA[,1], pch=16, col="red")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
```

```
plot(xB[,2], xB[,1], pch=16, col="blue")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
par(oldpar)

### Only selected components with non-zero weights
logweight2 <- c(-1, -0.5, 1)
knotInd <- c(0, 5, 14)
yA <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
    sigma2=sigma2, weight=logweight2, knotInd=knotInd,
    intcpt=intcpt, scale=scale, logw=TRUE)
yB <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
    sigma2=sigma2, weight=exp(logweight2), knotInd=knotInd,
    intcpt=intcpt, scale=scale, logw=FALSE)
oldpar <- par(mfrow=c(1, 2), bty="n")
plot(yA[,2], yA[,1], pch=16, col="red")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
plot(yB[,2], yB[,1], pch=16, col="blue")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
par(oldpar)

logweight3 <- log(weight)[-c(5,8,11)]
knotInd3 <- c(0,1,2, 3,5, 6,8, 9,11, 12,13,14)
zA <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
    sigma2=sigma2, weight=logweight3, knotInd=knotInd3,
    intcpt=intcpt, scale=scale, logw=TRUE)
zB <- rgspline2(1000, mu1=knots1, mu2=knots2, sigma1=sigma1,
    sigma2=sigma2, weight=exp(logweight3), knotInd=knotInd3,
    intcpt=intcpt, scale=scale, logw=FALSE)
oldpar <- par(mfrow=c(1, 2), bty="n")
plot(zA[,2], zA[,1], pch=16, col="red")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
plot(zB[,2], zB[,1], pch=16, col="blue")
abline(h=intcpt[1]+scale[1]*knots1, col="orange")
abline(v=intcpt[2]+scale[2]*knots2, col="orange")
par(oldpar)
```

---

logpoisson   *Poisson log-linear regression model*

Description

Fits the poisson log-linear regression model using the maximum-likelihood. The log-likelihood is maximized using the Newton-Raphson algorithm (the same as Fisher scoring in this case). The function returns the inverse of the observed and expected information matrix.

Usage

```
logpoisson(y, x, offset=0, epsilon=1e-08, maxit=25, trace=FALSE)

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

## S3 method for class 'logpoisson'
summary(object, ...)
```

Arguments

y	response vector taking integer values or zero.
x	matrix or data.frame with covarites.
	Intercept is included by default in the model and should not be included in x.
offset	possible offset term. It is assumed to be equal to zero if not specified.
epsilon	positive convergence tolerance ε. The iterations converge when

$$\left| \frac{\ell_{new} - \ell_{old}}{\ell_{new}} \right| \leq \varepsilon,$$

where ℓ denotes the value of the log-likelihood.

maxit	integer giving the maximal number of iterations.
trace	logical indicating if output should be produced for each iteration.
object	an object of class "logpoisson".
...	other arguments passed to print or summary.

Value

An object of class "logpoisson". This has components

coefficients	the coefficients of the linear predictor.
loglik	the value of the log-likelihood.
score	the score vector.
vcov	the inverse of the information matrix.
linear.predictors	
	the values of the linear predictor for each observation.
fitted.values	the values of fitted counts for each observation.
converged	logical indicating whether the optimization routine converged.
iter	number of iterations performed
y	
x	

Author(s)

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

References

Agresti, A. (2002). *Categorical Data Analysis. Second edition.* Hoboken: John Wiley \& Sons. Section 7.2.

See Also

glm.

Examples

```
set.seed(1977)
n <- 100
x1 <- rbinom(n, 1, 0.4)
x2 <- runif(n, 0, 1)
eta <- 5 + 0.1*x1 -0.2*x2
mu <-  exp(eta)
y <- rpois(n, mu)

### Fit the model using poisson
Xmat <- data.frame(x1=x1, x2=x2)
fit <- logpoisson(y=y, x=Xmat)
summary(fit)

### Fit the model using standard glm
fit0 <- glm(y~x1+x2, family=poisson(link="log"))
summary(fit0)
```

---

logpoissonRE   *Poisson log-linear regression with random effects*

Description

This function implements MCMC sampling for the Poisson log-linear model. Details are given in Komárek and Lesaffre (2008). On as many places as possible, the same notation as in this paper is used also in this manual page.

In general, the following log-linear model for response Y is assumed:

$$\log\{\mathrm{E}(Y)\} = \eta,$$

where the form of the linear predictor η depends on whether a hierarchical centering is used or not. In the following, β denotes fixed effects and b random effects.

No hierarchical centering (DEFAULT)
The linear predictor has the following form

$$\eta = \beta'(x',\ x_b') + b'x_b,$$

where b is a vector of random effects with zero location.

Hierarchical centering

The linear predictor has the following form

$$\eta = \beta' x + b' x_b,$$

b is a vector of random effects with location α.

For description of the rest of the model, see [cumlogitRE](#).

Usage

```
logpoissonRE(y, x, xb, offset=0, cluster,
      intcpt.random=FALSE,
      hierar.center=FALSE,
      drandom=c("normal", "gspline"),
      prior.fixed,
      prior.random,
      prior.gspline,
      init.fixed,
      init.random,
      init.gspline,
      nsimul = list(niter=10, nthin=1, nburn=0, nwrite=10),
      store = list(ecount=FALSE, b=FALSE, alloc=FALSE, acoef=FALSE),
      dir=getwd(),
      precision=8)
```

Arguments

y	response vector taking integer values or zero.
x	vector, matrix or data.frame with covariates for **fixed** effects.
xb	vector, matrix or data.frame with covariates for **random** effects.
	If you want to include **random intercept**, do it by setting the argument intcpt.random to TRUE. The intercept column should not be included in xb.
offset	optional vector of the offset term.
cluster	see [cumlogitRE](#).
intcpt.random	see [cumlogitRE](#).
hierar.center	see [cumlogitRE](#).
drandom	see [cumlogitRE](#).
prior.fixed	see [cumlogitRE](#).
prior.random	see [cumlogitRE](#).
prior.gspline	see [cumlogitRE](#).
init.fixed	see [cumlogitRE](#).
init.random	see [cumlogitRE](#).
init.gspline	see [cumlogitRE](#).
nsimul	see [cumlogitRE](#).

store   list indicating which chains (out of these not stored by default) should be com-
               pulsory stored. The list has the logical components with the following names.

> **ecount** if TRUE values of individual predictive (expected) counts are stored.
>
> **b** if TRUE values of cluster specific random effects are stored.
>
> **alloc** if TRUE values of allocation indicators are stored.
>
> **acoef** if TRUE and distribution of random effects is given as a **bivariate** G-spline
> values of log-G-spline weights (a coefficients) are stored for all compo-
> nents.

dir   see [cumlogitRE](#).

precision   see [cumlogitRE](#).

Value

See [cumlogitRE](#).

Files created

iteration.sim see [cumlogitRE](#).

betaF.sim sampled values of the fixed effects β.

> **Note** that in models with **G-spline** distributed random effects which are not hierarchically cen-
> tered, the average effect of the covariates involved in the random effects (needed for inference)
> is obtained as a sum of the corresponding β coefficient and a scaled mean of the G-spline. β
> coefficients adjusted in this way are stored in the file 'betaRadj.sim' (see below).

betaR.sim sampled values of the location parameters α of the random effects when the **hierarchi-
cal centering** was used.

> **Note** that in models with **G-spline** distributed random effects which are hierarchically cen-
> tered, the average effect of the covariates involved in the random effects (needed for infer-
> ence) is obtained as a sum of the corresponding α coefficient and a mean of the G-spline. α
> coefficients adjusted in this way are stored in the file 'betaRadj.sim' (see below).

varR.sim see [cumlogitRE](#).

loglik.sim see [cumlogitRE](#).

expectcount.sim sampled values of predictive (expected) counts for each observations.

> Created only if store$ecount is TRUE.

b.sim see [cumlogitRE](#).

Files created for models with G-spline distributed random effects

See [cumlogitRE](#).

Author(s)

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

References

Agresti, A. (2002). *Categorical Data Analysis. Second edition*. Hoboken: John Wiley \& Sons.

Gelfand, A. E., Sahu, S. K., and Carlin, B. P. (1995). Efficient parametrisations for normal linear mixed models. *Biometrika*, **82**, 479–488.

Gilks, W. R. and Wild, P. (1992). Adaptive rejection sampling for Gibbs sampling. *Applied Statistics,* **41**, 337–348.

Neal, R. M. (2003). Slice sampling (with Discussion). *The Annals of Statistics,* **31**, 705–767.

Komárek, A. and Lesaffre, E. (2008). Generalized linear mixed model with a penalized Gaussian mixture as a random-effects distribution. *Computational Statistics and Data Analysis*, **52**, 3441–3458.

Molenberghs, G. and Verbeke, G. (2005). *Models for Discrete Longitudinal Data*. New York: Springer Science+Business Media.

See Also

[logpoisson](), [cumlogitRE](), [poisson](), [glm]().

Examples

```
### See ex-Epileptic.pdf and ex-Epileptic.R
### available in the documentation
### to the package
```

---

logpoissonRE.predict   *Prediction for Poisson log-linear regression with random effects*

Description

This function compute predictive (expected) counts for specified combinations of covariates. It is based on the MCMC output obtained using [logpoissonRE]().

Usage

```
logpoissonRE.predict(nobs, x, xb, offset, cluster,
      intcpt.random=FALSE, hierar.center=FALSE,
      drandom=c("normal", "gspline"),
      betaF, betaR, varR, is.varR=TRUE,
      prior.gspline,
      probs, values=FALSE,
      dir=getwd(), wfile, indfile, header=TRUE, logw, is.indfile,
      skip=0, nwrite)
```

Arguments

nobs	number of covariate combinations for which we want to perform a prediction
x	covariate combinations for which we want to perform a prediction. It should have the same structure as in [logpoissonRE](#) used to obtain the MCMC output
xb	covariate combinations for which we want to perform a prediction. It should have the same structure as in [logpoissonRE](#) used to obtain the MCMC output
offset	optional offset vector.
cluster	vector defining pertinence of the single observations to clusters. It is useful when we want to predict longitudinal profiles. See also the same argument in [logpoissonRE](#).
intcpt.random	see the same argument in [logpoissonRE](#)
hierar.center	see the same argument in [logpoissonRE](#)
drandom	see the same argument in [logpoissonRE](#)
betaF	sampled values of the fixed effects. This should be a (sub)sample from the MCMC output stored in the file 'betaF.sim'
betaR	sampled values of the mean of random effects. This should be a (sub)sample from the MCMC output stored in the file 'betaR.sim' It is only needed if hierar.center is TRUE.
varR	sampled values of either (co)variance matrices or precision (matrices) for random effects if there are any. This should be a (sub)sample of either the first or second half of the columns stored in the file 'varR.sim'
is.varR	logical indicating whether varR gives (co)variance (is.varR TRUE) or precisions (inverse variances) (is.varR FALSE)
prior.gspline	if drandom is **gspline** this is a list specifying the G-splines. It should have the same structure as the same argument in [logpoissonRE](#) used to obtain the MCMC output. However, it is satisfactory if the items **K**, **delta** and **sigma** are given.
probs	probabilities for which the (pointwise) sample quantiles of the predictive counts should be computed. If not given only average (and values) of the predictive counts are computed
values	if TRUE also values of the predictive counts at each (MCMC) iteration are returned. If FALSE only sample mean (and quantiles) of the predictive probabilities are returned
dir	character giving the directory where the file with (sampled) G-spline (log-)weights is stored. Needed only if drandom is **gspline**.
wfile	character giving the name of the file with (sampled) G-spline (log-)weights. Needed only if drandom is **gspline**. In most cases, for **univariate** G-spline this argument will be equal to "logweight.sim" and for **bivariate** G-spline equal to "weight.sim".

indfile	character giving the name of the file where we stored indeces of these G-spline components for which the weights are stored in the file given by wfile. The corresponding file should have the same structure as 'knotInd.sim' created by logpoissonRE. Needed only if is.indfile is TRUE. In most cases, for **univariate** G-spline it does not have to be specified and for **bivariate** G-spline it will be equal to "knotInd.sim".
header	logical indicating whether the files wfile, indfile contain a header. Needed only if drandom is **gspline**.
logw	logical indicating whether the file wfile contains logarithms of the weights. Needed only if drandom is **gspline**. In most cases, for **univariate** G-spline it will be TRUE and for **bivariate** G-spline it will be FALSE.
is.indfile	logical. If TRUE then wfile contains only the non-zero weights and the G-spline is reconstructed using indfile. If FALSE then wfile must contain on each row weights of all components and indfile is ignored. Needed only if drandom is **gspline** and random effects are bivariate.
skip	number of data rows that should be skipped at the beginning of the files wfile, indfile.
nwrite	frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change)

Value

A list with the following components (description below applies for the case with prob=0.5)

Mean	a matrix with 1 column giving in each row posterior predictive mean of the count $E(Y)$ for a given covariate combination.
50%	a matrix with 1 column giving in each row posterior predictive quantile (here 50% quantile) of the count for a given covariate combination. There is one component of this type in the resulting list for each value of probs.
values	a matrix with n columns, where n denotes the number of covariate combinations for which we perform the prediction, and number of rows equal to the length of the MCMC. Each column gives sampled counts a given covariate combination. It is returned only if values is TRUE.

Author(s)

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

References

Komárek, A. and Lesaffre, E. (2008). Generalized linear mixed model with a penalized Gaussian mixture as a random-effects distribution. *Computational Statistics and Data Analysis*, **52**, 3441–3458.

See Also

logpoissonRE, logpoisson, [glm](glm).

---

maxPosterProb   *G-spline utility*

Description

For given G-spline basis and given data, it determines for each data point the G-spline component for which the value of the basis density is maximal.

Usage

```
maxPosterProb(data, intercept, std.dev, K, delta, sigma)
```

Arguments

data	numeric vector or matrix with data. If given as a matrix then rows correspond to observations and columns to margins.
intercept	numeric vector of length 1 or ncol(data) with intercepts for each margin. If given as a number, it is recycled.
std.dev	numeric vector of length 1 or ncol(data) with standard deviations for each margin. If given as a number, it is recycled.
K	numeric vector of length 1 or ncol(data) which specifies, for each marginal G-spline, then number of knots on each side of the zero knot. That is, the i-th marginal G-spline has $2K_i + 1$ knots. If given as a number, it is recycled.
delta	numeric vector of length 1 or ncol(data) which specifies the distance between two consecutive knots for each marginal G-spline. That is, the i-th marginal G-spline has the following knots $$\mu_{i,j} = j\,\delta_i,\ j = -K_i, \ldots, K_i.$$ If given as a number, it is recycled.
sigma	numeric vector of length 1 or ncol(data) with basis standard deviations for marginal G-splines. If given as a number, it is recycled.

Value

Matrix which specifies determined components (indeces are on scale $-K_i, \ldots, K_i$).

Author(s)

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

See Also

cumlogitRE, logpoissonRE.

Examples

```
N <- 100
intcpt <- c(0, 5, 15)
std.dev <- c(1, 0.5, 3)
data <- data.frame(b1=rnorm(N, intcpt[1], std.dev[1]),
                   b2=rnorm(N, intcpt[2], std.dev[2]),
                   b3=rnorm(N, intcpt[3], std.dev[3]))
alloc <- maxPosterProb(data=data, intercept=intcpt, std.dev=std.dev,
                       K=15, delta=0.3, sigma=0.2)

par(mfrow=c(1, 3), bty="n")
for (i in 1:3) hist(alloc[,i], prob=TRUE, col="seagreen3",
                    xlab="Allocation", breaks=(-15):15,
                    main=paste("Margin ", i, sep=""))
```

QuantileFun   *Sample quantiles*

Description

This is (almost) the same as quantile(x), or apply(x, 1, quantile) or apply(x, 2, quantile).
The motivation to write it was to validate my C++ function.

Usage

```
QuantileFun(x, probs=seq(0, 1, 0.25), vals.in.cols=TRUE)
```

Arguments

x	values of the function
probs	numeric vector of probabilities with values in $[0, 1]$
vals.in.cols	if TRUE then it is assumed that function f evaluated in a specific grid point over (MCMC) iterations is stored in a column of x. That is (MCMC) iterations correspond to rows.

Value

A data.frame with 1 row for each probs value.

Author(s)

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

See Also

quantile.

Examples

```
probs <- c(0, 0.25, 0.354, 0.5, 0.75, 1)

x <- rnorm(1001)
QuantileFun(x, probs=probs)
quantile(x, probs=probs)

n <- 1001
xx <- data.frame(x1=rnorm(n), x2=rgamma(n, shape=1, rate=1), x3=1:n)
QuantileFun(xx, probs=probs, vals.in.cols=TRUE)
apply(xx, 2, quantile, probs=probs)

xx2 <- t(xx)
QuantileFun(xx2, probs=probs, vals.in.cols=FALSE)
apply(xx2, 1, quantile, probs=probs)
```

---

scanFH   *Read Data Values*

Description

Read numeric data into a data frame from a file. Header is assumed to be present in the file.

Usage

```
scanFH(file, quiet=FALSE)
```

Arguments

file   the name of a file to read data values from. If the specified file is `""`, then input
                is taken from the keyboard (or stdin if input is redirected). (In this case input
                can be terminated by a blank line or an EOF signal, `Ctrl-D` on Unix and `Ctrl-Z`
                on Windows.)

                Otherwise, the file name is interpreted *relative* to the current working directory
                (given by getwd()), unless it specifies an *absolute* path. Tilde-expansion is
                performed where supported.

                Alternatively, file can be a connection, which will be opened if necessary,
                and if so closed at the end of the function call. Whatever mode the connection
                is opened in, any of LF, CRLF or CR will be accepted as the EOL marker for a
                line and so will match sep = `"\n"`.

                `file` can also be a complete URL.

                To read a data file not in the current encoding (for example a Latin-1 file in a
                UTF-8 locale or conversely) use a file connection setting the encoding argu-
                ment.

quiet   logical: if FALSE (default), scan() will print a line, saying how many items have been read.

Details

See [scan](#).

Value

data.frame with read data values.

Author(s)

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

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth \& Brooks/Cole.

See Also

[scan](#)

Examples

```
cat("x y z", "1 2 3", "1 4 6", "10 20 30", file="ex.data", sep="\n")
pp <- scanFH("ex.data", quiet=FALSE)
pp <- scanFH("ex.data", quiet= TRUE)
print(pp)
unlink("ex.data") # tidy up
```

---

summaryGspline1   *Summary for a univariate G-spline (penalized Gaussian mixture)*

Description

This function is primarily designed to work out the MCMC output from functions [cumlogitRE](#) and [logpoissonRE](#) in which a distribution of the univariate random effect was specified as a G-spline. It computes posterior pointwise mean and quantiles for a G-spline density based on the MCMC output.

Usage

```
summaryGspline1(x, mu, sigma,
   standard=TRUE, intcpt, scale,
   probs, values=FALSE,
   dir=getwd(), wfile="logweight.sim", header=TRUE, logw=TRUE,
   skip=0, nwrite)
```

Arguments

x	grid of values at which we want to evaluate the G-spline density
mu	a vector with G-spline knots (means of basis G-splines)
sigma	basis standard deviation(s). If a single number is supplied then it is assumed that all basis G-splines have the same standard deviation. Alternatively a vector of the same length as mu can be given in which case the basis G-splines do not necessarily have the same standard deviations
standard	logical, if TRUE then the standardized (zero-mean, unit-variance) G-spline densities are computed and summarized
intcpt	a vector with sampled intercept values. If not supplied it is assumed that all intercepts are equal to zero. It does not have to be supplied if standard=TRUE.
scale	a vector with sampled values of the G-spline scale (in most of my papers denoted by τ) If not supplied it is assumed that all scale values are equal to one. It does not have to be supplied if standard=TRUE.
probs	probabilities for which the (pointwise) sample quantiles of the G-spline density should be computed. If not given only average (and values) of the G-spline are computed
values	if TRUE also values of the G-spline at each (MCMC) iteration are returned. If FALSE only sample mean (and quantiles) of the G-spline are returned
dir	character giving the directory where the file with (sampled) G-spline (log-)weights is stored
wfile	character giving the name of the file with (sampled) G-spline (log-)weights
header	logical indicating whether the file wfile contains a header
logw	logical indicating whether the file wfile contains logarithms of the weights
skip	number of data rows that should be skipped at the beginning of the file wfile
nwrite	frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change)

Value

A list with the following components (component 'values' is present only when the argument values was TRUE):

summary	a data.frame with the following columns (the description below applies to the situation when probs=0.5):

	'x'	a grid of values at which the G-spline density is evaluated
	'Mean'	pointwise posterior mean of the G-spline density
	'50%'	pointwise posterior 50% quantile of the G-spline density. There is one column of this type for each probs value

values	a matrix with one column for each x value and number of rows equal to the

length of the MCMC. In each row, there is a G-spline density evaluated at one MCMC iteration.

Author(s)

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

Examples

```
### See ex-Toenail.pdf, ex-Toenail.R
### and ex-Epileptic.pdf, ex-Epileptic.R
### available in the documentation
### to the package
```

---

summaryGspline2   *Summary for a bivariate G-spline (penalized Gaussian mixture)*

Description

This function is primarily designed to work out the MCMC output from functions `cumlogitRE` and `logpoissonRE` in which a distribution of the bivariate random effect was specified as a G-spline. It computes posterior pointwise mean and quantiles for a G-spline density based on the MCMC output.

Besides the summary for the joint bivariate G-spline density it also directly computes summaries for both marginal G-splines.

Usage

```
summaryGspline2(x1, x2, mu1, mu2, sigma1, sigma2,
   standard=TRUE, intcpt, scale,
   probs, values=FALSE,
   dir=getwd(), wfile="weight.sim", indfile="knotInd.sim",
   header=TRUE, logw=FALSE, is.indfile=TRUE,
   skip=0, nwrite)
```

Arguments

x1	grid of values for the first margin at which we evaluate the G-spline density
x2	grid of values for the second margin at which we evaluate the G-spline density
mu1	a vector with G-spline knots (means of basis G-splines) for the first margin
mu2	a vector with G-spline knots (means of basis G-splines) for the second margin
sigma1	basis standard deviation(s) for the first margin. If a single number is supplied then it is assumed that all basis G-splines in the first margin have the same standard deviation. Alternatively a vector of the same length as mu1 can be given in which case the basis G-splines in the first margin do not necessarily have the same standard deviations

sigma2	basis standard deviation(s) for the second margin. If a single number is supplied then it is assumed that all basis G-splines in the second margin have the same standard deviation. Alternatively a vector of the same length as mu2 can be given in which case the basis G-splines in the second margin do not necessarily have the same standard deviations
standard	if TRUE then the standardized (zero-mean, unit-variance) G-splines are computed and summarized
intcpt	a two-column matrix with sampled intercept values. If not supplied it is assumed that all intercepts are equal to zero. It does not have to be supplied if standard=TRUE.
scale	a two-column matrix with sampled intercept values of the G-spline scale (in most of my papers denoted by τ) If not supplied it is assumed that all scale values are equal to one. It does not have to be supplied if standard=TRUE.
probs	probabilities for which the (pointwise) sample quantiles of the G-spline should be computed. If not given only average (and values) of the G-spline are computed
values	if TRUE also values of the G-spline at each (MCMC) iteration are returned. If FALSE only sample mean (and quantiles) of the G-spline are returned
dir	character giving the directory where the file with (sampled) G-spline (log-)weights is stored
wfile	character giving the name of the file with (sampled) G-spline (log-)weights
indfile	character giving the name of the file where it is indicated which G-spline components correspond to non-zero weights. It does not have to be supplied if is.indfile=FALSE.
header	logical indicating whether the files wfile, indfile contain a header
logw	logical indicating whether the file wfile contains logarithms of the weights
is.indfile	logical. If TRUE then wfile contains only the non-zero weights and the G-spline is reconstructed using indfile. If FALSE then wfile must contain on each row weights of all components and indfile is ignored.
skip	number of data rows that should be skipped at the beginning of the files wfile, indfile
nwrite	frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change)

Value

A list with the following components (component 'values', 'values1', 'values2' are present only when the argument values was TRUE). The description below applies to the situation when probs=0.5.

summary	a list with the components

'x1'	grid of values for the first margin at which we evaluate the G-spline density
'x2'	grid of values for the second margin at which we evaluate the G-spline density
'Mean'	a matrix with `length(x1)` rows and `length(x2)` columns giving the pointwise posterior mean of the joint G-spline density
'50%'	a matrix with `length(x1)` rows and `length(x2)` columns giving the pointwise posterior 50% quantile of the joint G-spline density. There is a matrix of this type for each `probs` value.

summary1	a `data.frame` with the following columns

'x'	a grid of values at which the first marginal G-spline density is evaluated
'Mean'	pointwise posterior mean of the first marginal G-spline density
'50%'	pointwise posterior 50% quantile of the first marginal G-spline density. There is one column of this type for each `probs` value

summary2	a `data.frame` for the second marginal G-spline having the same structure as 'summary1'.
values	a matrix with one column for each (x1, x2) value and number of rows equal to the length of the MCMC. In each row, there is a joint G-spline density evaluated at one MCMC iteration.
values1	a matrix with one column for each x1 value and number of rows equal to the length of the MCMC. In each row, there is the first marginal G-spline density evaluated at one MCMC iteration.
values1	a matrix with one column for each x1 value and number of rows equal to the length of the MCMC. In each row, there is the second marginal G-spline density evaluated at one MCMC iteration.

Author(s)

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

Examples

```
### See ex-Toenail.pdf, ex-Toenail.R
### and ex-Epileptic.pdf, ex-Epileptic.R
### available in the documentation
### to the package
```

---

toenail	*Toenail infection*

Description

This data set considers information from a longitudinal clinical trial in dermatology which was set up to compare the efficacy of two oral treatments for toenail infection (De Backer at al., 1998). One of the end points of the study was the degree of onycholysis which expresses the degree of separation of the nail plate from the nail-bed (0, absent; 1, mild; 2, moderate; 3, severe) and was evaluated at seven visits (approximately on weeks 0, 4, 8, 12, 24, 36 and 48). In total, 1 908 measurements on 294 patients are available. In this dataset, only a dichotomized onycholysis (0, absent or mild; 1, moderate or severe) is given (variable `infect`).

IMPORTANT NOTICE: The data have kindly been made available by Novartis, Belgium. The source of the data must be acknowledged in any publication which uses them (see Lesaffre and Spiessens, 2001 for more details).

Usage

```
data(toenail)
```

Format

A data frame with 1908 observations on the following 5 variables.

`idnr` identification number of the patient

`infect` binary response

`trt` treatment group

`time` time of measurement (in months)

`visit` visit number

Source

Lesaffre, E. and Spiessens, B. (2001). On the effect of the number of quadrature points in a logistic random-effects model: An example. *Applied Statistics*, **50**, 325–335.

References

De Backer, M., De Vroey, C., Lesaffre, E., Scheys, I., and De Keyser, P. (1998). Twelve weeks of continuous onychomycosis caused by dermatophytes: A double blind comparative trial of terbafine 250 mg/day versus itraconazole 200 mg/day. *Journal of the American Academy of Dermatology*, **38**, S57–S63.

Examples

```
data(toenail)
## maybe str(toenail); plot(toenail) ...
```

Index