Package 'bayesSurv'

December 18, 2023

Version 3.7

Date 2023-12-16

Title Bayesian Survival Regression with Flexible Error and Random Effects Distributions

Depends R (>= 3.0.0), survival, coda, smoothSurv

Imports graphics, stats, utils

Description Contains Bayesian implementations of the Mixed-Effects Accelerated Failure Time (MEAFT) models for censored data. Those can be not only right-censored but also interval-censored, doubly-interval-censored or misclassified interval-censored. The methods implemented in the package have been published in Komárek and Lesaffre (2006, Stat. Modelling) <doi:10.1191/1471082X06st107oa>, Komárek, Lesaffre and Legrand (2007, Stat. in Medicine) <doi:10.1002/sim.3083>, Komárek and Lesaffre (2007, Stat. Sinica) <https://www3.stat.sinica.edu.tw/statistica/oldpdf/A17n27.pdf>, Komárek and Lesaffre (2008, JASA) <doi:10.1198/016214507000000563>, García-Zattera, Jara and Komárek (2016, Biometrics) <doi:10.1111/biom.12424>.

Encoding UTF-8

License GPL (>= 2)

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

ZipData no

NeedsCompilation yes

Author Arnošt Komárek [aut, cre] (<https://orcid.org/0000-0001-8778-3762>)

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

Repository CRAN

Date/Publication 2023-12-18 18:00:04 UTC

R topics documented:

`bayesBisurvreg` *Population-averaged accelerated failure time model for bivariate, possibly doubly-interval-censored data. The error distribution is expressed as a penalized bivariate normal mixture with high number of components (bivariate G-spline).*

Description

A function to estimate a regression model with bivariate (possibly right-, left-, interval- or doubly-interval-censored) data. In the case of doubly interval censoring, different regression models can be specified for the onset and event times.

The error density of the regression model is specified as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variance matrices). This function performs an MCMC sampling from the posterior distribution of unknown quantities.

For details, see Komárek (2006) and Komárek and Lesaffre (2006).

We explain first in more detail a model without doubly censoring. Let $T_{i,l},\ i = 1, \ldots, N,\ l = 1, 2$ be event times for ith cluster and the first and the second unit. The following regression model is assumed:

$$\log(T_{i,l}) = \beta' x_{i,l} + \varepsilon_{i,l}, \quad i = 1, \ldots, N,\ l = 1, 2$$

where β is unknown regression parameter vector and $x_{i,l}$ is a vector of covariates. The bivariate error terms $\varepsilon_i = (\varepsilon_{i,1},\ \varepsilon_{i,2})',\ i = 1, \ldots, N$ are assumed to be i.i.d. with a bivariate density $g_\varepsilon(e_1,\ e_2)$. This density is expressed as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variance matrices). We distinguish two, theoretically equivalent, specifications.

Specification 1

$$(\varepsilon_1,\ \varepsilon_2)' \sim \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} N_2(\mu_{(j_1,j_2)},\ \text{diag}(\sigma_1^2,\ \sigma_2^2))$$

where $\sigma_1^2,\ \sigma_2^2$ are **unknown** basis variances and $\mu_{(j_1,j_2)} = (\mu_{1,j_1},\ \mu_{2,j_2})'$ is an equidistant grid of knots symmetric around the **unknown** point $(\gamma_1,\ \gamma_2)'$ and related to the unknown basis variances through the relationship

$$\mu_{1,j_1} = \gamma_1 + j_1 \delta_1 \sigma_1, \quad j_1 = -K_1, \ldots, K_1,$$

$$\mu_{2,j_2} = \gamma_2 + j_2 \delta_2 \sigma_2, \quad j_2 = -K_2, \ldots, K_2,$$

where $\delta_1,\ \delta_2$ are fixed constants, e.g. $\delta_1 = \delta_2 = 2/3$ (which has a justification of being close to cubic B-splines).

Specification 2

$$(\varepsilon_1,\ \varepsilon_2)' \sim (\alpha_1,\ \alpha_2)' + \boldsymbol{S}\,(V_1,\ V_2)'$$

where $(\alpha_1,\ \alpha_2)'$ is an **unknown** intercept term and $\boldsymbol{S}$ is a diagonal matrix with τ_1 and τ_2 on a diagonal, i.e. $\tau_1,\ \tau_2$ are **unknown** scale parameters. $(V_1,\ V_2)')$ is then standardized bivariate error term which is distributed according to the bivariate normal mixture, i.e.

$$(V_1,\ V_2)' \sim \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} N_2(\mu_{(j_1,j_2)},\ \text{diag}(\sigma_1^2, \sigma_2^2))$$

where $\mu_{(j_1,j_2)} = (\mu_{1,j_1},\ \mu_{2,j_2})'$ is an equidistant grid of **fixed** knots (means), usually symmetric about the **fixed** point $(\gamma_1,\ \gamma_2)' = (0, 0)'$ and $\sigma_1^2,\ \sigma_2^2$ are **fixed** basis variances. Reasonable values for the numbers of grid points K_1 and K_2 are $K_1 = K_2 = 15$ with the distance between the two knots equal to $\delta = 0.3$ and for the basis variances $\sigma_1^2 \sigma_2^2 = 0.2^2$.

Personally, I found Specification 2 performing better. In the paper Komárek and Lesaffre (2006) only Specification 2 is described.

The mixture weights $w_{j_1,j_2},\ j_1 = -K_1,\ldots,K_1,\ j_2 = -K_2,\ldots,K_2$ are not estimated directly. To avoid the constraints $0 < w_{j_1,j_2} < 1$ and $\sum_{j_1=-K_1}^{K_1}\sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} = 1$ transformed weights $a_{j_1,j_2},\ j_1 = -K_1,\ldots,K_1,\ j_2 = -K_2,\ldots,K_2$ related to the original weights by the logistic transformation:

$$a_{j_1,j_2} = \frac{\exp(w_{j_1,j_2})}{\sum_{m_1}\sum_{m_2}\exp(w_{m_1,m_2})}$$

are estimated instead.

A Bayesian model is set up for all unknown parameters. For more details I refer to Komárek and Lesaffre (2006) and to Komárek (2006).

If there are doubly-censored data the model of the same type as above can be specified for both the onset time and the time-to-event.

Usage

```
bayesBisurvreg(formula, formula2, data = parent.frame(),
   na.action = na.fail, onlyX = FALSE,
   nsimul = list(niter = 10, nthin = 1, nburn = 0, nwrite = 10),
   prior, prior.beta, init = list(iter = 0),
   mcmc.par = list(type.update.a = "slice", k.overrelax.a = 1,
                   k.overrelax.sigma = 1, k.overrelax.scale = 1),
   prior2, prior.beta2, init2,
   mcmc.par2 = list(type.update.a = "slice", k.overrelax.a = 1,
                    k.overrelax.sigma = 1, k.overrelax.scale = 1),
   store = list(a = FALSE, a2 = FALSE, y = FALSE, y2 = FALSE,
                r = FALSE, r2 = FALSE),
   dir)
```

Arguments

`formula`	model formula for the regression. In the case of doubly-censored data, this is the model formula for the onset time. Data are assumed to be sorted according to subjects and within subjects according to the types of the events that determine the bivariate survival distribution, i.e. the response vector must be $t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}, t_{3,1}, t_{3,2}, \ldots, t_{n,1}, t_{n,2}$. The rows of the design matrix with covariates must be sorted analogically. The left-hand side of the formula must be an object created using `Surv`.
`formula2`	model formula for the regression of the time-to-event in the case of doubly-censored data. Ignored otherwise. The same remark as for `formula` concerning the sort order applies here.
`data`	optional data frame in which to interpret the variables occuring in the formulas.
`na.action`	the user is discouraged from changing the default value `na.fail`.
`onlyX`	if `TRUE` no MCMC sampling is performed and only the design matrix (matrices) are returned. This can be useful to set up correctly priors for regression parameters in the presence of `factor` covariates.
`nsimul`	a list giving the number of iterations of the MCMC and other parameters of the simulation.

niter total number of sampled values after discarding thinned ones, burn-up included;

nthin thinning interval;

nburn number of sampled values in a burn-up period after discarding thinned values. This value should be smaller than `niter`. If not, `nburn` is set to `niter - 1`. It can be set to zero;

nwrite an interval at which information about the number of performed iterations is print on the screen and during the burn-up period an interval with which the sampled values are writen to files;

`prior` a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by `formula`. See `prior` argument of `bayesHistogram` function for more detail. In this list also 'Specification' as described above is specified.

`prior.beta` prior specification for the regression parameters, in the case of doubly censored data for the regression parameters of the onset time. I.e. it is related to `formula`.

This should be a list with the following components:

mean.prior a vector specifying a prior mean for each `beta` parameter in the model.

var.prior a vector specifying a prior variance for each `beta` parameter.

It is recommended to run the function bayesBisurvreg first with its argument `onlyX` set to `TRUE` to find out how the betas are sorted. They must correspond to a design matrix X taken from `formula`.

`init` an optional list with initial values for the MCMC related to the model given by `formula`. The list can have the following components:

iter the number of the iteration to which the initial values correspond, usually zero.

beta a vector of initial values for the regression parameters. It must be sorted in the same way as are the columns in the design matrix. Use `onlyX=TRUE` if you do not know how the columns in the design matrix are created.

a a matrix of size $(2K_1 + 1) \times (2K_2 + 1)$ with the initial values of transformed mixture weights.

lambda initial values for the Markov random fields precision parameters. According to the chosen prior for the transformed mixture weights, this is either a number or a vector of length 2.

gamma a vector of length 2 of initial values for the middle knots γ_1, γ_2 in each dimension.

If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended (for easier interpretation of the results) to set `init$gamma` to zero for all dimensions (default behavior).

If 'Specification' is 1 `init$gamma` should be approximately equal to the mean value of the residuals in each margin.

sigma a vector of length 2 of initial values of the basis standard deviations σ_1, σ_2.

If 'Specification' is 2 this value will not be changed by the MCMC and it is recommended to set it approximately equal to the range of standardized

data (let say 4 + 4) divided by the number of knots in each margin and multiplied by something like 2/3.
If 'Specification' is 1 this should be approximately equal to the range of the residuals divided by the number of knots in each margin and multiplied again by something like 2/3.

intercept a vector of length 2 of initial values of the intercept terms α_1, α_2.
If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to zero for both dimensions.

scale a vector of length 2 of initial values of the scale parameters τ_1, τ_2.
If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to one for both dimensions.

y a matrix with 2 columns and N rows with initial values of log-event-times for each cluster in rows.

r a matrix with 2 columns and N rows with initial component labels for each bivariate residual in rows. All values in the first column must be between $-K_1$and K_1 and all values in the second column must be between $-K_2$ and K_2. See argument `init` of the function `bayesHistogram` for more details.

`mcmc.par` a list specifying how some of the G-spline parameters related to `formula` are to be updated. The list can have the following components (all of them have their default values):

type.update.a G-spline transformed weights a can be updated using one of the following algorithms:

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

Default is `slice`.

k.overrelax.a if `type.update.a == "slice"` some updates are overrelaxed. Then every `k.overrelax.a`th iteration is not overrelaxed. Default is `k.overrelax.a = 1`, i.e. no overrelaxation

k.overrelax.sigma G-spline basis standard deviations are updated using the slice sampler of Neal (2003). At the same time, overrelaxation can be used. Then every k.overrelax.sigma th update is not overrelaxed. Default is `k.overrelax.sigma = 1`, i.e. no overrelaxation

k.overrelax.scale G-spline scales are updated using the slice sampler of Neal (2003). At the same time, overrelaxation can be used. Then every k.overrelax.scale th update is not overrelaxed. Default is `k.overrelax.scale = 1`, i.e. no overrelaxation

`prior2` a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by `formula2`. See `prior` argument of `bayesHistogram` function for more detail.

`prior.beta2` prior specification for the regression parameters of time-to-event in the case of doubly censored data (related to `formula2`). This should be a list with the same structure as `prior.beta`.

`init2` an optional list with initial values for the MCMC related to the model given by `formula2`. The list has the same structure as `init`.

`mcmc.par2` a list specifying how some of the G-spline parameters related to `formula2` are to be updated. The list has the same structure as `mcmc.par`.

`store` a list of logical values specifying which chains that are not stored by default are to be stored. The list can have the following components.

a if `TRUE` then all the transformed mixture weights $a_{k_1,\,k_2}$, $k_1 = -K_1, \ldots, K_1$, $k_2 = -K_2, \ldots, K_2$, related to the G-spline of `formula` are stored.

a2 if `TRUE` and there are doubly-censored data then all the transformed mixture weights $a_{k_1,\,k_2}$, $k_1 = -K_1, \ldots, K_1$, $k_2 = -K_2, \ldots, K_2$, related to the G-spline of `formula2` are stored.

y if `TRUE` then augmented log-event times for all observations related to the `formula` are stored.

y2 if `TRUE` then augmented log-event times for all observations related to `formula2` are stored.

r if `TRUE` then labels of mixture components for residuals related to `formula` are stored.

r2 if `TRUE` then labels of mixture components for residuals related to `formula2` are stored.

`dir` a string that specifies a directory where all sampled values are to be stored.

Value

A list of class `bayesBisurvreg` containing an information concerning the initial values and prior choices.

Files created

Additionally, the following files with sampled values are stored in a directory specified by `dir` argument of this function (some of them are created only on request, see `store` parameter of this function).

Headers are written to all files created by default and to files asked by the user via the argument `store`. During the burn-in, only every `nsimul$nwrite` value is written. After the burn-in, all sampled values are written in files created by default and to files asked by the user via the argument `store`. In the files for which the corresponding `store` component is `FALSE`, every `nsimul$nwrite` value is written during the whole MCMC (this might be useful to restart the MCMC from some specific point).

The following files are created:

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

mixmoment.sim columns labeled `k`, `Mean.1`, `Mean.2`, `D.1.1`, `D.2.1`, `D.2.2`, where

k = number of mixture components that had probability numerically higher than zero;

Mean.1 = $\mathrm{E}(\varepsilon_{i,1})$;

Mean.2 = $\mathrm{E}(\varepsilon_{i,2})$;

D.1.1 = $\mathrm{var}(\varepsilon_{i,1})$;

D.2.1 = cov($\varepsilon_{i,1}$, $\varepsilon_{i,2}$);

D.2.2 = var($\varepsilon_{i,2}$);

all related to the distribution of the error term from the model given by `formula`.

mixmoment_2.sim in the case of doubly-censored data, the same structure as `mixmoment.sim`, however related to the model given by `formula2`.

mweight.sim sampled mixture weights $w_{k_1,\,k_2}$ of mixture components that had probabilities numerically higher than zero. Related to the model given by `formula`.

mweight_2.sim in the case of doubly-censored data, the same structure as `mweight.sim`, however related to the model given by `formula2`.

mmean.sim indeces $k_1,\ k_2$, $k_1 \in \{-K_1, \ldots, K_1\}$, $k_2 \in \{-K_2, \ldots, K_2\}$ of mixture components that had probabilities numerically higher than zero. It corresponds to the weights in `mweight.sim`. Related to the model given by `formula`.

mmean_2.sim in the case of doubly-censored data, the same structure as `mmean.sim`, however related to the model given by `formula2`.

gspline.sim characteristics of the sampled G-spline (distribution of $(\varepsilon_{i,1},\, \varepsilon_{i,2})'$) related to the model given by `formula`. This file together with `mixmoment.sim`, `mweight.sim` and `mmean.sim` can be used to reconstruct the G-spline in each MCMC iteration.

The file has columns labeled `gamma1`, `gamma2`, `sigma1`, `sigma2`, `delta1`, `delta2`, `intercept1`, `intercept2`, `scale1`, `scale2`. The meaning of the values in these columns is the following:

gamma1 = the middle knot γ_1 in the first dimension. If 'Specification' is 2, this column usually contains zeros;

gamma2 = the middle knot γ_2 in the second dimension. If 'Specification' is 2, this column usually contains zeros;

sigma1 = basis standard deviation σ_1 of the G-spline in the first dimension. This column contains a fixed value if 'Specification' is 2;

sigma2 = basis standard deviation σ_2 of the G-spline in the second dimension. This column contains a fixed value if 'Specification' is 2;

delta1 = distance $delta_1$ between the two knots of the G-spline in the first dimension. This column contains a fixed value if 'Specification' is 2;

delta2 = distance δ_2 between the two knots of the G-spline in the second dimension. This column contains a fixed value if 'Specification' is 2;

intercept1 = the intercept term α_1 of the G-spline in the first dimension. If 'Specification' is 1, this column usually contains zeros;

intercept2 = the intercept term α_2 of the G-spline in the second dimension. If 'Specification' is 1, this column usually contains zeros;

scale1 = the scale parameter τ_1 of the G-spline in the first dimension. If 'Specification' is 1, this column usually contains ones;

scale2 = the scale parameter τ_2 of the G-spline in the second dimension. 'Specification' is 1, this column usually contains ones.

gspline_2.sim in the case of doubly-censored data, the same structure as `gspline.sim`, however related to the model given by `formula2`.

mlogweight.sim fully created only if `store$a = TRUE`. The file contains the transformed weights $a_{k_1,\,k_2}$, $k_1 = -K_1, \ldots, K_1$, $k_2 = -K_2, \ldots, K_2$ of all mixture components, i.e. also of components that had numerically zero probabilities. This file is related to the model given by `formula`.

mlogweight_2.sim fully created only if `store$a2 = TRUE` and in the case of doubly-censored data, the same structure as `mlogweight.sim`, however related to the model given by `formula2`.

r.sim fully created only if `store$r = TRUE`. The file contains the labels of the mixture components into which the residuals are intrinsically assigned. Instead of double indeces $(k_1,\, k_2)$, values from 1 to $(2\,K_1 + 1) \times (2\,K_2 + 1)$ are stored here. Function `vecr2matr` can be used to transform it back to double indeces.

r_2.sim fully created only if `store$r2 = TRUE` and in the case of doubly-censored data, the same structure as `r.sim`, however related to the model given by `formula2`.

lambda.sim either one column labeled `lambda` or two columns labeled `lambda1` and `lambda2`. These are the values of the smoothing parameter(s) λ (hyperparameters of the prior distribution of the transformed mixture weights $a_{k_1,\,k_2}$). This file is related to the model given by `formula`.

lambda_2.sim in the case of doubly-censored data, the same structure as `lambda.sim`, however related to the model given by `formula2`.

beta.sim sampled values of the regression parameters β related to the model given by `formula`. The columns are labeled according to the `colnames` of the design matrix.

beta_2.sim in the case of doubly-censored data, the same structure as `beta.sim`, however related to the model given by `formula2`.

Y.sim fully created only if `store$y = TRUE`. It contains sampled (augmented) log-event times for all observations in the data set.

Y_2.sim fully created only if `store$y2 = TRUE` and in the case of doubly-censored data, the same structure as `Y.sim`, however related to the model given by `formula2`.

logposter.sim columns labeled `loglik`, `penalty` or `penalty1` and `penalty2`, `logprw`. This file is related to the model given by `formula`. The columns have the following meaning.

loglik $= -N\Big\{\log(2\pi) + \log(\sigma_1) + \log(\sigma_2)\Big\} - 0.5\sum_{i=1}^{N}\Big\{(\sigma_1^2\,\tau_1^2)^{-1}\;(y_{i,1} - x'_{i,1}\beta - \alpha_1 - \tau_1\mu_{1,\,r_{i,1}})^2 + (\sigma_2^2\,\tau_2^2)^{-1}\;(y_{i,2} - x'_{i,2}\beta - \alpha_2 - \tau_2\mu_{2,\,r_{i,2}})^2\Big\}$

where $y_{i,l}$ denotes (augmented) *(i,l)*th true log-event time. In other words, `loglik` is equal to the conditional log-density $\sum_{i=1}^{N}\,\log\Big\{p\big((y_{i,1},\,y_{i,2})\;\big|\;r_i,\;\beta,\;\text{G-spline}\big)\Big\}$;

penalty1: If `prior$neighbor.system = "uniCAR"`: the penalty term for the first dimension not multiplied by `lambda1`;

penalty2: If `prior$neighbor.system = "uniCAR"`: the penalty term for the second dimension not multiplied by `lambda2`;

penalty: If `prior$neighbor.system` is different from `"uniCAR"`: the penalty term not multiplied by `lambda`;

logprw $= -2\,N\,\log\big\{\sum_{k_1}\sum_{k_2}a_{k_1,\,k_2}\big\} + \sum_{k_1}\sum_{k_2}N_{k_1,\,k_2}\,a_{k_1,\,k_2}$, where $N_{k_1,\,k_2}$ is the number of residuals assigned intrinsincally to the $(k_1,\,k_2)$th mixture component.

In other words, `logprw` is equal to the conditional log-density $\sum_{i=1}^{N}\log\big\{p(r_i\;|\;\text{G-spline weights})\big\}$.

logposter_2.sim in the case of doubly-censored data, the same structure as `lambda.sim`, however related to the model given by `formula2`.

Author(s)

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

References

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

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failure time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3 - 22.

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

Examples

```
## See the description of R commands for
## the population averaged AFT model
## with the Signal Tandmobiel data,
## analysis described in Komarek and Lesaffre (2006),
##
## R commands available in the documentation
## directory of this package as
## - see ex-tandmobPA.R and
##   https://www2.karlin.mff.cuni.cz/ komarek/software/bayesSurv/ex-tandmobPA.pdf
##
```

`bayesDensity` *Summary for the density estimate based on the mixture Bayesian AFT model.*

Description

Function to summarize the results obtained using `bayessurvreg1` function.

Compute the conditional (given the number of mixture components) and unconditional estimate of the density function based on the values sampled using the reversible jumps MCMC (MCMC average evaluated in a grid of values).

Give also the values of each sampled density evaluated at that grid (returned as the attribute of the resulting object). Methods for printing and plotting are also provided.

Usage

```
bayesDensity(dir, stgrid, centgrid, grid, n.grid = 100,
    skip = 0, by = 1, last.iter,
    standard = TRUE, center = TRUE, unstandard = TRUE)
```

Arguments

dir	directory where to search for files 'mixmoment.sim', 'mweight.sim', mmean.sim', 'mvariance.sim' with the McMC sample.
stgrid	grid of values at which the sampled standardized densities are to be evaluated. If missing, the grid is automatically computed.
centgrid	grid of values at which the sampled centered (but not scaled) densities are to be evaluated. If missing. the grid is automatically computed.
grid	grid of values at which the sampled densities are to be evaluated. If missing, the grid is guessed from the first 20 sampled mixtures as the sequence starting with the minimal sampled mixture mean minus 3 standard deviations of the appropriate mixture component, ending with the maximal sampled mixture mean plus 3 standard deviations of the appropriate mixture component, of the length given by n.grid.
n.grid	the length of the grid if grid = NULL.
skip	number of rows that should be skipped at the beginning of each *.sim file with the stored sample.
by	additional thinning of the sample.
last.iter	index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file mixmoment.sim.
standard	if TRUE then also standardized (zero mean, unit variance) sampled densities are evaluated.
center	if TRUE then also centered (zero mean) sampled densities are evaluated.
unstandard	of TRUE then also original (unstandardized) sampled densities are evaluated.

Value

An object of class bayesDensity is returned. This object is a list and has potentially three components: standard, center and unstandard. Each of these three components is a data.frame with as many rows as number of grid points at which the density was evaluated and with columns called 'grid', 'unconditional' and 'k = 1', ..., 'k = k.max' giving a predictive errr density, either averaged over all sampled ks (unconditional) or averaged over a psecific number of mixture components.

Additionally, the object of class bayesDensity has three attributes:

sample.size	a vector of length 1 + kmax giving the frequency of each k in the sample.
moments	a data.frame with columns called 'intercept' and 'scale' giving the mean and variance of the sampled mixture at each iteration of the McMC.
k	a data.frame with one column called 'k' giving number of mixture components at each iteration.

There exist methods to print and plot objects of the class bayesDensity.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2007). Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution. *Statistica Sinica*, **17**, 549–569.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2007),
##
## R commands available
## in the documentation
## directory of this package
## - ex-cgd.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-cgd.pdf
##
## - ex-tandmobMixture.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobMixture.pdf
##
```

`bayesGspline` *Summary for the density estimate based on the model with Bayesian G-splines.*

Description

Compute the estimate of the density function based on the values sampled using the MCMC (MCMC average evaluated in a grid of values) in a model where density is specified as a Bayesian G-spline.

This function serves to summarize the MCMC chains related to the distributional parts of the considered models obtained using the functions: `bayesHistogram`, `bayesBisurvreg`, `bayessurvreg2`, `bayessurvreg3`.

If asked, this function returns also the values of the G-spline evaluated in a grid at each iteration of MCMC.

Usage

```
bayesGspline(dir, extens="", extens.adjust="_b",
   grid1, grid2, skip = 0, by = 1, last.iter, nwrite,
   only.aver = TRUE, standard = FALSE, version = 0)
```

Arguments

`dir` directory where to search for files ('mixmoment.sim', 'mweight.sim', 'mmean.sim', 'gspline.sim') with the MCMC sample.

`extens` an extension used to distinguish different sampled G-splines if more G-splines were used in one simulation (e.g. with doubly-censored data or in the model where both the error term and the random intercept were defined as the G-splines). According to which `bayes*survreg*` function was used, specify the argument `extens` in the following way.

bayesHistogram: always `extens = ""`

bayesBisurvreg:
- to compute the bivariate distribution of the *error* term for the *onset* time: `extens = ""`;
- to compute the bivariate distribution of the *error* term for the *event* time if there was doubly-censoring: `extens = "_2"`;

bayessurvreg2:
- to compute the distribution of the *error* term for the *onset* time: `extens = ""`;
- to compute the distribution of the *error* term for the *event* time if there was doubly-censoring: `extens = "_2"`;

bayessurvreg3:
- to compute the distribution of the *error* term for the *onset* time: `extens = ""`;
- to compute the distribution of the *error* term for the *event* time if there was doubly-censoring: `extens = "_2"`;
- to compute the distribution of the *random intercept* for the *onset* time: `extens = "_b"`;
- to compute the distribution of the *random intercept* term for the *event* time if there was doubly-censoring: `extens = "_b2"`;

`extens.adjust` this argument is applicable for the situation when the MCMC chains were created using the function `bayessurvreg3`, and when both the distribution of the error term and the random intercept was specified as the G-spline.

In that case the location of the error term and the random intercept are separately not identifiable. Only the location of the sum $\varepsilon + b$ can be estimated. For this reason, the function `bayesGspline` always centers the distribution of the random intercept to have a zero mean and adds its original mean to the mean of the distribution of the error term.

Argument `extens.adjust` is used to match correctly the files containing the G-spline of the random intercept corresponding to the particular error term.

The following values of `extens.adjust` should be used in the following situations:

- if there are no doubly-censored data or if we are computing the distribution of the error term/random intercept from the model for the *onset* time then

  `extens.adjust = "_b"`

- if there are doubly-censored data and we are computing the distribution of the error term/random intercept from the model for the *event* time then

extens.adjust = "_b2"

grid1 — grid of values from the first dimension at which the sampled densities are to be evaluated.

grid2 — grid of values from the second dimension (if the G-spline was bivariate) at which the sampled densities are to be evaluated. This item is missing if the G-spline is univariate.

skip — number of rows that should be skipped at the beginning of each *.sim file with the stored sample.

by — additional thinning of the sample.

last.iter — index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file mixmoment.sim.

nwrite — frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change).

only.aver — TRUE/FALSE, if TRUE only MCMC average is returned otherwise also values of the G-spline at each iteration are returned (which might ask for quite lots of memory).

standard — TRUE/FALSE, if TRUE, each G-spline is standardized to have zero mean and unit variance. Only applicable if version = 30 or 31, otherwise standard is always set to FALSE.

version — this argument indicates by which bayes*survreg* function the chains used by bayesGspline were created. Use the following:

bayesHistogram: version = 0;

bayesBisurvreg: version = 0;

bayessurvreg2: version = 0;

bayessurvreg3: version = 30 or 31.

Use version = 30 if you want to compute the density of the *error* term.
Use version = 31 if you want to compute the density of the *random intercept*.
Use version = 32 if you want to compute the density of the *error* term in the model with doubly-interval-censored data and bivariate normal distribution for random intercepts in the onset and time-to-event parts of the model OR if you have just interval-censored data and a simple AFT model without random effects and you want to compute the density of the *error* term of the model.

Value

An object of class bayesGspline is returned. This object is a list with components grid, average for the univariate G-spline and components grid1, grid2, average for the bivariate G-spline.

grid — this is a grid of values (vector) at which the McMC average of the G-spline was computed.

average — these are McMC averages of the G-spline (vector) evaluated in grid.

grid1 — this is a grid of values (vector) for the first dimension at which the McMC average of the G-spline was computed.

`grid2` this is a grid of values (vector) for the second dimension at which the McMC average of the G-spline was computed.

`average` this is a matrix `length(grid1)` times `length(grid2)` with McMC averages of the G-spline evaluated in

$$x1 = \quad (\text{grid1} \quad \ldots \quad \text{grid1})$$

and

$$x2 = \begin{matrix} (\quad \text{grid2} \quad) \\ (\quad \ldots \quad) \\ (\quad \text{grid2} \quad) \end{matrix}$$

There exists a method to plot objects of the class `bayesGspline`.

Attributes

Additionally, the object of class `bayesGspline` has the following attributes:

`sample.size` a length of the McMC sample used to compute the McMC average.

`sample` G-spline evaluated in a grid of values. This attribute is present only if `only.aver = FALSE`.

For a univariate G-spline this is a matrix with `sample.size` columns and length(grid1) rows.

For a bivariate G-spline this is a matrix with `sample.size` columns and length(grid1)*length(grid2) rows.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3–22.

Komárek, A. and Lesaffre, E. (2008). Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions. *Journal of the American Statistical Association*, **103**, 523–533.

Komárek, A., Lesaffre, E., and Legrand, C. (2007). Baseline and treatment effect heterogeneity for survival times between centers using a random effects accelerated failure time model with flexible error distribution. *Statistics in Medicine,* **26**, 5457–5472.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2006),
## Komarek and Lesaffre (2008),
## Komarek, Lesaffre, and Legrand (2007).
##
## R commands available
## in the documentation
## directory of this package
##  - ex-tandmobPA.R and
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobPA.pdf
##  - ex-tandmobCS.R and
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobCS.pdf
##  - ex-eortc.R and
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-eortc.pdf
##
```

`bayesHistogram` *Smoothing of a uni- or bivariate histogram using Bayesian G-splines*

Description

A function to estimate a density of a uni- or bivariate (possibly censored) sample. The density is specified as a mixture of Bayesian G-splines (normal densities with equidistant means and equal variances). This function performs an MCMC sampling from the posterior distribution of unknown quantities in the density specification. Other method functions are available to visualize resulting density estimate.

This function served as a basis for further developed `bayesBisurvreg`, `bayessurvreg2` and `bayessurvreg3` functions. However, in contrast to these functions, `bayesHistogram` does not allow for doubly censoring.

Bivariate case:

Let $Y_{i,l},\ i = 1, \ldots, N,\ l = 1, 2$ be observations for the ith cluster and the first and the second unit (dimension). The bivariate observations $Y_i = (Y_{i,1}, Y_{i,2})',\ i = 1, \ldots, N$ are assumed to be i.i.d. with a~bivariate density $g_y(y_1, y_2)$. This density is expressed as a~mixture of Bayesian G-splines (normal densities with equidistant means and constant variance matrices). We distinguish two, theoretically equivalent, specifications.

Specification 1

$$(Y_1, Y_2)' \sim \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} N_2(\mu_{(j_1,j_2)}, \text{diag}(\sigma_1^2, \sigma_2^2))$$

where σ_1^2, σ_2^2 are **unknown** basis variances and $\mu_{(j_1,j_2)} = (\mu_{1,j_1}, \mu_{2,j_2})'$ is an~equidistant grid of knots symmetric around the **unknown** point $(\gamma_1, \gamma_2)'$ and related to the unknown basis variances through the relationship

$$\mu_{1,j_1} = \gamma_1 + j_1 \delta_1 \sigma_1, \quad j_1 = -K_1, \ldots, K_1,$$

$$\mu_{2,j_2} = \gamma_2 + j_2\delta_2\sigma_2, \quad j_2 = -K_2, \ldots, K_2,$$

where δ_1, δ_2 are fixed constants, e.g. $\delta_1 = \delta_2 = 2/3$ (which has a~justification of being close to cubic B-splines).

Specification 2

$$(Y_1,\, Y_2)' \sim (\alpha_1,\, \alpha_2)' + \boldsymbol{S}\,(Y_1,\, Y_2)'$$

where $(\alpha_1,\, \alpha_2)'$ is an **unknown** intercept term and $\boldsymbol{S}$ is a diagonal matrix with τ_1 and τ_2 on a diagonal, i.e. τ_1, τ_2 are **unknown** scale parameters. $(V_1,\, V_2)'$ is then standardized observational vector which is distributed according to the bivariate normal mixture, i.e.

$$(V_1,\, V_2)' \sim \sum_{j_1=-K_1}^{K_1} \sum_{j_2=-K_2}^{K_2} w_{j_1,j_2} N_2(\mu_{(j_1,j_2)},\, \mathrm{diag}(\sigma_1^2, \sigma_2^2))$$

where $\mu_{(j_1,j_2)} = (\mu_{1,j_1},\, \mu_{2,j_2})'$ is an~equidistant grid of **fixed** knots (means), usually symmetric about the **fixed** point $(\gamma_1,\, \gamma_2)' = (0, 0)'$ and σ_1^2, σ_2^2 are **fixed** basis variances. Reasonable values for the numbers of grid points K_1 and K_2 are $K_1 = K_2 = 15$ with the distance between the two knots equal to $\delta = 0.3$ and for the basis variances $\sigma_1^2\sigma_2^2 = 0.2^2$.

Univariate case:

It is a~direct simplification of the bivariate case.

Usage

```
bayesHistogram(y1, y2,
   nsimul = list(niter = 10, nthin = 1, nburn = 0, nwrite = 10),
   prior, init = list(iter = 0),
   mcmc.par = list(type.update.a = "slice", k.overrelax.a = 1,
                   k.overrelax.sigma = 1, k.overrelax.scale = 1),
   store = list(a = FALSE, y = FALSE, r = FALSE),
   dir)
```

Arguments

`y1`	response for the first dimension in the form of a survival object created using `Surv`.
`y2`	response for the second dimension in the form of a survival object created using `Surv`. If the response is one-dimensional this item is missing.
`nsimul`	a list giving the number of iterations of the MCMC and other parameters of the simulation.

niter total number of sampled values after discarding thinned ones, burn-up included;

nthin thinning interval;

nburn number of sampled values in a burn-up period after discarding thinned values. This value should be smaller than `niter`. If not, `nburn` is set to `niter - 1`. It can be set to zero;

nwrite an interval at which information about the number of performed iterations is print on the screen and during the burn-up period an interval with which the sampled values are writen to files;

`prior` a list that identifies prior hyperparameters and prior choices. See the paper Komárek and Lesaffre (2008) and the PhD. thesis Komárek (2006) for more details.

Some prior parameters can be guessed by the function itself. If you want to do so, set such parameters to `NULL`. Set to `NULL` also the parameters that are not needed in your model.

specification a~number giving which specification of the model is used. It can be one of the following numbers:

1 with this specification positions of the middle knots $\gamma_1, \ldots, \gamma_q$, where q is dimension of the G-spline and basis standard deviations $\sigma_{0,1}, \ldots, \sigma_{0,q}$ are estimated. At the same time the G-spline intercepts $\alpha_1, \ldots, \alpha_q$ and the G-spline scale parameters $s_1, \ldots, s_q$ are assumed to be fixed (usually, intercepts to zero and scales to 1). The user can specified the fixed quantities in the `init` parameter of this function

2 with this specification, G-spline intercepts $\alpha_1, \ldots, \alpha_q$ and the G-spline scale parameters $s_1, \ldots, s_q$ are estimated at the same time positions of the middle knots $\gamma_1, \ldots, \gamma_q$ and basis standard deviations $\sigma_{0,1}, \ldots, \sigma_{0,q}$ are assumed to be fixed (usually, middle knots to zero ans basis standard deviations to some smaller number like 0.2) The user can specified the fixed quantities in the `init` parameter of this function

K specification of the number of knots in each dimension, i.e. `K` is a vector of length equal to the dimension of the data q and $K_j, j = 1, \ldots, q$ determines that the subscript k_j of the knots runs over $-K_j, \ldots, 0, \ldots, K_j$. A value $K_j = 0$ is valid as well. There are only some restriction on the minimal value of K_j with respect to the choice of the neighbor system and possibly the order of the conditional autoregression in the prior of transformed weights (see below).

izero subscript $k_1 \ldots k_q$ of the knot whose transformed weight $a_{k_1 \ldots k_q}$ will constantly be equal to zero. This is here for identifiability. To avoid numerical problems it is highly recommended to set `izero=rep(0, q)`. `izero[j]` should be taken from the set $-K_j, \ldots, K_j$.

neighbor.system identification of the neighboring system for the Markov random field prior of transformed mixture weights $a_{k_1\, k_2}$. This can be substring of one of the following strings:

`uniCAR` "univariate conditional autoregression": a~prior based on squared differences of given order m (see argument `order`) in each row and column.

For univariate smoothing:

$$p(a) \propto \exp\Big\{-\frac{\lambda}{2} \sum_{k=-K+m}^{K} \big(\Delta^m a_k\big)^2\Big\},$$

where Δ^m denotes the difference operator of order m, i.e. $\Delta^1 a_k = a_k - a_{k-1}$ and $\Delta^m a_k = \Delta^{m-1} a_k - \Delta^{m-1} a_{k-1}$, $m \geq 2$.

For bivariate smoothing:

$$p(a) \propto \exp\Big\{-\frac{\lambda_1}{2} \sum_{k_1=-K_1}^{K_1} \sum_{k_2=-K_2+m}^{K_2} \big(\Delta_1^m a_{k_1,k_2}\big)^2 - \frac{\lambda_2}{2} \sum_{k_2=-K_2}^{K_2} \sum_{k_1=-K_1+m}^{K_1} \big(\Delta_2^m a_{k_1,k_2}\big)^2\Big\},$$

where Δ_l^m denotes the difference operator of order m acting in the lth margin, e.g.

$$\Delta_1^2 = a_{k_1,k_2} - 2a_{k_1,k_2-1} + a_{k_1,k_2-2}.$$

The precision parameters λ_1 and λ_2 might be forced to be equal (see argument `equal.lambda`.)

`eight.neighbors` this prior is based on eight nearest neighbors (i.e. except on edges, each full conditional depends only on eight nearest neighbors) and local quadratic smoothing. It applies only in the case of bivariate smoothing. The prior is then defined as

$$p(a) \propto \exp\Big\{-\frac{\lambda}{2}\sum_{k_1=-K_1}^{K_1-1}\sum_{k_2=-K_2}^{K_2-1}\big(\Delta a_{k_1,k_2}\big)^2\Big\},$$

where

$$\Delta a_{k_1,k_2} = a_{k_1,k_2} - a_{k_1+1,k_2} - a_{k_1,k_2+1} + a_{k_1+1,k_2+1}.$$

`twelve.neighbors` !!! THIS FEATURE HAS NOT BEEN IMPLEMENTED YET. !!!

order order of the conditional autoregression if `neighbor.system` = `uniCAR`. Implemented are 1, 2, 3. If `order` = `0` and `neighbor.system` = `uniCAR` then mixture weights are assumed to be fixed and equal to their initial values specified by the `init` parameter (see below). Note that the numbers K_j, $j = 1, \ldots, q$ must be all equal to or higher than `order`.

equal.lambda `TRUE/FALSE` applicable in the case when a density of bivariate observations is estimated and `neighbor.system` = `uniCAR`. It specifies whether there is only one common Markov random field precision parameter λ for all margins (dimensions) or whether each margin (dimension) has its own precision parameter λ. For all other neighbor systems is `equal.lambda` automatically `TRUE`.

prior.lambda specification of the prior distributions for the Markov random field precision parameter(s) λ (when `equal.lambda` = `TRUE`) or $\lambda_1, \ldots, \lambda_q$ (when `equal.lambda` = `TRUE`). This is a vector of substring of one of the following strings (one substring for each margin if `equal.lambda` = `FALSE`, otherwise just one substring):

`fixed` the λ parameter is then assumed to be fixed and equal to its initial values given by `init` (see below).

`gamma` a particular λ parameter has a priori gamma distribution with shape g_j and rate (inverse scale) h_j where $j = 1$ if `equal.lambda=TRUE` and $j = 1, \ldots, q$ if `equal.lambda=TRUE`. Shape and rate parameters are specified by `shape.lambda`, `rate.lambda` (see below).

`sduniform` a particular $1/\sqrt{\lambda}$ parameter (i.e.a standard deviation of the Markov random field) has a priori a uniform distribution on the interval $(0, S_j)$ where $j = 1$ if `equal.lambda=TRUE` and $j = 1, \ldots, q$ if `equal.lambda=TRUE`. Upper limit of intervals is specified by `rate.lambda` (see below).

prior.gamma specification of the prior distribution for a reference knot (intercept) γ in each dimension. This is a vector of substrings of one of the following strings (one substring for each margin):

`fixed` the γ parameter is then assumed to be fixed and equal to its initial values given by `init` (see below).

`normal` the γ parameter has a priori a normal distribution with mean and variance given by `mean.gamma` and `var.gamma`.

prior.sigma specification of the prior distribution for basis standard deviations of the G-spline in each dimension. This is a vector of substrings of one of the following strings (one substring for each margin):

`fixed` the σ parameter is then assumed to be fixed and equal to its initial values given by `init` (see below).

`gamma` a particular σ^{-2} parameter has a priori gamma distribution with shape ζ_j and rate (inverse scale) η_j where $j = 1, \ldots, q$. Shape and rate parameters are specified by `shape.sigma`, `rate.sigma` (see below).

`sduniform` a particular σ parameter has a priori a uniform distribution on the interval $(0, S_j)$. Upper limit of intervals is specified by `rate.sigma` (see below).

prior.intercept specification of the prior distribution for the intercept terms $\alpha_1, \ldots, \alpha_q$ (2nd specification) in each dimension. This is a vector of substrings of one of the following strings (one substring for each margin):

`fixed` the intercept parameter is then assumed to be fixed and equal to its initial values given by `init` (see below).

`normal` the intercept parameter has a priori a normal distribution with mean and variance given by `mean.intercept` and `var.intercept`.

prior.scale specification of the prior distribution for the scale parameter (2nd specification) of the G-spline in each dimension This is a vector of substrings of one of the following strings (one substring for each margin):

`fixed` the `scale` parameter is then assumed to be fixed and equal to its initial values given by `init` (see below).

`gamma` a particular $scale^{-2}$ parameter has a priori gamma distribution with shape ζ_j and rate (inverse scale) η_j where $j = 1, \ldots, q$. Shape and rate parameters are specified by `shape.scale`, `rate.scale` (see below).

`sduniform` a particular $scale$ parameter has a priori a uniform distribution on the interval $(0, S_j)$. Upper limit of intervals is specified by `rate.scale` (see below).

c4delta values of $c_1, \ldots, c_q$ which serve to compute the distance δ_j between two consecutive knots in each dimension. The knot $\mu_{j\,k}$, $j = 1, \ldots, q$, $k = -K_j, \ldots, K_j$ is defined as $\mu_{j\,k} = \gamma_j + k\,\delta_j$ with $\delta_j = c_j\,\sigma_j$.

mean.gamma these are means for the normal prior distribution of middle knots $\gamma_1, \ldots, \gamma_q$ in each dimension if this prior is normal. For fixed γ an appropriate element of the vector `mean.gamma` may be whatever.

var.gamma these are variances for the normal prior distribution of middle knots $\gamma_1, \ldots, \gamma_q$ in each dimension if this prior is normal. For fixed γ an appropriate element of the vector `var.gamma` may be whatever.

shape.lambda these are shape parameters for the gamma prior (if used) of Markov random field precision parameters $\lambda_1, \ldots, \lambda_q$ (if `equal.lambda = FALSE`) or λ_1 (if `equal.lambda = TRUE`).

rate.lambda these are rate parameters for the gamma prior (if `prior.lambda = gamma`) of Markov random field precision parameters $\lambda_1, \ldots, \lambda_q$ (if `equal.lambda = FALSE`) or λ_1 (if `equal.lambda = TRUE`) or upper limits of the uniform prior (if `prior.lambda = sduniform`) of Markov random field standard deviation parameters $\lambda_1^{-1/2}, \ldots, \lambda_q^{-1/2}$ (if `equal.lambda = FALSE`) or $\lambda_1^{-1/2}$ (if `equal.lambda = TRUE`).

shape.sigma these are shape parameters for the gamma prior (if used) of basis inverse variances $\sigma_1^{-2}, \ldots, \sigma_q^{-2}$.

rate.sigma these are rate parameters for the gamma prior (if `prior.sigma = gamma`) of basis inverse variances $\sigma_1^{-2}, \ldots, \sigma_q^{-2}$ or upper limits of the uniform prior (if `prior.sigma = sduniform`) of basis standard deviations $\sigma_1, \ldots, \sigma_q$.

mean.intercept these are means for the normal prior distribution of the G-spline intercepts (2nd specification) $\alpha_1, \ldots, \alpha_q$ in each dimension if this prior is normal. For fixed α an appropriate element of the vector `mean.intercept` may be whatever.

var.intercept these are variances for the normal prior distribution of the G-spline intercepts $\alpha_1, \ldots, \alpha_q$ in each dimension if this prior is normal. For fixed α an appropriate element of the vector `var.alpha` may be whatever.

shape.scale these are shape parameters for the gamma prior (if used) of the G-spline scale parameter (2nd specification) $scale_1^{-2}, \ldots, scale_q^{-2}$.

rate.scale these are rate parameters for the gamma prior (if `prior.scale = gamma`) of the G-spline inverse variances $scale_1^{-2}, \ldots, scale_q^{-2}$ or upper limits of the uniform prior (if `prior.scale = sduniform`) of the G-spline scale $scale_1, \ldots, scale_q$.

`init` a list of the initial values to start the McMC. Set to `NULL` such parameters that you want the program should itself sample for you or parameters that are not needed in your model.

iter the number of the iteration to which the initial values correspond, usually zero.

a vector/matrix of initial transformed mixture weights $a_{k_1}, k_1 = -K_1, \ldots, K_1$ if univariate density is estimated; $a_{k_1\,k_2}, k_1 = -K_1, \ldots, K_1, k_2 = -K_2, \ldots, K_2$, if bivariate density is estimated. This initial value can be guessed by the function itself.

lambda initial values for Markov random field precision parameter(s) λ (if `equal.lambda = TRUE`), $\lambda_1, \ldots, \lambda_q$ (if `equal.lambda = FALSE`.)

gamma initial values for the middle knots in each dimension.
If `prior$specification = 2` it is recommended (for easier interpretation of the results) to set `init$gamma` to zero for all dimensions.
If `prior$specification = 1` `init$gamma` should be approximately equal to the mean value of the data in each margin.

sigma initial values for basis standard deviations in each dimension.

If `prior$specification = 2` this should be approximately equal to the range of standardized data (let say 4 + 4) divided by the number of knots in each margin and multiplied by something like 2/3.
If `prior$specification = 1` this should be approximately equal to the range of your data divided by the number of knots in each margin and multiplied again by something like 2/3.

intercept initial values for the intercept term in each dimension.
Note that if `prior$specification = 1` this initial value is always changed to zero for all dimensions.

scale initial values for the G-spline scale parameter in each dimension.
Note that if `prior$specification = 1` this initial value is always changed to one for all dimensions.

y initial values for (possibly unobserved censored) observations. This should be either a vector of length equal to the sample size if the response is univariate or a matrix with as many rows as is the sample size and two columns if the response is bivariate. Be aware that `init$y` must be consistent with data supplied. This initial can be guessed by the function itself. Possible missing values in `init$y` tells the function to guess the initial value.

r initial values for labels of components to which the (augmented) observations belong. This initial can be guessed by the function itself. This should be either a vector of length equal to the sample size if the response is univariate or a matrix with as many rows as is the sample size and two columns if the response is bivariate. Values in the first column of this matrix should be between `-prior$K[1]` and `prior$K[1]`, values in the second column of this matrix between `-prior$K[2]` and `prior$K[2]`, e.g. when `init$r[i,1:2] = c(-3, 6)` it means that the ith observation is initially assigned to the component with the mean $\boldsymbol{\mu} = (\mu_1, \mu_2)'$ where

$$\mu_1 = \mu_{1,\,-3} = \gamma_1 - 3\,c_1\sigma_1$$

and

$$\mu_2 = \mu_{2,\,6} = \gamma_2 + 6\,c_2\sigma_2.$$

`mcmc.par` a list specifying further details of the McMC simulation. There are default values implemented for all components of this list.

type.update.a it specifies the McMC method to update transformed mixture weights a. It is a~substring of one of the following strings:

slice slice sampler of Neal (2003) is used (default choice);

ars.quantile adaptive rejection sampling of Gilks and Wild (1992) is used with starting abscissae equal to 15%, 50% and 85% quantiles of a~piecewise exponential approximation to the full conditional from the previous iteration;

ars.mode adaptive rejection sampling of Gilks and Wild (1992) is used with starting abscissae equal to the mode and plus/minus twice approximate standard deviation of the full conditional distribution

k.overrelax.a this specifies a frequency of overrelaxed updates of transformed mixture weights a when slice sampler is used. Every kth value is sampled in a usual way (without overrelaxation). If you do not want overrelaxation

at all, set k.overrelax.a to 1 (default choice). Note that overrelaxation can be only done with the slice sampler (and not with adaptive rejection sampling).

k.overrelax.sigma a vector of length equal to the dimension of the G-spline specifying a frequency of overrelaxed updates of basis G-spline variances. If you do not want overrelaxation at all, set all components of k.overrelax.sigma to 1 (default choice).

k.overrelax.scale a vector of length equal to the dimension of the G-spline specifying a frequency of overrelaxed updates of the G-spline scale parameters (2nd specification). If you do not want overrelaxation at all, set all components of k.overrelax.scale to 1 (default choice).

store a~list of logical values specifying which chains that are not stored by default are to be stored. The list can have the following components.

a if TRUE then all the transformed mixture weights $a_{k_1,\,k_2}$, $k_1 = -K_1, \ldots, K_1$, $k_2 = -K_2, \ldots, K_2$, related to the G-spline are stored.

y if TRUE then augmented log-event times for all observations are stored.

r if TRUE then labels of mixture components for residuals are stored.

dir a string that specifies a directory where all sampled values are to be stored.

Value

A list of class bayesHistogram containing an information concerning the initial values and prior choices.

Files created

Additionally, the following files with sampled values are stored in a directory specified by dir argument of this function (some of them are created only on request, see store parameter of this function).

Headers are written to all files created by default and to files asked by the user via the argument store. All sampled values are written in files created by default and to files asked by the user via the argument store. In the files for which the corresponding store component is FALSE, every nsimul$nwrite value is written during the whole MCMC (this might be useful to restart the MCMC from some specific point).

The following files are created:

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

mixmoment.sim columns labeled k, Mean.1, Mean.2, D.1.1, D.2.1, D.2.2 in the bivariate case and columns labeled k, Mean.1, D.1.1 in the univariate case, where

k = number of mixture components that had probability numerically higher than zero;

Mean.1 = $\mathrm{E}(Y_{i,1})$;

Mean.2 = $\mathrm{E}(Y_{i,2})$;

D.1.1 = $\mathrm{var}(Y_{i,1})$;

D.2.1 = $\mathrm{cov}(Y_{i,1},\, Y_{i,2})$;

D.2.2 = $\mathrm{var}(Y_{i,2})$.

mweight.sim sampled mixture weights $w_{k_1,\,k_2}$ of mixture components that had probabilities numerically higher than zero.

mmean.sim indeces $k_1,\ k_2$, $k_1 \in \{-K_1, \ldots, K_1\}$, $k_2 \in \{-K_2, \ldots, K_2\}$ of mixture components that had probabilities numerically higher than zero. It corresponds to the weights in `mweight.sim`.

gspline.sim characteristics of the sampled G-spline (distribution of $(Y_{i,1},\ Y_{i,2})'$). This file together with `mixmoment.sim`, `mweight.sim` and `mmean.sim` can be used to reconstruct the G-spline in each MCMC iteration.

The file has columns labeled `gamma1`, `gamma2`, `sigma1`, `sigma2`, `delta1`, `delta2`, `intercept1`, `intercept2`, `scale1`, `scale2`. The meaning of the values in these columns is the following:

gamma1 = the middle knot γ_1 in the first dimension. If 'Specification' is 2, this column usually contains zeros;

gamma2 = the middle knot γ_2 in the second dimension. If 'Specification' is 2, this column usually contains zeros;

sigma1 = basis standard deviation σ_1 of the G-spline in the first dimension. This column contains a~fixed value if 'Specification' is 2;

sigma2 = basis standard deviation σ_2 of the G-spline in the second dimension. This column contains a~fixed value if 'Specification' is 2;

delta1 = distance $delta_1$ between the two knots of the G-spline in the first dimension. This column contains a~fixed value if 'Specification' is 2;

delta2 = distance δ_2 between the two knots of the G-spline in the second dimension. This column contains a~fixed value if 'Specification' is 2;

intercept1 = the intercept term α_1 of the G-spline in the first dimension. If 'Specification' is 1, this column usually contains zeros;

intercept2 = the intercept term α_2 of the G-spline in the second dimension. If 'Specification' is 1, this column usually contains zeros;

scale1 = the scale parameter τ_1 of the G-spline in the first dimension. If 'Specification' is 1, this column usually contains ones;

scale2 = the scale parameter τ_2 of the G-spline in the second dimension. 'Specification' is 1, this column usually contains ones.

mlogweight.sim fully created only if `store$a` = `TRUE`. The file contains the transformed weights $a_{k_1,\,k_2}$, $k_1 = -K_1, \ldots, K_1$, $k_2 = -K_2, \ldots, K_2$ of all mixture components, i.e. also of components that had numerically zero probabilities.

r.sim fully created only if `store$r` = `TRUE`. The file contains the labels of the mixture components into which the observations are intrinsically assigned. Instead of double indeces $(k_1,\ k_2)$, values from 1 to $(2\,K_1 + 1) \times (2\,K_2 + 1)$ are stored here. Function `vecr2matr` can be used to transform it back to double indeces.

lambda.sim either one column labeled `lambda` or two columns labeled `lambda1` and `lambda2`. These are the values of the smoothing parameter(s) λ (hyperparameters of the prior distribution of the transformed mixture weights $a_{k_1,\,k_2}$).

Y.sim fully created only if `store$y` = `TRUE`. It contains sampled (augmented) log-event times for all observations in the data set.

logposter.sim columns labeled `loglik`, `penalty` or `penalty1` and `penalty2`, `logprw`. The columns have the following meaning (the formulas apply for the bivariate case).

loglik $= -N\Big\{\log(2\pi)+\log(\sigma_1)+\log(\sigma_2)\Big\}-0.5\sum_{i=1}^{N}\Big\{(\sigma_1^2\,\tau_1^2)^{-1}\;(y_{i,1}-\alpha_1-\tau_1\mu_{1,\,r_{i,1}})^2+(\sigma_2^2\,\tau_2^2)^{-1}\;(y_{i,2}-\alpha_2-\tau_2\mu_{2,\,r_{i,2}})^2\Big\}$

where $y_{i,l}$ denotes (augmented) *(i,l)*th true log-event time. In other words, `loglik` is equal to the conditional log-density $\sum_{i=1}^{N}\;\log\Big\{p\big((y_{i,1},\;y_{i,2})\;\big|\;r_i,\;\text{G-spline}\big)\Big\}$;

penalty1: If `prior$neighbor.system = "uniCAR"`: the penalty term for the first dimension not multiplied by `lambda1`;

penalty2: If `prior$neighbor.system = "uniCAR"`: the penalty term for the second dimension not multiplied by `lambda2`;

penalty: If `prior$neighbor.system` is different from `"uniCAR"`: the penalty term not multiplied by `lambda`;

logprw $= -2\,N\;\log\big\{\sum_{k_1}\sum_{k_2}a_{k_1,\,k_2}\big\}+\sum_{k_1}\sum_{k_2}N_{k_1,\,k_2}\,a_{k_1,\,k_2}$, where $N_{k_1,\,k_2}$ is the number of observations assigned intrinsincally to the $(k_1,\;k_2)$th mixture component.

In other words, `logprw` is equal to the conditional log-density $\sum_{i=1}^{N}\log\big\{p(r_i\;|\;\text{G-spline weights})\big\}$.

Author(s)

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

References

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

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2008). Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions. *Journal of the American Statistical Association*, **103**, 523 - 533.

Komárek, A. and Lesaffre, E. (2006b). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3 - 22.

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

`bayessurvreg1` *A Bayesian survival regression with an error distribution expressed as a~normal mixture with unknown number of components*

Description

A function to sample from the posterior distribution for a survival regression model

$$\log(T_{i,l}) = \beta^T x_{i,l} + b_i^T z_{i,l} + \varepsilon_{i,l}, \quad i = 1,\ldots,N,\; l = 1,\ldots,n_i,$$

where distribution of $\varepsilon_{i,l}$ is specified as a normal mixture with unknown number of components as in Richardson and Green (1997) and random effect b_i is normally distributed.

See Komárek (2006) or Komárek and Lesaffre (2007) for more detailed description of prior assumptions.

Sampled values are stored on a disk to be further worked out by e.g. coda or boa.

Usage

```
bayessurvreg1(formula, random,
   data = parent.frame(), subset,
   na.action = na.fail,
   x = FALSE, y = FALSE, onlyX = FALSE,
   nsimul = list(niter = 10, nthin = 1, nburn = 0,
                 nnoadapt = 0, nwrite = 10),
   prior = list(kmax = 5, k.prior = "poisson", poisson.k = 3,
                dirichlet.w = 1,
                mean.mu = NULL, var.mu = NULL,
                shape.invsig2 = 1.5,
                shape.hyper.invsig2 = 0.8, rate.hyper.invsig2 = NULL,
                pi.split = NULL, pi.birth = NULL,
                Eb0.depend.mix = FALSE),
   prior.beta, prior.b, prop.revjump,
   init = list(iter = 0, mixture = NULL, beta = NULL,
               b = NULL, D = NULL,
               y = NULL, r = NULL, otherp = NULL, u = NULL),
   store = list(y = TRUE, r = TRUE, b = TRUE, u = TRUE,
                MHb = FALSE, regresres = FALSE),
   dir,
   toler.chol = 1e-10, toler.qr = 1e-10, ...)
```

Arguments

formula	model formula for the 'fixed' part of the model, i.e. the part that specifies $\beta^T x_{i,l}$. See survreg for further details. Intercept is implicitely included in the model by estimation of the error distribution. As a consequence -1 in the model formula does not have any effect on the model. The left-hand side of the formula must be an~objecy created using Surv. If random is used then the formula must contain an identification of clusters in the form cluster(id), where id is a name of the variable that determines clusters, e.g. Surv(time, event)~gender + cluster(id).
random	formula for the 'random' part of the model, i.e. the part that specifies $b_i^T z_{i,l}$. If omitted, no random part is included in the model. E.g. to specify the model with a random intercept, say random=~1. All effects mentioned in random should also be mentioned on the right-hand side of formula. When some random effects are included the random intercept is added by default. It can be removed using e.g. random=~-1 + gender.
data	optional data frame in which to interpret the variables occuring in the formulas.

`subset` subset of the observations to be used in the fit.

`na.action` function to be used to handle any `NA`s in the data. The user is discouraged to change a default value `na.fail`.

`x` if `TRUE` then the `X` matrix is returned. This matrix contain all columns appearing in both `formula` and `random` parameters.

`y` if `TRUE` then the `y` matrix (of log-survival times) is returned.

`onlyX` if TRUE, no McMC is performed. The function returns only a design matrix of your model (intercept excluded). It might be useful to set up correctly a parameter for a block update of β (regression parameters related to the fixed effects) and γ (means of the random effects, random intercept excluded) parameters in the model if Metropolis-Hastings is to be used instead of default Gibbs.

`nsimul` a list giving the number of iterations of the McMC and other parameters of the simulation.

niter total number of sampled values after discarding thinned ones, burn-up included.

nthin thinning interval.

nburn number of sampled values in a burn-up period after discarding thinned values. This value should be smaller than `niter`. If not, `nburn` is set to `niter - 1`. It can be set to zero.

nnoadapt applicable if some blocks of parameters are updated using an adaptive Metropolis algorithm. This is a number of sampled values that are generated using an initial and fixed proposal covariance matrix. It should be smaller or equal to `nburn`. If not, `nnoadapt` is set to `nburn`.

nwrite an interval at which sampled values are written to files.

`prior` a list that identifies prior hyperparameters and prior choices. See accompanying paper for more details. Some prior parameters can be guessed by the function itself. If you want to do so, set such parameters to `NULL`. Set to `NULL` also the parameters that are not needed in your model.

kmax value of k_{max}, upper limit for the number of mixture components. Its high values like 100 will usually correspond to ∞.

k.prior a string specifying the prior distribution of k, number of mixture components. Valid are either "poisson", "uniform", or "fixed". When "fixed" is given then the number of mixture components is not sampled.

poisson.k prior hyperparameter λ for the number of mixture components k if Poisson prior for this parameter is used.

dirichlet.w prior hyperparameter δ for the Dirichlet distribution of mixture weights $w_1, \ldots, w_k$.

mean.mu prior hyperparameter ξ for the mean of the normal prior for mixture means $\mu_1, \ldots, \mu_k$.

var.mu prior hyperparameter κ for the variance of the normal prior for mixture means $\mu_1, \ldots, \mu_k$.

shape.invsig2 prior hyperparameter ζ for the shape of the inverse-gamma distribution for the mixture variances $\sigma_1^2, \ldots, \sigma_k^2$.

shape.hyper.invsig2 prior hyperparameter (shape) g for the gamma distribution of the parameter η. Remember, η is a scale parameter of the inverse-gamma distribution for the mixture variances $\sigma_1^2, \ldots, \sigma_k^2$.

rate.hyper.invsig2 prior hyperparameter (rate) h for the gamma distribution of the parameter η. Remember, η is a scale parameter of the inverse-gamma distribution for the mixture variances $\sigma_1^2, \ldots, \sigma_k^2$.

pi.split probabilities of a split move within the reversible jump McMC. It must be a vector of length equal to `kmax` with the first component equal to 1 and the last component equal to 0. If `NULL` 2nd to (k-1)th components are set to 0.5.

pi.birth probabilities of a birth move within the reversible jump McMC. It must be a vector of length equal to `kmax` with the first component equal to 1 and the last component equal to 0. If `NULL` 2nd to (k-1)th components are set to 0.5.

Eb0.depend.mix this will normally be `FALSE`. Setting this option to `TRUE` served for some experiments during the development of this function. In principle, when this is set to `TRUE` and the random intercept is included in the model then it is assumed that the mean of the random intercept is not zero but $\sum_{j=1}^{k} w_j \mu_j$, i.e. the mean of the random intercept depends on mixture. However, this did not werk too well.

`prior.beta` a list defining the blocks of β parameters (both fixed effects and means of random effects, except the random intercept) that are to be updated together (in a block), a description of how they are updated and a specification of priors. The list is assumed to have the following components.

mean.prior a vector specifying a prior mean for each β parameter in the model.

var.prior a vector specifying a prior variance for each β parameter. It is recommended to run the function `bayessurvreg1` first with its argument `onlyX` set to `TRUE` to find out how the βs are sorted. They must correspond to a design matrix X.

blocks a list with the following components.

ind.block a list with vectors with indeces of columns of the design matrix X defining the effect of βs in the block. If not specified, all β parameters corresponding to fixed effects are updated in one block and remaining β parameters (means of random effects) in the second block using the Gibbs move.

cov.prop a list with vectors with a lower triangle of the covariance matrix which is used in the normal proposal (use a command `lower.tri` with `diag = TRUE` to get a lower triangle from a matrix) when one of the Metopolis-like algorithms is used for a given block. This matrix is used at each iteration if the given block is updated using a standard random-walk Metropolis-Hastings step. If the block is updated using an adaptive Metropolis step this matrix is used only at start. If not specified and Metropolis-like algorith is required a diagonal matrix with prior variances for corresponding β on a diagonal is used. It is set to a vector of zeros of appropriate length when the Gibbs move is required for a given block.

type.upd a character vector specifying the type of the update that will be used for each block. Valid are substrings of either "gibbs" or "adaptive.metropolis" or "random.walk.metropolis". Default is "gibbs" for all blocks.

mean.sampled a vector of means of up to now sampled values. This component is useful when the adaptive Metropolis algorithm is used and we do not start from the beginning (e.g. already several iterations of McMC have already been performed). Otherwise, this component does not have to be filled.

eps.AM a vector with ϵ from the adaptive Metropolis algorithm for each block.

sd.AM a vector specifying $s_d, d = 1, \ldots, D$ numbers from the adaptive Metropolis algorithm for each dimension. This vector must be of length equal at least to the length of the longest block. Defaults values are $\frac{1}{d}2.4^2$ where d denotes a length of the block.

weight.unif a vector specifying the weight of the uniform component in the proposal for each block. If not specified, it is equal to 0.5 for all parameters.

half.range.unif a vector of same length as the number of columns in the design matrix X specifying the half range of the uniform component of the proposal.

`prior.b` a list defining the way in which the random effects are to be updated and the specification of priors for random effects related parameters. The list is assumed to have following components.

prior.D a string defining the prior distribution for the covariance matrix of random effects D. It can be either "inv.wishart" or "sduniform".

inv.wishart in that case is assumed that the prior distribution of the matrix D is Inverse-Wishart with degrees of freedom equal to τ and a scale matrix equal to S. When D is a matrix $q \times q$ a prior expectation of D is equal to $(\tau - q - 1)^{-1}S$ if $\tau > q + 1$. For $q - 1 < \tau \leq q + 1$ a prior expectation is not finite. Degrees of freedom parameter τ does not have to be an integer. It has to only satisfy a condition $\tau > q - 1$. `prior.b$df.D` gives a prior degrees of freedom parameter τ and `prior.b$scale.D` determines the scale matrix D. This is also the default choice.

sduniform this can be used only when the random effect is univariate. Then the matrix D is just a scalar and the prior of $\sqrt{D}$ (standard deviation of the univariate random effect) is assumed to be uniform on interval $(0, S)$. The upper limit S is given by `prior.b$scale.D`.

df.D degrees of freedom parameter τ in the case that the prior of the matrix D is inverse-Wishart.

scale.D a lower triangle of the scale matrix S in the case that the prior of the matrix D is inverse-Wishart or the upper limit S of the uniform distribution in the case that $\sqrt{D} \sim \text{Unif}(0, S)$.

type.upd a character vector specifying the type of the update. Valid are substrings of either "random.walk.metropolis" or "gibbs". Default is "gibbs". In contrast to β parameters, all random effects are updated using the same type of the move. If "random.walk.metropolis" is used, random effects may be divided into blocks in which they are updated. With "gibbs", there is only one block defined for all random effects. which are updated in one step using its full conditional distribution.

blocks a list with the following components. This is set to NULL if `type.upd = "gibbs"`.

ind.block a list with vectors with indeces of random effects defining the block. Random intercept has always an index 1, remaining random effects have subsequent indeces according to their appearance in the design matrix X.

cov.prop a list with vectors with a lower triangle of the covariance matrix which is used in the normal proposal (use a command `lower.tri` with `diag = TRUE` to get a lower triangle from a matrix) for a given block when

`type.upd = "random.walk.metropolis"`.

weight.unif a vector specifying the weight of the uniform component in the proposal for each block when

`type.upd = "random.walk.metropolis"`.

If not specified, it is equal to 0.5 for all parameters. It is set to NULL if `type.upd = "gibbs"`.

half.range.unif a vector of same length as the number of random effects specifying the half range of the uniform component of the proposal when `type.upd = "random.walk.metropolis"`. It is set to NULL if `type.upd = "gibbs"`.

`prop.revjump` a list of values defining in which way the reversible jumps will be performed.

algorithm a string defining the algorithm used to generate canonical proposal vectors $u = (u_{3k+1}, \ldots, u_{3k_{max}})'$ where $u_{3k+1}, u_{3k+2}, u_{3k+3}$ are directly used when a jump to a space of higher dimension is proposed. These canonical proposal vectors are further transformed to give desired parameters (mixture component's weight, mean and variance). Valid values of `prop.revjump$algorithm` are substrings of "basic", "independent.av", "correlated.av". `"basic"` means that both components of vectors u and vectors u in time are generated independently from a standard uniform distribution. This corresponds to a basic reversible jumps McMC algorithm of Green (1995). Other two methods implement an auxiliary variable method of Brooks et al. (2003). The first one an independent auxiliary variable method where vectors u may be correlated in time however their components are independent and the second one the correlated auxiliary method where vectors u are correlated in time and also their components may be correlated. In both cases components of vectors u follow marginally a standard uniform distribution. A moody ring method of Brooks et al. (2003) is used to generate u vectors.

moody.ring parameters for the moody ring when `algorithm` is either "independent.av" or "correlated.av". This is a two component vector with both components taking values between 0 and 0.5 defining the strength of a correlation in time and between the components of u vectors. This vector is ignored when `algorithm = "basic"`. The first component of this vector determines dependence between u vectors in time (ε in Brooks et al. (2003)), the second component determines dependence between components of u

vectors (δ in Brooks et al. (2003)). The second compoenent is ignored when `algorithm = "independent.av"`. Note that both ε and δ do not have a meaning of correlation. They determine a range of additional uniform distributions. So that their values equal to 0 mean perfect correlation and their values equal to 0.5 mean independence. I.e. "correlated.av" with $\delta = 0.5$ is same as "independent.av" and "correlated.av" with $\delta = 0.5, \varepsilon = 0.5$ is same as "basic".

transform.split.combine a description of how the canonical variables u are to be transformed to give new values of mixture component's weight, mean and variance when a split move is proposed. Possible values are substrings of "richardson.green", "brooks" and "identity". In all cases, the $(0, 1)$ canonical variables u are transformed to $(0, 1)$ variates v that are than used to compute new values of mixture component's weight, mean and variance using a method of moments matching described in Richardson and Green (1997). When "identity", no further transformation is performed, when "richardson.green", u vectors are transformed such that the components of resulting v vectors follow independently beta distributions with parameters given further by `p = prop.revjump$transform.split.combine.parms` such that in the triplet of v's used in a particular split move, $v_1 \sim beta(p_1, p_2), v_2 \sim beta(p_3, p_4), v_3 \sim beta(p_5, p_6)$. When "brooks" v_2 is further transformed by $|2v_2 - 1|$. Default values of

```
prop.revjump$transform.split.combine$parms
```

is `c(2, 2, 2, 2, 1, 1)`.

transform.split.combine.parms see above.

transform.birth.death a description of how the canonical variables u are to be transformed to give new values of mixture component's weight, mean and variance when a birth move is proposed. At this moment only one value is possible: "richardson.green" implementing the proposal as in Richardson and Green (1997).

`init` a list of the initial values to start the McMC. Set to `NULL` such parameters that you want the program should itself sample for you or parameters that are not needed in your model.

iter index of the iteration to which initial values correspond, usually zero.

mixture initial mixture for the error random variable ε. It must a vector of length `1 + 3*kmax`, where `mixture[1]` gives initial number of mixture of components k, `mixture[2:(k+1)]` gives initial mixture weights, `mixture[(2+kmax):(2+kmax+k-1)]` gives initial mixture means, `mixture[(2+2*kmax):(2+2*kmax+k-1)]` gives initial mixture variances. Remaining components of this vector are ignored.

beta initial values of regression parameters in the same order as columns of the design matrix `X`. Call the function `bayessurvreg1` with `onlyX = TRUE` to see how the columns are sorted. Remember, `beta` in this function contains both fixed effects β and means of random effect γ in the notation of the accompanying paper except the mean of the random intercept which is always zero.

b initial values of random effects b_i for each cluster. This must a matrix of size

$q \times N$ or a vector of length $q * N$, where q is a number of random effects and N number of clusters, one column per cluster.

D initial value for the covariance matrix of random effects D. Only its lower triangle must be given in a vector, e.g. `c(d[1,1], d[2,1], d[3,1], d[2,2], d[3,2], d[3,3])` for a matrix 3×3.

y initial values of true log-event times. This must be a vector of length $\sum_{i=1}^{N} n_i$.

r initial values of component labels $r_{i,l}$. This must be a vector of length $\sum_{i=1}^{N} n_i$.

otherp initial values for other parameters. At this moment, only a value of the parameter η is given here.

u initial canonical proposal vector of length $3k_{max}$. When initial number of compoents given by `init$mixture[1]` is k, effectively only last $3k_{max} - 3*k$ components of the initial u vector are used. Further, when `prop.revjump$algorithm = "correlated.av"`, the first component of `init$u` (`init$u[1]`) contains an initial mood parameter (C_0 in Brooks et al. (2003)) for the moody ring.

`store` a list that defines which sampled values besides regression parameters β, means of random effects γ (both stored in a file called `beta.sim`), a covariance matrix of random effects D (stored in a file `D.sim`), the mixture (stored in file `mixmoment.sim`, `mweight.sim`, `mmean.sim`, `mvariance.sim`), values of other parameters - η (stored in a file `otherp.sim`), values of log-likelihoods (stored in a file `loglik.sim`), information concerning the performance of the reversible jump McMC and acceptance of regression parameters (stored in a file `MHinfo.sim`), iteration indeces (stored in a file `iteration.sim`) are to be stored. The list `store` has the following components.

y if `TRUE` sampled true log-event times are stored.

r if `TRUE` sampled component labels are stored.

b if `TRUE` sampled values of random effects b_i are stored.

u if `TRUE` sampled values of canonical proposal vectors for the reversible jump McMC are stored.

MHb if `TRUE` information concerning the performance of the Metropolis-Hastings algorithm for the update of random effects (if used instead of a dafault Gibbs) is stored.

regresres if `TRUE` sampled values of regression residuals at each iteration are stored. The regression residual is defined as $res_{i,l} = \log(t_{i,l}) - \beta^T x_{i,l} - b_i^T z_{i,l}$.

In the case that either `store$y`, or `store$r`, or `store$b`, or `store$u` are `FALSE`, only the last values of either `y`, or `r`, or `b`, or `u` at the time of writting of remaining quantities are stored in appropriate files (without headers) to be possibly used by `bayessurvreg1.files2init` function.

`dir` a string that specifies a directory where all sampled values are to be stored.

`toler.chol` tolerance for the Cholesky decomposition.

`toler.qr` tolerance for the QR decomposition.

`...` who knows?

Value

A list of class `bayessurvreg` containing an information concerning the initial values and prior choices.

Files created

Additionally, the following files with sampled values are stored in a directory specified by `dir` parameter of this function (some of them are created only on request, see `store` parameter of this function).

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

loglik.sim two columns labeled `loglik` and `randomloglik`.

$$\texttt{loglik} = \sum_{i=1}^{N}\sum_{l=1}^{n_i}\left[\left\{\log\Big(\frac{1}{\sqrt{2\pi\sigma^2_{r_{i,l}}}}\Big) - \frac{(y_{i,l} - \beta^T x_{i,l} - b_i^T z_{i,l} - \mu_{r_{i,l}})^2}{2\sigma^2_{r_{i,l}}}\right\}\right],$$

where $y_{i,l}$ denotes (sampled) *(i,l)*th true log-event time, b_i sampled value of the random effect vector for the *i*th cluster, β sampled value of the regression parameter β and $k, w_j, \mu_j, \sigma^2_j, j = 1, \ldots, k$ sampled mixture at each iteration.

$$\texttt{randomloglik} = \sum_{i=1}^{N}\log\Big(g(b_i)\Big),$$

where g denotes a density of (multivariate) normal distribution $N(\gamma, D)$, where γ is a sampled value of the mean of random effect vector and D is a sampled value of the covariance matrix of the random effects at each iteration.

mixmoment.sim three columns labeled `k`, `Intercept` and `Scale`. These are the number of mixture components, mean and standard deviation of the sampled error distribution (mixture) at each iteration.

mweight.sim each row contains mixture weights $w_1, \ldots, w_k$ at each iteration. From the header of this file, maximal number of mixture components specified in the prior can be derived.

mmean.sim each row contains mixture means $\mu_1, \ldots, \mu_k$ at each iteration. From the header of this file, maximal number of mixture components specified in the prior can be derived.

mvariance.sim each row contains mixture variances $\sigma^2_1, \ldots, \sigma^2_k$ at each iteration. From the header of this file, maximal number of mixture components specified in the prior can be derived.

beta.sim columns labeled according to name of the design matrix. These are sampled values of regression parameters β and means of random effects γ (except the mean of the random intercept which is zero).

b.sim columns labeled `nameb[1].id[1], ...,nameb[q].id[1], ..., nameb[1].id[N], ..., nameb[q].id[N]`, where `q` is a dimension of the random effect vector b_i and `N` number of clusters. `nameb` is replaced by appropriate column name from the design matrix and `id` is replaced by identificator of the clusters. This gives sampled values of the random effects for each cluster.

D.sim columns labeled `det, D.s.t, s = 1,..., q, t = s,...,q`, where `q` is dimension of the random effect vector b_i. Column `det` gives a determinant of the covariance matrix D of the random effects at each iteration, remaining columns give a lower triangle of this matrix at each iteration.

Y.sim columns labeled `Y[m]` where `m` goes from 1 to $\sum_{i=1}^{N} n_i$. This gives sampled log-event times for each observation in the dataset at each iteration.

r.sim columns labeled `r[m]` where `m` goes from 1 to $\sum_{i=1}^{N} n_i$. This gives sampled mixture labels for each observation in the dataset at each iteration.

otherp.sim Currently only one column labeled `eta` that gives sampled values of the hyperparameter η.

MHinfo.sim this gives the information concerning the performance of reversible jump algorithm and a sampler of regression parameters β and means of random effects γ. It has columns

`accept.spl.comb` relative frequency of accepted split-combine moves up to that iteration.

`split` relative frequency of proposed split moves up to that iteration.

`accept.birth.death` relative frequency of accepted birth-death moves up to that iteration.

`birth` relative frequency of proposed birth moves up to that iteration.

`beta.block.m` with `m` going from 1 to number of defined blocks of beta parameters. This gives a relative frequency of accepted proposals for each block up to that iteration. When Gibbs move is used, these should be columns of ones.

MHbinfo.sim this gives the information concerning the performance of a sampler for random effects (relative frequency of accepted values for each cluster and each block of random effects updated together). When Gibbs move is used only ones are seen in this file.

u.sim Sampled values of canonical proposal variables for reversible jump algorithm are stored here. This file is useful only when trying to restart the simulation from some specific point.

regresres.sim columns labeled `res[m]` where `m` goes from 1 to $\sum_{i=1}^{N} n_i$ This stores so called regression residuals for each observation at each iteration. This residual is defined as

$$res_{i,l} = y_{i,l} - \beta^T x_{i,l} - b_i z_{i,l}, \qquad i = 1 \ldots, N, \quad l = 1, \ldots, n_i,$$

where $y_{i,l}$ is a (sampled) log-event time at each iteration.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2007). Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution. *Statistica Sinica*, **17**, 549 - 569.

Brooks, S. P., Giudici, P., and Roberts, G. O. (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distribution (with Discussion). *Journal of the Royal Statistical Society B,* **65,** 3 - 55.

Green, P. J. (1995). Reversible jump MCMC computation and Bayesian model determination. *Biometrika,* **82,** 711 - 732.

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

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2007).
##
## R commands available
## in the documentation
## directory of this package as
## - ex-cgd.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-cgd.pdf
##
## - ex-tandmobMixture.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobMixture.pdf
##
```

---

bayessurvreg1.files2init

Read the initial values for the Bayesian survival regression model to the list.

Description

This function creates the list of initial values as required by the `init` argument of the function `bayessurvreg1`. The initials are taken from the files that are of the form of the files where the simulated values from the McMC run performed by the function `bayessurvreg1` are stored. The files are assumed to have the following names: "iteration.sim", "mixmoment.sim", "mweight.sim", "mmean.sim", "mvariance.sim", "beta.sim", "b.sim", "Y.sim", "r.sim", "D.sim", "otherp.sim", "u.sim". Some of these files may be missing. In that case, the corresponding initial is filled by `NULL`.

Usage

```
bayessurvreg1.files2init(dir = getwd(), row, kmax)
```

Arguments

`dir`	string giving the directory where it will be searched for the files with initial values.
`row`	the row (possible header does not count) from the files with the values that will be considered to give the initial values. By default, it is the last row from the files.
`kmax`	maximal number of mixture components. This must be given only if `header == FALSE`.

Value

A list with components called "iter", "mixture", "beta", "b", "D", "y", "r", "otherp", "u" in the form as required by the argument `init` of the function `bayessurvreg1`.

Author(s)

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

`bayessurvreg2` *Cluster-specific accelerated failure time model for multivariate, possibly doubly-interval-censored data. The error distribution is expressed as a penalized univariate normal mixture with high number of components (G-spline). The distribution of the vector of random effects is multivariate normal.*

Description

A function to estimate a regression model with possibly clustered (possibly right, left, interval or doubly-interval censored) data. In the case of doubly-interval censoring, different regression models can be specified for the onset and event times.

(Multivariate) random effects, normally distributed and acting as in the linear mixed model, normally distributed, can be included to adjust for clusters.

The error density of the regression model is specified as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variances). This function performs an MCMC sampling from the posterior distribution of unknown quantities.

For details, see Komárek (2006), and Komárek, Lesaffre and Legrand (2007).

We explain first in more detail a model without doubly censoring. Let $T_{i,l},\ i = 1, \ldots, N,\ l = 1, \ldots, n_i$ be event times for ith cluster and the units within that cluster The following regression model is assumed:

$$\log(T_{i,l}) = \beta' x_{i,l} + b_i' z_{i,l} + \varepsilon_{i,l}, \quad i = 1, \ldots, N,\ l = 1, \ldots, n_i$$

where β is unknown regression parameter vector, $x_{i,l}$ is a vector of covariates. b_i is a (multivariate) cluster-specific random effect vector and $z_{i,l}$ is a vector of covariates for random effects.

The random effect vectors $b_i,\ i = 1, \ldots, N$ are assumed to be i.i.d. with a (multivariate) normal distribution with the mean β_b and a covariance matrix D. Hierarchical centring (see Gelfand, Sahu, Carlin, 1995) is used. I.e. β_b expresses the average effect of the covariates included in $z_{i,l}$. Note that covariates included in $z_{i,l}$ may not be included in the covariate vector $x_{i,l}$. The covariance matrix D is assigned an inverse Wishart prior distribution in the next level of hierarchy.

The error terms $\varepsilon_{i,l},\ i = 1, \ldots, N, l = 1, \ldots, n_i$ are assumed to be i.i.d. with a univariate density $g_\varepsilon(e)$. This density is expressed as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variances). We distinguish two, theoretically equivalent, specifications.

Specification 1

$$\varepsilon \sim \sum_{j=-K}^{K} w_j N(\mu_j,\, \sigma^2)$$

where σ^2 is the **unknown** basis variance and $\mu_j,\ j = -K, \ldots, K$ is an equidistant grid of knots symmetric around the **unknown** point γ and related to the unknown basis variance through the relationship

$$\mu_j = \gamma + j\delta\sigma, \quad j = -K, \ldots, K,$$

where δ is fixed constants, e.g. $\delta = 2/3$ (which has a justification of being close to cubic B-splines).

Specification 2

$$\varepsilon \sim \alpha + \tau V$$

where α is an **unknown** intercept term and τ is an **unknown** scale parameter. V is then standardized error term which is distributed according to the univariate normal mixture, i.e.

$$V \sim \sum_{j=-K}^{K} w_j N(\mu_j, \sigma^2)$$

where $\mu_j,\ j = -K, \ldots, K$ is an equidistant grid of **fixed** knots (means), usually symmetric about the **fixed** point $\gamma = 0$ and σ^2 is **fixed** basis variance. Reasonable values for the numbers of grid points K is $K = 15$ with the distance between the two knots equal to $\delta = 0.3$ and for the basis variance $\sigma^2 = 0.2^2$.

Personally, I found Specification 2 performing better. In the paper Komárek, Lesaffre and Legrand (2007) only Specification 2 is described.

The mixture weights $w_j,\ j = -K, \ldots, K$ are not estimated directly. To avoid the constraints $0 < w_j < 1$ and $\sum_{j=-K}^{K} w_j = 1$ transformed weights $a_j,\ j = -K, \ldots, K$ related to the original weights by the logistic transformation:

$$a_j = \frac{\exp(w_j)}{\sum_m \exp(w_m)}$$

are estimated instead.

A Bayesian model is set up for all unknown parameters. For more details I refer to Komárek (2006) and to Komárek, Lesafre, and Legrand (2007).

If there are doubly-censored data the model of the same type as above can be specified for both the onset time and the time-to-event.

Usage

```
bayessurvreg2(formula, random, formula2, random2,
   data = parent.frame(),
   na.action = na.fail, onlyX = FALSE,
   nsimul = list(niter = 10, nthin = 1, nburn = 0, nwrite = 10),
   prior, prior.beta, prior.b, init = list(iter = 0),
   mcmc.par = list(type.update.a = "slice", k.overrelax.a = 1,
                   k.overrelax.sigma = 1, k.overrelax.scale = 1),
   prior2, prior.beta2, prior.b2, init2,
   mcmc.par2 = list(type.update.a = "slice", k.overrelax.a = 1,
                    k.overrelax.sigma = 1, k.overrelax.scale = 1),
   store = list(a = FALSE, a2 = FALSE, y = FALSE, y2 = FALSE,
                r = FALSE, r2 = FALSE, b = FALSE, b2 = FALSE),
   dir)
```

Arguments

`formula` model formula for the regression. In the case of doubly-censored data, this is the model formula for the onset time.

The left-hand side of the `formula` must be an object created using `Surv`.

In the formula all covariates appearing both in the vector $x_{i,l}$ and $z_{i,l}$ must be mentioned. Intercept is implicitely included in the model by the estimation of the error distribution. As a consequence `-1` in the model formula does not have any effect on the model specification.

If `random` is used then the formula must contain an identification of clusters in the form `cluster(id)`, where `id` is a name of the variable that determines clusters, e.g.

```
Surv(time, event) gender + cluster(id).
```

`random` formula for the 'random' part of the model, i.e. the part that specifies the covariates $z_{i,l}$. In the case of doubly-censored data, this is the `random` formula for the onset time.

If omitted, no random part is included in the model. E.g. to specify the model with a random intercept, say `random= 1`. All effects mentioned in `random` should also be mentioned on the right-hand side of `formula`.

When some random effects are included the random intercept is added by default. It can be removed using e.g. `random= -1 + gender`.

`formula2` model formula for the regression of the time-to-event in the case of doubly-censored data. Ignored otherwise. The same structure as for `formula` applies here.

`random2` specification of the 'random' part of the model for time-to-event in the case of doubly-censored data. Ignored otherwise. The same structure as for `random` applies here.

`data` optional data frame in which to interpret the variables occuring in the `formula`, `formula2`, `random`, `random2` statements.

`na.action` the user is discouraged from changing the default value `na.fail`.

`onlyX` if `TRUE` no MCMC sampling is performed and only the design matrix (matrices) are returned. This can be useful to set up correctly priors for regression parameters in the presence of `factor` covariates.

`nsimul` a list giving the number of iterations of the MCMC and other parameters of the simulation.

niter total number of sampled values after discarding thinned ones, burn-up included;

nthin thinning interval;

nburn number of sampled values in a burn-up period after discarding thinned values. This value should be smaller than `niter`. If not, `nburn` is set to `niter - 1`. It can be set to zero;

nwrite an interval at which information about the number of performed iterations is print on the screen and during the burn-up period an interval with which the sampled values are writen to files;

`prior` a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by `formula` and `random`. See `prior` argument of `bayesHistogram` function for more detail. In this list also 'Specification' as described above is specified.

The item `prior$neighbor.system` can only be equal to `uniCAR` here.

`prior.b` a list defining the way in which the random effects involved in `formula` and `random` are to be updated and the specification of priors for parameters related to these random effects. The list is assumed to have the following components.

prior.D a string defining the prior distribution for the covariance matrix of random effects D. It can be either "inv.wishart" or "sduniform".

inv.wishart in that case is assumed that the prior distribution of the matrix D is Inverse-Wishart with degrees of freedom equal to τ and a scale matrix equal to S. When D is a matrix $q \times q$ a prior expectation of D is equal to $(\tau - q - 1)^{-1}S$ if $\tau > q + 1$. For $q - 1 < \tau \leq q + 1$ a prior expectation is not finite. Degrees of freedom parameter τ does not have to be an integer. It has to only satisfy a condition $\tau > q-1$. `prior.b$df.D` gives a prior degrees of freedom parameter τ and `prior.b$scale.D` determines the scale matrix D. Inverse-Wishart is also the default choice.

sduniform this can be used only when the random effect is univariate (e.g. only random intercept in the model). Then the matrix D is just a scalar and the prior of $\sqrt{D}$ (standard deviation of the univariate random effect) is assumed to be uniform on interval $(0, S)$. The upper limit S is given by `prior.b$scale.D`.

df.D degrees of freedom parameter τ in the case that the prior of the matrix D is inverse-Wishart.

scale.D a lower triangle of the scale matrix S in the case that the prior of the matrix D is inverse-Wishart or the upper limit S of the uniform distribution in the case that $\sqrt{D} \sim \text{Unif}(0, S)$.

`prior.beta` prior specification for the regression parameters, in the case of doubly-censored data for the regression parameters of the onset time, i.e. it is related to `formula` and `random`. Note that the `beta` vector contains both the fixed effects β and the means of the random effects (except the random intercept) β_b.

This should be a list with the following components:

mean.prior a vector specifying a prior mean for each `beta` parameter in the model.

var.prior a vector specifying a prior variance for each `beta` parameter.

It is recommended to run the function bayessurvreg2 first with its argument `onlyX` set to `TRUE` to find out how the betas are sorted. They must correspond to a design matrix X taken from `formula`.

`init` an optional list with initial values for the MCMC related to the model given by `formula` and `random`. The list can have the following components:

iter the number of the iteration to which the initial values correspond, usually zero.

beta a vector of initial values for the regression parameters (both the fixed effects and means of the random effects). It must be sorted in the same way as are the columns in the design matrix. Use `onlyX=TRUE` if you do not know how the columns in the design matrix are created.

a a vector of length $2K + 1$ with the initial values of transformed mixture weights.

lambda initial values for the Markov random fields precision parameter.

gamma an initial values for the middle knot γ.

If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended (for easier interpretation of the results) to set `init$gamma` to zero (default behavior).

If 'Specification' is 1 `init$gamma` should be approximately equal to the mean value of the residuals.

sigma an initial values of the basis standard deviation σ.

If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended to set it approximately equal to the range of standardized data (let say 4 + 4) divided by the number of knots and multiplied by something like 2/3.

If 'Specification' is 1 this should be approximately equal to the range of the residuals divided by the number of knots $(2K + 1)$ and multiplied again by something like 2/3.

intercept an initial values of the intercept term α.

If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to zero.

scale an initial value of the scale parameter τ.

If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to one.

D initial value for the covariance matrix of random effects D. Only its lower triangle must be given in a vector, e.g. `c(d[1,1], d[2,1], d[3,1], d[2,2], d[3,2], d[3,3])` for a matrix 3×3.

b a vector or matrix of the initial values of random effects $b_i,\ i = 1, \ldots, N$ for each cluster. The matrix should be of size $q \times N$, where q is the number of random effects. I.e. each column of the matrix contains the initial values for one cluster.

y a vector of length $\sum_{i=1}^{N} n_i$ with initial values of log-event-times.

r a vector of length $\sum_{i=1}^{N} n_i$ with initial component labels for each residual. All values must be between $-K$ and K. See argument `init` of the function `bayesHistogram` for more details.

`mcmc.par` a list specifying how some of the G-spline parameters related to the distribution of the error term from `formula` are to be updated. See `bayesBisurvreg` for more details.

In contrast to `bayesBisurvreg` function argument `mcmc.par$type.update.a` can also be equal to `"block"` in which case all a coefficients are updated in 1 block using the Metropolis-Hastings algorithm.

`prior2` a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by `formula2` and `random2`. See `prior` argument of `bayesHistogram` function for more detail.

`prior.b2` prior specification for the parameters related to the random effects from `formula2` and `random2`. This should be a list with the same structure as `prior.b`.

`prior.beta2` prior specification for the regression parameters of time-to-event in the case of doubly censored data (related to `formula2` and `random2`). This should be a list with the same structure as `prior.beta`.

`init2` an optional list with initial values for the MCMC related to the model given by `formula2` and `random2`. The list has the same structure as `init`.

`mcmc.par2` a list specifying how some of the G-spline parameters related to `formula2` are to be updated. The list has the same structure as `mcmc.par`.

`store` a list of logical values specifying which chains that are not stored by default are to be stored. The list can have the following components.

a if `TRUE` then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline (error distribution) of `formula` are stored.

a2 if `TRUE` and there are doubly-censored data then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline (error distribution) of `formula2` are stored.

y if `TRUE` then augmented log-event times for all observations related to the `formula` are stored.

y2 if `TRUE` then augmented log-event times for all observations related to `formula2` are stored.

r if `TRUE` then labels of mixture components for residuals related to `formula` are stored.

r2 if `TRUE` then labels of mixture components for residuals related to `formula2` are stored.

b if `TRUE` then the sampled values of the random effects related to `formula` and `random` are stored.

b2 if `TRUE` then the sampled values of the random effects related to `formula2` and `random2` are stored.

`dir` a string that specifies a directory where all sampled values are to be stored.

Value

A list of class `bayessurvreg2` containing an information concerning the initial values and prior choices.

Files created

Additionally, the following files with sampled values are stored in a directory specified by `dir` argument of this function (some of them are created only on request, see `store` parameter of this function).

Headers are written to all files created by default and to files asked by the user via the argument `store`. During the burn-in, only every `nsimul$nwrite` value is written. After the burn-in, all sampled values are written in files created by default and to files asked by the user via the argument `store`. In the files for which the corresponding `store` component is `FALSE`, every `nsimul$nwrite` value is written during the whole MCMC (this might be useful to restart the MCMC from some specific point).

The following files are created:

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

mixmoment.sim columns labeled `k`, `Mean.1`, `D.1.1`, where

k = number of mixture components that had probability numerically higher than zero;

Mean.1 = $\mathrm{E}(\varepsilon_{i,l})$;

D.1.1 = $\mathrm{var}(\varepsilon_{i,l})$;

all related to the distribution of the error term from the model given by `formula`.

mixmoment_2.sim in the case of doubly-censored data, the same structure as `mixmoment.sim`, however related to the model given by `formula2`.

mweight.sim sampled mixture weights w_k of mixture components that had probabilities numerically higher than zero. Related to the model given by `formula`.

mweight_2.sim in the case of doubly-censored data, the same structure as `mweight.sim`, however related to the model given by `formula2`.

mmean.sim indeces k, $k \in \{-K, \ldots, K\}$ of mixture components that had probabilities numerically higher than zero. It corresponds to the weights in `mweight.sim`. Related to the model given by `formula`.

mmean_2.sim in the case of doubly-censored data, the same structure as `mmean.sim`, however related to the model given by `formula2`.

gspline.sim characteristics of the sampled G-spline (distribution of $\varepsilon_{i,l}$) related to the model given by `formula`. This file together with `mixmoment.sim`, `mweight.sim` and `mmean.sim` can be used to reconstruct the G-spline in each MCMC iteration.

The file has columns labeled `gamma1`, `sigma1`, `delta1`, `intercept1`, `scale1`, The meaning of the values in these columns is the following:

gamma1 = the middle knot γ If 'Specification' is 2, this column usually contains zeros;

sigma1 = basis standard deviation σ of the G-spline. This column contains a fixed value if 'Specification' is 2;

delta1 = distance $delta$ between the two knots of the G-spline. This column contains a fixed value if 'Specification' is 2;

intercept1 = the intercept term α of the G-spline. If 'Specification' is 1, this column usually contains zeros;

scale1 = the scale parameter τ of the G-spline. If 'Specification' is 1, this column usually contains ones;

gspline_2.sim in the case of doubly-censored data, the same structure as `gspline.sim`, however related to the model given by `formula2`.

mlogweight.sim fully created only if `store$a` = `TRUE`. The file contains the transformed weights a_k, $k = -K, \ldots, K$ of all mixture components, i.e. also of components that had numerically zero probabilities. This file is related to the error distribution of the model given by `formula`.

mlogweight_2.sim fully created only if `store$a2` = `TRUE` and in the case of doubly-censored data, the same structure as `mlogweight.sim`, however related to the error distribution of the model given by `formula2`.

r.sim fully created only if `store$r` = `TRUE`. The file contains the labels of the mixture components into which the residuals are intrinsically assigned. Instead of indeces on the scale $\{-K, \ldots, K\}$ values from 1 to $(2\,K + 1)$ are stored here. Function `vecr2matr` can be used to transform it back to indices from $-K$ to K.

r_2.sim fully created only if `store$r2 = TRUE` and in the case of doubly-censored data, the same structure as `r.sim`, however related to the model given by `formula2`.

lambda.sim one column labeled `lambda`. These are the values of the smoothing parameterλ (hyperparameters of the prior distribution of the transformed mixture weights a_k). This file is related to the model given by `formula`.

lambda_2.sim in the case of doubly-censored data, the same structure as `lambda.sim`, however related to the model given by `formula2`.

beta.sim sampled values of the regression parameters, both the fixed effects β and means of the random effects β_b (except the random intercept which has always the mean equal to zero). This file is related to the model given by `formula`. The columns are labeled according to the `colnames` of the design matrix.

beta_2.sim in the case of doubly-censored data, the same structure as `beta.sim`, however related to the model given by `formula2`.

D.sim sampled values of the covariance matrix D of the random effects. The file has $1+0.5\,q\,(q+1)$ columns (q is the dimension of the random effect vector b_i). The first column labeled `det` contains the determinant of the sampled matrix, additional columns labeled `D.1.1`, `D.2.1`, ..., `D.q.1`, ...`D.q.q` contain the lower triangle of the sampled matrix. This file is related to the model specified by `formula` and `random`.

D_2.sim in the case of doubly-censored data, the same structure as `D.sim`, however related to the model given by `formula2` and `random2`.

b.sim fully created only if `store$b = TRUE`. It contains sampled values of random effects for all clusters in the data set. The file has $q \times N$ columns sorted as $b_{1,1}, \ldots, b_{1,q}, \ldots, b_{N,1}, \ldots, b_{N,q}$. This file is related to the model given by `formula` and `random`.

b_2.sim fully created only if `store$b2 = TRUE` and in the case of doubly-censored data, the same structure as `b.sim`, however related to the model given by `formula2` and `random2`.

Y.sim fully created only if `store$y = TRUE`. It contains sampled (augmented) log-event times for all observations in the data set.

Y_2.sim fully created only if `store$y2 = TRUE` and in the case of doubly-censored data, the same structure as `Y.sim`, however related to the model given by `formula2`.

logposter.sim columns labeled `loglik`, `penalty`, and `logprw`. This file is related to the model given by `formula`. The columns have the following meaning.

loglik $= -(\sum_{i=1}^{N}\, n_i)\,\Big\{\log(\sqrt{2\pi}) + \log(\sigma)\Big\} - 0.5\,\sum_{i=1}^{N}\,\sum_{l=1}^{n_i}\Big\{(\sigma^2\,\tau^2)^{-1}\;(y_{i,l} - x'_{i,l}\beta - z'_{i,l}b_i - \alpha - \tau\mu_{r_{i,l}})^2\Big\}$

where $y_{i,l}$ denotes (augmented) *(i,l)*th true log-event time.

In other words, `loglik` is equal to the conditional log-density

$$\sum_{i=1}^{N}\sum_{l=1}^{n_i} \log\Big\{p\big(y_{i,l} \;\big|\; r_{i,l},\, \beta,\, b_i,\, \text{G-spline}\big)\Big\};$$

penalty: the penalty term

$$-\frac{1}{2}\sum_{k}\Big(\Delta\, a_k\Big)^2$$

(not multiplied by λ);

logprw $= -2\,(\sum_i n_i)\,\log\{\sum_k a_k\} + \sum_k N_k\,a_k$, where N_k is the number of residuals assigned intrinsincally to the kth mixture component.

In other words, `logprw` is equal to the conditional log-density $\sum_{i=1}^{N}\sum_{l=1}^{n_i}\log\{p(r_{i,l} \mid \text{G-spline weights})\}$.

logposter_2.sim in the case of doubly-censored data, the same structure as `logposter.sim`, however related to the model given by `formula2`.

Author(s)

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

References

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

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A., Lesaffre, E., and Legrand, C. (2007). Baseline and treatment effect heterogeneity for survival times between centers using a random effects accelerated failure time model with flexible error distribution. *Statistics in Medicine,* **26**, 5457-5472.

Examples

```
## See the description of R commands for
## the model with EORTC data,
## analysis described in Komarek, Lesaffre and Legrand (2007).
##
## R commands available in the documentation
## directory of this package
## as ex-eortc.R and
## https://www2.karlin.mff.cuni.cz/ komarek/software/bayesSurv/ex-eortc.pdf
##
```

`bayessurvreg3` *Cluster-specific accelerated failure time model for multivariate, possibly doubly-interval-censored data with flexibly specified random effects and/or error distribution.*

Description

A function to estimate a regression model with possibly clustered (possibly right, left, interval or doubly-interval censored) data. In the case of doubly-interval censoring, different regression models can be specified for the onset and event times.

A univariate random effect (random intercept) with the distribution expressed as a penalized normal mixture can be included in the model to adjust for clusters.

The error density of the regression model is specified as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variances). This function performs an MCMC sampling from the posterior distribution of unknown quantities.

For details, see Komárek (2006) and Komárek and Lesaffre (2008).

SUPPLEMENTED IN 06/2013: Interval-censored times might be subject to misclassification. In case of doubly-interval-censored data, the event time might be subject to misclassification. For details, see García-Zattera, Jara and Komárek (2016).

We explain first in more detail a model without doubly censoring. Let $T_{i,l},\ i = 1, \ldots, N,\ l = 1, \ldots, n_i$ be event times for ith cluster and the units within that cluster The following regression model is assumed:

$$\log(T_{i,l}) = \beta' x_{i,l} + b_i + \varepsilon_{i,l}, \quad i = 1, \ldots, N,\ l = 1, \ldots, n_i$$

where β is unknown regression parameter vector, $x_{i,l}$ is a vector of covariates. b_i is a cluster-specific random effect (random intercept).

The random effects $b_i,\ i = 1, \ldots, N$ are assumed to be i.i.d. with a univariate density $g_b(b)$. The error terms $\varepsilon_{i,l},\ i = 1, \ldots, N, l = 1, \ldots, n_i$ are assumed to be i.i.d. with a univariate density $g_\varepsilon(e)$.

Densities g_b and g_ε are both expressed as a mixture of Bayesian G-splines (normal densities with equidistant means and constant variances). We distinguish two, theoretically equivalent, specifications.

In the following, the density for ε is explicitely described. The density for b is obtained in an analogous manner.

Specification 1

$$\varepsilon \sim \sum_{j=-K}^{K} w_j N(\mu_j,\, \sigma^2)$$

where σ^2 is the **unknown** basis variance and $\mu_j,\ j = -K, \ldots, K$ is an equidistant grid of knots symmetric around the **unknown** point γ and related to the unknown basis variance through the relationship

$$\mu_j = \gamma + j\delta\sigma, \quad j = -K, \ldots, K,$$

where δ is fixed constants, e.g. $\delta = 2/3$ (which has a justification of being close to cubic B-splines).

Specification 2

$$\varepsilon \sim \alpha + \tau\, V$$

where α is an **unknown** intercept term and τ is an **unknown** scale parameter. V is then standardized error term which is distributed according to the univariate normal mixture, i.e.

$$V \sim \sum_{j=-K}^{K} w_j N(\mu_j,\, \sigma^2)$$

where $\mu_j,\ j = -K, \ldots, K$ is an equidistant grid of **fixed** knots (means), usually symmetric about the **fixed** point $\gamma = 0$ and σ^2 is **fixed** basis variance. Reasonable values for the numbers of grid points K is $K = 15$ with the distance between the two knots equal to $\delta = 0.3$ and for the basis variance $\sigma^2 = 0.2^2$.

Personally, I found Specification 2 performing better. In the paper Komárek and Lesaffre (2008) only Specification 2 is described.

The mixture weights $w_j,\ j = -K, \ldots, K$ are not estimated directly. To avoid the constraints $0 < w_j < 1$ and $\sum_{j=-K}^{K} w_j = 1$ transformed weights $a_j,\ j = -K, \ldots, K$ related to the original weights by the logistic transformation:

$$a_j = \frac{\exp(w_j)}{\sum_m \exp(w_m)}$$

are estimated instead.

A Bayesian model is set up for all unknown parameters. For more details I refer to Komárek and Lesaffre (2008).

If there are doubly-censored data the model of the same type as above can be specified for both the onset time and the time-to-event.

In the case one wishes to link the random intercept of the onset model and the random intercept of the time-to-event model, there are the following possibilities.

Bivariate normal distribution
It is assumed that the pair of random intercepts from the onset and time-to-event part of the model are normally distributed with zero mean and an unknown covariance matrix D.

A priori, the inverse covariance matrix D^{-1} is addumed to follow a Wishart distribution.

Unknown correlation between the basis G-splines
Each pair of basis G-splines describing the distribution of the random intercept in the onset part and the time-to-event part of the model is assumed to be correlated with an unknown correlation coefficient ϱ. Note that this is just an experiment and is no more further supported.

Prior distribution on ϱ is assumed to be uniform. In the MCMC, the Fisher Z transform of the ϱ given by

$$Z = -\frac{1}{2}\log\Big(\frac{1-\varrho}{1+\varrho}\Big) = \text{atanh}(\varrho)$$

is sampled. Its prior is derived from the uniform prior Unif$(-1,\ 1)$ put on ϱ.

The Fisher Z transform is updated using the Metropolis-Hastings alhorithm. The proposal distribution is given either by a normal approximation obtained using the Taylor expansion of the full conditional distribution or by a Langevin proposal (see Robert and Casella, 2004, p. 318).

Usage

```
bayessurvreg3(formula, random, formula2, random2,
   data = parent.frame(),
   classification,
   classParam = list(Model = c("Examiner", "Factor:Examiner"),
                     a.sens = 1, b.sens = 1, a.spec = 1, b.spec = 1,
                     init.sens = NULL, init.spec = NULL),
   na.action = na.fail, onlyX = FALSE,
   nsimul = list(niter = 10, nthin = 1, nburn = 0, nwrite = 10),
   prior, prior.beta, prior.b, init = list(iter = 0),
   mcmc.par = list(type.update.a = "slice", k.overrelax.a = 1,
                   k.overrelax.sigma = 1, k.overrelax.scale = 1,
```

```
                    type.update.a.b = "slice", k.overrelax.a.b = 1,
                    k.overrelax.sigma.b = 1, k.overrelax.scale.b = 1),
    prior2, prior.beta2, prior.b2, init2,
    mcmc.par2 = list(type.update.a = "slice", k.overrelax.a = 1,
                     k.overrelax.sigma = 1, k.overrelax.scale = 1,
                     type.update.a.b = "slice", k.overrelax.a.b = 1,
                     k.overrelax.sigma.b = 1, k.overrelax.scale.b = 1),
    priorinit.Nb,
    rho = list(type.update = "fixed.zero", init=0, sigmaL=0.1),
    store = list(a = FALSE, a2 = FALSE, y = FALSE, y2 = FALSE,
                 r = FALSE, r2 = FALSE, b = FALSE, b2 = FALSE,
                 a.b = FALSE, a.b2 = FALSE, r.b = FALSE, r.b2 = FALSE),
    dir)

bayessurvreg3Para(formula, random, formula2, random2,
    data = parent.frame(),
    classification,
    classParam = list(Model = c("Examiner", "Factor:Examiner"),
                      a.sens = 1, b.sens = 1, a.spec = 1, b.spec = 1,
                      init.sens = NULL, init.spec = NULL),
    na.action = na.fail, onlyX = FALSE,
    nsimul = list(niter = 10, nthin = 1, nburn = 0, nwrite = 10),
    prior, prior.beta, prior.b, init = list(iter = 0),
    mcmc.par = list(type.update.a = "slice", k.overrelax.a = 1,
                    k.overrelax.sigma = 1, k.overrelax.scale = 1,
                    type.update.a.b = "slice", k.overrelax.a.b = 1,
                    k.overrelax.sigma.b = 1, k.overrelax.scale.b = 1),
    prior2, prior.beta2, prior.b2, init2,
    mcmc.par2 = list(type.update.a = "slice", k.overrelax.a = 1,
                     k.overrelax.sigma = 1, k.overrelax.scale = 1,
                     type.update.a.b = "slice", k.overrelax.a.b = 1,
                     k.overrelax.sigma.b = 1, k.overrelax.scale.b = 1),
    priorinit.Nb,
    rho = list(type.update = "fixed.zero", init=0, sigmaL=0.1),
    store = list(a = FALSE, a2 = FALSE, y = FALSE, y2 = FALSE,
                 r = FALSE, r2 = FALSE, b = FALSE, b2 = FALSE,
                 a.b = FALSE, a.b2 = FALSE, r.b = FALSE, r.b2 = FALSE),
    dir)
```

Arguments

formula	model formula for the regression. In the case of doubly-censored data, this is the model formula for the onset time. The left-hand side of the formula must be an object created using Surv. Intercept is implicitely included in the model by the estimation of the error distribution. As a consequence -1 in the model formula does not have any effect on the model specification. If random is used then the formula must contain an identification of clusters

in the form `cluster(id)`, where `id` is a name of the variable that determines clusters, e.g.

`Surv(time, event) gender + cluster(id)`.

`random` formula for the 'random' part of the model. In the case of doubly-censored data, this is the `random` formula for the onset time. With this version of the function only

`random = 1`

is allowed. If omitted, no random part is included in the model.

`formula2` model formula for the regression of the time-to-event in the case of doubly-censored data. Ignored otherwise. The same structure as for `formula` applies here.

`random2` specification of the 'random' part of the model for time-to-event in the case of doubly-censored data. Ignored otherwise. The same structure as for `random` applies here.

`data` optional data frame in which to interpret the variables occuring in the `formula`, `formula2`, `random`, `random2` statements.

`classification` `data.frame` with the information for a model which considers misclassification of the event times. It is assumed to have the following columns where the position of columns is important, not their names:

1. **idUnit**: variable which determines the rows of `classification` matrix pertaining to one unit in `formula/formula2 data`. Number of unique `idUnit` values must be the same as in `formula/formula2 data`, `classification` matrix must be sorted in the same order as `formula/formula2 data` and having all rows pertaining to one unit in its consecutive rows.
2. **Time**: variable with the examination times. It is assumed that the `Times` are sorted in an increasing order for each `idUnit`.
3. **Examiner**: variable which determines the examiner who performed evaluation at a specific visit. Number of unique `Examiner` values determines the number of examiners.
4. **Status**: 0/1 variable giving the event status according to examiner, 0 = no event, 1 = event.
5. **Factor**: possible factor (e.g., tooth in our dental application which may influence the misclassification). Numeric or character variables are converted to a `factor`. This column is obligatory only if `classModel` is "Factor:Examiner".

Possible additional columns are ignored.

If missing, no misclassification is considered.

`classParam` a `list` with additional parameters for the misclassification model. It is ignored if there is no `classification` argument specified.

The following components of the `list` `classParam` are expected.

Model a character string which specifies the model considered. It can be 1. "Examiner": sensitivity and specificity depend only on `Examiner`, 2. "Factor:Examiner": sensitivity and specificity is for each examiner generally different for different levels of a factor `Factor`.

a.sens parameter 'a' (`shape1`) of the beta prior distributions for sensitivities.

b.sens parameter 'b' (`shape2`) of the beta prior distributions for sensitivities.

a.spec parameter 'a' (`shape1`) of the beta prior distributions for specificities.

b.spec parameter 'b' (`shape2`) of the beta prior distributions for specificities.

init.sens a vector or matrix with initial values of sensitivities. A vector is expected if `Model` is "Examiner" in which case each component of the vector corresponds to each examiner. A matrix is expected if `Model` is "Factor:Examiner" in which case rows of the matrix correspond to the values of `Factor` and columns to examiners.
If not given then the initial sensitivities are sampled from a uniform distribution on (0.8, 0.9).

init.spec a vector or matrix with initial values of specificities. The structure is the same as for `init.sens`.

`na.action` the user is discouraged from changing the default value `na.fail`.

`onlyX` if `TRUE` no MCMC sampling is performed and only the design matrix (matrices) are returned. This can be useful to set up correctly priors for regression parameters in the presence of `factor` covariates.

`nsimul` a list giving the number of iterations of the MCMC and other parameters of the simulation.

niter total number of sampled values after discarding thinned ones, burn-up included;

nthin thinning interval;

nburn number of sampled values in a burn-up period after discarding thinned values. This value should be smaller than `niter`. If not, `nburn` is set to `niter - 1`. It can be set to zero;

nwrite an interval at which information about the number of performed iterations is print on the screen and during the burn-up period an interval with which the sampled values are writen to files;

`prior` a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by `formula` and `random`. See `prior` argument of `bayesHistogram` function for more detail. In this list also 'Specification' as described above is specified.

The item `prior$neighbor.system` can only be equal to `uniCAR` here.

`prior.b` a list specifying the prior distribution of the G-spline defining the distribution of the random intercept in the regression model given by `formula` and `random`. See `prior` argument of `bayesHistogram` function for more detail. In this list also 'Specification' as described above is specified.

It is ignored if the argument `priorinit.Nb` is given.

The item `prior.b$neighbor.system` can only be equal to `uniCAR` here.

`prior.beta` prior specification for the regression parameters, in the case of doubly-censored data for the regression parameters of the onset time, i.e. it is related to `formula` and `random`.

This should be a list with the following components:

mean.prior a vector specifying a prior mean for each `beta` parameter in the model.

var.prior a vector specifying a prior variance for each `beta` parameter.

It is recommended to run the function bayessurvreg3 first with its argument `onlyX` set to `TRUE` to find out how the betas are sorted. They must correspond to a design matrix X taken from `formula`.

`init` an optional list with initial values for the MCMC related to the model given by `formula` and `random`. The list can have the following components:

iter the number of the iteration to which the initial values correspond, usually zero.

beta a vector of initial values for the regression parameters. It must be sorted in the same way as are the columns in the design matrix. Use `onlyX=TRUE` if you do not know how the columns in the design matrix are created.

a a vector of length $2K + 1$ with the initial values of transformed mixture weights for the G-spline defining the distribution of the error term ε.

lambda initial values for the Markov random fields precision parameter for the G-spline defining the distribution of the error term ε.

gamma an initial values for the middle knot γ for the G-spline defining the distribution of the error term ε.
If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended (for easier interpretation of the results) to set `init$gamma` to zero (default behavior).
If 'Specification' is 1 `init$gamma` should be approximately equal to the mean value of the residuals.

sigma an initial values of the basis standard deviation σ for the G-spline defining the distribution of the error term ε.
If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended to set it approximately equal to the range of standardized data (let say 4 + 4) divided by the number of knots and multiplied by something like 2/3.
If 'Specification' is 1 this should be approximately equal to the range of the residuals divided by the number of knots $(2K + 1)$ and multiplied again by something like 2/3.

intercept an initial values of the intercept term α for the G-spline defining the distribution of the error term ε.
If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to zero.

scale an initial value of the scale parameter τ for the G-spline defining the distribution of the error term ε.
If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to one.

a.b a vector of length $2K + 1$ with the initial values of transformed mixture weights for the G-spline defining the distribution of the random intercept b.

lambda.b initial values for the Markov random fields precision parameter for the G-spline defining the distribution of the random intercept b.

gamma.b an initial values for the middle knot γ for the G-spline defining the distribution of the random intercept b.
Due to identifiability reasons, this value is always changed to zero and is for neither 'Specification' updated by the MCMC.

sigma.b an initial values of the basis standard deviation σ for the G-spline defining the distribution of the random intercept b.
If 'Specification' is 2, this value will not be changed by the MCMC and it is recommended to set it approximately equal to the range of standardized data (let say 4 + 4) divided by the number of knots and multiplied by something like 2/3.
If 'Specification' is 1 this should be approximately equal to the range of the residuals divided by the number of knots $(2K + 1)$ and multiplied again by something like 2/3.

intercept.b an initial values of the intercept term α for the G-spline defining the distribution of the random intercept b.
Due to identifiability reasons, this value is always changed to zero and is for neither 'Specification' updated by the MCMC.

scale.b an initial value of the scale parameter τ for the G-spline defining the distribution of the random intercept b.
If 'Specification' is 1 this value is not changed by the MCMC and the initial value is always changed to one.

b a vector of length N of the initial values of random effects $b_i,\ i = 1, \ldots, N$ for each cluster.

y a vector of length $\sum_{i=1}^{N} n_i$ with initial values of log-event-times.

r a vector of length $\sum_{i=1}^{N} n_i$ with initial component labels for each residual. All values must be between $-K$ and K. See argument `init` of the function `bayesHistogram` for more details.

r.b a vector of length N with initial component labels for each random intercept. All values must be between $-K$ and K. See argument `init` of the function `bayesHistogram` for more details.

`mcmc.par` a list specifying how some of the G-spline parameters related to the distribution of the error term and of the random intercept from `formula` and `random` are to be updated. See `bayesBisurvreg` for more details.

Compared to the mcmc.par argument of the function `bayesBisurvreg` additional components related to the G-spline for the random intercept can be present, namely

```
type.update.a.b
k.overrelax.a.b
k.overrelax.sigma.b
k.overrelax.scale.b
```

In contrast to bayesBisurvreg function arguments mcmc.par$type.update.a and mcmc.par$type.update.a.b can also be equal to "block" in which case all a coefficients are updated in 1 block using the Metropolis-Hastings algorithm.

prior2 a list specifying the prior distribution of the G-spline defining the distribution of the error term in the regression model given by formula2 and random2. See prior argument of bayesHistogram function for more detail.

prior.b2 prior specification for the parameters related to the random effects from formula2 and random2. This should be a list with the same structure as prior.b.

It is ignored if the argument priorinit.Nb is given.

prior.beta2 prior specification for the regression parameters of time-to-event in the case of doubly censored data (related to formula2 and random2). This should be a list with the same structure as prior.beta.

init2 an optional list with initial values for the MCMC related to the model given by formula2 and random2. The list has the same structure as init.

mcmc.par2 a list specifying how some of the G-spline parameters related to formula2 and random2 are to be updated. The list has the same structure as mcmc.par.

priorinit.Nb a list specifying the prior of the random intercepts in the case of the AFT model with doubly-interval-censored data and onset, time-to-event random intercepts following bivariate normal distribution.

The list should have the following components.

init.D initial value for the covariance matrix of the onset random intercept and time-to-event random intercept.
It can be specified either as a vector of length 3 giving the lower triangle of the matrix or as a matrix 2 x 2.

df.Di degrees of freedom ν for the Wishart prior of the matrix D^{-1}.
Note that it must be higher than 1.

scale.Di scale matrix S for the Wishart prior of the matrix D^{-1}.
It can be specified either as a vector of length 3 giving the lower triangle of the matrix or as a matrix 2 x 2.
Note that a priori

$$\mathrm{E}(D^{-1}.) = \nu S$$

rho a list specifying possible correlation between the onset random intercept and the time-to-event random intercept in the experimental version of the model. If not given correlation is fixed to 0.

It is ignored if the argument priorinit.Nb is given. Ordinary users should not care about this argument.

The list can have the following components.

type.update character specifying how the Fisher Z transform of the correlation coefficient is updated. Possible values are:
"fixed.zero": correlation coefficient is fixed to 0 and it is not updated.
"normal.around.mode": at each iteration of MCMC, 1 Newton-Raphson step from the current point Z of the full conditional distribution is performed, normal approximation is formed by Taylor expansion and new point Z is sampled from that normal approximation.

Note that this proposal does not work too well if the current point Z lies in the area of low posterior mass. The reason is that even 1 Newton-Raphson step usually leads to the area of high posterior probability mass and the proposal is "too ambisious".
`"langevin"`. at each iteration of MCMC, new point Z is sampled using the Langevin algorithm. A scale parameter (see below) must cerefully be chosen for this algorithm to ensure that the acceptance rate is about 50–60% (Robert, Casella, 2004, p. 319).

`store` a list of logical values specifying which chains that are not stored by default are to be stored. The list can have the following components.

a if `TRUE` then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline defining the error distribution of `formula` are stored.

a.b if `TRUE` then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline defining the distribution of the random intercept from `formula` and `random` are stored.

a2 if `TRUE` and there are doubly-censored data then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline defining the error distribution of `formula2` are stored.

a.b2 if `TRUE` then all the transformed mixture weights a_k, $k = -K, \ldots, K$, related to the G-spline defining the distribution of the random intercept from `formula2` and `random2` are stored.

y if `TRUE` then augmented log-event times for all observations related to the `formula` are stored.

y2 if `TRUE` then augmented log-event times for all observations related to `formula2` are stored.

r if `TRUE` then labels of mixture components for residuals related to `formula` are stored.

r.b if `TRUE` then labels of mixture components for random intercepts related to `formula` and `random` are stored.

r2 if `TRUE` then labels of mixture components for residuals related to `formula2` are stored.

r.b2 if `TRUE` then labels of mixture components for random intercepts related to `formula2` and `random2` are stored.

b if `TRUE` then the sampled values of the random interceptss related to `formula` and `random` are stored.

b2 if `TRUE` then the sampled values of the random interceptss related to `formula2` and `random2` are stored.

`dir` a string that specifies a directory where all sampled values are to be stored.

Value

A list of class `bayessurvreg3` containing an information concerning the initial values and prior choices.

Files created

Additionally, the following files with sampled values are stored in a directory specified by `dir` argument of this function (some of them are created only on request, see `store` parameter of this function).

Headers are written to all files created by default and to files asked by the user via the argument `store`. During the burn-in, only every `nsimul$nwrite` value is written. After the burn-in, all sampled values are written in files created by default and to files asked by the user via the argument `store`. In the files for which the corresponding `store` component is `FALSE`, every `nsimul$nwrite` value is written during the whole MCMC (this might be useful to restart the MCMC from some specific point).

The following files are created:

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

mixmoment.sim this file is related to the density of the error term from the model given by `formula`.

Columns labeled `k`, `Mean.1`, `D.1.1`, where

k = number of mixture components that had probability numerically higher than zero;

Mean.1 = $\mathrm{E}(\varepsilon_{i,l})$;

D.1.1 = $\mathrm{var}(\varepsilon_{i,l})$.

mixmoment_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

The same structure as `mixmoment.sim`.

mixmoment_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

The same structure as `mixmoment.sim`.

mixmoment_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

The same structure as `mixmoment.sim`.

mweight.sim this file is related to the density of the error term from the model given by `formula`.

Sampled mixture weights w_k of mixture components that had probabilities numerically higher than zero.

mweight_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

The same structure as `mweight.sim`.

mweight_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

The same structure as `mweight.sim`.

mweight_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

The same structure as `mweight.sim`.

mmean.sim this file is related to the density of the error term from the model given by `formula`.

Indeces k, $k \in \{-K, \ldots, K\}$ of mixture components that had probabilities numerically higher than zero. It corresponds to the weights in `mweight.sim`.

mmean_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

The same structure as `mmean.sim`.

mmean_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

The same structure as `mmean.sim`.

mmean_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

The same structure as `mmean.sim`.

gspline.sim this file is related to the density of the error term from the model given by `formula`.

Characteristics of the sampled G-spline. This file together with `mixmoment.sim`, `mweight.sim` and `mmean.sim` can be used to reconstruct the G-spline in each MCMC iteration.

The file has columns labeled `gamma1`, `sigma1`, `delta1`, `intercept1`, `scale1`, The meaning of the values in these columns is the following:

gamma1 = the middle knot γ If 'Specification' is 2, this column usually contains zeros;

sigma1 = basis standard deviation σ of the G-spline. This column contains a fixed value if 'Specification' is 2;

delta1 = distance $delta$ between the two knots of the G-spline. This column contains a fixed value if 'Specification' is 2;

intercept1 = the intercept term α of the G-spline. If 'Specification' is 1, this column usually contains zeros;

scale1 = the scale parameter τ of the G-spline. If 'Specification' is 1, this column usually contains ones;

gspline_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

The same structure as `gspline.sim`.

gspline_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

The same structure as `gspline.sim`.

gspline_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

The same structure as `gspline.sim`.

mlogweight.sim this file is related to the density of the error term from the model given by `formula`.

Fully created only if `store$a = TRUE`. The file contains the transformed weights a_k, $k = -K, \ldots, K$ of all mixture components, i.e. also of components that had numerically zero probabilities.

mlogweight_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

Fully created only if `store$a.b = TRUE`.

The same structure as `mlogweight.sim`.

mlogweight_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

Fully created only if `store$a2 = TRUE`.

The same structure as `mlogweight.sim`.

mlogweight_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

Fully created only if `store$a.b2 = TRUE`.

The same structure as `mlogweight.sim`.

r.sim this file is related to the density of the error term from the model given by `formula`.

Fully created only if `store$r = TRUE`. The file contains the labels of the mixture components into which the residuals are intrinsically assigned. Instead of indeces on the scale $\{-K, \ldots, K\}$ values from 1 to $(2\,K + 1)$ are stored here. Function `vecr2matr` can be used to transform it back to indices from $-K$ to K.

r_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

Fully created only if `store$r.b = TRUE`.

The same structure as `r.sim`.

r_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

Fully created only if `store$r2 = TRUE`.

The same structure as `r.sim`.

r_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

Fully created only if `store$r.b2 = TRUE`.

The same structure as `r.sim`.

lambda.sim this file is related to the density of the error term from the model given by `formula`.

One column labeled `lambda`. These are the values of the smoothing parameterλ (hyperparameters of the prior distribution of the transformed mixture weights a_k).

lambda_b.sim this file is related to the density of the random intercept from the model given by `formula` and `random`.

The same structure as `lambda.sim`.

lambda_2.sim in the case of doubly-censored data. This file is related to the density of the error term from the model given by `formula2`.

The same structure as `lambda.sim`.

lambda_b2.sim in the case of doubly-censored data. This file is related to the density of the random intercept from the model given by `formula2` and `random2`.

The same structure as `lambda.sim`.

beta.sim this file is related to the model given by `formula`.

Sampled values of the regression parameters β.

The columns are labeled according to the `colnames` of the design matrix.

beta_2.sim in the case of doubly-censored data, the same structure as `beta.sim`, however related to the model given by `formula2`.

b.sim this file is related to the model given by `formula` and `random`.

Fully created only if `store$b = TRUE`. It contains sampled values of random intercepts for all clusters in the data set. The file has N columns.

b_2.sim fully created only if `store$b2 = TRUE` and in the case of doubly-censored data, the same structure as `b.sim`, however related to the model given by `formula2` and `random2`.

Y.sim this file is related to the model given by `formula`.

Fully created only if `store$y = TRUE`. It contains sampled (augmented) log-event times for all observations in the data set.

Y_2.sim fully created only if `store$y2 = TRUE` and in the case of doubly-censored data, the same structure as `Y.sim`, however related to the model given by `formula2`.

logposter.sim This file is related to the residuals of the model given by `formula`.

Columns labeled `loglik`, `penalty`, and `logprw`. The columns have the following meaning.

loglik $= -(\sum_{i=1}^{N} n_i)\left\{\log(\sqrt{2\pi}) + \log(\sigma)\right\} - 0.5\sum_{i=1}^{N}\sum_{l=1}^{n_i}\left\{(\sigma^2\,\tau^2)^{-1}\;(y_{i,l} - x'_{i,l}\beta - b_i - \alpha - \tau\mu_{r_{i,l}})^2\right\}$

where $y_{i,l}$ denotes (augmented) *(i,l)*th true log-event time.

In other words, `loglik` is equal to the conditional log-density

$$\sum_{i=1}^{N}\sum_{l=1}^{n_i}\log\Big\{p\big(y_{i,l} \mid r_{i,l},\, \beta,\, b_i,\, \text{error-G-spline}\big)\Big\};$$

penalty: the penalty term

$$-\frac{1}{2}\sum_{k}\Big(\Delta\, a_k\Big)^2$$

(not multiplied by λ);

logprw $= -2\,(\sum_i n_i)\,\log\{\sum_k a_k\} + \sum_k N_k\, a_k$, where N_k is the number of residuals assigned intrinsincally to the kth mixture component.

In other words, `logprw` is equal to the conditional log-density

$$\sum_{i=1}^{N}\sum_{l=1}^{n_i}\log\big\{p(r_{i,l} \mid \text{error-G-spline weights})\big\}.$$

logposter_b.sim This file is related to the random intercepts from the model given by `formula` and `random`.

Columns labeled `loglik`, `penalty`, and `logprw`. The columns have the following meaning.

loglik $= -N\left\{\log(\sqrt{2\pi}) + \log(\sigma)\right\} - 0.5\sum_{i=1}^{N}\left\{(\sigma^2\,\tau^2)^{-1}\;(b_i - \alpha - \tau\mu_{r_i})^2\right\}$

where b_i denotes (augmented) *i*th random intercept.

In other words, `loglik` is equal to the conditional log-density

$$\sum_{i=1}^{N}\log\Big\{p\big(b_i \mid r_i,\, \text{b-G-spline}\big)\Big\};$$

The columns `penalty` and `logprw` have the analogous meaning as in the case of logposter.sim file.

logposter_2.sim in the case of doubly-censored data, the same structure as `logposter.sim`, however related to the model given by `formula2`.

logposter_b2.sim in the case of doubly-censored data, the same structure as `logposter_b.sim`, however related to the model given by `formula2` and `random2`.

Author(s)

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

References

García-Zattera, M. J., Jara, A., and Komárek, A. (2016). A flexible AFT model for misclassified clustered interval-censored data. *Biometrics*, **72**, 473 - 483.

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2008). Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions. *Journal of the American Statistical Association*, **103**, 523 - 533.

Robert C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods, Second Edition.* New York: Springer Science+Business Media.

Examples

```
## See the description of R commands for
## the cluster specific AFT model
## with the Signal Tandmobiel data,
## analysis described in Komarek and Lesaffre (2007).
##
## R commands available in the documentation
## directory of this package
## - see ex-tandmobCS.R and
##   https://www2.karlin.mff.cuni.cz/ komarek/software/bayesSurv/ex-tandmobCS.pdf
##
```

---

cgd *Chronic Granulomatous Disease data*

Description

Dataset from Fleming and Harrington (1991).

Data from a multicenter placebo-controlled randomized trial of gamma inferon in patients with chronic granulomatous disease (CGD). 128 patients were randomized to either gamma inferon ($n = 63$) or placebo ($n = 65$). For each patient the times from study entry to initial and any recurrent serious infections are available. There is a minimum of one and a maximum of eight (recurrent) infection times per patient, with a total of 203 records.

Usage

```
data(cgd)
```

Format

a~data frame with 203 rows and 17 columns. There are the following variables contained in the data frame.

hospit code of the hospital

ID case identification number

RDT date randomization onto study (mmddyy)

IDT date of onset of serious infection, or date the patient was taken off the study (mmddyy)

trtmt treatment code, 1 = rIFN-gamma, 2 = placebo

inherit pattern of inheritance, 1 = X-linked, 2 = autosomal recessive

age age in years

height height of the patient in cm

weight weight of the patient in kg

cortico using corticosteroids at time of study entry, 1 = yes, 2 = no

prophy using prophylatic antibiotics at time of study entry, 1 = yes, 2 = no

gender 1 = male, 2 = female

hcat hospital category, 1 = US-NIH, 2 = US-other, 2 = Europe - Amsterdam, 4 = Europe - other

T1 elapsed time (in days) from randomization (from sequence = 1 record) to diagnosis of a serious infection or, if a censored observation, elapsed time from randomization to censoring date; computed as IDT - RDT (from sequence = 1 record)

T2 0, for sequence = 1 record, if sequence > 1, T2 = T1(from previous record) + 1

event censoring indicator, 1 = non-censored observation, 2 = censored observation

sequence sequence number, for each patient, the infection recods are in sequence number order

Source

Appendix D.2 of Fleming and Harrington (1991).

References

Fleming, T. R. and Harrington, D. P. (1991). *Counting Processes and Survival Analysis*. New York: John Wiley and Sons.

`credible.region` *Compute a simultaneous credible region (rectangle) from a sample for a vector valued parameter.*

Description

See references below for more details.

Usage

```
credible.region(sample, probs=c(0.90, 0.975))
```

Arguments

`sample`	a data frame or matrix with sampled values (one column = one parameter)
`probs`	probabilities for which the credible regions are to be computed

Value

A list (one component for each confidence region) of length equal to `length(probs)`. Each component of the list is a matrix with two rows (lower and upper limit) and as many columns as the number of parameters giving the confidence region.

Author(s)

Arnošt Komárek <arnost.komarek@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, page 30

Held, L. (2004). Simultaneous inference in risk assessment; a Bayesian perspective *In: COMPSTAT 2004, Proceedings in Computational Statistics (J. Antoch, Ed.),* 213 - 222, page 214

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

Examples

```
m <- 10000
sample <- data.frame(x1=rnorm(m), x2=rnorm(m), x3=rnorm(m))
probs <- c(0.70, 0.90, 0.95)
CR <- credible.region(sample, probs=probs)

for (kk in 1:length(CR)){
 suma <- sum(sample$x1 >= CR[[kk]]["Lower", "x1"] & sample$x1 <= CR[[kk]]["Upper", "x1"] &
          sample$x2 >= CR[[kk]]["Lower", "x2"] & sample$x2 <= CR[[kk]]["Upper", "x2"] &
           sample$x3 >= CR[[kk]]["Lower", "x3"] & sample$x3 <= CR[[kk]]["Upper", "x3"])
  show <- c(suma/m, probs[kk])
```

```
  names(show) <- c("Empirical", "Desired")
  print(show)
}
```

---

densplot2 *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

```
densplot2(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.

Value

No return value.

Author(s)

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

`files2coda` *Read the sampled values from the Bayesian survival regression model to a coda mcmc object.*

Description

This function creates a coda `mcmc` object from values found in files where sampled values from `bayessurvreg1` function are stored or from data.frames.

Usage

```
files2coda(files, data.frames, variant = 1, dir = getwd(),
   start = 1, end, thin = 1, header = TRUE, chain)
```

Arguments

`files`	a vector of strings giving the names of files that are to be converted to coda objects. If missing and `data.frames` is also missing, all appropriate files found in a directory `dir` are converted to coda objects. File "iteration.sim" is always used (if found) to index the sampled values. If this file is not found the sampled values are indexed from 1 to the sample size. If "mixture.sim" appeares here, only the column with number of mixture components is converted to the coda object.
`data.frames`	a vector of strings giving the names of data.frames that are to be converted to coda objects.
`variant`	a variant of `bayessurvreg` function used to generate sampled values. This argument is only used to identify appropriate files when `files` argument is missing. Currently only 1 is supported to cooperate with `bayessurvreg1`.
`dir`	string giving the directory where it will be searched for the files with sampled values.
`start`	the first row (possible header does not count) from the files with the sampled values that will be converted to coda objects.
`end`	the last row from the files with the sampled values that will be converted to coda objects. If missing, it is the last row in files.
`thin`	additional thinning of sampled values (i.e. only every `thin` value from `files` and `data.frames` is considered).
`header`	TRUE or FALSE indicating whether the files with the sampled values contain also the header on the first line or not.
`chain`	parameter giving the number of the chain if parallel chains were created and sampled values stored in data.frames further stored in lists(). If `missing`, data.frames are not assumed to be stored in lists.

Value

A list with `mcmc` objects. One object per file or data.frame.

Author(s)

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

Examples

```
## *** illustration of usage of parameters 'data.frames' and 'chain' ***
## ********************************************************************
## Two parallel chains with four variables, stored in data.frames
## data.frames are further stored in lists
library("coda")

group1 <- list();  group2 <- list();  group3 <- list()
    ## first chain of first two variables:
  group1[[1]] <- data.frame(var1 = rnorm(100, 0, 1), var2 = rnorm(100, 5, 4))
    ## second chain of first two variables:
  group1[[2]] <- data.frame(var1 = rnorm(100, 0, 1), var2 = rnorm(100, 5, 4))
    ## first chain of the third variable:
  group2[[1]] <- data.frame(var3 = rgamma(100, 1, 1))
    ## second chain of the third variable:
  group2[[2]] <- data.frame(var3 = rgamma(100, 1, 1))
    ## first chain of the fourth variable:
  group3[[1]] <- data.frame(var4 = rbinom(100, 1, 0.4))
    ## second chain of the fourth variable:
  group3[[2]] <- data.frame(var4 = rbinom(100, 1, 0.4))

  ## Create mcmc objects for each chain separately
mc.chain1 <- files2coda(data.frames = c("group1", "group2", "group3"), chain = 1)
mc.chain2 <- files2coda(data.frames = c("group1", "group2", "group3"), chain = 2)

  ## Create mcmc.list to represent two parallel chains
mc <- mcmc.list(mc.chain1, mc.chain2)
rm(mc.chain1, mc.chain2)


## *** illustration of usage of parameter 'data.frames' without 'chain' ***
## *************************************************************************
## Only one chain for four variables was sampled and stored in three data.frames
    ## chain of first two variables:
  group1 <- data.frame(var1 = rnorm(100, 0, 1), var2 = rnorm(100, 5, 4))
    ## chain of the third variable:
  group2 <- data.frame(var3 = rgamma(100, 1, 1))
    ## chain of the fourth variable:
  group3 <- data.frame(var4 = rbinom(100, 1, 0.4))

  ## Create an mcmc object
mc <- files2coda(data.frames = c("group1", "group2", "group3"))
```

`give.summary` *Brief summary for the chain(s) obtained using the MCMC.*

Description

This function computes a sample mean, quantiles and a Bayesian p-value which is defined as

$$p = 2 \times \min(n_-, n_+),$$

where n_- is the number of the sampled values which are negative and n_+ is the number of sampled values which are positive.

Usage

```
give.summary(sample, probs=c(0.5, 0.025, 0.975))
```

Arguments

sample	a data frame or a vector with sampled values
probs	probabilities of the quantiles that are to be computed

Value

A matrix or a vector with the sample mean, quantiles and a Bayesian p-value.

Author(s)

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

Examples

```
## Example with a sample stored in a vector:
sample <- rnorm(1000)
give.summary(sample)

## Example with a sample stored in a data.frame:
sample <- data.frame(x=rnorm(1000), y=rgamma(1000, shape=1, rate=1))
give.summary(sample)
```

---

marginal.bayesGspline *Summary for the marginal density estimates based on the bivariate model with Bayesian G-splines.*

Description

Compute the estimate of the marginal density function based on the values sampled using the MCMC (MCMC average evaluated in a grid of values) in a model where density is specified as a bivariate Bayesian G-spline.

This function serves to summarize the MCMC chains related to the distributional parts of the considered models obtained using the functions: bayesHistogram and bayesBisurvreg.

If asked, this function returns also the values of the marginal G-spline evaluated in a grid at each iteration of MCMC.

Usage

```
marginal.bayesGspline(dir, extens = "", K, grid1, grid2,
   skip = 0, by = 1, last.iter, nwrite, only.aver = TRUE)
```

Arguments

dir	directory where to search for files ('mixmoment.sim', 'mweight.sim', 'mmean.sim', 'gspline.sim') with the MCMC sample.
extens	an extension used to distinguish different sampled G-splines if more G-splines were used in one simulation (e.g. with doubly-censored data). According to which bayes*survreg* function was used, specify the argument extens in the following way. **bayesHistogram:** always extens = "" **bayesBisurvreg:** • to compute the marginals of the bivariate distribution of the *error* term for the *onset* time: extens = ""; • to compute the marginals of the bivariate distribution of the *error* term for the *event* time if there was doubly-censoring: extens = "_2";
K	a~vector of length 2 specifying the number of knots at each side of the middle knot for each dimension of the G-spline.
grid1	grid of values from the first dimension at which the sampled marginal densities are to be evaluated.
grid2	grid of values from the second dimension at which the sampled marginal densities are to be evaluated.
skip	number of rows that should be skipped at the beginning of each *.sim file with the stored sample.
by	additional thinning of the sample.
last.iter	index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file mixmoment.sim.
nwrite	frequency with which is the user informed about the progress of computation (every nwriteth iteration count of iterations change).
only.aver	TRUE/FALSE, if TRUE only MCMC average is returned otherwise also values of the marginal G-spline at each iteration are returned (which might ask for quite lots of memory).

Value

An object of class marginal.bayesGspline is returned. This object is a list with components margin1 and margin2 (for two margins).

Both margin1 and margin2 components are data.frames with columns named grid and average where

grid	is a grid of values (vector) at which the McMC average of the marginal G-spline was computed.
average	are McMC averages of the marginal G-spline (vector) evaluated in grid.

There exists a method to plot objects of the class marginal.bayesGspline.

Attributes

Additionally, the object of class `marginal.bayesGspline` has the following attributes:

`sample.size` a length of the McMC sample used to compute the McMC average.

`sample1` marginal (margin = 1) G-spline evaluated in a grid of values. This attribute is present only if `only.aver = FALSE`.

This a matrix with `sample.size` columns and length(grid1) rows.

`sample2` marginal (margin = 2) G-spline evaluated in a grid of values. This attribute is present only if `only.aver = FALSE`.

This a matrix with `sample.size` columns and length(grid2) rows.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3 - 22.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2006),
##
## R commands available
## in the documentation
## directory of this package
## - see ex-tandmobPA.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobPA.pdf
##
```

---

plot.bayesDensity *Plot an object of class bayesDensity*

Description

This function plots an object created by `bayesDensity`.

Usage

```
## S3 method for class 'bayesDensity'
plot(x, k.cond, dim.plot = TRUE, over = TRUE,
   alegend = TRUE, standard = TRUE, center = FALSE,
   type = "l", bty = "n",
   xlab = expression(epsilon), ylab = expression(f(epsilon)),
   lty, xlim, ylim, xleg, yleg, main, ...)
```

Arguments

x	an object of class bayesDensity.
k.cond	a numerical vector giving the numbers of mixture components for which the conditional densities are to be plotted. 0 states for the unconditional (overall) density, averaged over the mixture with all possible numbers of components. If NULL, all conditional and the unconditional density found in x will be plotted.
dim.plot	an indicator whether the dimension of the plot used in par(mfrow) should be computed automatically. If dim.plot = FALSE and over = FALSE the user has to determine himself using par(mfrow) how to put the plots on the page.
over	an indicator whether all densities should be drawn into one plot using different types of lines. If FALSE a separate plot for each density is created.
alegend	an indicator whether an automatic legend should be added to the plot.
standard	logical, do we want to plot standardized density?
center	logical, do we want to plot centered density?, set both standard and center to FALSE if you wish to plot unstandardized density.
xleg, yleg	position of the legend if over = TRUE.
type, bty, xlab, ylab, lty, xlim, ylim, main, ...	other arguments passed to the plot.default function.

Value

No return value.

Author(s)

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

---

plot.bayesGspline *Plot an object of class bayesGspline*

Description

This function plots an object created by bayesGspline.

Usage

```
## S3 method for class 'bayesGspline'
plot(x, add = FALSE, type = "l", lty=1, bty = "n",
     xlab, ylab, main, sub, ...)
```

Arguments

x	an object of class bayesGspline.
add	if TRUE a new plot is produced, otherwise it is drawn to an existing plot.
type, lty, bty, xlab, ylab, main, sub, ...	other arguments passed to the plot.default function.

Value

No return value.

Author(s)

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

---

plot.marginal.bayesGspline

Plot an object of class marginal.bayesGspline

Description

This function plots an object created by marginal.bayesGspline.

Usage

```
## S3 method for class 'marginal.bayesGspline'
plot(x, type = "l", lty=1, bty = "n",
   xlab1, ylab1, main1, xlab2, ylab2, main2, sub, ...)
```

Arguments

x	an object of class marginal.bayesGspline.
type, lty, bty, xlab1, ylab1, main1, xlab2, ylab2, main2, sub, ...	other arguments passed to the plot.default function.

Value

No return value.

Author(s)

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

predictive *Compute predictive quantities based on a Bayesian survival regression model fitted using bayessurvreg1 function.*

Description

This function runs additional McMC to compute predictive survivor and hazard curves and predictive event times for specified values of covariates.

Firstly, the function bayessurvreg1 has to be used to obtain a sample from the posterior distribution of unknown quantities.

Directly, posterior predictive quantiles and means of asked quantities are computed and stored in files.

Function predictive.control serves only to perform some input checks inside the main function predictive.

Usage

```
predictive(formula, random, time0 = 0, data = parent.frame(),
     grid, type = "mixture", subset, na.action = na.fail,
     quantile = c(0, 0.025, 0.5, 0.975, 1),
     skip = 0, by = 1, last.iter, nwrite, only.aver = FALSE,
    predict = list(Et=TRUE, t=FALSE, Surv=TRUE, hazard=FALSE, cum.hazard=FALSE),
   store = list(Et=TRUE, t = FALSE, Surv = FALSE, hazard = FALSE, cum.hazard=FALSE),
     Eb0.depend.mix = FALSE,
     dir, toler.chol = 1e-10, toler.qr = 1e-10)

predictive.control(predict, store, only.aver, quantile)
```

Arguments

formula	the same formula as that one used to sample from the posterior distribution of unknown quantities by the function bayessurvreg1.
random	the same random statement as that one used to sample from the posterior distribution of unknown quantities by the function bayessurvreg1.
time0	starting time for the survival model. This option is used to get correct hazard function in the case that the original model was $\log(T - time0) = \ldots$.
data	optional data frame in which to interpret the variables occuring in the formulas. Usually, you create a new data.frame similar to that one used to obtain a sample from the posterior distribution. In this new data.frame, put covariate values equal to these for which predictive quantities are to be obtained. If cluster statement was used, assign a unique cluster identification to each observation. Response variable and a censoring indicator may be set to arbitrary values. They are only used in formula but are ignored for computation.

`grid` a list of length as number of observations in `data` or a vector giving grids of values where predictive survivor functions, hazards, cumulative hazards are to be evaluated. If it is a vector, same grid is used for all observations from `data`. Not needed if only `predict$t` or `predict$Et` are `TRUE`. If `time0` is different from zero your grid should start at `time0` and not at zero.

`type` a character string giving the type of assumed error distribution. Currently, valid are substrings of "mixture". In the future, "spline", "polya.tree" might be also implemented.

`subset` subset of the observations from the `data` to be used. This option will normally not be needed.

`na.action` function to be used to handle any `NA`s in the data. The user is discouraged to change a default value `na.fail`.

`quantile` a vector of quantiles that are to be computed for each predictive quantity.

`skip` number of rows that should be skipped at the beginning of each *.sim file with the stored sample.

`by` additional thinning of the sample.

`last.iter` index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file `mixmoment.sim`.

`nwrite` frequency with which is the user informed about the progress of computation (every `nwrite`th iteration count of iterations change).

`only.aver` if `TRUE` only posterior predictive mean is computed for all quantities and no quantiles.

`predict` a list of logical values indicating which predictive quantities are to be sampled. Components of the list:

Et predictive expectations of survivor times

t predictive survivor times

Surv predictive survivor functions

hazard predictive hazard functions

cum.hazard predictive cumulative hazard functions

`store` a list of logical values indicating which predictive quantities are to be stored in files as 'predET*.sim', 'predT*.sim', 'predS*.sim', 'predhazard*.sim', 'predcumhazard*.sim'. If you are interested only in posterior means or quantiles of the predictive quantities you do not have to store sampled values. Posterior means and quantiles are stored in files 'quantET*.sim', 'quantT*.sim', 'quantS*.sim', 'quanthazard*.sim', 'quantpredhazard*.sim'.

`Eb0.depend.mix` a logical value indicating whether the mean of the random intercept (if included in the model) was given in a hierarchical model as an overall mean of the mixture in the error term. With `FALSE` (default) you have the same model as that one described in an accompanying paper. An ordinary user is discouraged from setting this to `TRUE`.

`dir` a string giving a directory where previously simulated values were stored and where newly obtained quantities will be stored. On Unix, do not use '~/' to specify your home directory. A full path must be given, e.g. '/home/arnost/'.

`toler.chol` tolerance for the Cholesky decomposition.

`toler.qr` tolerance for the QR decomposition.

Value

An integer which should be equal to zero if everything ran fine.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2007). Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution. *Statistica Sinica*, **17**, 549 - 569.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2007).
##
## R commands available
## in the documentation
## directory of this package as
## - ex-cgd.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-cgd.pdf
##
## - ex-tandmobMixture.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobMixture.pdf
##
```

`predictive2` *Compute predictive quantities based on a Bayesian survival regression model fitted using bayesBisurvreg or bayessurvreg2 or bayessurvreg3 functions.*

Description

This function computes predictive densities, survivor and hazard curves for specified combinations of covariates.

Firstly, either the function `bayesBisurvreg` or the function `bayessurvreg2` or the function `bayessurvreg3` has to be used to obtain a sample from the posterior distribution of unknown quantities.

Function `predictive2.control` serves only to perform some input checks inside the main function `predictive2`.

Usage

```
predictive2(formula, random, obs.dim, time0, data = parent.frame(),
     grid, na.action = na.fail, Gspline,
     quantile = c(0, 0.025, 0.5, 0.975, 1),
     skip = 0, by = 1, last.iter, nwrite,
     only.aver = TRUE,
     predict = list(density=FALSE, Surv=TRUE,
                    hazard=FALSE, cum.hazard=FALSE),
     dir, extens = "", extens.random="_b", version=0)

predictive2Para(formula, random, obs.dim, time0, data = parent.frame(),
     grid, na.action = na.fail, Gspline,
     quantile = c(0, 0.025, 0.5, 0.975, 1),
     skip = 0, by = 1, last.iter, nwrite,
     only.aver = TRUE,
     predict = list(density=FALSE, Surv=TRUE,
                    hazard=FALSE, cum.hazard=FALSE),
     dir, extens = "", extens.random="_b", version=0)

predictive2.control(predict, only.aver, quantile, obs.dim,
     time0, Gspline, n)
```

Arguments

formula	the same formula as that one used to sample from the posterior distribution of unknown quantities by the function bayesBisurvreg or bayessurvreg2 or bayessurvreg3. On the left hand side whichever Surv object of a~proper length can be used (it is ignored anyway). REMARK: the prediction must be asked for at least two combinations of covariates. This is the restriction imposed by one of the internal functions I use.
random	the same random statement as that one used to sample from the posterior distribution of unknown quantities by the function bayessurvreg2 or bayessurvreg3, not applicable if bayesBisurvreg was used to sample from the posterior distribution.
obs.dim	a vector that has to be supplied if bivariate data were used for estimation (using the function bayesBisurvreg). This vector has to be of the same length as the number of covariate combinations for which the predictive quantities are to be computed. It determines to which dimension (1 or 2) each observation belong.
time0	a~vector of length Gspline$dim giving the starting time for the survival model. It does not have to be supplied if equal to zero (usually). This option is used to get hazard and density functions on the original time scale in the case that the model was $\log(T - time0) = \ldots$. Note that time0 IS NOT the starting time of doubly censored observation since there after subtracting the onset time, the starting time is (usually) equal to zero.
data	optional data frame in which to interpret the variables occuring in the formulas. Usually, you create a new data.frame similar to that one used to obtain

a sample from the posterior distribution. In this new `data.frame`, put covariate values equal to these for which predictive quantities are to be obtained. If `cluster` statement was used, assign a unique cluster identification to each observation. Response variable and a censoring indicator may be set to arbitrary values. They are only used in `formula` but are ignored for computation.

`grid` a~vector giving the grid of values where predictive quantities are to be evaluated. The grid should normally start at some value slightly higher than `time0`.

`na.action` function to be used to handle any `NA`s in the data. The user is discouraged to change a default value `na.fail`.

`Gspline` a~list specifying the G-spline used for the error distribution in the model. It is a~list with the following components:

dim dimension of the G-spline, in the case when the function `bayesBisurvreg` was used to fit the model this will usually be equal to 2, in the case when the function `bayessurvreg2` was used to fit the model this MUST be equal to 1.

K a~vector of length `Gspline$dim` specifying the number of knots at each side of the middle knot for each dimension of the G-spline.

`quantile` a vector of quantiles that are to be computed for each predictive quantity.

`skip` number of rows that should be skipped at the beginning of each *.sim file with the stored sample.

`by` additional thinning of the sample.

`last.iter` index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file `mixmoment.sim`.

`nwrite` frequency with which is the user informed about the progress of computation (every `nwrite`th iteration count of iterations change).

`only.aver` if `TRUE` only posterior predictive mean is computed for all quantities and no quantiles.

The word of warning: with `only.aver` set to `FALSE`, all quantities must be stored for all iterations of the MCMC to be able to compute the quantiles. This might require quite lots of memory.

`predict` a list of logical values indicating which predictive quantities are to be computed. Components of the list:

density predictive density

Surv predictive survivor functions

hazard predictive hazard functions

cum.hazard predictive cumulative hazard functions

`dir` directory where to search for files ('mixmoment.sim', 'mweight.sim', mmean.sim', gspline.sim', 'beta.sim', 'D.sim', ...) with the McMC sample.

`extens` an extension used to distinguish different sampled G-splines if more formulas were used in one MCMC simulation (e.g. with doubly-censored data).

- if the data were not doubly-censored or if you wish to compute predictive quantities for the *onset* time of the doubly-censored data then

`extens = ""`

- if the data were doubly-censored and you wish to compute predictive quantities for the *event* time then

`extens = "_2"`

`extens.random` only applicable if the function `bayessurvreg3` was used to generate the MCMC sample.

This is an extension used to distinguish different sampled G-splines determining the distribution of the random intercept (under the presence of doubly-censored data).

- if the data were not doubly-censored or if you wish to compute predictive quantities for the *onset* time of the doubly-censored data then

`extens.random = "_b"`

- if the data were doubly-censored and you wish to compute predictive quantities for the *event* time then

`extens.random = "_b2"`

`version` this argument indicates by which `bayes*survreg*` function the chains used by `bayesGspline` were created. Use the following:

bayesBisurvreg: `version = 0`;

bayessurvreg2: `version = 0`;

bayessurvreg3: with all distributions specified as G-splines: `version = 3`;

bayessurvreg3: with error distributions specified as G-splines and bivariate normal random intercepts: `version = 32`.

`n` number of covariate combinations for which the prediction will be performed.

Value

A list with possibly the following components (what is included depends on the value of the arguments `predict` and `only.aver`):

`grid` a~vector with the grid values at which the survivor function, survivor density, hazard and cumulative hazard are computed.

`Surv` predictive survivor functions.

A~matrix with as many columns as length(grid) and as many rows as the number of covariate combinations for which the predictive quantities were asked. One row per covariate combination.

`density` predictive survivor densities.

The same structure as `Surv` component of the list.

`hazard` predictive hazard functions.

The same structure as `Surv` component of the list.

`cum.hazard` predictive cumulative hazard functions.
The same structure as `Surv` component of the list.

`quant.Surv` pointwise quantiles for the predictive survivor functions.
This is a list with as many components as the number of covariate combinations. One component per covariate combination.
Each component of this list is a~matrix with as many columns as length(grid) and as many rows as the length of the argument `quantile`. Each row of this matrix gives values of one quantile. The rows are also labeled by the probabilities (in %) of the quantiles.

`quant.density` pointwise quantiles for the predictive survivor densities.
The same structure as `quant.Surv` component of the list.

`quant.hazard` pointwise quantiles for the predictive hazard functions.
The same structure as `quant.Surv` component of the list.

`quant.cum.hazard`
pointwise quantiles for the predictive cumulative hazard functions.
The same structure as `quant.Surv` component of the list.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2008). Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions. *Journal of the American Statistical Association*, **103**, 523 - 533.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3 - 22.

Komárek, A., Lesaffre, E., and Legrand, C. (2007). Baseline and treatment effect heterogeneity for survival times between centers using a random effects accelerated failure time model with flexible error distribution. *Statistics in Medicine*, **26**, 5457 - 5472.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2006),
## Komarek and Lesaffre (2008),
## Komarek, Lesaffre, and Legrand (2007).
##
## R commands available in the documentation
## directory of this package
##  - ex-tandmobPA.R and
```

```
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobPA.pdf
##  - ex-tandmobCS.R and
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobCS.pdf
##  - ex-eortc.R and
##    https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-eortc.pdf
```

`print.bayesDensity` *Print a summary for the density estimate based on the Bayesian model.*

Description

This function prints a~object created by `bayesDensity`.

Usage

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

Arguments

`x`	an object of class `bayesDensity`.
`...`	this is here for a consistency with a generic function.

Value

No return value.

Author(s)

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

`rMVNorm` *Sample from the multivariate normal distribution*

Description

According to the parametrization used, sample from the multivariate normal distribution.

The following parametrization can be specified

standard In this case we sample from either $\mathcal{N}(\mu, \Sigma)$ or from $\mathcal{N}(\mu, Q^{-1})$.

canonical In this case we sample from $\mathcal{N}(Q^{-1}b,\ Q^{-1})$.

Generation of random numbers is performed by Algorithms 2.3-2.5 in Rue and Held (2005, pp. 34-35).

Usage

```
rMVNorm(n, mean=0, Sigma=1, Q, param=c("standard", "canonical"))
```

Arguments

n	number of observations to be sampled.
mean	For `param="standard"`, it is a vector μ of means. If `length(mean)` is equal to 1, it is recycled and all components have the same mean. For `param="canonical"`, it is a vector b of canonical means. If `length(mean)` is equal to 1, it is recycled and all components have the same mean.
Sigma	covariance matrix of the multivariate normal distribution. It is ignored if `Q` is given at the same time.
Q	precision matrix of the multivariate normal distribution. It does not have to be supplied provided `Sigma` is given and `param="standard"`. It must be supplied if `param="canonical"`.
param	a character which specifies the parametrization.

Value

Matrix with sampled points in rows.

Author(s)

Arnošt Komárek <arnost.komarek@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

rnorm, Mvnorm.

Examples

```
### Mean, covariance matrix, its inverse
### and the canonical mean
mu <- c(0, 2, 0.5)
L <- matrix(c(1, 1, 1,  0, 0.5, 0.5,  0, 0, 0.3), ncol=3)
Sigma <- L %*% t(L)
Q <- chol2inv(t(L))
b <- Q %*% mu

print(Sigma)
print(Q)
print(Sigma %*% Q)

### Sample using different parametrizations
```

```
set.seed(775988621)
n <- 10000

### Sample from N(mu, Sigma)
xx1 <- rMVNorm(n=n, mean=mu, Sigma=Sigma)
apply(xx1, 2, mean)
var(xx1)

### Sample from N(mu, Q^{-1})
xx2 <- rMVNorm(n=n, mean=mu, Q=Q)
apply(xx2, 2, mean)
var(xx2)

### Sample from N(Q^{-1}*b, Q^{-1})
xx3 <- rMVNorm(n=n, mean=b, Q=Q, param="canonical")
apply(xx3, 2, mean)
var(xx3)
```

`rWishart` *Sample from the Wishart distribution*

Description

Sample from the Wishart distribution

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

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

$$\text{E}(W) = \nu S$$

.

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

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

Usage

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

Arguments

`n`	number of observations to be sampled.
`df`	degrees of freedom of the Wishart distribution.
`S`	scale matrix of the Wishart distribution.

Value

Matrix with sampled points (lower triangles of W) in rows.

Author(s)

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

References

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis, Second edition*. Boca Raton: Chapman and Hall/CRC.

Ripley, B. D. (1987). *Stochastic Simulation*. New York: John Wiley and Sons.

Examples

```
### The same as rgamma(n, shape=df/2, rate=1/(2*S))
n <- 1000
df <- 1
S  <- 3
w <- rWishart(n=n, df=df, S=S)
mean(w)    ## should be close to df*S
var(w)     ## should be close to 2*df*S^2

### Multivariate Wishart
n <- 1000
df <- 2
S <- matrix(c(1,3,3,13), nrow=2)
w <- rWishart(n=n, df=df, S=S)
apply(w, 2, mean)                ## should be close to df*S
df*S

df <- 2.5
S <- matrix(c(1,2,3,2,20,26,3,26,70), nrow=3)
w <- rWishart(n=n, df=df, S=S)
apply(w, 2, mean)                ## should be close to df*S
df*S
```

---

sampleCovMat *Compute a sample covariance matrix.*

Description

This function computes a sample covariance matrix.

Usage

```
sampleCovMat(sample)
```

Arguments

sample	a matrix or data.frame with sampled values in rows. I.e. number of rows of sample determines a sample size, number of columns of sample determines a dimension of the distribution from which it was sampled.

Details

When $y_1, \ldots, y_n$ is a sequence of p-dimensional vectors y_i the sample covariance matrix S is equal to

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - m)(y_i - m)^T$$

where

$$m = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

When $n = 1$ the function returns just sum of squares.

Value

This function returns a matrix.

Author(s)

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

Examples

```
## Sample some values
z1 <- rnorm(100, 0, 1)          ## first components of y
z2 <- rnorm(100, 5, 2)          ## second components of y
z3 <- rnorm(100, 10, 0.5)       ## third components of y

## Put them into a data.frame
sample <- data.frame(z1, z2, z3)

## Compute a sample covariance matrix
sampleCovMat(sample)
```

---

sampled.kendall.tau *Estimate of the Kendall's tau from the bivariate model*

Description

This function computes an estimate of the residual (after adjustment for covariates) Kendall's tau for the bivariate survival model fitted using the functions bayesHistogram or bayesBisurvreg.

For both these function their argument prior$specification must be equal to 2!

When G is a bivariate distribution function, the population version of the Kendall's tau is defined as

$$\tau = 4 \int G dG - 1$$

.

For the model estimated using one of the above mentioned functions the value of Kendall's tau at each iteration of MCMC is equal to

$$\tau = 4 \sum_{i=-K_1}^{K_1} \sum_{j=-K_2}^{K_2} \sum_{k=-K_1}^{K_1} \sum_{l=-K_2}^{K_2} w_{i,j} w_{k,l} \Phi\left(\frac{\mu_{1,i} - \mu_{1,k}}{\sqrt{2}\sigma_1}\right) \Phi\left(\frac{\mu_{2,j} - \mu_{2,l}}{\sqrt{2}\sigma_2}\right) - 1,$$

where $\mu_{1,-K_1}, \ldots, \mu_{1,K_1}$ are knots in the first margin, $\mu_{2,-K_2}, \ldots, \mu_{2,K_2}$ are knots in the second margin, σ_1 is the basis standard deviation in the first margin, σ_2 is the basis standard deviation in the second margin, and $w_{i,j},\ i = -K_1, \ldots, K_1,\ j = -K_2, \ldots, K_2$ are the G-spline weights.

Usage

```
sampled.kendall.tau(dir = getwd(), extens = "", K,
  skip = 0, by = 1, last.iter, nwrite)
```

Arguments

`dir`	directory where to search for files ('mixmoment.sim', 'mweight.sim', 'mmean.sim', 'gspline.sim') with the MCMC sample.
`extens`	an extension used to distinguish different sampled G-splines if more G-splines were used in one simulation (with doubly-censored data) According to which `bayes*survreg*` function was used, specify the argument `extens` in the following way. **bayesHistogram:** always `extens = ""` **bayesBisurvreg:** • to compute the bivariate distribution of the *error* term for the *onset* time: `extens = ""`; • to compute the bivariate distribution of the *error* term for the *event* time if there was doubly-censoring: `extens = "_2"`;
`K`	a~vector of length 2 specifying the number of knots at each side of the middle knot for each dimension of the G-spline.
`skip`	number of rows that should be skipped at the beginning of each *.sim file with the stored sample.
`by`	additional thinning of the sample.
`last.iter`	index of the last row from *.sim files that should be used. If not specified than it is set to the maximum available determined according to the file `mixmoment.sim`.
`nwrite`	frequency with which is the user informed about the progress of computation (every `nwrite`th iteration count of iterations change).

Value

A vector with sampled values of the Kendall's tau.

Author(s)

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

References

Komárek, A. (2006). *Accelerated Failure Time Models for Multivariate Interval-Censored Data with Flexible Distributional Assumptions*. PhD. Thesis, Katholieke Universiteit Leuven, Faculteit Wetenschappen.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3 - 22.

Examples

```
## See the description of R commands for
## the models described in
## Komarek (2006),
## Komarek and Lesaffre (2006),
##
## R commands available
## in the documentation
## directory of this package
## - see ex-tandmobPA.R and
##   https://www2.karlin.mff.cuni.cz/~komarek/software/bayesSurv/ex-tandmobPA.pdf
##
```

---

scanFN *Read Data Values*

Description

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

Usage

```
scanFN(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` argument.

`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@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 <- scanFN("ex.data", quiet=FALSE)
pp <- scanFN("ex.data", quiet= TRUE)
print(pp)
unlink("ex.data") # tidy up
```

`simult.pvalue` *Compute a simultaneous p-value from a sample for a vector valued parameter.*

Description

The p-value is computed as 1 - the credible level of the credible region which just cover the point (0, 0, ..., 0)'.

The function returns also the simultaneous credible region (rectangle) with a specified credible level.

Usage

```
simult.pvalue(sample, precision=0.001, prob=0.95)
## S3 method for class 'simult.pvalue'
print(x, ...)
```

Arguments

sample	a data frame or matrix with sampled values (one column = one parameter)
precision	precision with which the p-value is to be computed
prob	probability for which the credible region is to be computed
x	an object of class simult.pvalue
...	who knows

Value

An object of class 'simult.pvalue'.

Author(s)

Arnošt Komárek <arnost.komarek@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. page 30

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

Examples

```
m <- 1000
sample <- data.frame(x1=rnorm(m), x2=rnorm(m), x3=rnorm(m))
simult.pvalue(sample)

sample <- data.frame(x1=rnorm(m), x2=rnorm(m), x3=rnorm(m, mean=2))
simult.pvalue(sample)

sample <- data.frame(x1=rnorm(m), x2=rnorm(m), x3=rnorm(m, mean=5))
simult.pvalue(sample, prob=0.99, precision=0.0001)
```

tandmob2 *Signal Tandmobiel data, version 2*

Description

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

The dataset presented here contains mainly the information on the emergence and caries times summarized in the interval-censored observations. Some baseline covariates are also included here.

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

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

IMPORTANT NOTICE: It is possible to use these data for your research work under the condition that each manuscript is first approved by
Prof. Emmanuel Lesaffre
Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat)
Katholieke Universiteit Leuven
Kapucijnenvoer 35
B-3000 Leuven
Belgium
<emmanuel.lesaffre@kuleuven.be>

Usage

```
data(tandmob2)
```

Format

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

IDNR identification number of a child

GENDER character *boy* or *girl*

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

DOB character, date of birth in the format DDmmmYY

PROVINCE factor, code of the province with

- **0 =** Antwerpen
- **1 =** Vlaams Brabant
- **2 =** Limburg
- **3 =** Oost Vlaanderen
- **4 =** West Vlaanderen

EDUC factor, code of the educational system with

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

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

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

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

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

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

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

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

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

xx takes values as for the variable `EBEG.xx`.

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

xx takes values as for the variable `EBEG.xx`.

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

xx takes values as for the variable `EBEG.xx`.

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

Txx.DMF indicator whether a deciduous tooth xx was ***d****ecayed* or ***m****issing due to caries* or ***f****illed* on at most the last examination before the first examination when the emergence of the permanent successor was recorded.

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

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

Source

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

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

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

References

Komárek, A., Lesaffre, E., Härkänen, T., Declerck, D., and Virtanen, J. I. (2005). A Bayesian analysis of multivariate doubly-interval-censored dental data. *Biostatistics,* **6**, 145–155.

Komárek, A. and Lesaffre, E. (2007). Bayesian accelerated failure time model for correlated interval-censored data with a normal mixture as an error distribution. *Statistica Sinica,* **17**, 549–569.

Lesaffre, E., Komárek, A., and Declerck, D. (2005). An overview of methods for interval-censored data with an emphasis on applications in dentistry. *Statistical Methods in Medical Research*, **14**, 539–552.

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

`tandmobRoos` *Signal Tandmobiel data, version Roos*

Description

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

The dataset presented here contains mainly the information on the emergence and caries times summarized in the interval-censored observations. Some baseline covariates are also included here.

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

This is the version of the dataset used first by Leroy et al. (2005) and contains a subset of the `tandmob2`. Some children were removed to satisfy inclusion criteria given in Leroy et al. (2005). Additionally, left-censored emergence times of the permanent first molars are adjusted according to the eruption stage (see Leroy et al., 2005).

This data set was then used in the analyses presented in Komárek and Lesaffre (2006, 2008).

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

Prof. Emmanuel Lesaffre
Leuven Biostatistics and statistical Bioinformatics Centre (L-BioStat)
Katholieke Universiteit Leuven
Kapucijnenvoer 35
B-3000 Leuven
Belgium
<emmanuel.lesaffre@kuleuven.be>

Usage

```
data(tandmobRoos)
```

Format

a~data frame with 4\,394 rows and the following variables

IDNR identification number of a child

GENDER character *boy* or *girl*

DOB character, date of birth in the format DDmmmYY

PROVINCE factor, code of the province with

- **0 =** Antwerpen
- **1 =** Vlaams Brabant
- **2 =** Limburg
- **3 =** Oost Vlaanderen
- **4 =** West Vlaanderen

EDUC factor, code of the educational system with

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

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

EBEG.xx lower limit of the emergence (in years of age) of the permanent tooth xx. In contrast to `tandmob2`, the lower emergence limit for the permanent first molars that were originally left-censored, are adjusted according to the eruption stage (see Leroy, 2005 for more details).

xx takes values 16, 26, 36, 46 (permanent first molars).

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

xx takes values as for the variable `EBEG.xx`.

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

xx takes values as for the variable `EBEG.xx`.

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

xx takes values as for the variable EBEG.xx.

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

TOOTH.xx numeric, 0 or 1. Equal to 1 if the information concerning the permanent tooth was available, 0 if the permanent tooth xx was removed from the dataset by Kris.

xx takes values 16, 26, 36, 46.

These variables are almost useless for ordinary users.

Txxd numeric, 0 or 1. It is equal to 1 if the deciduous tooth xx was decayed, 0 otherwise.

xx takes values 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

Txxm numeric, 0 or 1. It is equal to 1 if the deciduous tooth xx was missing due to caries, 0 otherwise.

xx takes values 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

Txxf numeric, 0 or 1. It is equal to 1 if the deciduous tooth xx was filled, 0 otherwise.

xx takes values 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

Txxs numeric, 0 or 1. It is equal to 1 if the deciduous tooth xx was sound, 0 otherwise.

xx takes values 54, 64, 74, 84 (deciduous first molars), 55, 65, 75, 85 (deciduous second molars).

SEAL.xx numeric, 0 or 1. It is equal to 1 if the permanent first molar xx was sealed in pits and fissures (a form of protection), 0 otherwise.

xx takes values 16, 26, 36, 46 (permanent first molars).

FREQ.BR numeric, 0 or 1. It is equal to 1 if the child brushes daily the teeth, equal to 0 if he/she brushes less than once a day.

PLAQUE.xx.1 numeric, 0 or 1. It is equal to 1 if there was occlusal plaque in pits and fissures of the permanent tooth xx. It is equal to 0 if there was either no plaque present or the plaque was present on the total occlusal surface.

xx takes values 16, 26, 36, 46 (permanent first molars).

PLAQUE.xx.2 numeric, 0 or 1. It is equal to 1 if there was occlusal plaque on the total occlusal surface of the permanent tooth xx. It is equal to 0 if there was either no plaque present or the plaque was present only in pits and fissures.

xx takes values 16, 26, 36, 46 (permanent first molars).

Source

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

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

Data collection was supported by Unilever, Belgium. The Signal Tandmobiel project comprises the following partners: D. Declerck (Dental School, Catholic University Leuven), L. Martens (Dental School, University Ghent), J. Vanobbergen (Oral Health Promotion and Prevention, Flemish Dental

Association), P. Bottenberg (Dental School, University Brussels), E. Lesaffre (Biostatistical Centre, Catholic University Leuven), K. Hoppenbrouwers (Youth Health Department, Catholic University Leuven; Flemish Association for Youth Health Care).

References

Komárek, A. and Lesaffre, E. (2008). Bayesian accelerated failure time model with multivariate doubly-interval-censored data and flexible distributional assumptions. *Journal of the American Statistical Association*, **103**, 523–533.

Komárek, A. and Lesaffre, E. (2006). Bayesian semi-parametric accelerated failurew time model for paired doubly interval-censored data. *Statistical Modelling*, **6**, 3–22.

Leroy, R., Bogaerts, K., Lesaffre, E., and Declerck, D. (2005). Effect of caries experience in primary molars on cavity formation in the adjacent permanent first molar. *Caries Research*, **39**, 342–349.

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

`traceplot2` *Trace plot of MCMC output.*

Description

Displays a plot of iterations vs. sampled values for each variable in the chain, with a separate plot per variable.

This is slightly modified version of `traceplot` function of a `coda` package to conform to my personal preferences.

Usage

```
traceplot2(x, chains, bty = "n", main, xlab, ...)
```

Arguments

`x`	an `mcmc` or `mcmc.list` object.
`chains`	indeces of chains from the object that are to be plotted.
`bty, main, xlab, ...`	further arguments passed to the `plot.default` function.

Value

No return value.

Author(s)

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

vecr2matr *Transform single component indeces to double component indeces*

Description

Components of a bivariate G-spline can be indexed in several ways. Suppose that the knots in the first dimension are $\mu_{1,-K_1}, \ldots, \mu_{1,K_1}$ and the knots in the second dimension $\mu_{2,-K_2}, \ldots, \mu_{2,K_2}$. I.e. we have $2K_1 + 1$ knots in the first dimension and $2K_2 + 1$ knots in the second dimension. Each G-spline component can have a double index (k_1, k_2) assigned which means that it corresponds to the knot $(\mu_{1,k_1}, \mu_{2,k_2})$ or alternatively the same G-spline component can have a~single index

$$r = (k_2 + K_2) \times (2K_1 + 1) + k_1 + K_1 + 1$$

assigned where r takes values from $1, \ldots, K_1 \times K_2$. Single indexing is used for example by files `r.sim` and `r_2.sim` generated by functions `bayesHistogram`, `bayesBisurvreg`, `bayessurvreg2` to save some space.

This function serves to translate single indeces to double indeces using the relationship

$$k_1 = (r - 1) \bmod (2K_1 + 1) - K_1$$

$$k_2 = (r - 1) \operatorname{div} (2K_1 + 1) - K_2$$

The function can be used also in one dimensional case when a~simple relationship holds

$$r = k + K + 1$$

$$k = r - 1 - K$$

Usage

```
vecr2matr(vec.r, KK)
```

Arguments

`vec.r`	a~vector of single indeces
`KK`	a~vector with numbers of knots on each side of the central knot for each dimension of the G-spline. The length of `KK` determines dimension of the G-spline

Value

In bivariate case: a~matrix with two columns and as many rows as the length of `vec.r`.

In univariate case: a~vector with as ,amy components as the length of `vec.r`.

Author(s)

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

Examples

```
### Bivariate G-spline
### with 31 knots in each dimension
KK <- c(15, 15)

### First observation in component (-15, -15),
### second observation in component (15, 15),
### third observation in component (0, 0)
vec.r <- c(1, 961, 481)
vecr2matr(vec.r, KK)
```

Index