Package ‘rmsb’

April 13, 2025

Title Bayesian Regression Modeling Strategies

Version 1.1-2

Date 2025-04-13

Description
A Bayesian companion to the 'rms' package, 'rmsb' provides Bayesian model fitting, post-fit estimation, and graphics. It implements Bayesian regression models whose fit objects can be processed by 'rms' functions such as 'contrast()', 'summary()', 'Predict()', 'nomogram()', and 'latex()'. The fitting function currently implemented in the package is 'blrm()' for Bayesian logistic binary and ordinal regression with optional clustering, censoring, and departures from the proportional odds assumption using the partial proportional odds model of Peterson and Harrell (1990) <https://www.jstor.org/stable/2347760>.

License GPL (>= 3)

Encoding UTF-8

URL https://hbiostat.org/R/rmsb/

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.4.0), rms (>= 8.0-0)

Imports methods, Rcpp (>= 0.12.0), rstan (>= 2.26.23), Hmisc (>= 5.2-3), survival (>= 3.1-12), ggplot2, MASS, cluster, digest, knitr, loo

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

Suggests cmdstanr, bayesplot, mice

Additional_repositories https://mc-stan.org/r-packages/

SystemRequirements GNU make

NeedsCompilation yes

Author Frank Harrell [aut, cre] (<https://orcid.org/0000-0002-8271-5493>),
Ben Goodrich [ctb] (contributed Stan code),
Ben Bolker [ctb] (wrote original code that is folded into the

pdensityContour function),
Doug Bates [ctb] (write original code for highest posterior density
interval that is folded into the HPDint function)

Maintainer Frank Harrell <fh@fharrell.com>

Repository CRAN

Date/Publication 2025-04-13 16:30:02 UTC

Contents

`rmsb-package` *The 'rmsb' package.*

Description

Regression Modeling Strategies Bayesian

The **rmsb** package is an appendage to the **rms** package that implements Bayesian regression models whose fit objects can be processed by **rms** functions such as `contrast`, `summary`, `Predict`, `nomogram`, and `latex`. The fitting function currently implemented in the package is `blrm` for Bayesian logistic binary and ordinal regression with optional clustering, censoring, and departures from the proportional odds assumption using the partial proportional odds model of Peterson and Harrell (1990).

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.19.3. https://mc-stan.org

See Also

- `https://hbiostat.org/R/rmsb/` for the package's main web page
- `https://hbiostat.org/R/examples/blrm/blrm.html` for a vignette with many examples of using the `blrm` function

`blrm` *Bayesian Binary and Ordinal Logistic Regression*

Description

Uses `rstan` with pre-compiled Stan code, or `cmdstan` to get posterior draws of parameters from a binary logistic or proportional odds semiparametric ordinal logistic model. The Stan code internally using the qr decompositon on the design matrix so that highly collinear columns of the matrix do not hinder the posterior sampling. The parameters are transformed back to the original scale before returning results to R. Design matrix columns are centered before running Stan, so Stan diagnostic output will have the intercept terms shifted but the results of `blrm()` for intercepts are for the original uncentered data. The only prior distributions for regression betas are normal with mean zero. Priors are specified on contrasts so that they can be specified on a meaningful scale and so that more complex patterns can be imposed. Parameters that are not involved in any contrasts in `pcontrast` or `npcontrast` have non-informative priors. Contrasts are automatically converted to the QR space used in Stan code.

Usage

```
blrm(
  formula,
  ppo = NULL,
  cppo = NULL,
  data = environment(formula),
  subset,
  na.action = na.delete,
  priorsdppo = rep(100, pppo),
  iprior = 0,
  conc = 1/(0.8 + 0.35 * max(k, 3)),
  ascale = 1,
  psigma = 1,
  rsdmean = if (psigma == 1) 0 else 1,
  rsdsd = 1,
  normcppo = FALSE,
  pcontrast = NULL,
  npcontrast = NULL,
  backend = c("rstan", "cmdstan"),
  iter = 2000,
  warmup = iter/2,
  chains = 4,
  refresh = 0,
  progress = if (refresh > 0) "stan-progress.txt" else "",
  x = TRUE,
  y = TRUE,
  loo = n <= 1000,
  ppairs = NULL,
  method = c("both", "sampling", "optimizing"),
  inito = if (length(ppo)) 0 else "random",
  inits = inito,
  standata = FALSE,
  file = NULL,
  debug = FALSE,
  sampling.args = NULL,
  showopt = FALSE,
  ...
)
```

Arguments

`formula`	a R formula object that can use `rms` package enhancements such as the restricted interaction operator
`ppo`	formula specifying the model predictors for which proportional odds is not assumed
`cppo`	a function that if present causes a constrained partial PO model to be fit. The function specifies the values in the Gamma vector in Peterson and Harrell (1990)

equation (6). Sometimes to make posterior sampling better behaved, the function should be scaled and centered. This is done by wrapping `cppo` in a function that scales the `cppo` result before returning the vector value, when `normcppo` is `TRUE`. The default normalization is based on the mean and standard deviation of the function values over the distribution of observed Y. For getting predicted values and estimates post-`blrm()`, `cppo` must not reference any functions that are not available at such later times.

`data`	a data frame; defaults to using objects from the calling environment
`subset`	a logical vector or integer subscript vector specifying which subset of data whould be used
`na.action`	default is `na.delete` to remove missings and report on them
`priorsdppo`	vector of prior standard deviations for non-proportional odds parameters. The last element is the only one for which the SD corresponds to the original data scale. This only applies to the unconstrained PPO model.
`iprior`	specifies whether to use a Dirichlet distribution for the cell probabilities, which induce a more complex prior distribution for the intercepts (`iprior=0`, the default), non-informative priors (`iprior=1`) directly on the intercept parameters, or to directly use a t-distribution with 3 d.f. and scale parameter `ascale` (`iprior=2`).
`conc`	the Dirichlet distribution concentration parameter for the prior distribution of cell probabilities at covariate means. The default is the reciprocal of 0.8 + 0.35 max(k, 3) where k is the number of Y categories. The default is chosen to make the posterior mean of the intercepts more closely match the MLE. For optimizing, the concentration parameter is always 1.0 when `iprior=0` to obtain results very close to the MLE for providing the posterior mode.
`ascale`	scale parameter for the t-distribution for priors for the intercepts if `iprior=2`, defaulting to 1.0
`psigma`	defaults to 1 for a half-t distribution with 4 d.f., location parameter `rsdmean` and scale parameter `rsdsd`. Set `psigma=2` to use the exponential distribution.
`rsdmean`	the assumed mean of the prior distribution of the standard deviation of random effects. When `psigma=2` this is the mean of an exponential distribution and defaults to 1. When `psigma=1` this is the mean of the half-t distribution and defaults to zero.
`rsdsd`	applies only to `psigma=1` and is the scale parameter for the half t distribution for the SD of random effects, defaulting to 1.
`normcppo`	set to `TRUE` to modify the `cppo` function automatically centering and scaling the result
`pcontrast`	a list specifying contrasts that are to be given Gaussian prior distributions. The predictor combinations specified in `pcontrast` are run through `rms::gendata()` so that contrasts are specified in units of original variables, and unspecified variables are set to medians or modes as saved by `rms::datadist()`. Thanks to `Stan`, putting priors on combinations and transformations of model parameters has the same effect of putting different priors on the original parameters without figuring out how to do that. The syntax used here allows specification of differences, double differences (e.g., interactions or nonlinearity), triple differences (e.g., to put contraints on nonlinear interactions), etc. The requested predictor

	combinations must be named so they may be referred to inside `contrast`. The syntax is `pcontrast=list(..., contrast=expression(...), mu=, sd=, weights=, expand=)`. `...` denotes one or more `list()`s with predictor combinations, and each `list()` must be named, e.g., `pcontrast=list(c1=list(sex='female'), c2=list(sex='male'))` to set up for a `female - male` contrast specified as `contrast=expression(c1 - c2)`. The `c1 - c2` subtraction will operate on the design matrices generated by the covariate settings in the `list()`s. For `weights, expand` see `rms::Xcontrast()` and `rms::contrast.rms()`. `mu` is a vector of prior means associated with the rows of the stacked contrasts, and `sd` is a corresponding vector of Gaussian prior SDs. When `mu` is not given it defaults to 0.0, and `sd` defaults to 100.0. Values of `mu` and/or `sd` are repeated to the number of contrasts if they are of length 1. Full examples are given here.
`npcontrast`	like `pcontrast` but applies to the non-proportional odds submodel in ppo. Priors for the amount of departure from proportional odds are isolated from the priors of the "main effects" in `formula`. The mean and standard deviation for the non-PO contrasts are on the scale of Z*tau before `cppo` is applied. If `cppo` picks off a single condition, i.e., death is the highest level of Y and you want a special effect of treatment on death, then `cppo` will be something like `function(y) y == 4` and the contrast prior will be on the scale of the additional treatment effect for death. If `cppo` is more of a continuous function you will have to take into account the values of that function when figuring the prior mean and SD. For example, if y ranges from 10-90 and `cppo` is `sqrt(y)`, and you want to specify a prior on the log odds ratio for y=10 vs. y=90 you'll need to divide the prior standard deviation in `npcontrast` by `sqrt(90) - sqrt(10)`.
`backend`	set to `cmdstan` to use `cmdstan` through the R `cmdstanr` package instead of the default `rstan`. You can also specify this with a global option `rmsb.backend`.
`iter`	number of posterior samples per chain for `rstan::sampling()` to run, counting warmups
`warmup`	number of warmup iterations to discard. Default is `iter`/2.
`chains`	number of separate chains to run
`refresh`	see `rstan::sampling()` and `cmdstanr::sample()`. The default is 0, indicating that no progress notes are output. If `refresh > 0` and `progress` is not `''`, progress output will be appended to file `progress`. The default file name is `'stan-progress.txt'`.
`progress`	see `refresh`. Defaults to `''` if `refresh = 0`. Note: If running interactively but not under RStudio, `rstan` will open a browser window for monitoring progress.
`x`	set to `FALSE` to not store the design matrix in the fit. `x=TRUE` is needed if running `blrmStats` for example.
`y`	set to `FALSE` to not store the response variable in the fit
`loo`	set to `FALSE` to not run `loo` and store its result as object `loo` in the returned object. `loo` defaults to `FALSE` if the sample size is greater than 1000, as `loo` requires the per-observation likelihood components, which creates a matrix N times the number of posterior draws.
`ppairs`	set to a file name to run `rstan` `pairs` or, if `backend='cmdstan'` `bayesplot::mcmc_pairs` and store the resulting png plot there. Set to `TRUE` instead to directly plot these diagnostics. The default is not to run pair plots.

`method`	set to `'optimizing'` to run the Stan optimizer and not do posterior sampling, `'both'` (the default) to run both the optimizer and posterior sampling, or `'sampling'` to run only the posterior sampling and not compute posterior modes. Running `optimizing` is a way to obtain maximum likelihood estimates and allows one to quickly study the effect of changing the prior distributions. When `method='optimizing'` is used the result returned is not a standard `blrm()` object but is instead the parameter estimates, -2 log likelihood, and optionally the Hession matrix (if you specify `hessian=TRUE` in ...; not available with `cmdstan`). When `method='both'` is used, `rstan::sampling()` and `rstan::optimizing()` are both run, and parameter estimates (posterior modes) from `optimizing` are stored in a matrix `param` in the fit object, which also contains the posterior means and medians, and other results from `optimizing` are stored in object `opt` in the `blrm()` fit object. When random effects are present, `method` is automatically set to `'sampling'` as maximum likelihood estimates without marginalizing over the random effects do not make sense. When you specify `method='optimizing'` specify `iprior=` to get regular MLEs in which no prior is put on the intercepts.
`inito`	intial value for optimization. The default is the `rstan` default `'random'`. Frequently specifying `init=0` will benefit when the number of distinct Y categories grows or when using ppo hence 0 is the default for that.
`inits`	initial value for sampling, defaults to `inito`
`standata`	set to `TRUE` to return the Stan data list and not run the model
`file`	a file name for a `saveRDS`-created file containing or to contain the saved fit object. If `file` is specified and the file does not exist, it will be created right before the fit object is returned, less the large `rstan` object. If the file already exists, its stored `md5` hash string `datahash` fit object component is retrieved and compared to that of the current `rstan` inputs. If the data to be sent to `rstan`, the priors, and all sampling and optimization options and stan code are identical, the previously stored fit object is immediately returned and no new calculatons are done.
`debug`	set to `TRUE` to output timing and progress information to /tmp/debug.txt
`sampling.args`	a list containing parameters to pass to `rstan::sampling()` or to the `rcmdstan` `sample` function, other than these arguments: `iter, warmup, chains, refresh, init` which are already arguments to `blrm`. A good use of this is `sampling.args=list(seed=3)` to get reproducible sampling.
`showopt`	set to `TRUE` to show Stan optimizer output
`...`	passed to `rstan::optimizing()` or the `rcmdstan` optimizing function. `sampling.args` is usually used instead.

Details

The partial proportional odds model of Peterson and Harrell (1990) is implemented, and is invoked when the user specifies a second model formula as the `ppo` argument. This formula has no left-hand-side variable, and has right-side variables that are a subset of those in `formula` specifying for which predictors the proportional odds assumption is relaxed.

The Peterson and Harrell (1990) constrained partial proportional odds is also implemented, and is usually preferred to the above unconstrained PPO model as it adds a vector of coefficients instead of a matrix of coefficients. In the constrained PPO model the user provides a function `cppo` that computes a score for all observed values of the dependent variable. For example with a discrete ordinal outcome cppo may return a value of 1.0 for a specific value of Y and zero otherwise. That will result in a departure from the proportional odds assumption for just that one level of Y. The value returned by cppo at the lowest Y value is never used in any case.

blrm() also handles single-level hierarchical random effects models for the case when there are repeated measurements per subject which are reflected as random intercepts, and a different experimental model that allows for AR(1) serial correlation within subject. For both setups, a cluster term in the model signals the existence of subject-specific random effects.

When using the cmdstan backend, cmdstanr will need to compile the Stan code once per computer, only recompiling the code when the Stan source code changes. By default the compiled code is stored in directory .rmsb under your home directory. Specify options(rmsbdir=) to specify a different location. You should specify rmsbdir to be in a project-specific location if you want to archive code for old projects.

If you want to run MCMC sampling even when no inputs or Stan code have changed, i.e., to use a different random number seed for the sampling process when you did not specify sampling.args(seed=...), remove the file before running blrm.

Set options(rmsbmsg=FALSE) to suppress certain information messages.

See here and here for multiple examples with results.

Value

an rms fit object of class blrm, rmsb, rms that also contains rstan or cmdstanr results under the name rstan. In the rstan results, which are also used to produce diagnostics, the intercepts are shifted because of the centering of columns of the design matrix done by blrm(). With method='optimizing' a class-less list is return with these elements: coefficients (MLEs), beta (non-intercept parameters on the QR decomposition scale), deviance (-2 log likelihood), return_code (see rstan::optimizing()), and, if you specified hessian=TRUE to blrm(), the Hessian matrix. To learn about the scaling of orthogonalized QR design matrix columns, look at the xqrsd object in the returned object. This is the vector of SDs for all the columns of the transformed matrix. The returned element sampling_time is the elapsed time for running posterior samplers, in seconds. This will be just a little more than the time for running one CPU core for one chain.

Author(s)

Frank Harrell and Ben Goodrich

See Also

print.blrm(), blrmStats(), stanDx(), stanGet(), coef.rmsb(), vcov.rmsb(), print.rmsb(), coef.rmsb()

Examples

```
## Not run:
  getHdata(titanic3)
  dd <- datadist(titanic3); options(datadist='dd')
  f <- blrm(survived ~ (rcs(age, 5) + sex + pclass)^2, data=titanic3)
  f                   # model summary using print.blrm
  coef(f)             # compute posterior mean parameter values
  coef(f, 'median')   # compute posterior median values
  stanDx(f)           # print basic Stan diagnostics
  s <- stanGet(f)     # extract rstan object from fit
  plot(s, pars=f$betas)       # Stan posteriors for beta parameters
  stanDxplot(s)       # Stan diagnostic plots by chain
  blrmStats(f)        # more details about predictive accuracy measures
  ggplot(Predict(...))   # standard rms output
  summary(f, ...)     # invokes summary.rms
  contrast(f, ...)    # contrast.rms computes HPD intervals
  plot(nomogram(f, ...)) # plot nomogram using posterior mean parameters

  # Fit a random effects model to handle multiple observations per
  # subject ID using cmdstan
  # options(rmsb.backend='cmdstan')
  f <- blrm(outcome ~ rcs(age, 5) + sex + cluster(id), data=mydata)

## End(Not run)
```

blrmStats *Compute Indexes of Predictive Accuracy and Their Uncertainties*

Description

For a binary or ordinal logistic regression fit from `blrm()`, computes several indexes of predictive accuracy along with highest posterior density intervals for them. Optionally plots their posterior densities. When there are more than two levels of the outcome variable, computes Somers' Dxy and c-index on a random sample of 10,000 observations.

Usage

```
blrmStats(fit, ns = 400, prob = 0.95, pl = FALSE, dist = c("density", "hist"))
```

Arguments

`fit`	an object produced by `blrm()`
`ns`	number of posterior draws to use in the calculations (default is 400)
`prob`	HPD interval probability (default is 0.95)
`pl`	set to `TRUE` to plot the posterior densities using base graphics
`dist`	if `pl` is `TRUE` specifies whether to plot the density estimate (the default) or a histogram

Value

list of class `blrmStats` whose most important element is `Stats`. The indexes computed are defined below, with gp, B, EV, and vp computed using the intercept corresponding to the median value of Y. See `https://fharrell.com/post/addvalue` for more information.

"Dxy" Somers' Dxy rank correlation between predicted and observed. The concordance probability (c-index; AUROC in the binary Y case) may be obtained from the relationship Dxy=2(c-0.5).

"g" Gini's mean difference: the average absolute difference over all pairs of linear predictor values

"gp" Gini's mean difference on the predicted probability scale

"B" Brier score

"EV" explained variation

"v" variance of linear predictor

"vp" variable of estimated probabilities

Author(s)

Frank Harrell

See Also

`Hmisc::rcorr.cens()`

Examples

```
## Not run:
  f <- blrm(...)
  blrmStats(f, pl=TRUE)   # print and plot

## End(Not run)
```

`cluster` *cluster*

Description

Cluster Function for Random Effects

Usage

```
cluster(x)
```

Arguments

`x`	a vector representing a categorical vector

Details

Used by `blrm` to signal a categorical variable to generate random effects.

Value

x unchanged

Author(s)

Frank Harrell

`coef.rmsb` *Extract Bayesian Summary of Coefficients*

Description

Computes either the posterior mean (default), posterior median, or posterior mode of the parameters in an `rms` Bayesian regression model

Usage

```
## S3 method for class 'rmsb'
coef(object, stat = c("mean", "median", "mode"), ...)
```

Arguments

`object`	an object created by an `rms` package Bayesian fitting function
`stat`	name of measure of posterior distribution central tendency to compute
`...`	ignored

Value

a vector of intercepts and regression coefficients

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(...)
  coef(f, stat='mode')

## End(Not run)
```

compareBmods *Compare Bayesian Model Fits*

Description

Uses `loo::loo_model_weights()` to compare a series of models such as those created with `blrm()`

Usage

```
compareBmods(..., method = "stacking", r_eff_list = NULL)
```

Arguments

`...`	a series of model fits
`method`	see `loo::loo_model_weights()`
`r_eff_list`	see `loo::loo_model_weights()`

Value

a `loo::loo_model_weights()` object

Author(s)

Frank Harrell

distSym *Distribution Symmetry Measure*

Description

From a sample from a distribution computes a symmetry measure. By default it is the gap between the mean and the 0.95 quantile divided by the gap between the 0.05 quantile and the mean.

Usage

```
distSym(x, prob = 0.9, na.rm = FALSE)
```

Arguments

`x`	a numeric vector representing a sample from a continuous distribution
`prob`	quantile interval coverage
`na.rm`	set to `TRUE` to remove `NA`s before proceeding.

Value

a scalar with a value of 1.0 indicating symmetry

Author(s)

Frank Harrell

`ExProb.blrm` *Function Generator for Exceedance Probabilities for* `blrm()`

Description

For a `blrm()` object generates a function for computing the estimates of the function Prob(Y>=y) given one or more values of the linear predictor using the reference (median) intercept. This function can optionally be evaluated at only a set of user-specified `y` values, otherwise a right-step function is returned. There is a plot method for plotting the step functions, and if more than one linear predictor was evaluated multiple step functions are drawn. `ExProb` is especially useful for `nomogram()`. The linear predictor argument is a posterior summarized linear predictor lp (e.g. using posterior mean of intercepts and slopes) computed at the reference intercept. `lptau` must be provided when call the created function if the model is a partial proportional odds model.

Usage

```
## S3 method for class 'blrm'
ExProb(object, posterior.summary = c("mean", "median"), ...)
```

Arguments

`object`	a `blrm()` fit
`posterior.summary`	defaults to posterior mean; may also specify `"median"`. Must be consistent with the summary used when creating `lp`.
`...`	ignored

Value

an R function

Author(s)

Frank Harrell

`getParamCoef` *Get a Bayesian Parameter Vector Summary*

Description

Retrieves posterior mean, median, or mode (if available)

Usage

```
getParamCoef(
  fit,
  posterior.summary = c("mean", "median", "mode"),
  what = c("both", "betas", "taus")
)
```

Arguments

`fit`	a Bayesian model fit from `rmsb`
`posterior.summary`	which summary statistic (Bayesian point estimate) to fetch
`what`	specifies which coefficients to include. Default is all. Specify `what="betas"` to include only intercepts and betas if the model is a partial proportional odds model (i.e.,, exclude the tau parameters). Specify `what="taus"` to include only the tau parameters.

Value

vector of regression coefficients

Author(s)

Frank Harrell

`HPDint` *Highest Posterior Density Interval*

Description

Adapts code from `coda::HPDinterval()` to compute a highest posterior density interval from posterior samples for a single parameter. Quoting from the coda help file, for each parameter the interval is constructed from the empirical cdf of the sample as the shortest interval for which the difference in the ecdf values of the endpoints is the nominal probability. Assuming that the distribution is not severely multimodal, this is the HPD interval.

Usage

```
HPDint(x, prob = 0.95)
```

Arguments

`x`	a vector of posterior draws
`prob`	desired probability coverage

Value

a 2-vector with elements `Lower` and `Upper`

Author(s)

Douglas Bates and Frank Harrell

`Mean.blrm` *Function Generator for Mean Y for* `blrm()`

Description

Creates a function to turn a posterior summarized linear predictor lp (e.g. using posterior mean of intercepts and slopes) computed at the reference intercept into e.g. an estimate of mean Y using the posterior mean of all the intercept. `lptau` must be provided when call the created function if the model is a partial proportional odds model.

Usage

```
## S3 method for class 'blrm'
Mean(object, codes = FALSE, posterior.summary = c("mean", "median"), ...)
```

Arguments

`object`	a `blrm()` fit
`codes`	if `TRUE`, use the integer codes $1, 2, \ldots, k$ for the k-level response in computing the predicted mean response.
`posterior.summary`	defaults to posterior mean; may also specify `"median"`. Must be consistent with the summary used when creating `lp`.
`...`	ignored

Value

an R function

Author(s)

Frank Harrell

`pdensityContour` *Bivariate Posterior Contour*

Description

Computes coordinates of a highest density contour containing a given probability volume given a sample from a continuous bivariate distribution, and optionally plots. The default method assumes an elliptical shape, but one can optionally use a kernel density estimator. Code adapted from `embbook::HPDregionplot`. See https://www.sumsar.net/blog/2014/11/how-to-summarize-a-2d-posterior-usin

Usage

```
pdensityContour(
  x,
  y,
  method = c("ellipse", "kernel"),
  prob = 0.95,
  otherprob = c(0.01, 0.1, 0.25, 0.5, 0.75, 0.9),
  h = c(1.3 * MASS::bandwidth.nrd(x), 1.3 * MASS::bandwidth.nrd(y)),
  n = 70,
  pl = FALSE
)
```

Arguments

`x`	a numeric vector
`y`	a numeric vector the same length of x
`method`	defaults to `'ellipse'`, can be set to `'kernel'`
`prob`	main probability coverage (the only one for `method='ellipse'`)
`otherprob`	vector of other probability coverages for `method='kernel'`
`h`	vector of bandwidths for x and y. See `MASS::kde2d()`.
`n`	number of grid points in each direction, defaulting to normal reference bandwidth (see `bandwidth.nrd`).
`pl`	set to `TRUE` to plot contours

Value

a 2-column matrix with x and y coordinates unless `pl=TRUE` in which case a `ggplot2` graphic is returned

Author(s)

Ben Bolker and Frank Harrell

`plot.PostF` *Plot Posterior Density of* `PostF`

Description

Computes highest posterior density and posterior mean and median as vertical lines, and plots these on the density function. You can transform the posterior draws while plotting.

Usage

```
## S3 method for class 'PostF'
plot(
  x,
  ...,
  cint = 0.95,
  label = NULL,
  type = c("linetype", "facet"),
  ltitle = ""
)
```

Arguments

`x`	result of running a function created by `PostF`
`...`	other results created by such functions
`cint`	interval probability
`label`	x-axis label if not the expression originally evaluated. When more than one result is plotted, `label` is a vector of character strings, one for each result.
`type`	when plotting more than one result specifies whether to make one plot distinguishing results by line type, or whether to make separate panels
`ltitle`	used of `type='linetype'` to specify name of legend for the line types

Value

`ggplot2` object

Author(s)

Frank Harrell

plot.rmsb *Plot Posterior Densities and Summaries*

Description

For an `rms` Bayesian fit object, plots posterior densities for selected parameters along with posterior mode, mean, median, and highest posterior density interval. If the fit was produced by `stackMI` the density represents the distribution after stacking the posterior draws over imputations, and the per-imputation density is also drawn as pale curves. If exactly two parameters are being plotted and `bivar=TRUE`, hightest bivariate posterior density contrours are plotted instead, for a variety of `prob` values including the one specified, using

Usage

```
## S3 method for class 'rmsb'
plot(
  x,
  which = NULL,
  nrow = NULL,
  ncol = NULL,
  prob = 0.95,
  bivar = FALSE,
  bivarmethod = c("ellipse", "kernel"),
  ...
)
```

Arguments

`x`	an `rms` Bayesian fit object
`which`	names of parameters to plot, defaulting to all non-intercepts. Can instead be a vector of integers.
`nrow`	number of rows of plots
`ncol`	number of columns of plots
`prob`	probability for HPD interval
`bivar`	set to `TRUE` to plot bivariate density contours instead of univariate results (ignored if the number of parameters plotted is not exactly two)
`bivarmethod`	passed as `method` argument to `pdensityContour`
`...`	passed to `pdensityContour`

Value

`ggplot2` object

Author(s)

Frank Harrell

`PostF` *Function Generator for Posterior Probabilities of Assertions*

Description

From a Bayesian fit object such as that from `blrm()` generates an R function for evaluating the probability that an assertion is true. The probability, within simulation error, is the proportion of times the assertion is true over the posterior draws. If the assertion does not evaluate to a logical or 0/1 quantity, it is taken as a continuous derived parameter and the vector of draws for that parameter is returned and can be passed to the `PostF` plot method. `PostF` can also be used on objects created by `contrast.rms`

Usage

```
PostF(fit, name = c("short", "orig"), pr = FALSE)
```

Arguments

`fit`	a Bayesian fit or `contrast.rms` object
`name`	specifies whether assertions will refer to shortened parameter names (the default) or original names. Shorted names are of the form `a1, ..., ak` where `k` is the number of intercepts in the model, and `b1, ..., bp` where `p` is the number of non-intercepts. When using original names that are not legal R variable names, you must enclose them in backticks. For `contrast` objects, `name` is ignored and you must use contrast names. The `cnames` argument to `contrast.rms` is handy for assigning your own names.
`pr`	set to `TRUE` to have a table of short names and original names printed when `name='short'`. For `contrasts` the contrast names are printed if `pr=TRUE`.

Value

an R function

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(y ~ age + sex)
  P <- PostF(f)
  P(b2 > 0)     # Model is a1 + b1*age + b2*(sex == 'male')
  P(b1 < 0 & b2 > 0)   # Post prob of a compound assertion
  # To compute probabilities using original parameter names:
  P <- PostF(f, name='orig')
  P(age < 0)    # Post prob of negative age effect
  P(`sex=male` > 0)
  f <- blrm(y ~ sex + pol(age, 2))
  P <- PostF(f)
  # Compute and plot posterior density of the vertex of the
  # quadratic age effect
  plot(P(-b2 / (2 * b3)))

  # The following would be useful in age and sex interacted
  k <- contrast(f, list(age=c(30, 50), sex='male'),
                   list(age=c(30, 50), sex='female'),
                cnames=c('age 30 M-F', 'age 50 M-F'))
  P <- PostF(k)
  P(`age 30 M-F` > 0 & `age 50 M-F` > 0)
##'
## End(Not run)
```

`predict.blrm` *Make predictions from a* `blrm()` *fit*

Description

Predict method for `blrm()` objects

Usage

```
## S3 method for class 'blrm'
predict(
  object,
  ...,
  kint = NULL,
  ycut = NULL,
  zcppo = TRUE,
  Zmatrix = TRUE,
  fun = NULL,
  funint = TRUE,
 type = c("lp", "fitted", "fitted.ind", "mean", "x", "data.frame", "terms", "cterms",
   "ccterms", "adjto", "adjto.data.frame", "model.frame"),
  se.fit = FALSE,
  codes = FALSE,
  posterior.summary = c("mean", "median", "all"),
  cint = 0.95
)
```

Arguments

`object, ..., type, se.fit, codes`
see `rms::predict.lrm()`

`kint`	This is only useful in a multiple intercept model such as the ordinal logistic model. There to use to second of three intercepts, for example, specify `kint=2`. The default is the middle intercept corresponding to the median `y`. You can specify `ycut` instead, and the intercept corresponding to Y >= `ycut` will be used for `kint`.
`ycut`	for an ordinal model specifies the Y cutoff to use in evaluating departures from proportional odds, when the constrained partial proportional odds model is used. When omitted, `ycut` is implied by `kint`. The only time it is absolutely mandatory to specify `ycut` is when computing an effect (e.g., odds ratio) at a level of the response variable that did not occur in the data. This would only occur when the cppo function given to `blrm` is a continuous function. If `type='x'` and neither `kint` nor `ycut` are given, the partial PO part of the design matrix is not multiplied by the cppo function. If `type='x'`, the number of predicted observations is 1, `ycut` is longer than 1, and `zcppo` is `TRUE`, predictions are duplicated to the length of `ycut` as it is assumed that the user wants to see the effect of varying `ycut`, e.g., to see cutoff-specific odds ratios.
`zcppo`	applies only to `type='x'` for a constrained partial PO model. Set to `FALSE` to prevent multiplication of Z matrix by `cppo(ycut)`.
`Zmatrix`	set to `FALSE` to exclude the partial PO Z matrix from the returned design matrix if `type='x'`
`fun`	a function to evaluate on the linear predictor, e.g. a function created by `Mean.blrm()` or `Quantile.blrm()`
`funint`	set to `FALSE` if `fun` is not a function such as the result of `Mean.blrm()`, `Quantile.blrm()`, or `ExProb.blrm()` that contains an `intercepts` argument
`posterior.summary`	set to `'median'` or `'mode'` to use posterior median/mode instead of mean. For some `type`s set to `'all'` to compute the needed quantity for all posterior draws, and return one more dimension in the array.
`cint`	probability for highest posterior density interval. Set to `FALSE` to suppress calculation of the interval.

Value

a data frame, matrix, or vector with posterior summaries for the requested quantity, plus an attribute `'draws'` that has all the posterior draws for that quantity. For `type='fitted'` and `type='fitted.ind'` this attribute is a 3-dimensional array representing draws x observations generating predictions x levels of Y.

Author(s)

Frank Harrell

See Also

`rms::predict.lrm()`

Examples

```
## Not run:
  f <- blrm(...)
  predict(f, newdata, type='...', posterior.summary='median')

## End(Not run)
```

---

print.blrm *Print* blrm() *Results*

Description

Prints main results from blrm() along with indexes and predictive accuracy and their highest posterior density intervals computed from blrmStats.

Usage

```
## S3 method for class 'blrm'
print(
  x,
  dec = 4,
  coefs = TRUE,
  intercepts = x$non.slopes < 10,
  prob = 0.95,
  ns = 400,
  title = NULL,
  ...
)
```

Arguments

x	object created by blrm()
dec	number of digits to print to the right of the decimal
coefs	specify FALSE to suppress printing parameter estimates, and in integer k to print only the first k
intercepts	set to FALSE to suppress printing intercepts. Default is to print them unless there are more than 9.
prob	HPD interval probability for summary indexes
ns	number of random samples of the posterior draws for use in computing HPD intervals for accuracy indexes
title	title of output, constructed by default
...	passed to prModFit

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(...)
  options(lang='html')   # default is lang='plain'; also can be latex
  f               # print using defaults
  print(f, posterior.summary='median')   # instead of post. means

## End(Not run)
```

---

print.blrmStats — *Print Details for* `blrmStats` *Predictive Accuracy Measures*

Description

Prints results of `blrmStats` with brief explanations

Usage

```
## S3 method for class 'blrmStats'
print(x, dec = 3, ...)
```

Arguments

`x`	an object produced by `blrmStats`
`dec`	number of digits to round indexes
`...`	ignored

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(...)
  s <- blrmStats(...)
  s    # print with defaults
  print(s, dec=4)

## End(Not run)
```

print.predict.blrm *Print Predictions for* blrm()

Description

Prints the summary portion of the results of `predict.blrm`

Usage

```
## S3 method for class 'predict.blrm'
print(x, digits = 3, ...)
```

Arguments

`x`	result from `predict.blrm`
`digits`	number of digits to round numeric results
`...`	ignored

Author(s)

Frank Harrell

print.rmsb *Basic Print for Bayesian Parameter Summary*

Description

For a Bayesian regression fit prints the posterior mean, median, SE, highest posterior density interval, and symmetry coefficient from the posterior draws. For a given parameter, the symmetry measure is computed using the `distSym` function.

Usage

```
## S3 method for class 'rmsb'
print(x, prob = 0.95, dec = 4, intercepts = TRUE, pr = TRUE, ...)
```

Arguments

`x`	an object created by an `rms` Bayesian fitting function
`prob`	HPD interval coverage probability (default is 0.95)
`dec`	amount of rounding (digits to the right of the decimal)
`intercepts`	set to `FALSE` to not print intercepts
`pr`	set to `FALSE` to return an unrounded matrix and not print
`...`	ignored

Value

matrix (rounded if `pr=TRUE`)

Author(s)

Frank Harrell

Examples

```
## Not run: 
  f <- blrm(...)
  print.rmsb(f)

## End(Not run)
```

`Quantile.blrm` *Function Generator for Quantiles of Y for* `blrm()`

Description

Creates a function to turn a posterior summarized linear predictor lp (e.g. using posterior mean of intercepts and slopes) computed at the reference intercept into e.g. an estimate of a quantile of Y using the posterior mean of all the intercepts. `lptau` must be provided when call the created function if the model is a partial proportional odds model.

Usage

```
## S3 method for class 'blrm'
Quantile(object, codes = FALSE, posterior.summary = c("mean", "median"), ...)
```

Arguments

`object`	a `blrm()` fit
`codes`	if `TRUE`, use the integer codes $1, 2, \ldots, k$ for the k-level response in computing the quantile
`posterior.summary`	defaults to posterior mean; may also specify `"median"`. Must be consistent with the summary used when creating `lp`.
`...`	ignored

Value

an R function

Author(s)

Frank Harrell

selectedQr *QR Decomposition Preserving Selected Columns*

Description

Runs a matrix through the QR decomposition and returns the transformed matrix and the forward and inverse transforming matrices `R, Rinv`. If columns of the input matrix `X` are centered the QR transformed matrix will be orthogonal. This is helpful in understanding the transformation and in scaling prior distributions on the transformed scale. `not` can be specified to keep selected columns as-is. `cornerQr` leaves the last column of `X` alone (possibly after centering). When `not` is specified, the square transforming matrices have appropriate identity submatrices inserted so that recreation of original `X` is automatic.

Usage

```
selectedQr(X, not = NULL, corner = FALSE, center = TRUE)
```

Arguments

`X`	a numeric matrix
`not`	an integer vector specifying which columns of `X` are to be kept with their original values
`corner`	set to `FALSE` to not treat the last column specially. You may not specify both `not` and `corner`.
`center`	set to `FALSE` to not center columns of `X` first

Value

list with elements `X, R, Rinv, xbar` where `xbar` is the vector of means (vector of zeros if `center=FALSE`)

Author(s)

Ben Goodrich and Frank Harrell

Examples

```
x <- 1 : 10
X <- cbind(x, x^2)
w <- selectedQr(X)
w
with(w, X %*% R)  # = scale(X, center=TRUE, scale=FALSE)
Xqr <- w$X
plot(X[, 1], Xqr[, 1])
plot(X[, 1], Xqr[, 2])
cov(X)
cov(Xqr)
```

```
X <- cbind(x, x^3, x^4, x^2)
w <- selectedQr(X, not=2:3)
with(w, X %*% R)
```

---

stackMI *Bayesian Model Fitting and Stacking for Multiple Imputation*

Description

Runs an rmsb package Bayesian fitting function such as blrm separately for each completed dataset given a multiple imputation result such as one produced by Hmisc::aregImpute. Stacks the posterior draws and diagnostics across all imputations, and computes parameter summaries on the stacked posterior draws.

Usage

```
stackMI(
  formula,
  fitter,
  xtrans,
  data = NULL,
  n.impute = xtrans$n.impute,
  dtrans = NULL,
  derived = NULL,
  subset = NULL,
  refresh = 0,
  progress = if (refresh > 0) "stan-progress.txt" else "",
  file = NULL,
  ...
)
```

Arguments

formula	a model formula
fitter	a Bayesian fitter
xtrans	an object created by transcan, aregImpute, or mice
data	data frame
n.impute	number of imputations to run, default is the number saved in xtrans
dtrans	see Hmisc::fit.mult.impute
derived	see Hmisc::fit.mult.impute
subset	an integer or logical vector specifying the subset of observations to fit
refresh	see rstan::sampling. The default is 0, indicating that no progress notes are output. If refresh > 0 and progress is not '', progress output will be appended to file progress. The default file name is 'stan-progress.txt'.

`progress`	see `refresh`. Defaults to `''` if `refresh = 0`. Note: If running interactively but not under RStudio, `rstan` will open a browser window for monitoring progress.
`file`	optional file name in which to store results in RDS format. If `file` is given and it already exists, and none of the arguments to `stackMI` have changed since that fit, the fit object from `file` is immediately returned. So if the model, data, and imputations have not changed nothing needs to be computed.
`...`	arguments passed to `fitter`

Value

an `rmsb` fit object with expanded posterior draws and diagnostics

Author(s)

Frank Harrell

`stanDx` *Print Stan Diagnostics*

Description

Retrieves the effect samples sizes and Rhats computed after a fitting function ran `rstan`, and prepares it for printing. If the fit was created by `stackImpute`, the diagnostics for all imputations are printed (separately).

Usage

```
stanDx(object)
```

Arguments

`object`	an object created by an `rms` package Bayesian fitting function such as `blrm()` or `stackMI()`

Value

matrix suitable for printing

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(...)
  stanDx(f)

## End(Not run)
```

stanDxplot *Diagnostic Trace Plots*

Description

For an rms Bayesian fit object, uses by default the stored posterior draws to check convergence properties of posterior sampling. If instead rstan=TRUE, calls the rstan traceplot function on the rstan object inside the rmsb object, to check properties of posterior sampling. If rstan=TRUE and the rstan object has been removed and previous=TRUE, attempts to find an already existing plot created by a previous run of the knitr chunk, assuming it was the plotno numbered plot of the chunk.

Usage

```
stanDxplot(
  x,
  which = NULL,
  rstan = FALSE,
  previous = TRUE,
  plotno = 1,
  rev = FALSE,
  stripsize = 8,
  ...
)
```

Arguments

x	an rms Bayesian fit object
which	names of parameters to plot, defaulting to all non-intercepts. When rstan=FALSE these are the friendly rms names, otherwise they are the rstan parameter names. If the model fit was run through stackMI for multiple imputation, the number of traces is multiplied by the number of imputations. Set to 'ALL' to plot all parameters.
rstan	set to TRUE to use rstan::traceplot() on a (presumed) stored rstan object in x, otherwise only real iterations are plotted and parameter values are shown as points instead of lines, with chains separated
previous	see details
plotno	see details
rev	set to TRUE to reverse direction for faceting chains
stripsize	specifies size of chain facet label text, default is 8
...	passed to rstan::traceplot()

Value

ggplot2 object if rstan object was in x

Author(s)

Frank Harrell

stanGet *Get Stan Output*

Description

Extracts the object created by `rstan::sampling()` so that standard Stan diagnostics can be run from it

Usage

```
stanGet(object)
```

Arguments

`object`	an objected created by an `rms` package Bayesian fitting function

Value

the object created by `rstan::sampling()`

Author(s)

Frank Harrell

Examples

```
## Not run:
  f <- blrm(...)
  s <- stanGet(f)

## End(Not run)
```

tauFetch — *Fetch Partial Proportional Odds Parameters*

Description

Fetches matrix of posterior draws for partial proportional odds parameters (taus) for a given intercept. Can also form a matrix containing both regular parameters and taus, or for just non-taus. For the constrained partial proportional odds model the function returns the appropriate `cppo` function value multiplied by tau (tau being a vector in this case and not a matrix).

Usage

```
tauFetch(fit, intercept, what = c("tau", "nontau", "both"))
```

Arguments

`fit`	an object created by `blrm()`
`intercept`	integer specifying which intercept to fetch
`what`	specifies the result to return

Value

matrix with number of raws equal to the numnber of original draws

Author(s)

Frank Harrell

vcov.rmsb — *Variance-Covariance Matrix*

Description

Computes the variance-covariance matrix from the posterior draws by compute the sample covariance matrix of the draws

Usage

```
## S3 method for class 'rmsb'
vcov(object, regcoef.only = TRUE, intercepts = "all", ...)
```

Arguments

`object`	an object produced by an `rms` package Bayesian fitting function
`regcoef.only`	set to `FALSE` to also include non-regression coefficients such as shape/scale parameters
`intercepts`	set to `'all'` to include all intercepts (the default), `'none'` to exclude them all, or a vector of integers to get selected intercepts
`...`	ignored

Value

matrix

Author(s)

Frank Harrell

See Also

`rms::vcov.rms()`

Examples

```
## Not run:
  f <- blrm(...)
  v <- vcov(f)

## End(Not run)
```

Index