Package: brmsmargins (via r-universe)

October 29, 2026

Title Bayesian Marginal Effects for 'brms' Models

Version 0.3.0

URL https://joshuawiley.com/brmsmargins/, https://github.com/JWiley/brmsmargins

BugReports https://github.com/JWiley/brmsmargins/issues

Description Calculate Bayesian marginal effects, average marginal effects, and marginal coefficients (also called population averaged coefficients) for models fit using the 'brms' package including fixed effects, mixed effects, and location scale models. These are based on marginal predictions that integrate out random effects if necessary (see for example <doi:10.1186/s12874-015-0046-6> and <doi:10.1111/biom.12707>).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports methods, stats, data.table (>= 1.12.0), extraoperators (>= 0.1.1), brms, bayestestR, Rcpp, posterior

Suggests testthat (>= 3.0.0), covr, withr, knitr, rmarkdown, margins, betareg

Config/testthat/edition 3

Config/testthat/parallel true

LinkingTo RcppArmadillo, Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Joshua F. Wiley [aut, cre] (ORCID: <https://orcid.org/0000-0002-0271-6702>), Donald Hedeker [aut] (ORCID: <https://orcid.org/0000-0001-8134-6094>)

Maintainer Joshua F. Wiley <jwiley.psych@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-04-06 09:44:41 UTC

RemoteUrl https://github.com/cran/brmsmargins

RemoteRef HEAD

RemoteSha 4ca4fbb3a3ce22a92dc023ad911a40e57c0daf53

Contents

brmsmargins 2
bsummary 7
integratemvn 8
integratemvt 9
integratere 10
lmcpp 11
marginalcoef 12
prediction 13
rowBootMeans 15
tab2mat 16

Index **17**

`brmsmargins` *Calculate Marginal Effects from 'brms' Models*

Description

This function is designed to help calculate marginal effects including average marginal effects (AMEs) from `brms` models. Arguments are labeled as *required* when it is required that the user directly specify the argument. Arguments are labeled as *optional* when either the argument is optional or there are sensible default values so that users do not typically need to specify the argument.

Usage

```
brmsmargins(
  object,
  at = NULL,
  wat = NULL,
  add = NULL,
  newdata = model.frame(object),
  CI = 0.99,
  CIType = "HDI",
  contrasts = NULL,
  ROPE = NULL,
  MID = NULL,
  subset = NULL,
  dpar = NULL,
  seed,
```

```
  verbose = FALSE,
  ...
)
```

Arguments

`object`	A *required* argument specifying a fitted `brms` model object.
`at`	An *optional* argument (but note, either `at` or `add` are *required*) specifying an object inheriting from data frame indicating the values to hold specific variables at when calculating average predictions. This is intended for AMEs from categorical variables.
`wat`	An *optional* list with named elements including one element named, "ID" with a single character string, the name of the variable in the model frame that is the ID variable. Additionally, there should be one or more named elements, named after variables in the model (and specified in the `at` argument), that contain a `data.table` or `data.frame` with three variables: (1) the ID variable giving IDs, (2) the values specified for the variable in the `at` argument, and (3) the actual values to be substituted for each ID. `wat` cannot be non null unless `at` also is non null.
`add`	An *optional* argument (but note, either `at` or `add` are *required*) specifying an object inheriting from data frame indicating the values to add to specific variables at when calculating average predictions. This is intended for AMEs for continuous variables.
`newdata`	An *optional* argument specifying an object inheriting from data frame indicating the baseline values to use for predictions and AMEs. It uses a sensible default: the model frame from the `brms` model object passed on the `object` argument.
`CI`	An *optional* argument with a numeric value specifying the width of the credible interval. Defaults to `0.99`. This default is arbitrary, but is purposefully higher than the common `0.95` to encourage science with greater acknowledgment of uncertainty or larger sample sizes (ideally).
`CIType`	An *optional* argument, a character string specifying the type of credible interval (e.g., highest density interval). It is passed down to `bsummary()` which in turn passes it to `bayestestR::ci()`. Defaults to "HDI".
`contrasts`	An *optional* argument specifying a contrast matrix. The posterior predictions matrix is post multiplied by the contrast matrix, so they must be conformable. The posterior predictions matrix has a separate column for each row in the `at` or `add` object, so the contrast matrix should have the same number of rows. It can have multiple columns, if you desire multiple specific contrasts.
`ROPE`	An *optional* argument, that can either be left as `NULL`, the default, or a numeric vector of length 2, specifying the lower and upper thresholds for the Region of Practical Equivalence (ROPE).
`MID`	An *optional* argument, that can either left as `NULL`, the default, or a numeric vector of length 2, specifying the lower and upper thresholds for a Minimally Important Difference (MID). Unlike the ROPE, percentages for the MID are calculated as at or exceeding the bounds specified by this argument, whereas the ROPE is the percentage of the posterior at or inside the bounds specified.

`subset` An *optional* argument, a character string that is a valid `R` expression used to subset the dataset passed in `newdata`, prior to analysis. Defaults to `NULL`.

`dpar` An *optional* argument giving the parameter passed on to the `dpar` argument of `fitted()` in brms. Defaults to `NULL`, indicating the mean or location parameter typically.

`seed` An *optional* argument that controls whether (and if so what) random seed to use. This does not matter when using fixed effects only. However, when using Monte Carlo integration to integrate out random effects from mixed effects models, it is critical if you are looking at a continuous marginal effect with some small offset value as otherwise the Monte Carlo error from one set of predictions to another may exceed the true predicted difference. If `seed` is left missing, the default, than a single, random integer between +\- 1e7 is chosen and used to set the seed before each prediction. If manually chosen (recommended for reproducibility), the seed should either be a single value, in which case this single value is used to set the seed before each prediction. Alternately, it can be a vector of seeds with either the same length as the number of rows in `at` or `add`, whichever was specified. This is probably generally not what you want, as it means that even for the same input data, you would get slightly different predictions (when integrating out random effects) due to Monte Carlo variation. Finally, rather than being missing, you can explicitly set `seed = NULL`, if you do not want any seed to be set. This would be fine, for instance, when only using fixed effects, or if you know what you are doing and intend that behavior when integrating out random effects.

`verbose` An *optional* argument, a logical value whether to print more verbose messages. Defaults to `FALSE` which is quieter. Set to `TRUE` for more messages to be printed where relevant.

`...` An *optional* argument, additional arguments passed on to `prediction()`. In particular, the `effects` argument of `prediction()` is important for mixed effects models to control how random effects are treated in the predictions, which subsequently changes the marginal effect estimates.

Details

The main parts required for the function are a fitted model object, (via the `object` argument) a dataset to be used for prediction, (via the `newdata` argument which defaults to the model frame), and a dataset passed to either `at` or `add`. The steps are as follows:

1. Check that the function inputs (model object, data, etc.) are valid.
2. Take the dataset from the `newdata` argument and either add the values from the first row of `add` or replace the values using the first row of `at`. Only variables specified in `at` or `add` are modified. Other variables are left as is.
3. Use the `fitted()` function to generate predictions based on this modified dataset. If `effects` is set to "fixedonly" (meaning only generate predictions using fixed effects) or to "includeRE" (meaning generate predictions using fixed and random effects), then predictions are generated entirely using the `fitted()` function and are, typically back transformed to the response scale. For mixed effects models with fixed and random effects where `effects` is set to "integrateoutRE", then `fitted()` is only used to generate predictions using the fixed effects on the

linear scale. For each prediction generated, the random effects are integrated out by drawing k random samples from the model assumed random effect(s) distribution. These are added to the fixed effects predictions, back transformed, and then averaged over all k random samples to perform numerical Monte Carlo integration.

4. All the predictions for each posterior draw, after any back transformation has been applied, are averaged, resulting in one, marginal value for each posterior draw. These are marginal predictions. They are average marginal predictions if averaging over the sample dataset, or may be marginal predictions at the means, if the initial input dataset used mean values, etc.
5. Steps two to four are repeated for each row of `at` or `add`. Results are combined into a matrix where the columns are different rows from `at` or `add` and the rows are different posterior draws.
6. If contrasts were specified, using a contrast matrix, the marginal prediction matrix is post multiplied by the contrast matrix. Depending on the choice(s) of `add` or `at` and the values in the contrast matrix, these can then be average marginal effects (AMEs) by using numerical integration (`add` with 0 and a very close to 0 value) or discrete difference (`at` with say 0 and 1 as values) for a given predictor(s).
7. The marginal predictions and the contrasts, if specified are summarized.

Although `brmsmargins()` is focused on helping to calculate marginal effects, it can also be used to generate marginal predictions, and indeed these marginal predictions are the foundation of any marginal effect estimates. Through manipulating the input data, `at` or `add` and the contrast matrix, other types of estimates averaged or weighting results in specific ways are also possible.

Value

A list with four elements.

- `Posterior` Posterior distribution of all predictions. These predictions default to fixed effects only, but by specifying options to `prediction()` they can include random effects or be predictions integrating out random effects.
- `Summary` A summary of the predictions.
- `Contrasts` Posterior distribution of all contrasts, if a contrast matrix was specified.
- `ContrastSummary` A summary of the posterior distribution of all contrasts, if specified

References

Pavlou, M., Ambler, G., Seaman, S., & Omar, R. Z. (2015) doi:10.1186/s1287401500466 "A note on obtaining correct marginal predictions from a random intercepts model for binary outcomes" and Skrondal, A., & Rabe-Hesketh, S. (2009) doi:10.1111/j.1467985X.2009.00587.x "Prediction in multilevel generalized linear models" and Norton EC, Dowd BE, Maciejewski ML. (2019) doi:10.1001/jama.2019.1954 "Marginal Effects—Quantifying the Effect of Changes in Risk Factors in Logistic Regression Models"

Examples

```
## Not run:
#### Testing ####
## sample data and logistic model with brms
```

```
set.seed(1234)
Tx <- rep(0:1, each = 50)
ybin <- c(rep(0:1, c(40,10)), rep(0:1, c(10,40)))
logitd <- data.frame(Tx = Tx, ybin = ybin)
logitd$x <- rnorm(100, mean = logitd$ybin, sd = 2)

mbin <- brms::brm(ybin ~ Tx + x, data = logitd, family = brms::bernoulli())

summary(mbin)

## now check AME for Tx
tmp <- brmsmargins(
  object = mbin,
  at = data.table::data.table(Tx = 0:1),
  contrasts = matrix(c(-1, 1), nrow = 2),
  ROPE = c(-.05, +.05),
  MID = c(-.10, +.10))

tmp$Summary
tmp$ContrastSummary ## Tx AME


## now check AME for Tx with bootstrapping the AME population
tmpalt <- brmsmargins(
  object = mbin,
  at = data.table::data.table(Tx = 0:1),
  contrasts = matrix(c(-1, 1), nrow = 2),
  ROPE = c(-.05, +.05),
  MID = c(-.10, +.10),
  resample = 100L)

tmpalt$Summary
tmpalt$ContrastSummary ## Tx AME

## now check AME for continuous predictor, x
## use .01 as an approximation for first derivative
## 1 / .01 in the contrast matrix to get back to a one unit change metric
tmp2 <- brmsmargins(
  object = mbin,
  add = data.table::data.table(x = c(0, .01)),
  contrasts = matrix(c(-1/.01, 1/.01), nrow = 2),
  ROPE = c(-.05, +.05),
  MID = c(-.10, +.10))

tmp2$ContrastSummary ## x AME

if (FALSE) {
  library(lme4)
  data(sleepstudy)
  fit <- brms::brm(Reaction ~ 1 + Days + (1 + Days | Subject),
              data = sleepstudy,
              cores = 4)
```

```
summary(fit, prob = 0.99)

tmp <- brmsmargins(
  object = fit,
  at = data.table::data.table(Days = 0:1),
  contrasts = matrix(c(-1, 1), nrow = 2),
  ROPE = c(-.05, +.05),
  MID = c(-.10, +.10), CIType = "ETI", effects = "integrateoutRE", k = 5L)

tmp$Summary
tmp$ContrastSummary
}

## End(Not run)
```

`bsummary` *Personal Preference Based Bayesian Summary*

Description

Returns a summary of a posterior distribution for a single parameter / value. It is based on personal preference. Notably, it does not only use `bayestestR::describe_posterior()`, an excellent function, because of the desire to also describe the percentage of the full posterior distribution that is at or exceeding the value of a Minimally Important Difference (MID). MIDs are used in clinical studies with outcome measures where there are pre-defined differences that are considered clinically important, which is distinct from the ROPE or general credible intervals capturing uncertainty.

Usage

```
bsummary(x, CI = 0.99, CIType = "HDI", ROPE = NULL, MID = NULL)
```

Arguments

`x`	The posterior distribution of a parameter
`CI`	A numeric value indicating the desired width of the credible interval. Defaults to `0.99` currently, but this is subject to change. a 99% interval was chosen as the default as there have been recent arguments made in the realm of meta science that there are, essentially, too many false positives and that many of the "findings" in science are not able to be replicated. In any case, users should ideally specify a desired CI width, and not rely on defaults.
`CIType`	A character string indicating the type of credible interval, passed on to the `bayestestR::ci()` function as the method for CIs.
`ROPE`	Either left as `NULL`, the default, or a numeric vector of length 2, specifying the lower and upper thresholds for the Region of Practical Equivalence (ROPE).

MID — Either left as NULL, the default, or a numeric vector of length 2, specifying the lower and upper thresholds for a Minimally Important Difference (MID). Unlike the ROPE, percentages for the MID are calculated as at or exceeding the bounds specified by this argument, whereas the ROPE is the percentage of the posterior at or inside the bounds specified.

Value

A data.table with:

- M the mean of the posterior samples
- Mdn the median of the posterior samples
- LL the lower limit of the credible interval
- UL the upper limit of the credible interval
- PercentROPE the percentage of posterior samples falling into the ROPE
- PercentMID the percentage of posterior samples falling at or beyond the MID
- CI the width of the credible interval used
- CIType the type of credible interval used (e.g., highest density interval)
- ROPE a label describing the values included in the ROPE
- MID a label describing the values included in the MID

References

Kruschke, J. K. (2018). doi:10.1177/2515245918771304 "Rejecting or accepting parameter values in Bayesian estimation"

Examples

```
bsummary(rnorm(1000))

bsummary(rnorm(1000), ROPE = c(-.5, .5), MID = c(-1, 1))
```

---

integratemvn *Integrate over Multivariate Normal Random Effects*

Description

Used in the process of Monte Carlo integration over multivariate normal random effects. This generates the random draws from the multivariate normal distribution and multiplies these by the data. Not intended to be called directly by most users.

Usage

```
integratemvn(X, k, sd, chol)

integratemvnR(X, k, sd, chol)
```

Arguments

X	A numeric matrix of the data to be multiplied by the random effects
k	An integer, the number of random samples to be used for numerical integration
sd	A numeric vector of the standard deviations
chol	A numeric matrix, which should be the Cholesky decomposition of the correlation matrix of the multivariate normal distribution.

Value

A numeric matrix with random values

Functions

- `integratemvnR()`: Pure R implementation of `integratemvn()`.

Examples

```
integratemvn(
  X = matrix(1, 1, 2),
  k = 100L,
  sd = c(10, 5),
  chol = chol(matrix(c(1, .5, .5, 1), 2)))

integratemvn(matrix(1, 1, 1), 100L, c(5), matrix(1))
```

---

integratemvt *Integrate over Multivariate Student-t Random Effects*

Description

Used in the process of Monte Carlo integration over multivariate Student-t random effects. This generates the random draws from the multivariate Student-t distribution and multiplies these by the data. Not intended to be called directly by most users.

Usage

```
integratemvt(X, k, sd, chol, df)

integratemvtR(X, k, sd, chol, df)
```

Arguments

X	A numeric matrix of the data to be multiplied by the random effects
k	An integer, the number of random samples to be used for numerical integration
sd	A numeric vector of the standard deviations

chol	A numeric matrix, which should be the Cholesky decomposition of the correlation matrix of the multivariate Student-t distribution.
df	A numeric scalar giving the degrees of freedom of the (multivariate) Student-t distribution.

Value

A numeric matrix with random values

Functions

- integratemvtR(): Pure R implementation of integratemvt().

Examples

```
integratemvt(
  X = matrix(1, 1, 2),
  k = 100L,
  sd = c(10, 5),
  chol = chol(matrix(c(1, .5, .5, 1), 2)),
  df = 5)

integratemvt(matrix(1, 1, 1), 100L, c(5), matrix(1), df = 5)
```

---

integratere *Integrate over Random Effects*

Description

Used to conduct Monte Carlo integration over Gaussian random effects. Not intended to be called directly by most users.

Usage

```
integratere(d, sd, L, k, df, yhat, backtrans)

integratereR(d, sd, L, k, df, yhat, backtrans)
```

Arguments

d	A list with model matrices for each random effect block.
sd	A list with standard deviation matrices for each random effect block where rows are different posterior draws.
L	A list with matrices for each random effect block containing the parts of the L matrix, the Cholesky decomposition of the random effect correlation matrix.
k	An integer, the number of samples for Monte Carlo integration.

df	A list with either NULL for Gaussian random effect blocks or a numeric matrix of degrees of freedom for Student-t random effect blocks.
yhat	A matrix of the fixed effects predictions
backtrans	An integer, indicating the type of back transformation. 0 indicates inverse logit (e.g., for logistic regression). 1 indicates exponential (e.g., for poisson or negative binomial regression or if outcome was natural log transformed). 2 indicates square (e.g., if outcome was square root transformed). 3 indicates inverse (e.g., if outcome was inverse transformed such as Gamma regression) Any other integer results in no transformation. -9 is recommended as the option for no transformation as any future transformations supported will be other, positive integers.

Value

A numeric matrix with the Monte Carlo integral calculated.

Functions

- `integratereR()`: Pure R implementation of `integratere()`.

Examples

```
integratere(
  d = list(matrix(1, 1, 1)),
  sd = list(matrix(1, 2, 1)),
  L = list(matrix(1, 2, 1)),
  k = 10L,
  df = list(NULL),
  yhat = matrix(0, 2, 1),
  backtrans = 0L)
```

---

lmcpp *Fast Linear Regression*

Description

Used to get marginal coefficients off of a generalized linear mixed model.

Usage

```
lmcpp(X, y)
```

Arguments

X	A numeric model matrix. If intercept is desired, it must already have been added as a column.
y	A numeric matrix. A single column if one response variable or multiple columns where each column is a different response, such as a for marginal coefficients where each column is a different MCMC sample.

Value

A numeric matrix with the coefficient.

Examples

```
lmcpp(cbind(1, mtcars$hp, mtcars$am), as.matrix(mtcars[, c("mpg", "qsec")]))
```

`marginalcoef` *Marginal Coefficients from a 'brms' Model*

Description

Calculate marginal coefficients from a `brms` generalized linear mixed model using the method proposed by Hedeker (2018).

Usage

```
marginalcoef(
  object,
  summarize = TRUE,
  posterior = FALSE,
  index,
 backtrans = c("response", "linear", "identity", "invlogit", "exp", "square", "inverse"),
  k = 100L,
  seed,
  ...
)
```

Arguments

`object`	A fitted `brms` model object that includes random effects. Required.
`summarize`	A logical value, whether or not to calculate summaries of the posterior predictions. Defaults to `TRUE`.
`posterior`	A logical value whether or not to save and return the posterior samples. Defaults to `FALSE` as the assumption is a typical use case is to return the summaries only.
`index`	An optional integer vector, giving the posterior draws to be used in the calculations. If omitted, defaults to all posterior draws.
`backtrans`	A character string indicating the type of back transformation to be applied. Can be one of "response" meaning to use the response scale, "linear" or "identity" meaning to use the linear predictor scale, or a specific back transformation desired, from a possible list of "invlogit", "exp", "square", or "inverse". Custom back transformations should only be needed if, for example, the outcome variable was transformed prior to fitting the model.
`k`	An integer providing the number of random draws to use for integrating out the random effects. Only relevant when `effects = "integrateoutRE"`.

seed	An *optional* argument that controls whether (and if so what) random seed to use. This can help with reproducibility of results. It is missing by default.
...	Additional arguments passed to bsummary(), and only relevant if summarize is TRUE.

Value

A list with Summary and Posterior. Some of these may be NULL depending on the arguments used.

References

Hedeker, D., du Toit, S. H., Demirtas, H. & Gibbons, R. D. (2018) doi:10.1111/biom.12707. "A note on marginalization of regression parameters from mixed models of binary outcomes"

---

prediction *Marginal Posterior Predictions from a 'brms' Model*

Description

Calculate marginal predictions from a brms model. Marginal predictions average over the input data for each posterior draw. Marginal predictions for models with random effects will integrate over random effects. Arguments are labeled as *required* when it is required that the user directly specify the argument. Arguments are labeled as *optional* when either the argument is optional or there are sensible default values so that users do not typically need to specify the argument.

Usage

```
prediction(
  object,
  data,
  summarize = TRUE,
  posterior = FALSE,
  index,
  dpar = NULL,
  resample = 0L,
  resampleseed = FALSE,
  effects = c("fixedonly", "includeRE", "integrateoutRE"),
 backtrans = c("response", "linear", "identity", "invlogit", "exp", "square", "inverse"),
  k = 100L,
  raw = FALSE,
  ...
)
```

Arguments

object	A *required* argument specifying a fitted brms model object.
data	A *required* argument specifying a data frame or data table passed to fitted() as the new data to be used for predictions.
summarize	An *optional* argument, a logical value, whether or not to calculate summaries of the posterior predictions. Defaults to TRUE.
posterior	An *optional* argument, a logical value whether or not to save and return the posterior samples. Defaults to FALSE as the assumption is a typical use case is to return the summaries only.
index	An *optional* argument, an integer vector, giving the posterior draws to be used in the calculations. If omitted, defaults to all posterior draws.
dpar	An *optional* argument, the parameter passed on to the dpar argument of fitted() in brms. Defaults to NULL indicating the mean or location parameter typically.
resample	An *optional* argument, an integer indicating the number of bootstrap resamples of the posterior predictions to use when calculating summaries. Defaults to 0L. See documentation from .averagePosterior() for more details. This should be considered experimental.
resampleseed	An *optional* argument, a seed for random number generation. Defaults to FALSE, which means no seed is set. Only used if resample is a positive, non-zero integer. See documentation from .averagePosterior() for more details. This should be considered experimental.
effects	An *optional* argument, a character string indicating the type of prediction to be made. Can be one of "fixedonly" meaning only use fixed effects, "includeRE" meaning that random effects should be included in the predictions, or "integrateoutRE" meaning that random effects should be integrated out / over in the predictions. It defaults to "fixedonly" so is not typically required for a user to specify it.
backtrans	An *optional* argument, a character string indicating the type of back transformation to be applied. Can be one of "response" meaning to use the response scale, "linear" or "identity" meaning to use the linear predictor scale, or a specific back transformation desired, from a possible list of "invlogit", "exp", "square", or "inverse". Custom back transformations should only be needed if, for example, the outcome variable was transformed prior to fitting the model. It defaults to "response" so is not typically required for a user to specify it.
k	An *optional* argument, an integer providing the number of random draws to use for integrating out the random effects. Only relevant when effects = "integrateoutRE". It defaults to 100L, a rather arbitrary number attempting to balance the increased precision that comes from a larger value, with the increased computational cost of more Monte Carlo simulations when integrating out random effects.
raw	An *optional* argument, a logical value indicating whether to return the raw output or to average over the Monte Carlo samples. Defaults to FALSE. Setting it to TRUE can be useful if you want not only the full posterior distribution but also the k Monte Carlo samples used for the numerical integration. This cannot be used with summarize = TRUE.
...	Additional arguments passed to bsummary(), and only relevant if summarize is TRUE.

Value

A list with `Summary` and `Posterior`. Some of these may be `NULL` depending on the arguments used.

References

Pavlou, M., Ambler, G., Seaman, S., & Omar, R. Z. (2015) doi:10.1186/s1287401500466 "A note on obtaining correct marginal predictions from a random intercepts model for binary outcomes" and Skrondal, A., & Rabe-Hesketh, S. (2009) doi:10.1111/j.1467985X.2009.00587.x "Prediction in multilevel generalized linear models"

`rowBootMeans` *Bootstrap Row Means*

Description

This takes a numeric matrix, bootstrap resamples each row, and then calculates the mean. The intended use case is for Bayesian posterior predictions from sample data. Instead of directly calculating the average marginal effect (AME) across all observed values, these can be bootstrapped, so that uncertainty in the target population, and thus the AME in the target population, can be incorporated. Model uncertainty is already assumed to be handled by the different posterior samples, which are assumed to be across rows.

Usage

```
rowBootMeans(x)
```

Arguments

`x`	A numeric matrix

Value

A numeric vector with the simple bootstrapped row means of the matrix

Examples

```
x <- matrix(1:9, byrow = TRUE, 3)
replicate(10, rowBootMeans(x))
```

tab2mat *Convert a Row of a Table to a Square Matrix*

Description

Utility function to convert a row matrix to a square matrix. Used as the brms package returns things like the Cholesky decomposition matrix as separate columns where rows are posterior draws. Not intended to be called directly by most users.

Usage

```
tab2mat(X)

tab2matR(X)
```

Arguments

X	a matrix

Value

A numeric matrix with one row.

Functions

- tab2matR(): Pure R implementation of tab2mat().

Examples

```
tab2mat(matrix(1:4, 1))
tab2mat(matrix(1:9, 1))
```

Index

.averagePosterior(), *14*

bayestestR::ci(), *3*, *7*
bayestestR::describe_posterior(), *7*
brmsmargins, 2
brmsmargins(), *5*
bsummary, 7
bsummary(), *3*, *13*

fitted(), *4*, *14*

integratemvn, 8
integratemvn(), *9*
integratemvnR *(*integratemvn*)*, 8
integratemvt, 9
integratemvt(), *10*
integratemvtR *(*integratemvt*)*, 9
integratere, 10
integratere(), *11*
integratereR *(*integratere*)*, 10

lmcpp, 11

marginalcoef, 12

prediction, 13
prediction(), *4*, *5*

rowBootMeans, 15

tab2mat, 16
tab2mat(), *16*
tab2matR *(*tab2mat*)*, 16