Package ‘pema’

March 30, 2025

Title Penalized Meta-Analysis

Version 0.1.4

Description Conduct penalized meta-analysis, see Van Lissa, Van Erp, & Clapper (2023) <doi:10.31234/osf.io/6phs5>. In meta-analysis, there are often between-study differences. These can be coded as moderator variables, and controlled for using meta-regression. However, if the number of moderators is large relative to the number of studies, such an analysis may be overfit. Penalized meta-regression is useful in these cases, because it shrinks the regression slopes of irrelevant moderators towards zero.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/cjvanlissa/pema, https://cjvanlissa.github.io/pema/

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.4.0)

Imports methods, rstan (>= 2.26.0), Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstantools (>= 2.1.1), sn, shiny, ggplot2, cli

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

Suggests rmarkdown, knitr, mice, testthat (>= 3.0.0), webexercises, bain, metaforest, metafor

SystemRequirements GNU make

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation yes

Author Caspar J van Lissa [aut, cre] (<https://orcid.org/0000-0002-0808-5024>), Sara J van Erp [aut]

Maintainer Caspar J van Lissa <c.j.vanlissa@tilburguniversity.edu>

Repository CRAN

Date/Publication 2025-03-30 08:00:01 UTC

Contents

`pema-package` *pema: Conduct penalized meta-regression.*

Description

Penalized meta-regression shrinks the regression slopes of irrelevant moderators towards zero (Van Lissa & Van Erp, 2021).

Author(s)

Maintainer: Caspar J van Lissa <c.j.vanlissa@tilburguniversity.edu> (ORCID)

Authors:

- Sara J van Erp

References

Van Lissa, C. J., van Erp, S., & Clapper, E. B. (2023). Selecting relevant moderators with Bayesian regularized meta-regression. Research Synthesis Methods. doi:10.31234/osf.io/6phs5

Stan Development Team (NA). RStan: the R interface to Stan. R package version 2.26.2. https://mc-stan.org

See Also

Useful links:

- https://github.com/cjvanlissa/pema
- https://cjvanlissa.github.io/pema/

`as.stan` *Convert an object to stanfit*

Description

Create a `stanfit` object from an object for which a method exists, so that all methods for `stanfit` objects can be used.

Usage

```
as.stan(x, ...)
```

Arguments

`x`	An object for which a method exists.
`...`	Arguments passed to or from other methods.

Value

An object of class `stanfit`, as documented in rstan::stan.

Examples

```
stanfit <- "a"
class(stanfit) <- "stanfit"
converted <- as.stan(stanfit)
```

`bonapersona` *Data from 'The behavioral phenotype of early life adversity'*

Description

This meta-analysis of rodent studies examined whether early life adversity (ELA) alters cognitive performance in several domains. The data include over 400 independent experiments, involving approximately 8600 animals.

Usage

```
data(bonapersona)
```

Format

A data.frame with 734 rows and 65 columns.

References

Bonapersona, V., Kentrop, J., Van Lissa, C. J., van der Veen, R., Joels, M., & Sarabdjitsingh, R. A. (2019). The behavioral phenotype of early life adversity: A 3-level meta-analysis of rodent studies. Neuroscience & Biobehavioral Reviews, 102, 299–307. doi:10.1016/j.neubiorev.2019.04.021

---

brma — *Conduct Bayesian Regularized Meta-Analysis*

Description

This function conducts Bayesian regularized meta-regression (Van Lissa & Van Erp, 2021). It uses the `stan` function rstan::sampling to fit the model. A lasso or horseshoe prior is used to shrink the regression coefficients of irrelevant moderators towards zero. See Details.

Usage

```
brma(x, ...)

## S3 method for class 'formula'
brma(
  formula,
  data,
  vi = "vi",
  study = NULL,
  method = "hs",
  standardize = TRUE,
 prior = switch(method, lasso = c(df = 1, scale = 1), hs = c(df = 1, df_global = 1,
    df_slab = 4, scale_global = 1, scale_slab = 2, relevant_pars = NULL)),
  mute_stan = TRUE,
  ...
)

## Default S3 method:
brma(
  x,
  y,
  vi,
  study = NULL,
  method = "hs",
  standardize,
  prior,
  mute_stan = TRUE,
  intercept,
```

```
  ...
)
```

Arguments

x	An k x m numeric matrix, where k is the number of effect sizes and m is the number of moderators.
...	Additional arguments passed on to `rstan::sampling()`. Use this, e.g., to override default arguments of that function.
formula	An object of class `formula` (or one that can be coerced to that class), see `lm`.
data	Either a `data.frame` containing the variables in the model, see `lm`, or a `list` of multiple imputed `data.frames`, or an object returned by `mice`.
vi	Character. Name of the column in the `data` that contains the variances of the effect sizes. This column will be removed from the data prior to analysis. Defaults to `"vi"`.
study	Character. Name of the column in the `data` that contains the study id. Use this when the data includes multiple effect sizes per study. This column can be a vector of integers, or a factor. This column will be removed from the data prior to analysis. See `Details` for more information about analyzing dependent data.
method	Character, indicating the type of regularizing prior to use. Supports one of `c("hs", "lasso")`, see Details. Defaults to `"hs"`.
standardize	Either a logical argument or a list. If `standardize` is logical, it controls whether all predictors are standardized prior to analysis or not. Parameter estimates are restored to the predictors' original scale. Alternatively, users can provide a list to `standardize` to gain more control over the standardization process. In this case, it is assumed that the standardization has already taken place. This list must have two elements: `list(center = c(mean(X1) , mean(X2), mean(X...)), scale = c(sd(X1), sd(X2), sd(X...)))`. It is used only to restore parameter estimates to the original scale of the predictors. This is useful, e.g., to standardize continuous and dichotomous variables separately. Defaults to `TRUE`, which is recommended so that shrinking affects all parameters similarly.
prior	Numeric vector, specifying the prior to use. Note that the different `methods` require this vector to contain specific named elements.
mute_stan	Logical, indicating whether mute all 'Stan' output or not.
y	A numeric vector of k effect sizes.
intercept	Logical, indicating whether or not an intercept should be included in the model.

Details

The Bayesian regularized meta-analysis algorithm (Van Lissa & Van Erp, 2021) penalizes meta-regression coefficients either via the lasso prior (Park & Casella, 2008) or the regularized horseshoe prior (Piironen & Vehtari, 2017).

lasso The Bayesian equivalent of the lasso penalty is obtained when placing independent Laplace (i.e., double exponential) priors on the regression coefficients centered around zero. The scale of the Laplace priors is determined by a global scale parameter `scale`, which defaults to 1 and an inverse-tuning parameter $\frac{1}{\lambda}$ which is given a chi-square prior governed by a degrees of freedom parameter `df` (defaults to 1). If `standardize = TRUE`, shrinkage will affect all coefficients equally and it is not necessary to adapt the `scale` parameter. Increasing the `df` parameter will allow larger values for the inverse-tuning parameter, leading to less shrinkage.

hs One issue with the lasso prior is that it has relatively light tails. As a result, not only does the lasso have the desirable behavior of pulling small coefficients to zero, it also results in too much shrinkage of large coefficients. An alternative prior that improves upon this shrinkage pattern is the horseshoe prior (Carvalho, Polson & Scott, 2010). The horseshoe prior has an infinitely large spike at zero, thereby pulling small coefficients toward zero but in addition has fat tails, which allow substantial coefficients to escape the shrinkage. The regularized horseshoe is an extension of the horseshoe prior that allows the inclusion of prior information regarding the number of relevant predictors and can be more numerically stable in certain cases (Piironen & Vehtari, 2017). The regularized horseshoe has a global shrinkage parameter that influences all coefficients similarly and local shrinkage parameters that enable flexible shrinkage patterns for each coefficient separately. The local shrinkage parameters are given a Student's t prior with a default `df` parameter of 1. Larger values for `df` result in lighter tails and a prior that is no longer strictly a horseshoe prior. However, increasing `df` slightly might be necessary to avoid divergent transitions in Stan (see also `https://mc-stan.org/misc/warnings.html`). Similarly, the degrees of freedom for the Student's t prior on the global shrinkage parameter `df_global` can be increased from the default of 1 to, for example, 3 if divergent transitions occur although the resulting prior is then strictly no longer a horseshoe. The scale for the Student's t prior on the global shrinkage parameter `scale_global` defaults to 1 and can be decreased to achieve more shrinkage. Moreover, if prior information regarding the number of relevant moderators is available, it is recommended to include this information via the `relevant_pars` argument by setting it to the expected number of relevant moderators. When `relevant_pars` is specified, `scale_global` is ignored and instead based on the available prior information. Contrary to the horseshoe prior, the regularized horseshoe applies additional regularization on large coefficients which is governed by a Student's t prior with a `scale_slab` defaulting to 2 and `df_slab` defaulting to 4. This additional regularization ensures at least some shrinkage of large coefficients to avoid any sampling problems.

Value

A `list` object of class `brma`, with the following structure:

```
list(
  fit          # An object of class stanfit, for compatibility with rstan
  coefficients # A numeric matrix with parameter estimates; these are
               # interpreted as regression coefficients, except tau2 and tau,
               # which are interpreted as the residual variance and standard
               # deviation, respectively.
  formula      # The formula used to estimate the model
  terms        # The predictor terms in the formula
  X            # Numeric matrix of moderator variables
  Y            # Numeric vector with effect sizes
  vi           # Numeric vector with effect size variances
  tau2         # Numeric, estimated tau2
  R2           # Numeric, estimated heterogeneity explained by the moderators
```

```
  k             # Numeric, number of effect sizes
  study         # Numeric vector with study id numbers
)
```

References

Van Lissa, C. J., van Erp, S., & Clapper, E. B. (2023). Selecting relevant moderators with Bayesian regularized meta-regression. Research Synthesis Methods. doi:10.31234/osf.io/6phs5

Park, T., & Casella, G. (2008). The Bayesian Lasso. Journal of the American Statistical Association, 103(482), 681–686. doi:10.1198/016214508000000337

Carvalho, C. M., Polson, N. G., & Scott, J. G. (2010). The horseshoe estimator for sparse signals. Biometrika, 97(2), 465–480. doi:10.1093/biomet/asq017

Piironen, J., & Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. Electronic Journal of Statistics, 11(2). https://projecteuclid.org/journals/electronic-journal-of-statistics/volume-11/issue-2/Sparsity-information-and-regularizatio 10.1214/17-EJS1337SI.pdf

Examples

```
data("curry")
df <- curry[c(1:5, 50:55), c("d", "vi", "sex", "age", "donorcode")]
suppressWarnings({res <- brma(d~., data = df, iter = 10)})
```

---

check_workshop_data   *Check Data for BRMA Workshop*

Description

This function checks that argument `df` is a suitable `data.frame` for completing the workshop on Bayesian Regularized Meta-Regression, and checks that pema package dependencies are correctly installed. It suggests how to remedy any failed checks.

Usage

```
check_workshop_data(df)
```

Arguments

`df`	A `data.frame`.

Value

Invisibly returns a logical TRUE/FALSE, indicating whether all checks have passed.

Examples

```
check_workshop_data(fukkink_lont)
```

curry *Data from 'Happy to Help?'*

Description

A systematic review and meta-analysis of the effects of performing acts of kindness on the well-being of the actor.

Usage

```
data(curry)
```

Format

A data.frame with 56 rows and 18 columns.

study_id	`factor`	Unique identifier of the study
effect_id	`integer`	Unique identifier of the effect size
d	`numeric`	Standardized mean difference between the control group and intervention group
vi	`numeric`	Variance of the effect size
n1i	`numeric`	Number of participants in the intervention group
n1c	`numeric`	Number of participants in the control group
sex	`numeric`	Percentage of male participants
age	`numeric`	Mean age of participants
location	`character`	Geographical location of the study
donor	`character`	From what population did the donors (helpers) originate?
donorcode	`factor`	From what population did the donors (helpers) originate? Dichotomized to Anxious or Typ
interventioniv	`character`	Description of the intervention / independent variable
interventioncode	`factor`	Description of the intervention / independent variable, categorized to Acts of Kindness, Pro
control	`character`	Description of the control condition
controlcode	`factor`	Description of the control condition, categorized to Neutral Activity, Nothing, or Self Help
recipients	`character`	Who were the recipients of the act of kindness?
outcomedv	`character`	What was the outcome, or dependent variable, of the study?
outcomecode	`factor`	What was the outcome, or dependent variable, of the study? Categorized into Happiness, L

References

Curry, O. S., Rowland, L. A., Van Lissa, C. J., Zlotowitz, S., McAlaney, J., & Whitehouse, H. (2018). Happy to help? A systematic review and meta-analysis of the effects of performing acts of kindness on the well-being of the actor. Journal of Experimental Social Psychology, 76, 320-329. doi:10.1016/j.ecresq.2007.04.005

I2 — *Compute I2*

Description

I2 represents the amount of heterogeneity relative to the total amount of variance in the observed effect sizes (Higgins & Thompson, 2002). For three-level meta-analyses, it is additionally broken down into I2_w (amount of within-cluster heterogeneity) and I2_b (amount of between-cluster heterogeneity).

Usage

```
I2(x, ...)
```

Arguments

`x`	An object for which a method exists.
`...`	Arguments passed to other functions.

Value

Numeric matrix, with rows corresponding to I2 (total heterogeneity), and optionally I2_w and I2_b (within- and between-cluster heterogeneity).

Examples

```
I2(matrix(1:20, ncol = 1))
```

maxap — *Maximum a posteriori parameter estimate*

Description

Find the parameter estimate with the highest posterior probability density given a vector of samples.

Usage

```
maxap(x, dens = NULL, ...)
```

Arguments

`x`	Numeric vector.
`dens`	Optional object of class `density`. Defaults to `NULL`.
`...`	Arguments passed to `density`

Value

Atomic numeric vector with the maximum a-posteriori estimate of vector x.

Examples

```
maxap(c(1,2,3,4,5))
```

---

plot_sensitivity *Plot posterior distributions for BRMA models*

Description

To perform a rudimentary sensitivity analysis, plot the posterior distributions of multiple BRMA models and compare them visually.

Usage

```
plot_sensitivity(..., parameters = NULL, model_names = NULL)
```

Arguments

...	Objects of class brma. If the argument model_names is NULL, the names of these objects are used to label the plot.
parameters	Optional character vector with the names of parameters that exist in the models in ..., Default: NULL.
model_names	Optional character vector with the names used to label the models in ..., Default: NULL

Value

An object of class ggplot

Examples

```
plot_sensitivity(samples = list(
data.frame(Parameter = "b",
Value = rnorm(10),
Model = "M1"),
data.frame(Parameter = "b",
Value = rnorm(10, mean = 2),
Model = "M2")),
parameters = "b")
```

sample_prior *Sample from the Prior Distribution*

Description

Samples from a prior distribution with parameters defined in `prior`. The result can be plotted using the `plot` function.

Usage

```
sample_prior(
  method = c("hs", "lasso"),
  prior = switch(method, lasso = c(df = 1, scale = 1), hs = c(df = 1, df_global = 1,
    df_slab = 4, scale_global = 1, scale_slab = 2, par_ratio = NULL)),
  iter = 1000
)
```

Arguments

`method`	Character string, indicating which prior to sample from. Default: first element of `c("hs", "lasso")`.
`prior`	Numeric vector, specifying the prior to use. See brma for more details.
`iter`	A positive integer specifying the number of iterations to sample. Default: 1000

Value

NULL, function is called for its side-effect of plotting to the graphics device.

Examples

```
sample_prior("lasso", iter = 10)
```

shiny_prior *Interactively Sample from the Prior Distribution*

Description

Launches a `Shiny` app that allows interactive comparison of different priors for `brma`.

Usage

```
shiny_prior()
```

Value

NULL, function is called for its side-effect of launching a Shiny app.

Examples

```
## Not run:
shiny_prior()

## End(Not run)
```

---

simulate_smd *Simulates a meta-analytic dataset*

Description

This function simulates a meta-analytic dataset based on the random-effects model. The simulated effect size is Hedges' G, an estimator of the Standardized Mean Difference (Hedges, 1981; Li, Dusseldorp, & Meulman, 2017). The functional form of the model can be specified, and moderators can be either normally distributed or Bernoulli-distributed. See Van Lissa, in preparation, for a detailed explanation of the simulation procedure.

Usage

```
simulate_smd(
  k_train = 20,
  k_test = 100,
  mean_n = 40,
  es = 0.5,
  tau2 = 0.04,
  alpha = 0,
  moderators = 5,
  distribution = "normal",
  model = "es * x[, 1]"
)
```

Arguments

k_train	Atomic integer. The number of studies in the training dataset. Defaults to 20.
k_test	Atomic integer. The number of studies in the testing dataset. Defaults to 100.
mean_n	Atomic integer. The mean sample size of each simulated study in the meta-analytic dataset. Defaults to `40`. For each simulated study, the sample size n is randomly drawn from a normal distribution with mean mean_n, and sd mean_n/3.
es	Atomic numeric vector. The effect size, also known as beta, used in the model statement. Defaults to `.5`.
tau2	Atomic numeric vector. The residual heterogeneity. For a range of realistic values encountered in psychological research, see Van Erp, Verhagen, Grasman, & Wagenmakers, 2017. Defaults to `0.04`.
alpha	Vector of slant parameters, passed to sn::rsn.

`moderators` Atomic integer. The number of moderators to simulate for each study. Make sure that the number of moderators to be simulated is at least as large as the number of moderators referred to in the model parameter. Internally, the matrix of moderators is referred to as `"x"`. Defaults to 5.

`distribution` Atomic character. The distribution of the moderators. Can be set to either `"normal"` or `"bernoulli"`. Defaults to `"normal"`.

`model` Expression. An expression to specify the model from which to simulate the mean true effect size, mu. This formula may use the terms `"es"` (referring to the es parameter of the call to simulate_smd), and `"x\[, \]"` (referring to the matrix of moderators, x). Thus, to specify that the mean effect size, mu, is a function of the effect size and the first moderator, one would pass the value `model = "es * x\[ , 1\]"`. Defaults to `"es * x\[ , 1\]"`.

Value

List of length 4. The "training" element of this list is a data.frame with k_train rows. The columns are the variance of the effect size, vi; the effect size, yi, and the moderators, X. The "testing" element of this list is a data.frame with k_test rows. The columns are the effect size, yi, and the moderators, X. The "housekeeping" element of this list is a data.frame with k_train + k_test rows. The columns are n, the sample size n for each simulated study; mu_i, the mean true effect size for each simulated study; and theta_i, the true effect size for each simulated study.

Examples

```
set.seed(8)
simulate_smd()
simulate_smd(k_train = 50, distribution = "bernoulli")
simulate_smd(distribution = "bernoulli", model = "es * x[ ,1] * x[ ,2]")
```

Index