Package 'makemyprior'

August 23, 2024

Title Intuitive Construction of Joint Priors for Variance Parameters

Version 1.2.2

Description Tool for easy prior construction and visualization. It helps to formulates joint prior distributions for variance parameters in latent Gaussian models. The resulting prior is robust and can be created in an intuitive way. A graphical user interface (GUI) can be used to choose the joint prior, where the user can click through the model and select priors. An extensive guide is available in the GUI. The package allows for direct inference with the specified model and prior. Using a hierarchical variance decomposition, we formulate a joint variance prior that takes the whole model structure into account. In this way, existing knowledge can intuitively be incorporated at the level it applies to. Alternatively, one can use independent variance priors for each model components in the latent Gaussian model. Details can be found in the accompanying scientific paper: Hem, Fuglstad, Riebler (2024, Journal of Statistical Software, <doi:10.18637/jss.v110.i03>).

License GPL (>= 2)

URL `https://github.com/ingebogh/makemyprior`

BugReports `https://github.com/ingebogh/makemyprior/issues`

Encoding UTF-8

LazyData true

Imports ggplot2, Matrix, methods, shiny, shinyjs, shinyBS, visNetwork, rlang, MASS

Suggests rstan (>= 2.21.2), INLA (>= 20.03.17), knitr, rmarkdown

Depends R (>= 3.6.0)

Additional_repositories `https://inla.r-inla-download.org/R/stable/`

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author Ingeborg Hem [cre, aut] (<`https://orcid.org/0009-0008-3229-0184`>), Geir-Arne Fuglstad [aut] (<`https://orcid.org/0000-0003-4995-2152`>), Andrea Riebler [aut] (<`https://orcid.org/0009-0004-3998-4032`>)

Maintainer Ingeborg Hem <`ingeborg.hem@ntnu.no`>

Repository CRAN

Date/Publication 2024-08-23 11:10:06 UTC

Contents

compile_stan 3
create_stan_file 4
eval_joint_prior 5
eval_pc_prior 7
expit 8
extract_posterior_effect 8
extract_posterior_parameter 9
find_pc_prior_param 10
get_parameter_order 11
inference_inla 11
inference_stan 12
latin_data 14
logit 15
makemyprior_example_model 16
makemyprior_gui 17
makemyprior_models 18
makemyprior_plotting 19
make_prior 20
mc 21
neonatal_data 23
plot.mmp_prior 24
plot_marginal_prior 25
plot_posterior_fixed 26
plot_posterior_stan 27
plot_posterior_variance 28
plot_prior 29
plot_several_posterior_stan 29
plot_tree_structure 30
print.mmp_prior 31
scale_precmat 32
summary.mmp_prior 32
typical_variance 33
wheat_data 34

Index **36**

`compile_stan` *Compile stan-model*

Description

Function that compiles the stan-code for inference that is included in the model. The compiled version is stored in a .rds-file, which is by default stored in tempdir(). Can also be stored in the package (permanent = TRUE), or in a custom directory. In the latter case, this custom directory must be specified every time inference_stan is called. This will also be done by inference_stan, but this way can save some time if it is not already pre-compiled.

Usage

```
compile_stan(save = FALSE, permanent = FALSE, path = NULL)
```

Arguments

`save`	Whether to save stan-file to location in package or not, defaults to `FALSE`. Must be in interactive session to save the object to a file.
`permanent`	If `TRUE`, the file is stored in the package directory. If `FALSE` (default), the file is saved in `tempdir()`.
`path`	Only used if file cannot be saved in the package folder. This is a path to a folder where the file is stored, do not specify a name for the file! (It will be called `"full_file.rds"`, and should not be changed.) This argument makes the `permanent` argument being ignored.

Details

Note that you will get a message saying something about integer division. The PC priors on variance proportions are represented by splines, and to evaluate them in Stan we look up values, and use integer division for this. This does not cause problems.

Value

Returns the stan-model invisibly.

Examples

```
if (interactive() && requireNamespace("rstan")){
  compile_stan(save = TRUE) # saving in tempdir()
}
```

`create_stan_file` *Create a "skeleton" for custom Stan code*

Description

Makes and saves files with generic code for writing custom Stan code and still use the HD prior.

Usage

```
create_stan_file(save = FALSE, location = "")
```

Arguments

`save`	To confirm that files can be saved (default FALSE).
`location`	Path to location.

Details

Must be in an interactive session to store the code. A folder called `"my_stan_code"` will be created in `location`. If the folder already exists in the specified location, you get an error. The folder contains:

`main_file.stan` Main file. Can put all necessary functions here, but for a cleaner code that is easier to read, we put the functions in separate files.

`jacobian.stan` Function that automatically computes the Jacobian, needed to transform from weights and total variance parameterization to log-variance parameterization.

`prior_distributions.stan` Functions for computing the prior distributions.

The provided code is written so a random intercept model with an intercept, a group effect and a residual effect can be fitted:

`example_custom_stan.R` R script showing how one can fit a random intercept model using the provided code.

The code can be expanded to fit the desired model. This requires some knowledge about Stan. No more documentation is given, as this is merely an offer to users who want to use other models than what are provided in the package already, and will be highly model specific.

Value

Nothing.

Examples

```
## Not run:
create_stan_file(TRUE, "")

## End(Not run)
```

`eval_joint_prior` *Evaluate the joint variance prior*

Description

Function for evaluating the joint variance prior stored in `prior_obj`. To compute the joint prior, the functions needs to know the transformation from the total variance/variance proportion scale to log-variance scale. This is computed before inference, but is not stored in the `mmp_prior`-object. To avoid having to recompute this for every evaluation and thus improve the speed, we make a condensed data object with the function make_eval_prior_data.

Usage

```
eval_joint_prior(theta, prior_data)

make_eval_prior_data(prior_obj)
```

Arguments

`theta`	Vector of log variances. The order of the log variances is the same as specified in the formula, with the residual variance at the end for a Gaussian likelihood. To be sure, you can use get_parameter_order to check the order.
`prior_data`	An object from make_eval_prior_data.
`prior_obj`	Object of class `mmp_prior`, see make_prior.

Details

Note that a Jeffreys' prior is improper and sampling with the prior only will not work when it is used. For sampling from the prior (for example for debugging), use a proper prior for all parameters instead.

The `make_eval_prior_data` function is used to create a condensed version of the prior object from `make_prior`, that only contains what is needed to compute the joint prior. Since the HD prior is chosen on total variances and variance proportions, some additional information is needed to compute the Jacobian for the joint prior. To improve the speed, we do this once before evaluating the prior.

Expert option: `make_eval_prior_data` can also be used to extract the prior to be used with 'regular' INLA. See examples for how this can be done.

Value

Logarithm of the prior density.

Examples

```
ex_model <- makemyprior_example_model()
get_parameter_order(ex_model) # a, b, eps
prior_data <- make_eval_prior_data(ex_model)
eval_joint_prior(c(0, 0, 0), prior_data)
eval_joint_prior(c(-1, 0, 1), prior_data)

# a model with only 2 variance parameters
if (interactive()){

  data <- list(
    a = rep(1:10, each = 10)
  )
  set.seed(1); data$y <- rnorm(10, 0, 0.4)[data$a] + rnorm(100, 0, 1)

  # random intercept model
  ex_model2 <- make_prior(y ~ mc(a), data, family = "gaussian",
                        prior = list(tree = "s2 = (a, eps)",
                                     w = list(s2 = list(prior = "pc0", param = 0.25)),
                                 V = list(s2 = list(prior = "pc", param = c(3, 0.05)))),
                        intercept_prior = c(0, 1000))

  prior_data2 <- make_eval_prior_data(ex_model2)
  # evaluating the prior in a grid
  theta_a <- seq(-8, 4, 0.1)
  theta_eps <- seq(-8, 4, 0.1)
  res <- matrix(nrow = 0, ncol = 3)
  for (ind in 1:length(theta_a)){
    for (jnd in 1:length(theta_eps)){
      res <- rbind(res, c(theta_a[ind], theta_eps[jnd],
                      eval_joint_prior(c(theta_a[ind], theta_eps[jnd]), prior_data2)))
    }
  }

  # graph showing the prior
  if (requireNamespace("ggplot2")){
    res2 <- as.data.frame(res)
    names(res2) <- c("x", "y", "z")
   # Note from the "exp(z)" that we use the posterior, and not log posterior, in this plot
    ggplot(res2, aes(x = x, y = y, z = exp(z), fill = exp(z))) +
      geom_raster() +
      geom_contour(color = "black") +
      scale_fill_viridis_c(option = "E") +
      xlab("Log variance of 'a'") +
      ylab("Log residual variance") +
      labs(fill = "Density") +
      theme_bw()
  }

}

## Not run:
```

```
# How an HD prior can be computed with \code{makemyprior}, and then sent to regular INLA
# (expert option).
# Note the use of the hidden \code{make_jpr}-function.
# Also note that the order of the parameters must be the same as in the call to \code{make_prior}.
# The residual variance is put in the correct place by \code{make_jpr}.
data <- list(
  a = rep(1:10, each = 100),
  b = rep(1:100, times = 10)
)
set.seed(1); data$y <- rnorm(100, 0, 0.4)[data$a] + rnorm(100, 0, 0.6)[data$b] + rnorm(1000, 0, 1)
prior <- make_prior(y ~ mc(a) + mc(b), data, family = "gaussian",
                    prior = list(tree = "s1 = (a, b); s2 = (s1, eps)",
                                 w = list(s2 = list(prior = "pc0", param = 0.25)),
                                 V = list(s2 = list(prior = "pc", param = c(3, 0.05)))),
                    intercept_prior = c(0, 1000))
jpr_dat <- make_eval_prior_data(prior)
res <- inla(y ~ f(a) + f(b),
            data = data,
            control.fixed = list(prec.intercept = 1/1000^2),
            control.expert = list(jp = makemyprior:::make_jpr(jpr_dat)))

## End(Not run)
```

`eval_pc_prior` *Evaluate PC prior for variance proportion*

Description

Evaluate PC prior for a variance proportion.

Usage

```
eval_pc_prior(x, obj, param, logitscale = FALSE)
```

Arguments

`x`	Values to evaluate prior in.
`obj`	Prior object from make_prior
`param`	Which weight to plot, indicated using syntax shown when printing (do not need to include the `w[..]` part to indicate that it is a variance proportion, but can be included). Print the prior object to see syntax for each weight.
`logitscale`	Is the input `x` on logit-scale? (default `FALSE`).

Value

Returns density for the given variance proportion.

Examples

```
ex_prior <- makemyprior_example_model()
eval_pc_prior(seq(0, 1, 0.01), ex_prior, "eps/eps_a_b")
# or:
eval_pc_prior(seq(0, 1, 0.01), ex_prior, "w[eps/eps_a_b]")
```

`expit` *expit*

Description

Calculates inverse logit, exp(x)/(1+exp(x)) = 1/(1+exp(-x))

Usage

```
expit(x)
```

Arguments

`x`	Real number, or vector of reals

Value

expit value

Examples

```
expit(2)
expit(seq(-5, 5, 1))
```

`extract_posterior_effect` *Extract the posterior of a random effect*

Description

Extract the posterior of a random effect in the model for inference done with Stan

Usage

```
extract_posterior_effect(obj, effname)
```

Arguments

`obj`	An object from `inference_stan`.
`effname`	Name of the random effect, same name as in the data.

Value

Returns a matrix with the posterior samples of the chosen effect

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior <- makemyprior_example_model()
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  extract_posterior_effect(res_stan, "a")
}
```

`extract_posterior_parameter`

Extract the posterior parameter estimate

Description

Extract the posterior parameter estimate of a random effect variance or fixed effect coefficient when inference is done with Stan

Usage

```
extract_posterior_parameter(obj, param)
```

Arguments

`obj`	An object from `inference_stan`.
`param`	Name of the variance parameter, which is the same as the name of the corresponding fixed or random effect in the data. Intercept is denoted 'intercept', and residual variance is denoted 'eps'.

Value

Returns a vector with the posterior samples of the chosen parameter, on variance scale for variances parameters and original (the common) scale for fixed effect coefficients

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior <- makemyprior_example_model()
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  extract_posterior_parameter(res_stan, "intercept")
  extract_posterior_parameter(res_stan, "a")
}
```

---

find_pc_prior_param *Find suitable PC prior parameters*

Description

Returns the `U` value in P(`U` > sigma) = `alpha` for a PC prior on standard deviation given an equal-tailed credible interval P(`lower` < `func`(x) < `upper`) = `prob` where `x` is a Gaussian variable with zero mean standard deviation sigma. Note that this function uses sampling.

Usage

```
find_pc_prior_param(
  lower,
  upper,
  alpha = 0.05,
  func = exp,
  N = 10000,
  prob = 0.95
)
```

Arguments

`lower`	lower end of credible interval
`upper`	upper end of credible interval
`alpha`	tail probability of the PC prior (default = `0.05`)
`func`	function to scale Gaussian variables to match desired credible interval scale, default is the exponential function
`N`	number of samples to use when sampling sigma and `x`, default is `1e4`
`prob`	amount of mass to put in the credible interval, default is `0.95`

Value

The `U`-value to pass to the PC prior. NB! Store result to avoid rerunning this function, as it uses sampling. The function also prints (sampled) quantiles for the U-value that is returned.

Examples

```
find_pc_prior_param(0.1, 10)
```

`get_parameter_order` *Internal variance parameter order*

Description

Returns the internal order of the variance parameters related to each random effect in the model.

Usage

```
get_parameter_order(prior_obj)
```

Arguments

`prior_obj`	Object of class `mmp_prior`, see make_prior.

Value

Names of the random effects in the model in the order the prior object reads them.

Examples

```
ex_model <- makemyprior_example_model()
get_parameter_order(ex_model)
```

`inference_inla` *Run inference*

Description

This function helps you run inference with INLA using a prior object from make_prior. You must have INLA installed to run this. INLA can be installed with: `install.packages("INLA", repos = c(getOption("repos"), INLA = "https://inla.r-inla-download.org/R/stable"), dep = TRUE)`. Also see r-inla.org.

Usage

```
inference_inla(prior_obj, use_likelihood = TRUE, print_prior = TRUE, ...)
```

Arguments

`prior_obj`	An object from make_prior, from makemyprior_gui, from inference_stan, or from inference_inla (for refitting model)
`use_likelihood`	Whether to sample from the prior only (FALSE, can be used for e.g. debugging or to look at the priors on variance parameters when using an HD prior, see also Details), or to use the likelihood and data to get the posterior (TRUE, default).
`print_prior`	Whether to print a text with the chosen prior or not (default TRUE)
`...`	Other values to be sent to INLA. Useful arguments include `Ntrials` for the binomial likelihood and `E` (mean E_i in E_i*exp(eta_i)) for the Poisson likelihood. We set the default value of `num.threads` to 1 (this can however be changed). See inla for details. Can be anything sent to inla except for `control.expert` and arguments that specify priors.

Details

Jeffreys' prior is improper. If `use_likelihood = FALSE` and Jeffreys' prior is used for the total variance, the prior will be changed to a Gaussian(0,1) prior on the log total variance. This means that it does not make sense to look at the variances/standard deviations/precisions, but the variance proportions will be correct. Note that this is only an issue when sampling from the prior (i.e., not using the likelihood).

Value

A named list with a prior object (`prior`), an inla-object (`inla`) and some data inla requires (`inla_data`).

Examples

```
## Not run:

vignette("make_prior", package = "makemyprior")

## End(Not run)

ex_prior <- makemyprior_example_model()
if (interactive() && requireNamespace("INLA")){
  posterior <- inference_inla(ex_prior)
  plot(posterior)
}
```

`inference_stan` *Run inference*

Description

This function helps you run inference with rstan using a prior object from make_prior. Note that you must install Stan: `install.packages("rstan")`, see mc-stan.org.

Usage

```
inference_stan(
  prior_obj,
  use_likelihood = TRUE,
  print_prior = TRUE,
  path = NULL,
  ...
)
```

Arguments

`prior_obj`	An object from make_prior, from makemyprior_gui, from inference_stan, or from inference_inla (for refitting model)
`use_likelihood`	Whether to sample from the prior only (`FALSE`, can be used for e.g. debugging or to look at the priors on variance parameters when using an HD prior, see also Details), or to use the likelihood and data to get the posterior (`TRUE`, default).
`print_prior`	Whether to print a text with the chosen prior or not (default `TRUE`)
`path`	Path to folder. See compile_stan. Only necessary if compiled code is stored somewhere else than in `tempdir()` or the package directory (checking `tempdir()` first).
`...`	Other arguments to be sent to sampling. Useful arguments include: `iter` number of iterations for each chain (including burn-in, 2000 is the default) `warmup` number of iterations for the burn-in (default is `iter`/2) `chains` number of chains `init` initial value of the model parameters on internal parameterization (log-variances and covariate coefficients) `seed` seed value for random number generation. See sampling for more details. Note that for inference with `stan`, the `Ntrials` (the size parameter for binomial likelihood) and `E` (the mean E_i in E_i*exp(eta_i) for Poisson likelihood) argument must be included in the data object, and cannot be provided to this function.

Details

We cannot sample from a Jeffreys' prior since it is improper. If `use_likelihood = FALSE` and Jeffreys' prior is used for the total variance, the prior will be changed to a Gaussian(0,1) prior on the log total variance. This means that it does not make sense to look at the variances/standard deviations/precisions, but the variance proportions will be correct. Note that this is only an issue when sampling from the prior (i.e., not using the likelihood).

Value

A named list with a prior object (`prior`), a stan-object (`stan`) and some data stan requires (`stan_data`).

Examples

```
## Not run:

vignette("make_prior", package = "makemyprior")

## End(Not run)

ex_prior <- makemyprior_example_model()
if (interactive() && requireNamespace("rstan")){
  posterior <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  plot(posterior)
}

## Not run:

posterior <- inference_stan(ex_prior, use_likelihood = TRUE, iter = 1e4, chains = 1, seed = 1)
plot(posterior)

## End(Not run)
```

`latin_data` *Latin square experiment data*

Description

Simulated dataset for latin square experiment with 81 observations.

Usage

```
latin_data
```

Format

A list with the following variables:

y Response

lin Covariate for linear effect of treatment

row Row indexes

col Column indexes

treat_iid, treat_rw2 Treatment indexes

Examples

```
## Not run:

vignette("latin_square", package = "makemyprior")

## End(Not run)

if (interactive() && requireNamespace("rstan")){

  formula <- y ~ lin + mc(row) + mc(col) + mc(iid) +
    mc(rw2, model = "rw2", constr = TRUE, lin_constr = TRUE)

  prior <- make_prior(
    formula, latin_data,
    prior = list(tree = "s1 = (rw2, iid);
                                s2 = (row, col, s1); s3 = (s2, eps)",
                 w = list(s1 = list(prior = "pc0", param = 0.25),
                          s2 = list(prior = "dirichlet"),
                          s3 = list(prior = "pc0", param = 0.25))))

  posterior <- inference_stan(prior, iter = 150, warmup = 50,
                              seed = 1, init = "0", chains = 1)
  # Note: For reliable results, increase the number of iterations

  plot(prior)
  plot_tree_structure(prior)
  plot_posterior_fixed(posterior)
  plot_posterior_stan(posterior, param = "prior", prior = TRUE)
}

## Not run:

posterior <- inference_stan(prior, iter = 15000, warmup = 5000,
                            seed = 1, init = "0", chains = 1)

plot(prior)
plot_tree_structure(prior)
plot_posterior_fixed(posterior)
plot_posterior_stan(posterior, param = "prior", prior = TRUE)

## End(Not run)
```

`logit` *logit*

Description

Calculates logit, log(x/(1-x)) = log(x) - log(1-x)

Usage

```
logit(x)
```

Arguments

`x`	Value between 0 and 1, or vector of such values

Value

logit value

Examples

```
logit(0.5)
logit(seq(0, 1, 0.1))
```

`makemyprior_example_model`   *Returning a simple example prior object*

Description

Creating a simple prior object using make_prior. Used in examples of other functions in the package.

Usage

```
makemyprior_example_model(seed = 1)
```

Arguments

`seed`	A seed value for reproducing the data (default `seed = 1`).

Details

See the example for what model is made.

Value

An object of class `mmp_prior`.

Examples

```
ex_model <- makemyprior_example_model()

## Not run:
# The function corresponds to the following model call:

set.seed(1)

data <- list(
  a = rep(1:10, each = 10),
  b = rep(1:10, times = 10)
)
data$y <- rnorm(10, 0, 0.4)[data$a] + rnorm(10, 0, 0.6)[data$b] + rnorm(100, 0, 1)

formula <- y ~ mc(a) + mc(b)

prior <- make_prior(formula, data, family = "gaussian",
                    prior = list(tree = "s1 = (a, b); s2 = (s1, eps)",
                                 w = list(s2 = list(prior = "pc0", param = 0.25)),
                                 V = list(s2 = list(prior = "pc", param = c(3, 0.05)))),
                    intercept_prior = c(0, 1000))

## End(Not run)
```

`makemyprior_gui` *Graphical prior construction*

Description

This functions opens a shiny app where the specified prior can be seen, and changed.

Usage

```
makemyprior_gui(prior, guide = FALSE, no_pc = FALSE)
```

Arguments

`prior`	An object from make_prior.
`guide`	Logical, whether to open the guide directly when the app is started. Default is `FALSE`. The guide can be opened in the app at any time.
`no_pc`	Turn off computation of the PC prior on splits when using the shiny-app, to avoid slow computations. Upon closing, the PC priors will be computed. Default is `FALSE`.

Value

Returns an object that can be sent to inference_stan or inference_inla. Can also be sent to makemyprior_gui again.

Examples

```
## Not run:

vignette("make_prior", package = "makemyprior")


## End(Not run)

if (interactive()){
   ex_prior <- makemyprior_example_model()
   new_prior <- makemyprior_gui(ex_prior)
}
```

`makemyprior_models` *List available priors, latent models and likelihoods*

Description

Prints available priors, latent models and likelihoods to use with make_prior.

Usage

```
makemyprior_models(type = c("prior", "latent", "likelihood"), select = NULL)
```

Arguments

`type`	Which of priors, latent models and likelihoods to list. Options are `prior`, `latent` and `likelihood`
`select`	Which in each group to show details about. `NULL` gives only list what exists (default), `all` gives detailed information about everything in that category. Can also ask for one or more specific priors/latent models/likelihoods

Value

None.

Examples

```
makemyprior_models("prior", c("pc0", "pc1"))
makemyprior_models("latent")
makemyprior_models("likelihood", "all")
```

`makemyprior_plotting` *List of available plotting functions*

Description

Directs to this help page.

Usage

```
makemyprior_plotting()
```

Details

The available plotting functions are:

- plot_marginal_prior
- plot_posterior_fixed
- plot_posterior_precision
- plot_posterior_stan
- plot_posterior_stdev
- plot_posterior_variance
- plot_prior
- plot_several_posterior_stan
- plot_tree_structure

In addition the following functions can be used to extract posterior samples of effects and parameters:

- extract_posterior_effect
- extract_posterior_parameter

eval_pc_prior can be used to evaluate a PC prior for a weight parameter, and eval_joint_prior to evaluate the whole joint prior.

Value

None.

make_prior — *Making a prior object*

Description

Make a prior object with all necessary information about the prior and model. The object can either be sent to makemyprior_gui or used directly for inference with Stan (inference_stan) or INLA (inference_inla). eval_joint_prior can be used to evaluate the prior.

Usage

```
make_prior(
  formula,
  data,
  family = "gaussian",
  prior = list(),
  intercept_prior = c(),
  covariate_prior = list()
)
```

Arguments

`formula`	A formula object, using the function mc.
`data`	The data used in the model, as a `data.frame` or `list`. All elements must have the same length.
`family`	A string indicating the likelihood family. `gaussian` with identity link is the default. `binomial` with logit link and `poisson` with log link are also possible.
`prior`	`tree` The tree structure as a string. A split is specified as `s1 = (a,b)`, where `(s1)` represents a split node and can be any name except names of the input data in `data` and the reserved `(eps)`, which is used for residuals for a Gaussian likelihood. Short names are recommended. Note that these split names are just used in the initial specification, and will not be used later as they are changed by the function automatically. The child nodes for each split are included in parentheses separated by commas, and each split is separated by semicolons. Singletons are included as `(a)`. `V` A named list with information on the priors on each top node and singleton, i.e., all variances. The names in the list are the top node and singleton names from the `tree` argument. Syntax is `V = list(s1 = list(prior = prior_name, param = parameter_vector))`. `w` A named list with information on the priors on each split, i.e., all variance proportions. The names in the list are the split node names from the `tree` argument, and is specified in the same way as the variance priors in `V`. Prior on residuals can be defined using `eps` in this list in the case of a Gaussian likelihood.

`intercept_prior`
: Parameters for Gaussian prior on intercept, specified as a vector with mean and standard deviation. Default is (0, 1000).

`covariate_prior`
: Parameters for Gaussian prior on coefficients of covariates, specified as named list, each element is a vector with mean and standard deviation. Default is (0, 1000).

Details

See makemyprior_models for details on available priors and likelihoods.

Value

Prior object.

Examples

```
## Not run:

vignette("make_prior", package = "makemyprior")

## End(Not run)

p <- 10
m <- 10
n <- m*p

set.seed(1)
data <- list(a = rep(1:p, each = m),
             b = rep(1:m, times = p),
             x = runif(n))
data$y <- data$x + rnorm(p, 0, 0.5)[data$a] +
  rnorm(m, 0, 0.3)[data$b] + rnorm(n, 0, 1)

formula <- y ~ x + mc(a) + mc(b)

prior <- make_prior(formula, data, family = "gaussian",
                    intercept_prior = c(0, 1000),
                    covariate_prior = list(x = c(0, 100)))
prior
plot(prior)
```

`mc` *Define latent component*

Description

Function for defining a latent component for the HD prior package. All model components must be specified, and if an HD prior is used, this is specified later. See make_prior for more details and examples.

Usage

```
mc(
  label,
  model = "iid",
  constr = NULL,
  lin_constr = FALSE,
  Cmatrix = NULL,
  graph = NULL,
  ...
)
```

Arguments

`label`	Name of the component (short names is an advantage as they are used in the app), no default (MUST be provided)
`model`	Type of model, default is "iid" (see list of models: makemyprior_models, `makemyprior_models("latent`
`constr`	Sum-to-zero constraints on component (default TRUE)
`lin_constr`	Linear sum-to-zero constraint, TRUE/FALSE (only for rw2 and only for Stan)
`Cmatrix`	Precision for this component when `model = "generic0"`. We recommend that the matrixes are scaled to the typical variance of the corresponding covariance matrix is 1. This can be done with scale_precmat before sending matrix to mc.
`graph`	Path to graph file for besag effect (see details).
`...`	Additional arguments used for inference. For inference with rstan or inla, the following is useful (especially for the Besag model): `scale.model` if TRUE, the models are scaled so the geometric mean of the variance (typical variance) is 1 And some additional arguments that can be used by inla. Useful arguments include: `rankdef` number defining rank deficiency of the model `extraconstr` extra linear constraints (in addition to `constr`) See f for details.

Details

The graph argument is a path to a file describing neighbouring relationship on the following form: First row: number of elements The rest: First number is the index of this element, second is the number of neighbours, the rest is the index numbers of all neighbours for this element. If element 1 and 4 are neighbours, 1 should have 4 in its neighbour list, and 4 should have 1.

Value

For specifying details on this latent component.

Examples

```
## Not run:

vignette("make_prior", package = "makemyprior")

## End(Not run)
```

`neonatal_data` *Neonatal mortality data*

Description

Simulated neonatal mortality data with 323 observations.

Usage

```
neonatal_data
```

Format

A list with the following variables:

y Response

Ntrials Number of trials for each cluster

urban Covariate indicating if cluster is urban (1) or rural (0)

nu Cluster effect indexes

v County effect indexes for iid effect

u County effect indexes for Besag effect

Examples

```
## Not run:

vignette("neonatal_mortality", package = "makemyprior")

## End(Not run)

if (interactive() && requireNamespace("rstan")){

  graph_path <- paste0(path.package("makemyprior"), "/neonatal.graph")

  formula <- y ~ urban + mc(nu) + mc(v) +
    mc(u, model = "besag", graph = graph_path, scale.model = TRUE)

  set.seed(1)
  find_pc_prior_param(lower = 0.1, upper = 10, prob = 0.9, N = 2e5)

  prior <- make_prior(
    formula, neonatal_data, family = "binomial",
    prior = list(tree = "s1 = (u, v); s2 = (s1, nu)",
                 w = list(s1 = list(prior = "pc0", param = 0.25),
                          s2 = list(prior = "pc1", param = 0.75)),
                 V = list(s2 = list(prior = "pc",
                                    param = c(3.35, 0.05)))))

  posterior <- inference_stan(prior, iter = 150, warmup = 50,
                              seed = 1, init = "0", chains = 1)
  # Note: For reliable results, increase the number of iterations

  plot(prior)
  plot_tree_structure(prior)
  plot_posterior_fixed(posterior)
  plot_posterior_stan(posterior, param = "prior", prior = TRUE)
}

## Not run:

posterior <- inference_stan(prior, iter = 15000, warmup = 5000,
                            seed = 1, init = "0", chains = 1)

plot(prior)
plot_tree_structure(prior)
plot_posterior_fixed(posterior)
plot_posterior_stan(posterior, param = "prior", prior = TRUE)

## End(Not run)
```

`plot.mmp_prior` *Plotting*

Description

Plotting

Usage

```
## S3 method for class 'mmp_prior'
plot(x, ...)

## S3 method for class 'mmp_inla'
plot(x, ...)
```

```
## S3 method for class 'mmp_stan'
plot(x, ...)
```

Arguments

`x`	Object of class `mmp_prior`, `mmp_inla` or `mmp_stan`.
`...`	Additional arguments to plotting functions. Varies with what object is sent to function.

Details

See plot_prior (objects of class `mmp_prior`), plot_posterior_stan (objects of class `mmp_stan`), and plot_posterior_variance (objects of class `mmp_inla`),

Value

None.

Examples

```
pri <- makemyprior_example_model()
plot(pri)

if (interactive() && requireNamespace("rstan")){
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  plot(res_stan)
}

if (interactive() && requireNamespace("INLA")){
  res_inla <- inference_inla(pri)
 plot(res_inla)
}
```

`plot_marginal_prior` *Plotting prior for a single parameter (weight or variance (not standard deviation))*

Description

Following the parameterization of the prior.

Usage

```
plot_marginal_prior(x, obj, param, sd = FALSE)
```

Arguments

`x`	Values to evaluate prior in.
`obj`	An object from `make_prior`, `inference_stan`, `inference_inla`, or `makemyprior_gui`
`param`	Name of parameter to plot (see `print(obj)` for syntax). Note that only variances will be plotted, so `V[..]` and `sigma^2[..]` must be used to indicate those parameters.
`sd`	Whether to plot variance parameters on the standard deviation (`TRUE`) or variance (`FALSE`, default) scale

Value

A ggplot with the posterior distribution. See also makemyprior_plotting.

Examples

```
ex_prior <- makemyprior_example_model()
plot_marginal_prior(seq(0, 1, 0.001), ex_prior, "w[a/a_b]")
plot_marginal_prior(seq(0, 1, 0.001), ex_prior, "w[eps/eps_a_b]")
plot_marginal_prior(seq(0, 5, 0.01), ex_prior, "V[eps_a_b]")
```

`plot_posterior_fixed` *Plotting posterior distributions*

Description

Function for plotting the posterior distributions of the coefficients of the fixed effects

Usage

```
plot_posterior_fixed(obj)
```

Arguments

`obj`	An object from `inference_stan` or `inference_inla`

Value

A ggplot with the posterior distributions. See also makemyprior_plotting.

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior <- makemyprior_example_model()
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  plot_posterior_fixed(res_stan)
}
```

plot_posterior_stan *Plotting posterior distributions*

Description

Function for plotting the posterior distributions of the random effect variances on the scale of the tree parameterization.

Usage

```
plot_posterior_stan(
  obj,
  param = c("prior", "variance", "stdev", "precision"),
  prior = FALSE
)
```

Arguments

obj	An object from `inference_stan`
param	A string indicating parameterization of plot. `"prior"` for scale of parameters, `"variance"`, `"stdev"` and `"precision"` also possible.
prior	Include prior in the plot? Only possible for `param = "prior"`. Note that if Jeffreys' prior is used for the total variance, it will not be included in the plot.

Value

A ggplot with the posterior distributions. See also makemyprior_plotting.

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior <- makemyprior_example_model()
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  plot_posterior_stan(res_stan)
}
```

`plot_posterior_variance` *Plotting posterior variances, standard deviations or precisions*

Description

Plotting posterior variances, standard deviations or precisions

Usage

```
plot_posterior_variance(obj)

plot_posterior_stdev(obj)

plot_posterior_precision(obj)
```

Arguments

`obj`	An object from `inference_stan` or `inference_inla`.

Value

A ggplot object with the plot See also makemyprior_plotting.

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior <- makemyprior_example_model()
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  plot_posterior_variance(res_stan)
}

if (interactive() && requireNamespace("INLA")){
  ex_prior <- makemyprior_example_model()
  res_inla <- inference_inla(ex_prior)
  plot_posterior_variance(res_inla)
}
```

`plot_prior` *Plotting prior distributions*

Description

Function for plotting the prior distributions of the random effects on the scale of the parameters chosen

Usage

```
plot_prior(obj)
```

Arguments

`obj`	An object from `make_prior`, `inference_stan`, `inference_inla`, or `makemyprior_gui`

Value

A ggplot with the prior distributions. See also makemyprior_plotting.

Examples

```
ex_prior <- makemyprior_example_model()
plot_prior(ex_prior)
```

`plot_several_posterior_stan` *Plotting several posterior distributions*

Description

Function for plotting the posterior distributions of the random effect variances on the scale of the tree parameterization.

Usage

```
plot_several_posterior_stan(
  objs,
  param = c("prior", "variance", "stdev", "precision", "logvariance")
)
```

Arguments

`objs`	A names list with objects of class `mmp_stan` from `inference_stan`, can be any length (but typically length two for one prior (`use_likelihood = FALSE`) and posterior, or two posteriors).
`param`	A string indicating parameterization of plot. `"prior"` for scale of parameters, `"variance"`, `"stdev"`, `"precision"` and `"logvariance"` are also possible.

Details

We cannot sample from a Jeffreys' prior since it is improper. If Jeffreys' prior is used for the total variance, the prior will be changed to a Gaussian(0,1) prior on the log total variance. This means that it does not make sense to look at the variances/standard deviations/precisions, but the variance proportions will be correct. See also makemyprior_plotting.

Value

A ggplot with the posterior distributions.

Examples

```
if (interactive() && requireNamespace("rstan")){
  ex_prior1 <- makemyprior_example_model(seed = 1)
  ex_prior2 <- makemyprior_example_model(seed = 2)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  res_stan1 <- inference_stan(ex_prior1, iter = 100)
  res_stan2 <- inference_stan(ex_prior2, iter = 100)
  plot_several_posterior_stan(list(One = res_stan1, Two = res_stan2))
}
```

`plot_tree_structure` *Plotting the prior tree structure graph*

Description

Can only be used for visualization in R.

Usage

```
plot_tree_structure(obj, nodenames)
```

Arguments

`obj`	An object from `make_prior`, `inference_stan`, `inference_inla`, or `makemyprior_gui`
`nodenames`	Custom names for each node (optional). Given as a named list with the default names as list names, and the new names as list elements. Do not need to provide all.

Value

A visNetwork with the tree graph See also makemyprior_plotting.

Examples

```
ex_prior <- makemyprior_example_model()
plot_tree_structure(ex_prior)
```

---

print.mmp_prior *Print*

Description

Print

Usage

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

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

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

Arguments

x	Object of class mmp_prior, mmp_inla or mmp_stan.
...	For mmp_stan and mmp_inla, see print.data.frame.

Value

Returns input object invisible.

Examples

```
pri <- makemyprior_example_model()
pri # or print(pri)

if (interactive() && requireNamespace("rstan")){
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  res_stan # or print(res_stan)
}

if (interactive() && requireNamespace("INLA")){
  res_inla <- inference_inla(pri)
  res_inla # or print(res_inla)
}
```

`scale_precmat` *Scaling precision matrix*

Description

Scaling a precision matrix so the corresponding covariance matrix has typical variance (geometric mean) equal to 1.

Usage

```
scale_precmat(Q)
```

Arguments

`Q`	Precision matrix

Value

Precision matrix which is now scaled to have typical variance 1.

Examples

```
scale_precmat(diag(10))
```

`summary.mmp_prior` *Short summary*

Description

Short summary

Usage

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

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

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

Arguments

`object`	Object of class `mmp_prior`, `mmp_inla` or `mmp_stan`.
`...`	For `mmp_stan` and `mmp_inla`, see print.data.frame.

Value

Returns summary invisible.

Examples

```
pri <- makemyprior_example_model()
summary(pri)

if (interactive() && requireNamespace("rstan")){
  res_stan <- inference_stan(ex_prior, iter = 100)
  # Note: For reliable results, increase the number of iterations (e.g., 'iter = 2000')
  summary(res_stan)
}

if (interactive() && requireNamespace("INLA")){
  res_inla <- inference_inla(pri)
  summary(res_inla)
}
```

`typical_variance` *Compute the typical variance*

Description

Computing the typical variance (geometric mean) of a matrix.

Usage

```
typical_variance(mat)
```

Arguments

`mat`	Matrix.

Value

Typical variance.

Examples

```
typical_variance(diag(10))
```

wheat_data *Genomic wheat breeding model data*

Description

Simulated wheat yield data with 100 observations.

Usage

```
wheat_data
```

Format

A list with the following variables

y Response

a, b, x Indexes for the additive, dominance and epistasis genetic effects, respectively

Q_a, Q_d, Q_x Precision matrices for the genetic effects

Examples

```
## Not run:

vignette("wheat_breeding", package = "makemyprior")

## End(Not run)

if (interactive() && requireNamespace("rstan")){

  wheat_data_scaled <- wheat_data
  wheat_data_scaled$Q_a <- scale_precmat(wheat_data$Q_a)
  wheat_data_scaled$Q_d <- scale_precmat(wheat_data$Q_d)
  wheat_data_scaled$Q_x <- scale_precmat(wheat_data$Q_x)

  formula <- y ~
    mc(a, model = "generic0", Cmatrix = Q_a, constr = TRUE) +
    mc(d, model = "generic0", Cmatrix = Q_d, constr = TRUE) +
    mc(x, model = "generic0", Cmatrix = Q_x, constr = TRUE)

  prior <- make_prior(formula, wheat_data_scaled, prior = list(
    tree = "s1 = (d, x); s2 = (a, s1); s3 = (s2, eps)",
    w = list(s1 = list(prior = "pcM", param = c(0.67, 0.8)),
             s2 = list(prior = "pcM", param = c(0.85, 0.8)),
             s3 = list(prior = "pc0", param = 0.25))))

  posterior <- inference_stan(prior, iter = 150, warmup = 50,
                              chains = 1, seed = 1)
  # Note: For reliable results, increase the number of iterations
  plot(prior)
  plot_tree_structure(prior)
  plot_posterior_fixed(posterior)
  plot_posterior_stan(posterior, param = "prior", prior = TRUE)

}

## Not run:

posterior <- inference_stan(prior, iter = 150, warmup = 50,
                            chains = 1, seed = 1)

plot(prior)
plot_tree_structure(prior)
plot_posterior_fixed(posterior)
plot_posterior_stan(posterior, param = "prior", prior = TRUE)

## End(Not run)
```

Index

∗ **INLA**
  inference_inla, 11
∗ **datasets**
  latin_data, 14
  neonatal_data, 23
  wheat_data, 34
∗ **formula**
  mc, 21
∗ **gui**
  makemyprior_gui, 17
∗ **plot**
  plot_marginal_prior, 25
  plot_posterior_fixed, 26
  plot_posterior_stan, 27
  plot_posterior_variance, 28
  plot_prior, 29
  plot_several_posterior_stan, 29
  plot_tree_structure, 30
∗ **posterior**
  extract_posterior_effect, 8
  extract_posterior_parameter, 9
∗ **prior**
  make_prior, 20
∗ **rstan**
  inference_stan, 12

compile_stan, 3, *13*
create_stan_file, 4

eval_joint_prior, 5, *19, 20*
eval_pc_prior, 7, *19*
expit, 8
extract_posterior_effect, 8, *19*
extract_posterior_parameter, 9, *19*

f, *22*
find_pc_prior_param, 10

get_parameter_order, *5*, 11
ggplot, *26–30*

inference_inla, 11, *12, 13*, *17*, *20*
inference_stan, *3*, *12*, 12, *13*, *17*, *20*
inla, *12*, *22*

latin_data, 14
logit, 15

make_eval_prior_data, *5*
make_eval_prior_data
  *(*eval_joint_prior*)*, 5
make_prior, *5*, *7*, *11–13*, *16–18*, 20, *22*
makemyprior_example_model, 16
makemyprior_gui, *12, 13*, *17*, 17, *20*
makemyprior_models, 18, *21, 22*
makemyprior_plotting, 19, *26–31*
mc, *20*, 21, *22*

neonatal_data, 23

plot.mmp_inla *(*plot.mmp_prior*)*, 24
plot.mmp_prior, 24
plot.mmp_stan *(*plot.mmp_prior*)*, 24
plot_marginal_prior, *19*, 25
plot_posterior_fixed, *19*, 26
plot_posterior_precision, *19*
plot_posterior_precision
  *(*plot_posterior_variance*)*, 28
plot_posterior_stan, *19*, *25*, 27
plot_posterior_stdev, *19*
plot_posterior_stdev
  *(*plot_posterior_variance*)*, 28
plot_posterior_variance, *19*, *25*, 28
plot_prior, *19*, *25*, 29
plot_several_posterior_stan, *19*, 29
plot_tree_structure, *19*, 30
print.data.frame, *31*, *33*
print.mmp_inla *(*print.mmp_prior*)*, 31
print.mmp_prior, 31
print.mmp_stan *(*print.mmp_prior*)*, 31

rstan, *12*, *22*

sampling, *13*
scale_precmat, *22*, 32
summary.mmp_inla *(*summary.mmp_prior*)*, 32
summary.mmp_prior, 32
summary.mmp_stan *(*summary.mmp_prior*)*, 32

typical_variance, 33

visNetwork, *31*

wheat_data, 34