Package 'hbsaems'

July 22, 2025

Encoding UTF-8

Type Package

Title Hierarchical Bayes Small Area Estimation Model using 'Stan'

Version 0.1.1

Maintainer Achmad Syahrul Choir <madsyair@stis.ac.id>

Description Implementing Hierarchical Bayesian Small Area Estimation models using the 'brms' package as the computational backend. The modeling framework follows the methodological foundations described in area-level models. This package is designed to facilitate a principled Bayesian workflow, enabling users to conduct prior predictive checks, model fitting, posterior predictive checks, model comparison, and sensitivity analysis in a coherent and reproducible manner. It supports flexible model specifications via 'brms' and promotes transparency in model development, aligned with the recommendations of modern Bayesian data analysis practices, implementing methods described in Rao and Molina (2015) <doi:10.1002/9781118735855>.

Depends R (>= 3.6.0), brms

Imports bayesplot, bridgesampling, coda, DT, energy, ggplot2, grDevices, mice, minerva, priorsense, posterior, readxl, shiny, shinyWidgets, shinydashboard, stats, tools, XICOR

Suggests spelling, kableExtra, knitr, rmarkdown, Matrix, mockery, testthat (>= 3.0.0)

License GPL (>= 3)

URL https://github.com/madsyair/hbsaems

BugReports https://github.com/madsyair/hbsaems/issues

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Achmad Syahrul Choir [aut, cre] (ORCID: <https://orcid.org/0000-0001-7088-0646>),
Saniyyah Sri Nurhayati [aut],
Sofi Zamzanah [aut],
Arsyka Laila Oktalia Siregar [aut]

Repository CRAN

Date/Publication 2025-07-18 14:40:02 UTC

Contents

`adjacency_matrix_car` *Adjacency Matrix for Conditional Autoregressive (CAR)*

Description

The `adjacency_matrix_car` contains a symmetric adjacency matrix used for demonstrating *Hierarchical Bayesian Small Area Estimation* (HB SAE) with spatial structure under the Conditional Autoregressive (CAR) model. The matrix consists of 0s and 1s to represent neighborhood relationships between areas, where a value of 1 indicates direct adjacency. This structure is commonly used in CAR models to specify spatial dependence based on local neighborhood connections.

Usage

```
adjacency_matrix_car
```

Format

A 5×5 numeric matrix with row and column names `"1"` to `"5"`.

Source

Simulated example created for illustrating spatial structure in CAR models.

`data_betalogitnorm` *Simulation Data for Beta Logit Normal Model*

Description

The `data_betalogitnorm` is a simulation data created specifically to demonstrate the implementation of *Hierarchical Bayesian Small Area Estimation* (HB SAE) with Beta distribution. This data is suitable for testing Beta regression models with a hierarchical structure between areas. This data is also equipped with variables that apply spatial effects.

Usage

```
data_betalogitnorm
```

Format

A data frame with 100 rows and 9 variables:

y Response variable - The proportion of simulation results, has a value between 0 and 1, follows a Beta distribution.

theta True latent mean parameter on the logit scale, representing the underlying probability of success in each area.

x1, x2, x3 Predictors Variables

n The number of sample units for each region used in the survey

deff Design Effect

group Area ID (1–100) Random effects formula specifying the grouping structure in the data.

sre An optional grouping factor mapping observations to spatial locations.

Source

Simulated data based on a Beta-Logit-Normal model.

Examples

```
data(data_betalogitnorm)
head(data_betalogitnorm)
```

`data_binlogitnorm` *Simulated Binomial–Logit-Normal data (area-level)*

Description

The `data_binlogitnorm` dataset contains simulated data for 50 areas based on a Binomial–Logit-Normal model. It includes area-level covariates, true probability parameters, sample sizes, observed counts, direct estimators, sampling variances, and true latent values.

Usage

```
data_binlogitnorm
```

Format

A data frame with 50 rows and 13 variables:

n Sample size per area

y Observed success count per area

p Direct estimator of proportion

x1, x2, x3 Auxiliary area-level covariates

u_true True area-level random effect

eta_true True linear predictor (logit scale)

p_true True probability per area

psi_i Sampling variance of logit-transformed direct estimator

y_obs Simulated noisy version of eta (logit scale)

p_obs Estimated proportion via inverse logit of y_obs

group Area ID (1–100) for random effects formula specifying the grouping structure in the data.

sre An optional grouping factor mapping observations to spatial locations.

Details

This dataset is intended for evaluating small area estimation models under Binomial–Logit-Normal assumptions.

Source

Simulated data based on a Binomial–Logit-Normal model `data_fhnorm` *Simulated Fay-Herriot Normal Data (Area-Level)*

Description

The `data_fhnorm` dataset contains simulated data for n areas based on a Fay-Herriot Normal model. It includes area-level covariates, true latent values, direct estimators, sampling variances, and area random effects. This dataset is intended for evaluating small area estimation models under normality assumptions.

Usage

```
data_fhnorm
```

Format

A data frame with n rows and 8 variables:

y Observed direct estimator per area.

D Sampling variance for the direct estimator.

x1, x2, x3 Auxiliary area-level covariates.

theta_true True latent mean parameter per area.

u Area-level random effect.

group Area ID (1–n) specifying the grouping structure.

sre Optional grouping factor mapping observations to spatial locations (e.g. regions).

Source

Simulated data based on the Fay-Herriot Normal model.

Examples

```
data(data_fhnorm)
head(data_fhnorm)
```

`data_lnln` *Simulation Data for Lognormal-Lognormal Model*

Description

This dataset is a simulated data created for demonstrating the implementation of *Hierarchical Bayesian Small Area Estimation* (HB SAE) using a **lognormal-lognormal model**. It includes area-level covariates, random effects, direct estimates, and spatial components, for testing SAE models with lognormal assumptions and spatial correlation.

Usage

```
data_lnln
```

Format

A data frame with 100 rows and 13 variables:

group Area ID (1–100) for random effects formula specifying the grouping structure in the data.

x1, x2, x3 Auxiliary area-level covariates

u_true True unstructured area-level random effect on the log scale.

teta_true True linear predictor on the log scale (meanlog for lognormal distribution).

mu_orig_true True mean on the original scale, calculated from `eta_true` and `sigma_e`.

n Sample size per area.

y_obs Simulated observed mean per area, generated from a lognormal distribution.

lambda_dir Direct estimator of the mean per area (same as `y_obs`).

y_log_obs Log-transformed direct estimator.

psi_i Approximate sampling variance of `y_obs`.

sre An optional grouping factor mapping observations to spatial locations.

Source

Simulated data based on a Lognormal–Lognormal model.

hbcc *hbcc : Hierarchical Bayesian Convergence Checks*

Description

This function is designed to evaluate the convergence and quality of a Bayesian hierarchical model. It performs several diagnostic tests and generates various plots to assess Markov Chain Monte Carlo performance.

Usage

```
hbcc(
  model,
  diag_tests = c("rhat", "geweke", "heidel", "raftery"),
  plot_types = c("trace", "dens", "acf", "nuts_energy", "rhat", "neff")
)
```

Arguments

`model`	A `brmsfit` or `hbmfit` object.
`diag_tests`	Character vector of diagnostic tests (default:"rhat", "geweke", "raftery", "heidel")
`plot_types`	Character vector of plot types (default: trace","dens","acf", "nuts_energy", "rhat", "neff")

Details

Hierarchical Bayesian Convergence Checks

Value

An object of class `hbcc_results`, which is a list containing:

`rhat_ess`	Matrix of `Rhat`, `Bulk_ESS`, and `Tail_ESS` values for fixed and random effects.
`geweke`	Geweke diagnostic results (if selected).
`raftery`	Raftery-Lewis diagnostic results (if selected).
`heidel`	Heidelberger-Welch diagnostic results (if selected).
`plots`	A list of generated MCMC diagnostic plots, which may include:

- `"trace"` - Trace plot of the MCMC chains.
- `"dens"` - Density plot of the posterior distributions.
- `"acf"` - Autocorrelation function plot.
- `"nuts_energy"` - NUTS energy diagnostic plot.
- `"rhat"` - Rhat plot (if available).
- `"neff"` - Effective sample size plot.

Author(s)

Achmad Syahrul Choir and Saniyyah Sri Nurhayati

References

Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28.

Examples

```
library(hbsaems)
data("data_fhnorm")

# Prepare the dataset
data <- data_fhnorm

# Fit the Basic Model
model <- hbm(
formula = bf(y ~ x1 + x2 + x3), # Formula model
hb_sampling = "gaussian", # Gaussian family for continuous outcomes
hb_link = "identity", # Identity link function (no transformation)
data = data, # Dataset
chains = 4, # Number of MCMC chains
iter = 4000, # Total MCMC iterations
warmup = 2000, # Number of warmup iterations
cores = 2 # Parallel processing
)
summary(model)

# Convergence Checks
hbcc(model)
```

---

hbm *hbm : Hierarchical Bayesian Small Area Models*

Description

This function provide flexible modeling approaches to estimate area-level statistics while incorporating auxiliary information and spatial structures. This function allows users to fit Bayesian models using the brms package and supports Gaussian, Bernoulli, Poisson, and other distributions. It also accommodates spatial random effects (CAR and SAR) and missing data handling (deletion, model-based imputation, and multiple imputation).

Usage

```
hbm(
  formula,
  hb_sampling = "gaussian",
  hb_link = "identity",
  link_phi = "log",
  re = NULL,
  sre = NULL,
  sre_type = NULL,
  car_type = NULL,
  sar_type = NULL,
  M = NULL,
  data,
  prior = NULL,
  handle_missing = NULL,
  m = 5,
  control = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  cores = 1,
  sample_prior = "no",
  ...
)
```

Arguments

`formula`	Formula specifying the model structure of auxiliary variables and direct estimates The formula must be provided as a `brmsformula` or `formula` object. For multivariate models with multiple auxiliary variables, use the + operator to combine multiple `bf()` formulas. Example: `formula(y ~ x1 + x2 + x3)`, `bf(y ~ x1 + x2 + x3)`, or `bf(y	mi() ~ mi(x1)) + bf(x1	mi() ~ x2)`
`hb_sampling`	A character string naming the distribution family of the response variable to be used in the model (e.g., "gaussian", "bernoulli", "poisson")
`hb_link`	A specification for the model link function. This can be a name/expression or character string. See the 'Details' section for more information on link functions supported by each family.
`link_phi`	Link function for the second parameter (phi), typically representing precision, shape, or dispersion depending on the family used (e.g., "log", "identity")
`re`	Random effects formula specifying the grouping structure in the data. For example, re = ~(1|area), where "area" is the grouping variable or cluster ID indicating that observations within the same area share a common random effect. If not specified, each row will be treated as its own group, meaning a separate random effect is estimated for each observation.
`sre`	An optional grouping factor mapping observations to spatial locations. If not specified, each observation is treated as a separate location. It is recommended

	to always specify a grouping factor to allow for handling of new data in post-processing methods.
`sre_type`	Determines the type of spatial random effect used in the model. The function currently supports "sar" and "car"
`car_type`	Type of the CAR structure. Currently implemented are "escar" (exact sparse CAR), "esicar" (exact sparse intrinsic CAR), "icar" (intrinsic CAR), and "bym2".
`sar_type`	Type of the SAR structure. Either "lag" (for SAR of the response values) or "error" (for SAR of the residuals).
`M`	The M matrix in SAR is a spatial weighting matrix that shows the spatial relationship between locations with certain weights, while in CAR, the M matrix is an adjacency matrix that only contains 0 and 1 to show the proximity between locations. SAR is more focused on spatial influences with different intensities, while CAR is more on direct adjacency relationships. If sre is specified, the row names of M have to match the levels of the grouping factor
`data`	Dataset used for model fitting
`prior`	Priors for the model parameters (default: `NULL`). Should be specified using the `brms::prior()` function or a list of such objects. For example, `prior = prior(normal(0, 1), class = "b")` sets a Normal(0,1) prior on the regression coefficients. Multiple priors can be combined using `c()`, e.g., `prior = c(prior(normal(0, 1), class = "b"), prior(exponential(1), class = "sd"))`. If `NULL`, default priors from `brms` will be used.
`handle_missing`	Mechanism to handle missing data (NA values) to ensure model stability and avoid estimation errors. Three approaches are supported. The `"deleted"` approach performs complete case analysis by removing all rows with any missing values before model fitting. This is done using a simple filter such as `complete.cases(data)`. It is recommended when the missingness mechanism is Missing Completely At Random (MCAR). The `"multiple"` approach applies multiple imputation before model fitting. Several imputed datasets are created (e.g., using the `mice` package or the `brm_multiple()` function in `brms`), the model is fitted separately to each dataset, and the results are combined. This method is suitable when data are Missing At Random (MAR). The `"model"` approach uses model-based imputation within the Bayesian model itself. Missing values are incorporated using the `mi()` function in the model formula (e.g., `y ~ mi(x1) + mi(x2)`), allowing the missing values to be jointly estimated with the model parameters. This method also assumes a MAR mechanism and is applicable only for continuous variables. If data are suspected to be Missing Not At Random (MNAR), none of the above approaches directly apply. Further exploration, such as explicitly modeling the missingness process or conducting sensitivity analyses, is recommended.
`m`	Number of imputations to perform when using the `"multiple"` approach for handling missing data (default: 5). This parameter is only used if `handle_missing = "multiple"`. It determines how many imputed datasets will be generated. Each imputed dataset is analyzed separately, and the posterior draws are then combined to account for both within-imputation and between-imputation variability, following Rubin's rules. A typical choice is between 5 and 10 imputations, but more may be needed for higher missingness rates.

`control`	A list of control parameters for the sampler (default: `list()`)
`chains`	Number of Markov chains (default: 4)
`iter`	Total number of iterations per chain (default: 4000)
`warmup`	Number of warm-up iterations per chain (default: floor(iter/2))
`cores`	Number of CPU cores to use (default: 1)
`sample_prior`	Character. Indicates whether draws from priors should be sampled in addition to posterior draws. The options are: `"no"` (default): Do not draw from priors (only posterior draws are obtained). `"yes"`: Draw both from the prior and posterior. `"only"`: Draw solely from the prior, ignoring the likelihood. which allows among others to generate draws from the prior predictive distribution.
`...`	Additional arguments

Details

Hierarchical Bayesian Small Area Models

Value

A `hbmfit` object containing :

`model`	Summary of `brms` object.
`handle_missing`	Handle missing option used in the model.
`data`	Data passed to the `hbm` function.

Author(s)

Achmad Syahrul Choir, Saniyyah Sri Nurhayati, and Sofi Zamzanah

References

Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons. Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28.

Examples

```
# Load the example dataset
library(hbsaems)
data("data_fhnorm")

# Prepare the dataset
data <- data_fhnorm

# Fit the Basic Model
model <- hbm(
formula = bf(y ~ x1 + x2 + x3), # Formula model
hb_sampling = "gaussian", # Gaussian family for continuous outcomes
hb_link = "identity", # Identity link function (no transformation)
data = data, # Dataset
chains = 4, # Number of MCMC chains
iter = 4000, # Total MCMC iterations
warmup = 2000, # Number of warmup iterations
cores = 2 # Parallel processing
)
summary(model)

# Fit the Basic Model With Defined Random Effect
model_with_defined_re <- hbm(
formula = bf(y ~ x1 + x2 + x3), # Formula model
hb_sampling = "gaussian", # Gaussian family
hb_link = "identity", # Identity link
re = ~(1 | group), # Defined random effect
data = data,
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
summary(model_with_defined_re)

# Fit the Model with Missing Data
# a. Handling missing by deletion
data_miss <- data
data_miss$y[3:5] <- NA
model_deleted <- hbm(
formula = bf(y ~ x1 + x2 + x3),
hb_sampling = "gaussian",
hb_link = "identity",
re = ~(1 | group),
data = data,
handle_missing = "deleted",
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
summary(model_deleted)

# b. Handling missing using multiple imputation
model_multiple <- hbm(
formula = bf(y ~ x1 + x2 + x3),
hb_sampling = "gaussian",
hb_link = "identity",
re = ~(1 | group),
data = data,
handle_missing = "multiple",
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
```

```
summary(model_multiple)

# c. Handling missing during modeling
data_miss$y[3:5] <- NA
data_miss$x1[6:7] <- NA
model_model <- hbm(
formula = bf(y | mi() ~ mi(x1) + x2 + x3) +
bf(x1 | mi() ~ x2 + x3),
hb_sampling = "gaussian",
hb_link = "identity",
re = ~(1 | group),
data = data,
handle_missing = "model",
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
summary(model_model)

# Fit the Model with Spatial Effect
# a. CAR (Conditional Autoregressive)
data("adjacency_matrix_car")
adjacency_matrix_car

model_spatial_car <- hbm(
formula = bf(y ~ x1 + x2 + x3 ),
hb_sampling = "gaussian",
hb_link = "identity",
data = data,
sre = "sre",
sre_type = "car",
M = adjacency_matrix_car,
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
summary(model_spatial_car)

# b. SAR (Simultaneous Autoregressive)
data("spatial_weight_sar")
spatial_weight_sar

model_spatial_sar <- hbm(
formula = bf(y ~ x1 + x2 + x3 ),
hb_sampling = "gaussian",
hb_link = "identity",
data = data,
sre_type = "sar",
M = spatial_weight_sar,
chains = 4,
iter = 4000,
warmup = 2000,
cores = 2
)
```

---

hbmc *hbmc: Check Model Goodness of Fit and Prior Sensitivity*

Description

This function computes model fit diagnostics (WAIC, LOO) and performs posterior predictive checks for the primary model. If the 'model' argument is a list containing two or more brmsfit/hbmfit objects, it performs model comparison using selected metrics (LOO, WAIC, Bayes Factor) between the first model and each subsequent model in the list. Prior sensitivity analysis can be run for each model if specified. LOO calculation includes checks for problematic Pareto k values. If k > 0.7 and moment_match was FALSE, a warning is issued. If k > 0.7 and moment_match was TRUE, reloo is attempted. Additional arguments for moment matching and reloo can be passed via `moment_match_args` and `reloo_args`.

Usage

```
hbmc(
  model,
  ndraws_ppc = 100,
  moment_match = FALSE,
  moment_match_args = list(),
  reloo_args = list(),
  plot_types = c("pp_check", "params"),
  comparison_metrics = c("loo", "waic", "bf"),
  run_prior_sensitivity = FALSE,
  sensitivity_vars = NULL
)
```

Arguments

`model`	A single `brmsfit/hbmfit` object, or a list of `brmsfit/hbmfit` objects.
`ndraws_ppc`	Number of draws to plot in the posterior predictive check (default: 100).
`moment_match`	Logical; if `TRUE`, use moment matching for LOO computation directly. If `FALSE` (default), and problematic Pareto k values are detected, a warning will suggest re-running with `moment_match = TRUE`.
`moment_match_args`	A list of additional arguments to pass to `brms::loo` when `moment_match = TRUE`. Default is `list()`.
`reloo_args`	A list of additional arguments to pass to `brms::loo` when `reloo = TRUE` is attempted. Default is `list()`. Note that `k_threshold` is already set to 0.7 internally when reloo is attempted.

`plot_types` Character vector specifying types of plots to generate for the primary model (default: c("pp_check", "params")).

`comparison_metrics`
A character vector specifying which metrics to compute for model comparison when multiple models are provided. Options are "loo", "waic", "bf" (Bayes Factor). Default is `c("loo", "waic", "bf")` to compute all. If NULL or empty, no comparison metrics will be computed.

`run_prior_sensitivity`
Logical; if `TRUE`, attempt to run prior sensitivity analysis for each model (default: `FALSE`).

`sensitivity_vars`
A character vector of parameter names for which to run sensitivity analysis. Required if `run_prior_sensitivity` is `TRUE`.

Details

Check Model Goodness of Fit and Prior Sensitivity

Value

An object of class `hbmc_results`, which is a list containing:

`primary_model_diagnostics`
A list containing diagnostics for the primary model:

- `loo`: LOO diagnostics. May include notes about Pareto k values or reloo application.
- `waic`: WAIC value.
- `pp_check_plot`: Posterior predictive check plot.
- `params_plot`: Plot of marginal posterior distributions of parameters.

`comparison_results`
A list where each element contains specified comparison metrics (LOO, WAIC, Bayes Factor) for the primary model vs. another model. LOO results may include notes about Pareto k or reloo. NULL if fewer than two models or `comparison_metrics` is empty. Each element is named e.g., "model1_vs_model2".

`prior_sensitivity_results`
A list where each element (named by model name) contains results from prior sensitivity analysis if `run_prior_sensitivity` = `TRUE`. `NULL` otherwise. Each element in the list is itself a list with the following components:

result A data frame or list summarizing the sensitivity analysis results, such as changes in posterior estimates or model fit metrics (e.g., WAIC, LOO).

plot A plot object (usually a ggplot) visualizing the effect of different priors on model estimates or diagnostics. This plot can be displayed using `print()`.

If no sensitivity was conducted or available, this field may contain a character message or be `NULL`.

`models_compared_count`
Integer, the number of models involved in comparison.

Author(s)

Achmad Syahrul Choir, Saniyyah Sri Nurhayati, and Arsyka Laila Oktalia Siregar

References

Buerkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28. Kallioinen, N., Paananen, T., Buerkner, PC. et al. Detecting and diagnosing prior and likelihood sensitivity with power-scaling. Statistical Computing, 34, 57 (2024). Gabry,J., Simpson,D., Vehtari, A., Betancourt, M., Gelman, A., Visualization in Bayesian Workflow, Journal of the Royal Statistical Society Series A: Statistics in Society, Volume 182, Issue 2, February 2019, Pages 389-402

Examples

```
# Load the hbsaems package if not already loaded (for hbm function)
# library(hbsaems) # Assuming hbm is part of this package
# library(brms) # For bf()

# Prepare a sample dataset
set.seed(123)
sim_data <- data.frame(
y = rnorm(100, 50, 10),
x1 = rnorm(100, 5, 2),
x2 = rnorm(100, 10, 3),
grp = factor(sample(1:10, 100, replace = TRUE))
)

# Define formulas and priors
f1 <- bf(y ~ x1 + (1 | grp))
common_priors <- c(prior(normal(0,10), class="b"), prior(cauchy(0,1), class="sd"))

# Fit models using hbm (minimal iterations for example speed)
fit_hbm1 <- try(hbm(f1, data=sim_data, prior=common_priors, family=gaussian(),
chains=1, iter=1000, warmup=500, cores=1, refresh=0, save_model=NULL))

if (!inherits(fit_hbm1, "try-error")) {
# Example: Explicitly request moment_match = TRUE with specific moment_match_args
# (e.g., if you want to pass 'k_threshold' to moment_match itself, though usually
# k_threshold is for reloo)
# For reloo, you might pass other args like 'psis_object' if precomputed.
checks_custom_args <- try(hbmc(
model = fit_hbm1,
moment_match = TRUE,
moment_match_args = list(k_threshold = 0.6), # Example arg for moment_match
reloo_args = list(check = FALSE), # Example arg for reloo
comparison_metrics = "loo"
))
# if (!inherits(checks_custom_args, "try-error")) {
# print(summary(checks_custom_args))
# }
}
```

`hbm_betalogitnorm` *Small Area Estimation using Hierarchical Bayesian under Beta Distribution*

Description

This function is implemented a **Hierarchical Bayesian Small Area Estimation (HBSAE)** model under a **beta distribution** using **Bayesian inference** with the `brms` package.

The range of the variable data (y) that is intended as a beta distribution must be $0 < y < 1$. The data proportion is supposed to be implemented with this function.

The function utilizes the **Bayesian regression modeling framework** provided by `brms`, which interfaces with 'Stan' for efficient Markov Chain Monte Carlo sampling. The `brm()` function from `brms` is used to estimate posterior distributions based on user-defined hierarchical and spatial structures.

Usage

```
hbm_betalogitnorm(
  response,
  predictors,
  n = NULL,
  deff = NULL,
  link_phi = "identity",
  group = NULL,
  sre = NULL,
  sre_type = NULL,
  car_type = NULL,
  sar_type = NULL,
  M = NULL,
  data,
  handle_missing = NULL,
  m = 5,
  prior = NULL,
  control = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  cores = 1,
  sample_prior = "no",
  stanvars = NULL,
  ...
)
```

Arguments

`response`	The dependent (outcome) variable in the model. This variable represents the main response being predicted or analyzed.

`predictors`	A list of independent (explanatory) variables used in the model. These variables form the fixed effects in the regression equation.
`n`	The number of sample units for each region used in the survey
`deff`	Design Effect
`link_phi`	Link function for the second parameter (phi), typically representing precision, shape, or dispersion depending on the family used (e.g., "log", "identity")
`group`	The name of the grouping variable (e.g., area, cluster, region) used to define the hierarchical structure for random effects. This variable should correspond to a column in the input data and is typically used to model area-level variation through random intercepts
`sre`	An optional grouping factor mapping observations to spatial locations. If not specified, each observation is treated as a separate location. It is recommended to always specify a grouping factor to allow for handling of new data in post-processing methods.
`sre_type`	Determines the type of spatial random effect used in the model. The function currently supports "sar" and "car"
`car_type`	Type of the CAR structure. Currently implemented are "escar" (exact sparse CAR), "esicar" (exact sparse intrinsic CAR), "icar" (intrinsic CAR), and "bym2".
`sar_type`	Type of the SAR structure. Either "lag" (for SAR of the response values) or "error" (for SAR of the residuals).
`M`	The M matrix in SAR is a spatial weighting matrix that shows the spatial relationship between locations with certain weights, while in CAR, the M matrix is an adjacency matrix that only contains 0 and 1 to show the proximity between locations. SAR is more focused on spatial influences with different intensities, while CAR is more on direct adjacency relationships. If sre is specified, the row names of M have to match the levels of the grouping factor
`data`	Dataset used for model fitting
`handle_missing`	Mechanism to handle missing data (NA values) to ensure model stability and avoid estimation errors. Three approaches are supported. The `"deleted"` approach performs complete case analysis by removing all rows with any missing values before model fitting. This is done using a simple filter such as `complete.cases(data)`. It is recommended when the missingness mechanism is Missing Completely At Random (MCAR). The `"multiple"` approach applies multiple imputation before model fitting. Several imputed datasets are created (e.g., using the `mice` package or the `brm_multiple()` function in `brms`), the model is fitted separately to each dataset, and the results are combined. This method is suitable when data are Missing At Random (MAR). The `"model"` approach uses model-based imputation within the Bayesian model itself. Missing values are incorporated using the `mi()` function in the model formula (e.g., `y ~ mi(x1) + mi(x2)`), allowing the missing values to be jointly estimated with the model parameters. This method also assumes a MAR mechanism and is applicable only for continuous variables. If data are suspected to be Missing Not At Random (MNAR), none of the above approaches directly apply. Further exploration, such as explicitly modeling the missingness process or conducting sensitivity analyses, is recommended.

`m`	Number of imputations to perform when using the `"multiple"` approach for handling missing data (default: 5). This parameter is only used if `handle_missing = "multiple"`. It determines how many imputed datasets will be generated. Each imputed dataset is analyzed separately, and the posterior draws are then combined to account for both within-imputation and between-imputation variability, following Rubin's rules. A typical choice is between 5 and 10 imputations, but more may be needed for higher missingness rates.
`prior`	Priors for the model parameters (default: `NULL`). Should be specified using the `brms::prior()` function or a list of such objects. For example, `prior = prior(normal(0, 1), class = "b")` sets a Normal(0,1) prior on the regression coefficients. Multiple priors can be combined using `c()`, e.g., `prior = c(prior(normal(0, 1), class = "b"), prior(exponential(1), class = "sd"))`. If `NULL`, default priors from `brms` will be used.
`control`	A list of control parameters for the sampler (default: `list()`)
`chains`	Number of Markov chains (default: 4)
`iter`	Total number of iterations per chain (default: 4000)
`warmup`	Number of warm-up iterations per chain (default: floor(iter/2))
`cores`	Number of CPU cores to use (default: 1)
`sample_prior`	(default: "no")
`stanvars`	An optional `stanvar` or combination of `stanvar` objects used to define the hyperpriors for the hyperparameter `phi`. By default, if `phi` is not fixed, a gamma prior is used: `phi ~ gamma(alpha, beta)`, where `alpha` and `beta` can be defined via `stanvars`. Use `"+"` to combine multiple `stanvar` definitions. For example: `stanvar(scode = "alpha ~ gamma(2, 1);", block = "model") + stanvar(scode = "beta ~ gamma(1, 1);", block = "model")` To use the default hyperprior for `phi`, set `stanvars = NULL`.
`...`	Additional arguments passed to the `brm()` function.

Value

A `hbmfit` object

Author(s)

Sofi Zamzanah

References

Liu, B. (2009). *Hierarchical Bayes Estimation and Empirical Best Prediction of Small-Area Proportions*. College Park, University of Maryland. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons, page 390. Gelman, A. (2006). *Prior Distributions for Variance Parameters in Hierarchical Models (Comment on Article by Browne and Draper)*. Bayesian Analysis, 1(3), 527–528. Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y. S. (2008). *A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models*.

Examples

```
# Load the example dataset
library(hbsaems)
data("data_betalogitnorm")

# Prepare the dataset
data <- data_betalogitnorm

# Fit Beta Model
model1 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
data = data
)
summary(model1)

# if you have the information of n and deff values you can use the following model
model1 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
data = data
)
summary(model1)

# From this stage to the next will be explained the construction of the model with
# the condition that the user has information on the value of n and deff.
# If you do not have information related to the value of n and deff
# then simply delete the parameters n and deff in your model.

# Fit Beta Model with Grouping Variable as Random Effect
model2 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
group = "group",
data = data
)
summary(model2)

# Fit Beta Model With Missing Data
data_miss <- data
data_miss[5:7, "y"] <- NA

# a. Handling missing data by deleted (Only if missing in response)
model3 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
data = data_miss,
handle_missing = "deleted"
)
summary(model3)

# b. Handling missing data using multiple imputation (m=5)
model4 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
data = data_miss,
handle_missing = "multiple"
)
summary(model4)

# c. Handle missing data during model fitting using mi()
data_miss <- data
data_miss$x1[3:5] <- NA
data_miss$x2[14:17] <- NA
model5 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
group = "group",
data = data_miss,
handle_missing = "model"
)

# Fit Logit-Normal Model With Spatial Effect
data("adjacency_matrix_car")
M <- adjacency_matrix_car

model6 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
n = "n",
deff = "deff",
sre = "sre",
sre_type = "car",
M = M,
data = data
)
summary(model6)


# have input of argument stanvars as prior distribution of alpha and beta

model7 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
data = data,
stanvars = stanvar(scode = "alpha ~ gamma(2, 1);", block = "model") +
stanvar(scode = "beta ~ gamma(1, 1);", block = "model") #stanvars of alpha and beta
)

summary(model7)

# have input of argument stanvars as prior distribution of beta

model8 <- hbm_betalogitnorm(
response = "y",
predictors = c("x1", "x2", "x3"),
data = data,
stanvars = stanvar(scode = "beta ~ gamma(1, 1);", block = "model") #stanvars of beta

 )
summary(model8)
```

`hbm_binlogitnorm` *Small Area Estimation using Hierarchical Bayesian under Logit-Normal Model*

Description

This function implements a **Hierarchical Bayesian Small Area Estimation (HBSAE)** under a **Logit-Normal Model** using **Bayesian inference** with the `brms` package.

The model accounts for **fixed effects**, **random effects**, **spatial random effects (CAR/SAR models)**, and **measurement error correction**, allowing for robust small area estimation.

The function utilizes the **Bayesian regression modeling framework** provided by `brms`, which interfaces with 'Stan' for efficient Markov Chain Monte Carlo (MCMC) sampling. The `brm()` function from `brms` is used to estimate posterior distributions based on user-defined hierarchical and spatial structures.

Usage

```
hbm_binlogitnorm(
  response,
  trials,
  predictors,
  group = NULL,
  sre = NULL,
  sre_type = NULL,
  car_type = NULL,
  sar_type = NULL,
  M = NULL,
  data,
  handle_missing = NULL,
  m = 5,
  prior = NULL,
  control = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  cores = 1,
  sample_prior = "no",
  ...
)
```

Arguments

`response`	The dependent (outcome) variable in the model. This variable represents the count of successes in a Binomial distribution.
`trials`	Specifies the number of trials in a binomial model. This is required for binomial family models where the response variable is specified as a proportion.
`predictors`	A list of independent (explanatory) variables used in the model. These variables form the fixed effects in the regression equation.
`group`	The name of the grouping variable (e.g., area, cluster, region) used to define the hierarchical structure for random effects. This variable should correspond to a column in the input data and is typically used to model area-level variation through random intercepts
`sre`	An optional grouping factor mapping observations to spatial locations. If not specified, each observation is treated as a separate location. It is recommended to always specify a grouping factor to allow for handling of new data in post-processing methods.
`sre_type`	Determines the type of spatial random effect used in the model. The function currently supports "sar" and "car"
`car_type`	Type of the CAR structure. Currently implemented are "escar" (exact sparse CAR), "esicar" (exact sparse intrinsic CAR), "icar" (intrinsic CAR), and "bym2".
`sar_type`	Type of the SAR structure. Either "lag" (for SAR of the response values) or "error" (for SAR of the residuals).
`M`	The M matrix in SAR is a spatial weighting matrix that shows the spatial relationship between locations with certain weights, while in CAR, the M matrix is an adjacency matrix that only contains 0 and 1 to show the proximity between locations. SAR is more focused on spatial influences with different intensities, while CAR is more on direct adjacency relationships. If sre is specified, the row names of M have to match the levels of the grouping factor
`data`	Dataset used for model fitting
`handle_missing`	Mechanism to handle missing data (NA values) to ensure model stability and avoid estimation errors. Three approaches are supported. The `"deleted"` approach performs complete case analysis by removing all rows with any missing values before model fitting. This is done using a simple filter such as `complete.cases(data)`. It is recommended when the missingness mechanism

is Missing Completely At Random (MCAR). The `"multiple"` approach applies multiple imputation before model fitting. Several imputed datasets are created (e.g., using the `mice` package or the `brm_multiple()` function in `brms`), the model is fitted separately to each dataset, and the results are combined. This method is suitable when data are Missing At Random (MAR). The `"model"` approach uses model-based imputation within the Bayesian model itself. Missing values are incorporated using the `mi()` function in the model formula (e.g., `y ~ mi(x1) + mi(x2)`), allowing the missing values to be jointly estimated with the model parameters. This method also assumes a MAR mechanism and is applicable only for continuous variables. If data are suspected to be Missing Not At Random (MNAR), none of the above approaches directly apply. Further exploration, such as explicitly modeling the missingness process or conducting sensitivity analyses, is recommended.

`m`	Number of imputations to perform when using the `"multiple"` approach for handling missing data (default: 5). This parameter is only used if `handle_missing = "multiple"`. It determines how many imputed datasets will be generated. Each imputed dataset is analyzed separately, and the posterior draws are then combined to account for both within-imputation and between-imputation variability, following Rubin's rules. A typical choice is between 5 and 10 imputations, but more may be needed for higher missingness rates.
`prior`	Priors for the model parameters (default: `NULL`). Should be specified using the `brms::prior()` function or a list of such objects. For example, `prior = prior(normal(0, 1), class = "b")` sets a Normal(0,1) prior on the regression coefficients. Multiple priors can be combined using `c()`, e.g., `prior = c(prior(normal(0, 1), class = "b"), prior(exponential(1), class = "sd"))`. If `NULL`, default priors from `brms` will be used.
`control`	A list of control parameters for the sampler (default: `list()`)
`chains`	Number of Markov chains (default: 4)
`iter`	Total number of iterations per chain (default: 2000)
`warmup`	Number of warm-up iterations per chain (default: floor(iter/2))
`cores`	Number of CPU cores to use (default: 1)
`sample_prior`	(default: "no")
`...`	Additional arguments

Value

A `hbmfit` object

Author(s)

Saniyyah Sri Nurhayati

References

Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons, page 390. Gelman, A. (2006). Prior Distributions for Variance Parameters in Hierarchical Models (Comment on Article by Browne and Draper). Bayesian Analysis, 1(3), 527–528. Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y. S. (2008). A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models.

Examples

```
# Load the example dataset
library(hbsaems)
data("data_binlogitnorm")

# Prepare the dataset
data <- data_binlogitnorm

# Fit Logit-Normal Model
model1 <- hbm_binlogitnorm(
response = "y",
trials = "n",
predictors = c("x1", "x2", "x3"),
data = data
)
summary(model1)

# Fit Logit-Normal Model with Grouping Variable as Random Effect
model2 <- hbm_binlogitnorm(
response = "y",
trials = "n",
predictors = c("x1", "x2", "x3"),
group = "group",
data = data
)
summary(model2)

# Fit Logit-Normal Model With Missing Data
data_miss <- data
data_miss[5:7, "y"] <- NA

# a. Handling missing data by deleted (Only if missing in response)
model3 <- hbm_binlogitnorm(
response = "y",
trials = "n",
predictors = c("x1", "x2", "x3"),
data = data_miss,
handle_missing = "deleted"
)
summary(model3)

# b. Handling missing data using multiple imputation (m=5)
model4 <- hbm_binlogitnorm(
response = "y",
trials = "n",
predictors = c("x1", "x2", "x3"),
data = data_miss,
handle_missing = "multiple"
)
summary(model4)

# Fit Logit-Normal Model With Spatial Effect
data("adjacency_matrix_car")
M <- adjacency_matrix_car

model5 <- hbm_binlogitnorm(
response = "y",
trials = "n",
predictors = c("x1", "x2", "x3"),
sre = "sre",
sre_type = "car",
M = M,
data = data
)
summary(model5)
```

`hbm_lnln` *Small Area Estimation using Hierarchical Bayesian under Lognormal Distribution*

Description

This function implements a **Hierarchical Bayesian Small Area Estimation (HBSAE)** model under a **Lognormal distribution** using **Bayesian inference** with the `brms` package.

The response variable y_i in area i is assumed to follow a lognormal distribution:

$$y_i \mid \theta_i, \psi_i \sim \mathrm{Lognormal}(\theta_i, \psi_i)$$

which implies:

$$\log(y_i) \sim \mathcal{N}(\theta_i, \psi_i)$$

The function utilizes the **Bayesian regression modeling framework** provided by `brms`, which interfaces with 'Stan' for efficient Markov Chain Monte Carlo (MCMC) sampling. The `brm()` function from `brms` is used to estimate posterior distributions based on user-defined hierarchical and spatial structures.

Usage

```
hbm_lnln(
  response,
  predictors,
  group = NULL,
  sre = NULL,
  sre_type = NULL,
  car_type = NULL,
  sar_type = NULL,
  M = NULL,
  data,
  handle_missing = NULL,
  m = 5,
  prior = NULL,
  control = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  cores = 1,
  sample_prior = "no",
  ...
)
```

Arguments

`response`	A non-negative (x>0) dependent (outcome) variable assumed to follow a log-normal distribution.
`predictors`	A list of independent (explanatory) variables used in the model. These variables form the fixed effects in the regression equation.
`group`	The name of the grouping variable (e.g., area, cluster, region) used to define the hierarchical structure for random effects. This variable should correspond to a column in the input data and is typically used to model area-level variation through random intercepts.
`sre`	An optional grouping factor mapping observations to spatial locations. If not specified, each observation is treated as a separate location. It is recommended to always specify a grouping factor to allow for handling of new data in post-processing methods.
`sre_type`	Determines the type of spatial random effect used in the model. The function currently supports "sar" and "car"
`car_type`	Type of the CAR structure. Currently implemented are "escar" (exact sparse CAR), "esicar" (exact sparse intrinsic CAR), "icar" (intrinsic CAR), and "bym2".
`sar_type`	Type of the SAR structure. Either "lag" (for SAR of the response values) or "error" (for SAR of the residuals).
`M`	The M matrix in SAR is a spatial weighting matrix that shows the spatial relationship between locations with certain weights, while in CAR, the M matrix is an adjacency matrix that only contains 0 and 1 to show the proximity between locations. SAR is more focused on spatial influences with different intensities, while CAR is more on direct adjacency relationships. If sre is specified, the row names of M have to match the levels of the grouping factor
`data`	Dataset used for model fitting

`handle_missing`	Mechanism to handle missing data (NA values) to ensure model stability and avoid estimation errors. Three approaches are supported. The `"deleted"` approach performs complete case analysis by removing all rows with any missing values before model fitting. This is done using a simple filter such as `complete.cases(data)`. It is recommended when the missingness mechanism is Missing Completely At Random (MCAR). The `"multiple"` approach applies multiple imputation before model fitting. Several imputed datasets are created (e.g., using the `mice` package or the `brm_multiple()` function in `brms`), the model is fitted separately to each dataset, and the results are combined. This method is suitable when data are Missing At Random (MAR). The `"model"` approach uses model-based imputation within the Bayesian model itself. Missing values are incorporated using the `mi()` function in the model formula (e.g., `y ~ mi(x1) + mi(x2)`), allowing the missing values to be jointly estimated with the model parameters. This method also assumes a MAR mechanism and is applicable only for continuous variables. If data are suspected to be Missing Not At Random (MNAR), none of the above approaches directly apply. Further exploration, such as explicitly modeling the missingness process or conducting sensitivity analyses, is recommended.
`m`	Number of imputations to perform when using the `"multiple"` approach for handling missing data (default: 5). This parameter is only used if `handle_missing = "multiple"`. It determines how many imputed datasets will be generated. Each imputed dataset is analyzed separately, and the posterior draws are then combined to account for both within-imputation and between-imputation variability, following Rubin's rules. A typical choice is between 5 and 10 imputations, but more may be needed for higher missingness rates.
`prior`	Priors for the model parameters (default: `NULL`). Should be specified using the `brms::prior()` function or a list of such objects. For example, `prior = prior(normal(0, 1), class = "b")` sets a Normal(0,1) prior on the regression coefficients. Multiple priors can be combined using `c()`, e.g., `prior = c(prior(normal(0, 1), class = "b"), prior(exponential(1), class = "sd"))`. If `NULL`, default priors from `brms` will be used.
`control`	A list of control parameters for the sampler (default: `list()`)
`chains`	Number of Markov chains (default: 4)
`iter`	Total number of iterations per chain (default: 2000)
`warmup`	Number of warm-up iterations per chain (default: floor(iter/2))
`cores`	Number of CPU cores to use (default: 1)
`sample_prior`	(default: "no")
`...`	Additional arguments

Value

A `hbmfit` object

Author(s)

Arsyka Laila Oktalia Siregar

Source

Fabrizi, E., Ferrante, M. R., & Trivisano, C. (2018). Bayesian small area estimation for skewed business survey variables. Journal of the Royal Statistical Society. Series C (Applied Statistics), 67(4), 863–864. https://www.jstor.org/stable/26800576; Gelman, A. (2006). Prior Distributions for Variance Parameters in Hierarchical Models (Comment on Article by Browne and Draper). Bayesian Analysis, 1(3), 527–528; Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y. S. (2008). A Weakly Informative Default Prior Distribution for Logistic and Other Regression Models.

Examples

```
# Load necessary libraries
library(hbsaems)

# Load custom dataset
data <- data_lnln
head(data)

# --- 1. Prior Predictive Check ---
model.check_prior <- hbm_lnln(
response = "y_obs",
predictors = c("x1", "x2", "x3"),
group = "group",
data = data,
prior = c(
prior(normal(0.1, 0.1), class = "b"),
prior(normal(1, 1), class = "Intercept")
),
sample_prior = "only",
iter = 4000,
warmup = 2000,
chains = 2,
seed = 123
)
hbpc(model.check_prior, response_var = "y_obs")

# --- 2. Fit the Model with Data ---
model <- hbm_lnln(
response = "y_obs",
predictors = c("x1", "x2", "x3"),
group = "group",
data = data,
prior = c(
prior(normal(0.1, 0.1), class = "b"),
prior(normal(1, 1), class = "Intercept")
),
iter = 10000,
warmup = 5000,
chains = 1,
seed = 123
)
summary(model)
```

```
# --- 3. Fit Model with Spatial Effect (CAR) ---
M <- adjacency_matrix_car

model.spatial <- hbm_lnln(
response = "y_obs",
predictors = c("x1", "x2", "x3"),
group = "group",
sre = "sre", # Spatial grouping variable (must match rows/cols of M)
sre_type = "car", # Spatial random effect type
car_type = "icar", # CAR model type
M = M, # Adjacency matrix (must be symmetric)
data = data,
prior = c(
prior(normal(0.1, 0.1), class = "b"),
prior(normal(1, 1), class = "Intercept")
),
iter = 10000,
warmup = 5000,
chains = 1,
seed = 123
)
summary(model.spatial)
```

hbpc — *hbpc : Hierarchical Bayesian Prior Predictive Checking*

Description

This function facilitates prior predictive checking for Bayesian models. It is primarily used to visualize and summarize the implications of the chosen priors by examining distributions generated from the priors alone.

Usage

```
hbpc(model, data, response_var, ndraws_ppc = 50)
```

Arguments

`model`	A `brmsfit` object, which is the fitted Bayesian model from `brms`. Ideally, for prior predictive checks, this model should be fitted with the argument `sample_prior = "only"`.
`data`	The data that was used (or would be used) to fit the model. Required by `pp_check` for comparing with observed data, though for `sample_prior = "only"`, the focus is on `y_rep`.
`response_var`	A character string specifying the name of the response variable in the data. This is needed for `pp_check` to correctly identify the observed data `y`.
`ndraws_ppc`	An integer specifying the number of draws to use for the posterior predictive check (pp_check) plot. Default is 50.

Details

Hierarchical Bayesian Prior Predictive Checking

Value

A list of class `hbpc_results` containing:

`prior_summary` Summary of the priors used in the model (obtained from `brms::prior_summary(model)`).

`prior_predictive_plot`
: A `ggplot` object from `brms::pp_check` comparing the observed data (if available and appropriate) with draws from the prior predictive distribution.

`prior_draws_summary`
: (Optional) A summary of the parameter draws from the prior-only model, which can indicate the range and central tendency implied by the priors.

Author(s)

Achmad Syahrul Choir and Saniyyah Sri Nurhayati

References

Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28. Gabry,J., Simpson,D., Vehtari, A., Betancourt, M., Gelman, A., Visualization in Bayesian Workflow, Journal of the Royal Statistical Society Series A: Statistics in Society, Volume 182, Issue 2, February 2019, Pages 389–402

Examples

```
# This is a conceptual example. Actual usage requires a brms model
# fitted with sample_prior = "only".

library(brms)
 # Assume 'data_df' is your data frame with 'response_variable' and 'predictor_variable'
 data_df <- data.frame(
 response_variable = rnorm(100),
 predictor_variable = rnorm(100)
 )

 # Define some priors
 example_priors <- c(prior(normal(0, 10), class = "b"),
 prior(cauchy(0, 1), class = "sigma"))
#
 # Fit a model with sample_prior = "only"
 # Note: For actual prior predictive checks, you'd use your actual model
 # formula and data structure.
 # The data itself isn't used for fitting parameters when sample_prior = "only",
 # but its structure (like number of observations) can influence y_rep.
 fit_prior_only <- try(brm(
 bf(response_variable ~ predictor_variable),
 data = data_df,
 prior = example_priors,
 sample_prior = "only", # Crucial for prior predictive checks
 chains = 1, iter = 1000, warmup = 200, refresh = 0,
 control = list(adapt_delta = 0.8)
 ))
#
 if (!inherits(fit_prior_only, "try-error")) {
# # Perform prior predictive checking
 prior_check_output <- hbpc(
 model = fit_prior_only,
 data = data_df,
 response_var = "response_variable", # Specify the response variable name
 ndraws_ppc = 50
 )
#
 # Print the summary of priors
 print(prior_check_output$prior_summary)

 # Display the prior predictive plot
 print(prior_check_output$prior_predictive_plot)

 # Print summary of parameter draws from prior
 print(prior_check_output$prior_draws_summary)
 }
```

hbsae *hbsae : Hierarchical Bayesian Small Area Estimation*

Description

This function performs Hierarchical Bayesian Small Area Estimation (HBSAE). It estimates predictions and computes the Relative Standard Error (RSE) based on the posterior predictive sample from the fitted Bayesian model.

Usage

```
hbsae(model, newdata = NULL)
```

Arguments

`model`	A `brmsfit` or `hbmfit` object, a fitted model from the `brms` package and `hbsaems` package.
`newdata`	A dataset for making predictions.

Details

Hierarchical Bayesian Small Area Estimation

Value

An object of class `hbsae_results`, which is a list containing:

`rse_model`	A numeric value indicating the overall relative standard error (RSE) of the model.
`mse_model`	A numeric value indicating the overall mean squared error (MSE) of the model estimates, representing the average estimation error across areas.
`result_table`	A `data.frame` containing predictions and associated statistics for each small area.

Author(s)

Achmad Syahrul Choir and Saniyyah Sri Nurhayati

References

Bürkner, P. C. (2017). brms: An R package for Bayesian multilevel models using Stan. *Journal of Statistical Software*, 80(1), 1-28.

Examples

```
library(hbsaems)
data("data_fhnorm")

# Prepare the dataset
data <- data_fhnorm

# Fit the Basic Model
model <- hbm(
formula = bf(y ~ x1 + x2 + x3), # Formula model
hb_sampling = "gaussian", # Gaussian family for continuous outcomes
hb_link = "identity", # Identity link function (no transformation)
data = data, # Dataset
chains = 4, # Number of MCMC chains
iter = 4000, # Total MCMC iterations
warmup = 2000, # Number of warmup iterations
cores = 2 # Parallel processing
)
summary(model)

# Small Area Estimates
hbsae(model)
```

`print.hbcc_results` *Print a summary for a convergence check*

Description

Print a summary for a convergence check

Usage

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

Arguments

`x`	A `hbcc` object
`...`	Other potential arguments

Value

Summary of a convergence check

`print.hbmc_results` *Print a summary for a model goodness of fit and prior sensitivity*

Description

Print a summary for a model goodness of fit and prior sensitivity

Usage

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

Arguments

`x`	A `hbmc_results` object
`...`	Other potential arguments

Value

Summary of a model goodness of fit and prior sensitivity

`print.hbmfit` *Print a summary for a fitted model represented by a hbmfit object*

Description

Print a summary for a fitted model represented by a hbmfit object

Usage

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

Arguments

`x`	A `hbmfit` object (changed from object to x for consistency with generic)
`...`	Other potential arguments

Value

Summary of the model

`print.hbpc_results` *Print a summary for a prior predictive check*

Description

Print a summary for a prior predictive check

Usage

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

Arguments

`x`	A `hbpc_results` object
`...`	Other potential arguments

Value

Summary of a prior predictive check

`print.hbsae_results` *Print a summary for a prediction result*

Description

Print a summary for a prediction result

Usage

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

Arguments

`x`	A `hbsae` object
`...`	Other potential arguments

Value

Summary of a prediction result

`run_sae_app` *Launch the Shiny App for Small Area Estimation using Hierarchical Bayesian*

Description

This function launches an interactive **Shiny application** for performing **Hierarchical Bayesian Small Area Estimation (HBSAE)** using the `brms` package with `Stan` as the backend for Bayesian inference.

Usage

```
run_sae_app()
```

Value

Opens a Shiny app in the web browser. The function does not return a value.

Author(s)

Achmad Syahrul Choir and Arsyka Laila Oktalia Siregar

References

Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons.

Examples

```
# Launch the HBSAE Shiny application (run interactively only)
if (interactive()) {
  run_sae_app()
}

# The function will open an interactive web application in your default browser
# where you can:
# 1. Upload your small area data
# 2. Specify model parameters
# 3. Run Hierarchical Bayesian analysis
# 4. View and download results
# Note: This function requires an interactive R session
# and will open a web browser to display the Shiny application
```

`spatial_weight_sar` *Spatial Weight for Simultaneous Autoregressive (SAR)*

Description

The `spatial_weight_sar` dataset contains a standardized spatial weight matrix designed for use in *Hierarchical Bayesian Small Area Estimation* (HB SAE) under the Simultaneous Autoregressive (SAR) model. The matrix represents spatial influence between areas with continuous, non-negative weights, and is symmetric with zeros on the diagonal. This structure allows modeling of spatial autocorrelation with varying intensity across regions.

Usage

```
spatial_weight_sar
```

Format

A 100×100 numeric matrix.

Source

Simulated example created for illustrating spatial structure in SAR models.

`summary.hbcc_results` *Create a summary of a convergence check*

Description

Create a summary of a convergence check

Usage

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

Arguments

`object`	A `hbcc` object
`...`	Other potential arguments

Value

Summary of a convergence check

`summary.hbmc_results` *Create a summary of a model goodness of fit and prior sensitivity*

Description

Create a summary of a model goodness of fit and prior sensitivity

Usage

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

Arguments

`object`	A `hbmc_results` object
`...`	Other potential arguments

Value

Summary of a model goodness of fit and prior sensitivity

`summary.hbmfit` *Create a summary of a fitted model represented by a hbmfit object*

Description

Create a summary of a fitted model represented by a hbmfit object

Usage

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

Arguments

`object`	A `hbmfit` object
`...`	Other potential arguments

Value

Summary of the model

`summary.hbpc_results` *Create a summary of a prior predictive check*

Description

Create a summary of a prior predictive check

Usage

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

Arguments

`object`	A `hbpc_results` object
`...`	Other potential arguments

Value

Summary of a prior predictive check

summary.hbsae_results *Create a summary of a prediction result*

Description

Create a summary of a prediction result

Usage

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

Arguments

object	A hbsae object
...	Other potential arguments

Value

Summary of a prediction result

update_hbm *update_hbm : Update a Hierarchical Bayesian Model (hbm) object*

Description

This function updates an existing hbmfit object generated by hbm(), hbm_beta(), hbm_logitnormal(), and hbm_lognormal(). It allows updating the formula, data, and other arguments, following the behavior of brms::update().

Usage

```
update_hbm(
  model,
  newdata = NULL,
  iter = NULL,
  warmup = NULL,
  chains = NULL,
  cores = NULL,
  control = NULL,
  ...
)
```

Arguments

`model`	A `brmsfit`/`hbmfit` object
`newdata`	(optional) A new dataset with the same structure as the original
`iter`	(optional) Number of MCMC iterations
`warmup`	(optional) Number of warmup iterations
`chains`	(optional) Number of MCMC chains
`cores`	(optional) Number of cores to use for sampling
`control`	(optional) A named list of control parameters passed to Stan
`...`	Other arguments passed to `update.brmsfit`, except those modifying model structure

Details

Update a Hierarchical Bayesian Model (hbm) object

Value

An updated `hbmfit` object

Examples

```
library(hbsaems)
# Load example data
data("data_fhnorm")
# Fit initial model
model <- hbm(
formula = bf(y ~ x1 + x2 + x3),
hb_sampling = "gaussian",
hb_link = "identity",
data = data_fhnorm,
chains = 2,
iter = 10000,
warmup = 2000,
cores = 2
 )
# Update number of  iterations and warmup
updated_model <- update_hbm(
model,
newdata = data_fhnorm,
iter = 10000,
warmup = 2000,
 chains = 2,
 cores = 2
 )
# Check updated model summary
summary(updated_model)
```

Index