Package: fastsae (via r-universe)

October 17, 2026

Type Package

Title High-Performance Frequentist and Bayesian Small Area Estimation in R

Version 1.0.0

Description High-performance Small Area Estimation (SAE) in R, implementing frequentist EBLUP models via compiled C++ and Bayesian hierarchical models via 'INLA' (Integrated Nested Laplace Approximations) or, optionally, full MCMC with 'NumPyro' (Phan et al., 2019) <doi:10.48550/arXiv.1912.11554> on 'JAX' (CPU or NVIDIA GPU). Frequentist models include the area-level Fay-Herriot model (Fay and Herriot, 1979 <doi:10.1080/01621459.1979.10482505>), spatial Fay-Herriot (Pratesi and Salvati, 2008 <doi:10.1007/s10260-007-0061-9>), spatio-temporal Fay-Herriot (Marhuenda et al., 2013 <doi:10.1016/j.csda.2012.09.002>), unit-level Battese-Harter-Fuller (Battese et al., 1988 <doi:10.1080/01621459.1988.10478561>), and two-fold subarea models (Torabi and Rao, 2014 <doi:10.1016/j.jmva.2014.02.001>) for nested area-subarea hierarchies. Bayesian models use 'INLA' (Rue et al., 2009 <doi:10.1111/j.1467-9868.2008.00700.x>), supporting 6 likelihood families (Gaussian, Beta, Binomial, Poisson, Negative Binomial, Gamma), spatial structures (BYM2, Besag ICAR), temporal dynamics (RW1, RW2, AR1), and spatio-temporal interactions. Key features include analytical and bootstrap MSE estimation, benchmarking to aggregate targets (Rao and Molina, 2015 <doi:10.1002/9781118735855>), automatic handling of unsampled domains, and S3 methods for summaries, diagnostics and plots.

License GPL (>= 3)

URL https://ridsonap.github.io/fastsae/, https://github.com/ridsonap/fastsae

BugReports https://github.com/ridsonap/fastsae/issues

Encoding UTF-8

Language en-US

LazyData true

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

Imports cli, ggplot2, lifecycle, Matrix, methods, Rcpp, rlang, stats, utils

Suggests covr, doParallel, dplyr, emdi, foreach, INLA, lme4, MASS, parallel, reticulate, sae, spdep, spam, sf, sn, tipsae, rstan, rstantools, writexl, openxlsx, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements Python (>= 3.10) with 'jax' and 'numpyro' (optional, for hb_area(engine = ``jax''); see setup_numpyro_env())

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Depends R (>= 4.1.0)

Collate utils.R inla_utils.R RcppExports.R data.R methods.R autoplot.R gvf.R create_weights.R eblup_fh.R eblup_sfh.R eblup_stfh.R eblup_twofold.R eblup_bhf.R engine.R hb_area.R hb_area_jax.R python_env.R hb_unit.R hb_twofold.R hb_twofold_jax.R sim_spatial_weights.R sim_area_data.R sim_series_data.R diagnose.R spatial_test.R benchmark.R compare_sae.R export_sae.R map_sae.R

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Config/pak/sysreqs python3

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

Date/Publication 2026-10-17 12:10:50 UTC

RemoteUrl https://github.com/ridsonap/fastsae

RemoteRef HEAD

RemoteSha 1410521d62292a015bf19c66112e4e0ceca06c93

Contents

add_reliability_flags . . . 3
autoplot.fastsae . . . 4
autoplot.fastsae_diagnose . . . 5
benchmark_sae . . . 6
check_numpyro_available . . . 9
coef.fastsae . . . 10
compare_sae . . . 11
cornsoybean . . . 13
cornsoybeanmeans . . . 13

create_weights . . . 14
diagnose . . . 16
eblup_bhf . . . 18
eblup_fh . . . 20
eblup_sfh . . . 23
eblup_stfh . . . 26
eblup_twofold . . . 31
export_sae . . . 33
fitted.fastsae . . . 35
gvf_smooth . . . 35
hb_area . . . 37
hb_engine . . . 44
hb_twofold . . . 47
hb_unit . . . 51
map_sae . . . 54
mys . . . 58
mys_panel . . . 59
mys_proxmat . . . 60
plot.fastsae . . . 60
print.fastsae . . . 61
print.summary.fastsae . . . 61
residuals.fastsae . . . 62
setup_numpyro_env . . . 62
sim_area . . . 63
sim_area_data . . . 64
sim_panel . . . 65
sim_series_data . . . 66
sim_spatial_weights . . . 69
spatial_test . . . 70
summary.fastsae . . . 72

Index **73**

`add_reliability_flags` *Add Official Statistical Reliability Flags*

Description

Categorizes small area estimates into publication reliability tiers based on their relative standard error (RSE, %). The default thresholds follow the criteria used by BPS-Statistics Indonesia:

- **Reliable** (RSE $\leq 25\%$): suitable for publication.
- **Use with caution** ($25\% <$ RSE $\leq 50\%$): publish with a cautionary note on sampling variability.
- **Unreliable** (RSE $> 50\%$): suppress or aggregate to a higher level.

Other agencies use other cut-offs (for example 25% and 50% at the Australian Bureau of Statistics); set `thresholds` accordingly.

Usage

```
add_reliability_flags(object, rse_col = "rse", thresholds = c(25, 50))
```

Arguments

`object`	A `fastsae` model object or a data frame containing an RSE column.
`rse_col`	Character string specifying the name of the RSE column. Default is `"rse"`. If it is missing, the RSE is computed from the `mse` and estimate columns.
`thresholds`	Numeric vector of length 2 with the RSE (%) cut-offs. Default is `c(25, 50)`.

Value

The estimation table (a data frame) with an additional factor column `reliability`.

Examples

```
library(fastsae)
data(mys)

fit_fh <- eblup_fh(y ~ x1 + x2, vardir = "vardir", data = mys)
df_flagged <- add_reliability_flags(fit_fh)
table(df_flagged$reliability)
```

`autoplot.fastsae` *Autoplot Method for fastsae Objects*

Description

Creates diagnostic and comparison plots for Small Area Estimation (SAE) models fitted with **fastsae**. This extends the generic `autoplot` from **ggplot2**.

Usage

```
## S3 method for class 'fastsae'
autoplot(
  object,
  type = c("comparison", "ribbon", "mse", "estimates", "scatter", "rse", "map"),
  ...
)

## S3 method for class 'list'
autoplot(object, type = c("comparison", "mse", "scatter", "map"), ...)
```

Arguments

`object`	An object of class `fastsae`, or a (named) list of `fastsae` objects.
`type`	Type of plot to create. • For a single `fastsae` object: `"comparison"`, `"ribbon"`, `"mse"`, `"estimates"`, `"scatter"`, `"rse"`, or `"map"`. • For a list of `fastsae` objects: `"comparison"`, `"mse"`, `"scatter"`, or `"map"`.
`...`	Additional arguments passed to internal plotting helpers (e.g. `title`) or to **ggplot2** layers.

Value

A `ggplot` object.

Examples

```
library(fastsae)

# Single model plot
fit_fh <- eblup_fh(y ~ x1 + x2 + x3, data = mys, vardir = "vardir")
autoplot(fit_fh, type = "estimates")
autoplot(fit_fh, type = "rse")

# Compare two models
fit_sfh <- eblup_sfh(y ~ x1 + x2 + x3, data = mys, vardir = "vardir", W = mys_proxmat)
autoplot(list("Fay-Herriot" = fit_fh, "Spatial FH" = fit_sfh), type = "comparison")
```

`autoplot.fastsae_diagnose` *Autoplot Method for fastsae_diagnose Objects*

Description

Generates diagnostic visual inspections for small area estimation models evaluated via `diagnose`.

Usage

```
## S3 method for class 'fastsae_diagnose'
autoplot(
  object,
  type = c("all", "calibration", "rse", "residuals", "qq", "pit", "cpo"),
  ...
)
```

Arguments

`object`	An object of class `"fastsae_diagnose"` returned by `diagnose`.
`type`	Character string indicating the diagnostic plot type: • `"all"`: Combined multi-metric inspection (Calibration and RSE reduction). • `"calibration"`: Direct estimates vs SAE predictions with 1:1 identity line. • `"rse"`: RSE comparison between direct estimator and model-based predictions. • `"residuals"`: Standardized residuals versus fitted values. • `"qq"`: Normal Q-Q plot of standardized residuals.
`...`	Additional arguments passed to **ggplot2** layers.

Value

A `ggplot` object.

`benchmark_sae` *Benchmark Small Area Estimation Predictions to Aggregate Targets*

Description

Calibrates small area model predictions (from `hb_area`, `eblup_fh`, `eblup_sfh`, `eblup_stfh`, or `eblup_bhf`) so that their weighted aggregate matches known benchmark targets at higher administrative levels (e.g. provincial or national totals/means), as required in official statistics production.

Supports:

- **Single-Stage Benchmarking**: Calibrates small areas directly to an aggregate target (either globally or within sub-regions/provinces via `group`).
- **Two-Stage Hierarchical Benchmarking** (Rao and Molina, 2015, Sec 10.4; Steorts et al., 2014): Calibrates small areas (e.g. 500 Kabupaten/Kota) within overarching groups (e.g. 34 Provinces) while *simultaneously* harmonizing the provincial aggregates to match an overarching National Target (`national_target`). Guarantees simultaneous mathematical consistency across all administrative tiers: $\sum_{d \in \text{Prov}_g} \tilde{w}_{dg} \hat{\theta}_{dg}^{\text{BM}} = T_g^*$ and $\sum_g \tilde{W}_g T_g^* = T_{\text{nat}}$.

Methods implemented:

- `"ratio"`: Proportional / multiplicative adjustment. Preserves area proportions and non-negativity (You and Rao, 2002; Rao and Molina, 2015).
- `"difference"`: Uniform additive adjustment (Rao and Molina, 2015, Section 10.3).
- `"optimal"`: Variance-weighted quadratic loss benchmarking (Datta et al., 2011; Steorts et al., 2014). Domains with higher MSE absorb larger adjustments while precise domains remain minimally perturbed.
- `"logit"`: Logit-scale additive shift for bounded indicators $\theta_d \in (0, 1)$ (Berg and Fuller, 2014).

Usage

```
benchmark_sae(object, ...)

## S3 method for class 'fastsae'
benchmark_sae(
  object,
  target = NULL,
  weight = NULL,
  method = c("ratio", "difference", "optimal", "logit"),
  group = NULL,
  type = c("mean", "total"),
  national_target = NULL,
  outer_target = NULL,
  ...
)

## Default S3 method:
benchmark_sae(
  object,
  target = NULL,
  weight = NULL,
  method = c("ratio", "difference", "optimal", "logit"),
  group = NULL,
  type = c("mean", "total"),
  national_target = NULL,
  outer_target = NULL,
  ...
)

## S3 method for class 'fastsae_benchmark'
print(x, all_groups = FALSE, ...)

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

## S3 method for class 'fastsae_benchmark'
autoplot(object, ...)

## S3 method for class 'fastsae_benchmark'
plot(x, y = NULL, ...)
```

Arguments

`object`	A fitted `fastsae` model object (e.g., from `hb_area`, `eblup_fh`, etc.) or a numeric vector of model predictions.
`...`	Additional arguments passed to methods.
`target`	Numeric value, named vector, or data frame specifying the benchmark target(s) T_g. If `group` is specified, `target` can be:

- A single scalar numeric (constant target for all groups).
- A named numeric vector matching group identifiers.
- A data frame with columns `group` and `target`.
- `NULL` if `national_target` is provided in two-stage hierarchical mode (initial provincial targets are automatically derived from model aggregates).

`weight` Optional numeric vector or character string naming the domain benchmark weight column in `object$data` (e.g., population sizes N_d or population shares $N_d / \sum N_j$). Defaults to equal weights across domains if `NULL`.

`method` Character string specifying the benchmarking calibration method: `"ratio"` (default), `"difference"`, `"optimal"`, or `"logit"`.

`group` Optional vector or character string naming the grouping/stratum column (e.g., province or region) for multi-level hierarchical calibration.

`type` Character string: `"mean"` (default, benchmark target represents weighted average/rate, $\sum_{d \in g} w_d \hat{\theta}_d = T_g$ with normalized weights) or `"total"` (benchmark target represents population total, $\sum_{d \in g} w_d \hat{\theta}_d = T_g$).

`national_target` Optional numeric scalar defining the overarching national target (Level 0) for Two-Stage Hierarchical Benchmarking. When provided alongside `group`, first calibrates/harmonizes group (provincial) targets to match `national_target`, then calibrates small areas to the harmonized provincial targets.

`outer_target` Alias for `national_target`.

`x` An object of class `fastsae_benchmark` (for `print` and `plot` methods).

`all_groups` Logical; if `TRUE`, prints verification status for all groups in hierarchical benchmarking. Default is `FALSE`.

`y` Ignored argument for compatibility with the generic `plot` method.

Value

An object of class `c("fastsae_benchmark", "data.frame")` containing:

- `domain`: Domain identifier.
- `group`: Stratum / overarching group identifier (if specified).
- `weight`: Benchmark weight w_d.
- `original`: Model-based prediction before benchmarking ($\hat{\theta}_d$).
- `benchmarked`: Calibrated estimate satisfying the benchmark constraint ($\hat{\theta}_d^{\mathrm{BM}}$).
- `adjustment`: Absolute adjustment ($\hat{\theta}_d^{\mathrm{BM}} - \hat{\theta}_d$).
- `rel_adjustment_pct`: Relative adjustment percentage (%).
- `target`: Corresponding group benchmark target T_g^*.
- `national_target`: Overarching national target (if two-stage hierarchical).

References

1. Rao, J. N. K., and Molina, I. (2015). *Small Area Estimation* (2nd ed.). John Wiley & Sons. Chapter 10: "Benchmarking and Other Practical Issues", pp. 297-315.
2. You, Y., and Rao, J. N. K. (2002). A pseudo-empirical best linear unbiased prediction approach to small area estimation using survey weights. *The Canadian Journal of Statistics*, 30(3), 431-439.
3. Steorts, R. C., Hall, P., and Ghosh, M. (2014). General benchmarking under quadratic loss with applications to small area estimation. *Journal of Survey Statistics and Methodology*, 2(2), 173-193.
4. Datta, G. S., Ghosh, M., Steorts, R., and Maples, J. (2011). Bayesian benchmarking with applications to small area estimation. *Test*, 20(3), 574-588.
5. Berg, E., and Fuller, W. A. (2014). Small area prediction of proportions with a constrained multinomial logit model. *Journal of Survey Statistics and Methodology*, 2(3), 256-283.

Examples

```
library(fastsae)
data(mys)

# Assign province groups for hierarchical calibration example
mys$province <- rep(c("Prov_A", "Prov_B", "Prov_C"), length.out = nrow(mys))

# 1. Fit Fay-Herriot model
fit_fh <- eblup_fh(y ~ x1 + x2, vardir = "vardir", data = mys)

# 2. Ratio benchmarking to overall target (e.g. target = 6.5)
bm_ratio <- benchmark_sae(fit_fh, target = 6.5, method = "ratio")
head(bm_ratio)

# 3. Two-Stage Hierarchical Calibration (e.g. Regency -> Province -> National)
# Calibrate domains within provinces while matching overarching national target:
bm_hier <- benchmark_sae(
  fit_fh,
  group = "province",
  weight = mys$n,
  national_target = 6.2,
  method = "ratio"
)
print(bm_hier)
```

`check_numpyro_available`

Check whether the NumPyro / JAX engine is available

Description

Checks that `hb_area(engine = "jax")` can run: the package **reticulate** and a Python with the modules `jax` and `numpyro`. It uses the Python already initialised in the session, otherwise the one named by the environment variables `RETICULATE_PYTHON` or `RETICULATE_PYTHON_ENV`, otherwise the `r-numpyro-gpu` environment created by `setup_numpyro_env` (virtualenv, or conda when `conda` is on the `PATH`). When none of these exists it returns `FALSE` at once, without searching the system for other Python installations (which can take tens of seconds on some platforms). To use another Python, set `RETICULATE_PYTHON`, or select it with `reticulate::use_python()` and initialise it (e.g. `reticulate::py_config()`) before the check. The check never installs or downloads anything: the managed ('uv') Python of **reticulate** is not provisioned unless the `RETICULATE_USE_MANAGED_VENV` environment variable is set.

Usage

```
check_numpyro_available(device = "auto")
```

Arguments

`device`	Character: `"auto"`, `"metal"`, `"cuda"`, or `"cpu"`.

Value

Logical indicating whether the environment is properly configured.

See Also

`setup_numpyro_env`, `hb_area`

Examples

```
check_numpyro_available()
```

`coef.fastsae` *Extract coefficients from a fastsae object*

Description

Extract coefficients from a fastsae object

Usage

```
## S3 method for class 'fastsae'
coef(object, ...)
```

Arguments

`object`	An object of class `fastsae`.
`...`	Additional arguments.

Value

Named vector of estimated coefficients.

`compare_sae` *Compare Two Small Area Estimation Models*

Description

Provides a comprehensive evaluation and concordance comparison between two fitted Small Area Estimation (SAE) models (e.g. comparing Classical Fay-Herriot vs Spatial Fay-Herriot, Area-level vs Unit-level BHF, or EBLUP vs HB).

Computes empirical agreement and efficiency metrics including:

- Pearson linear correlation (r) and Spearman rank correlation (ρ).
- Mean Absolute Difference (MAE) and Root Mean Squared Difference (RMSD).
- Relative efficiency metrics: Mean MSE ratio ($\mathrm{MSE}_1/\mathrm{MSE}_2$), and the proportion of domains where Model 2 achieves greater precision.
- Average Relative Standard Error (RSE %) across models and precision gains.

Usage

```
compare_sae(model1, model2 = NULL, names = NULL, thresholds = c(25, 50), ...)

## Default S3 method:
compare_sae(model1, model2 = NULL, names = NULL, thresholds = c(25, 50), ...)

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

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

## S3 method for class 'fastsae_comparison'
plot(x, y = NULL, ...)

## S3 method for class 'fastsae_comparison'
autoplot(
  object,
  type = c("scatter", "comparison", "difference", "mse", "rse"),
  title = NULL,
  ...
)
```

Arguments

model1	A fitted fastsae model object, or a list containing two fastsae models.
model2	A second fitted fastsae model object. Ignored if model1 is a list.
names	Optional character vector of length 2 specifying descriptive labels for the models. Default is inferred from model types or call arguments.
thresholds	Numeric vector of length 2 with the RSE (%) cut-offs drawn in autoplot(type = "rse") (see add_reliability_flags). Default c(25, 50).
...	Additional arguments.
x	An object of class fastsae_comparison (for print and plot methods).
object	An object of class fastsae_comparison (for summary and autoplot methods).
y	Ignored argument for compatibility with the generic plot method.
type	Character string indicating comparison plot type: "scatter", "comparison", "difference", "mse", or "rse". Default is "scatter".
title	Optional character string specifying a custom plot title.

Value

An S3 object of class fastsae_comparison containing:

- metrics: A data frame of overall concordance and efficiency statistics.
- data: A domain-level data frame with aligned estimates, MSE, and RSE values.
- names: Vector of the two model names.
- thresholds: The RSE thresholds used.

See Also

autoplot.fastsae_comparison, map_sae

Examples

```
library(fastsae)
data(mys)

# Fit two models
fit_fh <- eblup_fh(y ~ x1 + x2, vardir = "vardir", data = mys)
fit_sfh <- eblup_sfh(y ~ x1 + x2, vardir = "vardir", data = mys, W = mys_proxmat)

# Compare models
comp <- compare_sae(fit_fh, fit_sfh, names = c("FH", "Spatial FH"))
print(comp)
summary(comp)

# Plot comparison
autoplot(comp, type = "scatter")
autoplot(comp, type = "difference")
```

cornsoybean — *Corn and Soybean Survey and Satellite Data in 12 Iowa Counties*

Description

Survey and satellite data for corn and soy beans in 12 Iowa counties, originally obtained from the 1978 June Enumerative Survey of the U.S. Department of Agriculture and from LANDSAT satellite observations during the 1978 growing season.

Usage

```
data(cornsoybean)
```

Format

A data frame with 37 observations on the following 5 variables:

County numeric county code.

CornHec reported hectares of corn from the survey.

SoyBeansHec reported hectares of soy beans from the survey.

CornPix number of pixels of corn in the sample segment within county, from satellite data.

SoyBeansPix number of pixels of soy beans in the sample segment within county, from satellite data.

Details

This dataset is included for demonstration purposes and is originally provided in the **sae** package.

Source

Battese, G.E., Harter, R.M., and Fuller, W.A. (1988). *An Error-Components Model for Prediction of County Crop Areas Using Survey and Satellite Data.* *Journal of the American Statistical Association*, 83, 28–36.

cornsoybeanmeans — *Corn and Soybean Mean Number of Pixels per Segment for 12 Iowa Counties*

Description

County means of number of pixels per segment of corn and soy beans, from satellite data, for 12 counties in Iowa. The dataset includes population size, sample size, and means of auxiliary variables used in the dataset cornsoybean.

Usage

```
data(cornsoybeanmeans)
```

Format

A data frame with 12 observations on the following 6 variables:

CountyIndex numeric county code.

CountyName name of the county.

SampSegments number of sample segments in the county (sample size).

PopnSegments number of population segments in the county (population size).

MeanCornPixPerSeg mean number of corn pixels per segment in the county.

MeanSoyBeansPixPerSeg mean number of soy beans pixels per segment in the county.

Details

This dataset is provided for demonstration purposes and is originally distributed with the **sae** package.

Source

Battese, G.E., Harter, R.M., and Fuller, W.A. (1988). *An Error-Components Model for Prediction of County Crop Areas Using Survey and Satellite Data.* *Journal of the American Statistical Association*, 83, 28–36.

---

create_weights *Create Spatial Proximity and Weights Matrix*

Description

Constructs spatial proximity, contiguity, and weight matrices (**W**) from spatial data (`sf` polygon or point objects, coordinate matrices, or neighbor lists). Supports Queen and Rook contiguity, k-nearest neighbors (k-NN), distance bands, and inverse distance weighting (IDW), with automatic fallback handling for isolated islands and disconnected domains.

Usage

```
create_weights(
  data,
  method = c("queen", "rook", "knn", "distance", "idw"),
  style = c("W", "B", "C", "U", "minmax"),
  k = 4,
  d_max = NULL,
  alpha = 1,
  island_fallback = TRUE,
  k_fallback = 1,
  domain = NULL
)
```

Arguments

`data` An `sf` polygon or point object, a numeric matrix or data frame of 2D coordinates (with columns `x`/`y` or longitude/latitude), or an `spdep` `nb` or `listw` object.

`method` Character string specifying the spatial weight method:

- `"queen"`: Queen contiguity (sharing at least one vertex or boundary edge).
- `"rook"`: Rook contiguity (sharing a common linear boundary edge).
- `"knn"`: k-nearest neighbors based on centroid/point distances.
- `"distance"`: Distance band where domains within `d_max` are neighbors.
- `"idw"`: Inverse distance weighting $w_{ij} = 1/d_{ij}^{\alpha}$.

`style` Character string specifying the normalization style:

- `"W"`: Row-standardized (row sums equal 1, default for SAR/SFH models).
- `"B"`: Binary adjacency (0/1 values, standard for INLA BYM2/Besag models).
- `"C"`: Globally standardized.
- `"U"`: Equal weighting ($1/m$).
- `"minmax"`: Min-max spectral normalization (divided by largest eigenvalue).

`k` Integer. Number of nearest neighbors when `method = "knn"` (default is 4).

`d_max` Numeric. Distance threshold when `method = "distance"`. If `NULL`, it is automatically set to the minimum distance ensuring no disconnected domains.

`alpha` Numeric. Distance decay power for `method = "idw"` (default is 1).

`island_fallback` Logical. If `TRUE` (default), disconnected domains or islands with zero contiguous neighbors are automatically connected to their nearest neighbor(s) based on centroid distance.

`k_fallback` Integer. Number of nearest neighbors to connect for isolated islands when `island_fallback = TRUE` (default is 1).

`domain` Optional vector, column name, or one-sided formula referencing domain IDs. If provided, used as row and column names for the resulting matrix.

Details

When working with real administrative boundaries (such as Indonesian provinces, regencies, or municipalities), island regencies (e.g. Kepulauan Seribu, Natuna, Sabang, Mentawai, Sangihe) lack contiguous land borders. If `island_fallback = TRUE`, `create_weights` detects these zero-neighbor domains and connects each to its closest `k_fallback` neighbor(s) based on centroid distances, ensuring a connected spatial graph without computational singularities.

Value

A square numeric matrix of class `c("fastsae_weights", "matrix")` of dimension $m \times m$. Attributes include `method`, `style`, `n_domains`, `n_islands`, and `call`.

Examples

```
if (requireNamespace("sf", quietly = TRUE) && requireNamespace("spdep", quietly = TRUE)) {
  # North Carolina counties, an sf polygon data set shipped with 'sf'
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
  W_queen <- create_weights(nc, method = "queen", style = "W")
  W_knn   <- create_weights(nc, method = "knn", k = 5, style = "W")
  W_bin   <- create_weights(nc, method = "queen", style = "B")
  W_queen
}

# Create spatial weights from 2D coordinates
coords <- matrix(runif(40), ncol = 2)
W_coord <- create_weights(coords, method = "knn", k = 3, style = "W")
dim(W_coord)
```

`diagnose` *Diagnostic Evaluation of Small Area Estimation Models*

Description

Conducts a comprehensive diagnostic evaluation of small area estimation models (HB or EBLUP). Evaluates estimation precision, efficiency gains over direct estimators, external calibration and bias diagnostics (Brown et al., 2001), goodness-of-fit tests, and residual spatial autocorrelation (Moran's I).

Usage

```
diagnose(
  object,
  W = NULL,
  truth = NULL,
  rse_threshold = 25,
  alpha_level = 0.05
)
```

Arguments

`object`	A fitted model object of class `"fastsae"` (e.g., from `hb_area`, `eblup_fh`, `eblup_sfh`, or `eblup_stfh`).
`W`	Optional spatial proximity or adjacency matrix of dimension $D \times D$. If `NULL` and the fitted model contains a spatial matrix (e.g. `eblup_sfh` or spatial `hb_area`), it is automatically extracted from `object`.
`truth`	Optional numeric vector containing true parameter values θ_d (useful for simulation studies).
`rse_threshold`	Numeric. Threshold for reliable Relative Standard Error (default is 25%).
`alpha_level`	Numeric. Significance level for hypothesis tests (default is 0.05).

Details

Goodness of fit. $W = \sum_d (y_d - \hat{\theta}_d)^2 / (\psi_d + \mathrm{MSE}_d)$ over the sampled domains is compared with χ^2_m (Brown et al., 2001); only large values indicate that the model estimates differ from the direct estimates more than their uncertainty allows, so the p-value is upper-tailed (as in `emdi`). The statistic ignores the positive correlation between y_d and $\hat{\theta}_d$, so it is conservative: under a correct model W is typically well below its degrees of freedom.

Bias regression. The OLS fit $y_d = \alpha + \beta \hat{\theta}_d$ is reported without a formal test of $\alpha = 0, \beta = 1$: for shrinkage estimators such as the EBLUP the slope exceeds 1 even when the model is correct (about $1/\gamma_d$ for the part of the variation not explained by the covariates), so that test would reject correct models. Inspect the regression (and `autoplot(x, type = "calibration")`) for systematic deviations instead.

Moran's I on the residuals $y_d - \hat{\theta}_d$ of the sampled domains uses the randomisation variance of Cliff and Ord (1981), two-sided.

Value

An object of class `c("fastsae_diagnose", "list")` containing:

model_info Basic model characteristics (model name, sample size, unsampled count).

precision Summary of direct vs SAE RSE, proportion of areas with RSE < threshold, and MSE reduction ratios.

brown_test Brown et al. (2001) diagnostics: `bias_regression` (descriptive OLS fit of the direct estimates on the model estimates, no test; see Details) and `goodness_of_fit` (the W statistic with its upper-tail χ^2 p-value).

spatial_test Moran's I test for spatial autocorrelation in model residuals (if `W` is available).

bayesian_metrics For Bayesian fits: DIC, WAIC, marginal log-likelihood and, when they are available per domain, CPO summaries and a Kolmogorov-Smirnov test of the PIT values.

simulation_metrics Mean relative bias, mean absolute relative error, relative RMSE across domains, and coverage rate (if `truth` is supplied).

df_diag Data frame combining direct estimates, SAE predictions, MSE, RSE, and residuals for diagnostics.

status Overall assessment summary string.

References

1. Brown, G., Chambers, R., Heady, P., & Heasman, D. (2001). Evaluation of small area estimation methods: An application to the British Labour Force Survey. *ONS Internal Report*.
2. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation* (2nd ed.). John Wiley & Sons.
3. Cliff, A. D., & Ord, J. K. (1981). *Spatial Processes: Models & Applications*. Pion London.

Examples

```
library(fastsae)
data(mys)
data(mys_proxmat)
```

```
# 1. Fit Fay-Herriot model
fit_fh <- eblup_fh(y ~ x1 + x2, data = mys, vardir = ~vardir, domain = ~area)
diag_fh <- diagnose(fit_fh)
print(diag_fh)

# 2. Fit Spatial HB model (requires INLA)
if (requireNamespace("INLA", quietly = TRUE)) {
  fit_hb <- hb_area(y ~ x1 + x2, data = mys, vardir = "vardir")
  diag_hb <- diagnose(fit_hb)
  print(diag_hb)
}
```

`eblup_bhf` *Empirical Best Linear Unbiased Prediction (EBLUP) for the Battese-Harter-Fuller Model*

Description

This function estimates small area means or totals using the unit-level model proposed by Battese, Harter, and Fuller (1988), which combines survey data (sample units) and auxiliary population information.

Usage

```
eblup_bhf(
  formula,
  unit_data,
  Xpop,
  domain_var,
  popsize_var,
  method = c("REML", "ML"),
  popnmean_xpop = NULL,
  B = 100,
  mse_type = c("bootstrap", "none"),
  n_threads = 1,
  seed = -1,
  print_result = TRUE,
  compute_mse = lifecycle::deprecated()
)
```

Arguments

`formula`	An object of class ‘formula‘ describing the model.
`unit_data`	A ‘data.frame‘ containing the unit-level survey data.
`Xpop`	A ‘data.frame‘ containing auxiliary variables and domain info.
`domain_var`	A character string giving the column name for domain identifier.

popsize_var	A character string for population size variable.
method	Fitting method: "REML" (default) or "ML".
popnmean_xpop	Population mean of auxiliary variables per domain.
B	Number of bootstrap replicates for MSE (if mse_type = "bootstrap").
mse_type	MSE estimator: "bootstrap" (default; parametric bootstrap of Gonzalez-Manteiga et al. (2008), as in sae::pbmseBHF()) or "none" (no MSE; mse and rse are NA).
n_threads	Number of OpenMP threads for the bootstrap MSE. Results depend on seed only, not on the number of threads.
seed	Random seed for reproducibility; -1 (default) sets no seed. A fixed seed does not change the random number stream of the session.
print_result	Print results (default TRUE).
compute_mse	Deprecated (fastsae 1.0.0); use mse_type. compute_mse = TRUE is mse_type = "bootstrap" and FALSE is "none".

Details

The nested error model $y_{dj} = x'_{dj}\beta + u_d + e_{dj}$ is fitted by REML or ML in compiled code. With $\gamma = \sigma_u^2/\sigma_e^2$, β and σ_e^2 are profiled out, so the fit is a one-dimensional maximisation over $\gamma \geq 0$ (including the boundary $\sigma_u^2 = 0$); each evaluation uses only within-domain cross-products and domain means. The estimates are those of lme4::lmer and sae::eblupBHF up to the optimiser tolerance of those packages. The standard errors in estcoef are from $(X'\hat{V}^{-1}X)^{-1}$.

The bootstrap MSE refits the model in each replicate with the same code and runs replicates in parallel (n_threads); random draws are generated sequentially from R's RNG, so a given seed gives the same result for any number of threads.

Value

List containing EBLUP estimates, fit, and optionally MSE. The fit element holds the variance components, beta, the predicted random effects and the unit-level conditional residuals.

References

Battese, G. E., Harter, R. M., and Fuller, W. A. (1988). An error-components model for prediction of county crop areas using survey and satellite data. *Journal of the American Statistical Association*, 83(401), 28-36.

Examples

```
library(dplyr)
df_meanpop <- cornsoybeanmeans |>
  rename(CornPix = MeanCornPixPerSeg, SoyBeansPix = MeanSoyBeansPixPerSeg)
df_cornsoybean <- cornsoybean |>
  rename(CountyIndex = County)

res <- eblup_bhf(
  formula = CornHec ~ CornPix + SoyBeansPix,
  Xpop = df_meanpop,
```

```
  unit_data = df_cornsoybean,
  domain_var = "CountyIndex",
  popsize_var = "PopnSegments"
)
```

`eblup_fh` *Empirical Best Linear Unbiased Prediction based on a Fay-Herriot Model.*

Description

This function gives the Empirical Best Linear Unbiased Prediction (EBLUP) or Empirical Best (EB) predictor under normality based on a Fay-Herriot model.

Usage

```
eblup_fh(
  formula,
  vardir,
  domain = NULL,
  data,
  method = c("REML", "ML"),
  transform = c("none", "log"),
  mse_type = c("analytical", "none"),
  maxiter = 100,
  precision = 1e-04,
  print_result = TRUE,
  self_benchmark = FALSE,
  benchmark_weight = NULL,
  benchmark_target = NULL,
  benchmark_group = NULL
)
```

Arguments

`formula`	an object of class formula that contains a description of the model to be fitted.
`vardir`	vector or column names from data that contain variance sampling from the direct estimator.
`domain`	vector, column name or one-sided formula referencing a domain names column in `data`. If NULL, the domains are numbered consecutively.
`data`	a data frame or a data frame extension (e.g. a tibble).
`method`	Fitting method can be chosen between 'ML' and 'REML'.
`transform`	Character string specifying data transformation for the response: `"none"` (default) or `"log"` (Log-Fay-Herriot model). With `"log"` the estimate is back-transformed as $\exp(\tilde{\mu} + \hat{g}_1/2)$ with $g_1 = \sigma_u^2(1-\gamma)$ ($\sigma_u^2/2$ for unsampled domains), i.e. the conditional-mean correction under the log-normal model (cf.

Slud & Maiti, 2006); it does not include the additional O(1/m) terms for the estimation of β and σ_u^2. The MSE on the original scale is the delta-method approximation $\hat{\theta}^2 \cdot mse_{log}$.

`mse_type` MSE estimator: `"analytical"` (default; Prasad-Rao, with the Datta-Lahiri bias correction for ML, as in `sae::mseFH()`) or `"none"` (no MSE; `mse` and `rse` are `NA`).

`maxiter` maximum number of iterations allowed in the Fisher-scoring algorithm.

`precision` convergence tolerance limit for the Fisher-scoring algorithm.

`print_result` print coefficient or not, default value is TRUE.

`self_benchmark` Logical. If `TRUE`, fits a self-benchmarking model via the augmented model approach (Wang, Fuller, and Qu, 2008; Rao and Molina, 2015, Sec 10.5.1), ensuring that the weighted aggregate of small area estimates automatically matches the direct survey aggregate or specified benchmark targets without post-hoc adjustment. With `transform = "log"` that constraint would hold on the log scale only, so instead the back-transformed estimates are ratio-benchmarked on the original scale (exact constraint; MSE scaled by the squared group ratio; the unbenchmarked estimate is kept in `eblup_unbenchmarked`).

`benchmark_weight`
Optional vector, column name, or one-sided formula referencing the domain benchmarking weights in `data` (e.g. population sizes or shares). Defaults to equal weights.

`benchmark_target`
Optional numeric scalar or named vector specifying fixed aggregate benchmark target(s). If `NULL` (default), calibrates to the weighted aggregate of the direct estimates.

`benchmark_group`
Optional vector, column name, or one-sided formula referencing the grouping / stratum column in `data` (e.g. province) for group-specific self-benchmarking.

Details

The model has a form that is response ~ auxiliary variables. where numeric type response variables can contain NA. When the response variable contains NA it will be estimated with synthetic estimator.

Value

The function returns a list with the following objects: `estcoef` a data frame with the estimated model coefficients (`beta`), their standard errors (`std.error`, repeated as `stderr_beta` for compatibility with fastsae 0.1.0), z-statistics (`zvalue`) and normal p-values (`pvalue`), `random_effect_var` estimated random effect variance, `goodness` named vector: `loglikelihood` (the full Gaussian log-likelihood at the estimates, for ML and REML fits alike), `AIC` and `BIC` computed from it, as **sae** does and as in Marhuenda, Morales and Pardo (2014). The restricted (REML) likelihood is not reported: it depends on the covariates through X, so it cannot compare models with different covariates., `df_eblup` a data frame that contains y, eblup, random_effect, vardir, mse, and rse.

- `y` variable response
- `eblup` estimated results for each area
- `random_effect` random effect for each area
- `vardir` variance sampling from the direct estimator for each area
- `mse` Mean Square Error
- `rse` Relative Standart Error (%)
  `self_benchmark` logical indicating whether self-benchmarking was active,
  `benchmark_summary` summary table of benchmark calibration (if self-benchmarking).

References

1. Rao, J. N., & Molina, I. (2015). Small area estimation. John Wiley & Sons.

1. Marhuenda, Y., Morales, D., & Pardo, M. C. (2014). Information criteria for Fay-Herriot model selection. *Computational Statistics & Data Analysis*, 70, 268-280.
2. Rao, J. N. K., & Molina, I. (2015). Small area estimation. John Wiley & Sons.
3. Slud, E. V., & Maiti, T. (2006). Small-area estimation via Fay-Herriot models with logit and log transformations. *Communications in Statistics - Theory and Methods*, 35(10), 1957-1971.
4. Wang, J., Fuller, W. A., and Qu, Y. (2008). Small area estimation under a restriction. *Survey Methodology*, 34(1), 29-36.

Examples

```
library(fastsae)

# Standard Fay-Herriot model
m1 <- eblup_fh(
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = "vardir"
)

# Log-Fay-Herriot with Slud-Maiti bias correction
m_log <- eblup_fh(
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = "vardir",
  transform = "log"
)

# Self-benchmarking Fay-Herriot model
m1_bench <- eblup_fh(
```

```
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = "vardir",
  self_benchmark = TRUE,
  benchmark_weight = "n"
)
```

---

eblup_sfh *Empirical Best Linear Unbiased Prediction based on a Spatial Fay-Herriot Model.*

Description

This function gives the Spatial Empirical Best Linear Unbiased Prediction (EBLUP) or Empirical Best (EB) predictor under normality based on a Fay-Herriot model.

Usage

```
eblup_sfh(
  formula,
  vardir,
  domain = NULL,
  data,
  method = c("REML", "ML"),
  mse_type = c("analytical", "pbmse", "npbmse", "none"),
  W = NULL,
  B = 100,
  n_threads = 1,
  seed = -1,
  maxiter = 100,
  precision = 1e-04,
  self_benchmark = FALSE,
  benchmark_weight = NULL,
  benchmark_target = NULL,
  benchmark_group = NULL,
  print_result = TRUE,
  mse_method = lifecycle::deprecated()
)
```

Arguments

formula	an object of class formula that contains a description of the model to be fitted. The variables included in the formula must be contained in the data.
vardir	vector or column names from data that contain variance sampling from the direct estimator for each area.
domain	vector, column name or one-sided formula referencing a domain names column in data. If NULL, the domains are numbered consecutively.

`data` a data frame or a data frame extension (e.g. a tibble).

`method` Fitting method can be chosen between 'ML' and 'REML'.

`mse_type` MSE estimator: "analytical" (default), "pbmse" (parametric bootstrap), "npbmse" (nonparametric bootstrap) or "none" (no MSE; `mse` and `rse` are `NA`). When there are unsampled domains, "pbmse"/"npbmse" bootstrap MSE is only defined for the sampled domains; unsampled domains automatically get a full-spatial synthetic (kriging) prediction and analytical MSE merged into the result.

`W` A square matrix with dimension equal to the TOTAL number of domains in `data` (including any unsampled domains where the response is `NA`). It should contain the row-standardized spatial weights (proximities) between ALL domains, with values ranging from 0 to 1. Rows and columns must be ordered consistently with the domain identifiers in `data`. Do NOT pre-subset `W` to sampled domains only – unsampled domains' spatial MSE/ prediction relies on their relation (in `W`) to sampled neighbors.

`B` Number of bootstrap replications when mse_type = "pbmse" or "npbmse".

`n_threads` Number of threads used in parallel computation (default 1). Values less than or equal to 0 use all available cores.

`seed` Integer seed for bootstrap resampling; `-1` (default) sets no seed. A fixed seed does not change the random number stream of the session. A value of -1 leaves the current R RNG state unchanged.

`maxiter` maximum number of iterations allowed in the Fisher-scoring algorithm. Default is 100 iterations.

`precision` convergence tolerance limit for the Fisher-scoring algorithm. Default value is 0.0001.

`self_benchmark` Logical. If `TRUE`, fits a self-benchmarking model via the augmented model approach (Wang, Fuller, and Qu, 2008; Rao and Molina, 2015, Sec 10.5.1), ensuring that the weighted aggregate of small area estimates automatically matches the direct survey aggregate or specified benchmark targets without post-hoc adjustment.

`benchmark_weight`
Optional vector, column name, or one-sided formula referencing the domain benchmarking weights in `data` (e.g. population sizes or shares). Defaults to equal weights.

`benchmark_target`
Optional numeric scalar or named vector specifying fixed aggregate benchmark target(s). If `NULL` (default), calibrates to the weighted aggregate of the direct estimates.

`benchmark_group`
Optional vector, column name, or one-sided formula referencing the grouping / stratum column in `data` (e.g. province) for group-specific self-benchmarking.

`print_result` print coefficient or not, default value is TRUE.

`mse_method` Deprecated (fastsae 1.0.0); use `mse_type`.

Details

The model has a form that is response ~ auxiliary variables. where numeric type response variables can contain NA. When the response variable contains NA, that domain is treated as unsampled: it is predicted with a full-spatial synthetic (kriging) estimator that borrows strength from sampled neighbors via `W`. The SAR process is defined on all domains of `W`, so the sampled domains have covariance $\sigma^2\Sigma[s,s]$ with $\Sigma = [(I-\rho W')(I-\rho W)]^{-1}$.

Analytical MSE. Sampled domains: $g_1+g_2+2g_3-g_4$ (Singh et al., 2005; as `sae::mseSFH`), minus $b'_{ML}\nabla g_1$ for ML. Unsampled domains: $g_1+g_2+g_3-\frac{1}{2}\mathrm{tr}(I^{-1}\nabla^2 g_1)$ (minus $b'_{ML}\nabla g_1$ for ML), with g_3 from the exact derivative of the kriging predictor; for sampled domains this second-order form agrees, up to o(1/m), with $g_1+g_2+2g_3-g_4$. The inverse information used for g_3 and b_{ML} is the one of the fitting method: REML information for REML and ML information for ML. (`sae::mseSFH` uses the REML information also for ML; the two agree asymptotically, but with few domains, or $\hat{\sigma}^2 \approx 0$ with $\hat{\rho} \approx 1$, the REML choice can make the ML bias term explode. Hence, for `method = "ML"`, the MSE differs from `sae::mseSFH`; REML results agree.)

When $\hat{\rho}$ lies on the boundary ($|\hat{\rho}| \geq 0.999$, or reported as -1), the Prasad-Rao expansion in ρ is not valid and can explode, so ρ is treated as fixed: only the σ^2 block of the information is inverted, which removes the ρ terms from g_3, g_4 and b_{ML} (the standard error of ρ is then `NA`). In this case the MSE differs from `sae::mseSFH`, which keeps those terms.

If the classical (sae-compatible) Fisher scoring fails, or converges to a negative σ^2 (which `sae::eblupSFH` truncates to 0 without re-estimating ρ), the (RE)ML likelihood is maximised under the constraints $\sigma^2 \geq 0$, $|\rho| \leq 0.999$. A lower-bound estimate $\rho = -0.999$ is reported as -1 (as `sae`) only when $I+W$ is non-singular (not for bipartite graphs such as a rook grid).

Computation. No $D \times D$ inverse is formed: with $A = (I-\rho W')(I-\rho W)$ and $M = A+\sigma^2 E'\Psi^{-1}E$ (E selects the sampled domains), $V^{-1} = \Psi^{-1} - \sigma^2\Psi^{-1}EM^{-1}E'\Psi^{-1}$, $g_1 = \sigma^2\,\mathrm{diag}(M^{-1})$ and the remaining terms follow from Cholesky solves with A and M. These are sparse for a sparse `W` (contiguity or k-nearest-neighbour weights), which makes models with a few thousand domains feasible; a dense `W` (e.g. inverse-distance weights) uses dense factorisations.

Value

The function returns a list with the following objects: `estcoef` a data frame with the estimated model coefficients (`beta`), their asymptotic standard errors (`std.error`, repeated as `stderr_beta` for compatibility with fastsae 0.1.0), the z-statistics (`zvalue`) and the normal p-values (`pvalue`)
`random_effect_var` estimated random effect variance
`rho` estimated spatial autocorrelation parameter
`estvarcomp` a data frame with parameter, estimate, and std.error
`goodness` named vector: `loglikelihood` (the full Gaussian log-likelihood at the estimates, for ML and REML fits alike), `AIC` and `BIC` computed from it, as **sae** does and as in Marhuenda, Morales and Pardo (2014). The restricted (REML) likelihood is not reported: it depends on the covariates through X, so it cannot compare models with different covariates.
`df_eblup` a data frame that contains the following columns:

- `y` variable response
- `eblup` estimated results for each area

- `random_effect` random effect for each area
- `vardir` variance sampling from the direct estimator for each area
- `mse` Mean Square Error
- `rse` Relative Standart Error (%)
- `mse_pb` or `mse_npb` Parametric / Non Parametric bootstrap MSE
- `mse_pbbc` or `mse_npbbc` Bias Corrected Parametric / Non Parametric bootstrap MSE

References

1. Marhuenda, Y., Morales, D., & Pardo, M. C. (2014). Information criteria for Fay-Herriot model selection. *Computational Statistics & Data Analysis*, 70, 268-280.
2. Rao, J. N., & Molina, I. (2015). Small area estimation. John Wiley & Sons.

Examples

```
library(fastsae)

# Spatial Fay-Herriot model
m1 <- eblup_sfh(
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = ~vardir,
  W = mys_proxmat
)

# Spatial Fay-Herriot model with Parametric Bootstrap MSE
m2 <- eblup_sfh(
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = ~vardir,
  mse_method = "pbmse",
  B = 50,
  W = mys_proxmat
)
```

`eblup_stfh` *Empirical Best Linear Unbiased Prediction based on a Spatio-Temporal Fay-Herriot Model.*

Description

This function gives the Spatio-Temporal Empirical Best Linear Unbiased Prediction (EBLUP) under normality based on a spatio-temporal Fay-Herriot model. It reimplements the same Fisher-scoring algorithm as `eblupSTFH()` (package **sae**, Marhuenda, Molina & Morales 2013), but the estimation loop runs in compiled C++/Armadillo, making it substantially faster and more memory-efficient than the original R implementation – especially for a large number of domains/time periods.

Usage

```
eblup_stfh(
  formula,
  vardir,
  data,
  domain,
  time,
  W,
  model = c("ST", "S"),
  maxiter = 100,
  precision = 1e-04,
  sigma21_start = NULL,
  rho1_start = 0.5,
  sigma22_start = NULL,
  rho2_start = 0.5,
  mse_type = c("analytical", "bootstrap", "none"),
  B = 100,
  n_threads = 1,
  seed = -1,
  self_benchmark = FALSE,
  benchmark_weight = NULL,
  benchmark_target = NULL,
  benchmark_group = NULL,
  print_result = TRUE,
  compute_mse = lifecycle::deprecated()
)
```

Arguments

`formula` an object of class formula describing the model to fit (response ~ auxiliary variables). Variables must be present in `data`.

`vardir` vector, column name or one-sided formula referencing a column in `data`, with the sampling variances of the direct estimator. Must be positive in the sampled cells; it is ignored (may be `NA`) where the response is `NA`.

`data` a data frame (or extension) with one row per domain and time period (`domain * time` rows), including the cells without a direct estimate: for those, the response (and `vardir`) is `NA`, while the auxiliary variables must be available (see Details). Rows may be in any order: they are sorted internally into the domain-major order required by the model (domains in order of first appearance, peri-

ods increasing), and the results are returned in the original row order. Note that `sae::eblupSTFH()` instead assumes a complete panel already in domain-major order.

`domain` vector, column name or one-sided formula referencing a domain names column in `data`. If NULL, the domains are numbered consecutively.

`time` vector, column name, or one-sided formula referencing a time names column in `data`.

`W` a square proximity/spatial weights matrix of dimension `domain` x `domain` (row-standardized, values typically in $[0, 1]$), with rows and columns in the order in which the domains first appear in `data`. It must cover all domains, including the ones that are never sampled: do not drop them from `W`, since their predictions borrow strength from their neighbours through it.

`model` character, either `"ST"` (spatio-temporal, default) or `"S"` (spatial only, no AR(1) temporal component).

`maxiter` maximum number of Fisher-scoring iterations. Default 100.

`precision` convergence tolerance for the Fisher-scoring algorithm. Default `1e-4`.

`sigma21_start`, `rho1_start`, `sigma22_start`, `rho2_start`
starting values for the variance/autocorrelation components. Defaults mirror `eblupSTFH(): 0.5 * median(vardir)` for the variances and `0.5` for the autocorrelations. `rho2_start` is ignored when `model = "S"`. The REML fit first uses the same Fisher-scoring iteration as `sae::eblupSTFH()`; if that fails (typically a variance iterate turns negative and the covariance matrix becomes indefinite) it falls back to constrained REML (variances ≥ 0, $|\rho| \leq 0.999$) with step halving. When a variance is estimated at 0, its autocorrelation is not identified and its standard error is reported as `NA`.

`mse_type` character, method for MSE estimation: `"analytical"` (default; fast Prasad-Rao type estimator g1 + g2 + 2 g3 - g4; g3 uses the exact derivative of the EBLUP, $\Psi P V_a P y$, and the REML information at the final estimates; correlations estimated on the boundary $|\rho| \geq 0.999$ are treated as fixed, so they do not enter g_3/g_4) or `"bootstrap"` (parametric bootstrap in which the variance components are re-estimated by REML in every replicate, as in Marhuenda et al., 2013) or `"none"` (no MSE; `mse` and `rse` are `NA`).

`B` number of bootstrap replicates for MSE computation. Only used when `mse_type = "bootstrap"`. Default `100`.

`n_threads` number of threads for parallel bootstrap MSE computation. Use `0` for all available cores. Default `1`.

`seed` random seed for bootstrap. Use `-1` for no seed. Default `-1`. A fixed seed does not change the random number stream of the session.

`self_benchmark` Logical. If `TRUE`, fits a self-benchmarking model via the augmented model approach (Wang, Fuller, and Qu, 2008; Rao and Molina, 2015, Sec 10.5.1), ensuring that the weighted aggregate of small area estimates automatically matches the direct survey aggregate or specified benchmark targets without post-hoc adjustment.

benchmark_weight
: Optional vector, column name, or one-sided formula referencing the domain benchmarking weights in `data` (e.g. population sizes or shares). Defaults to equal weights.

benchmark_target
: Optional numeric scalar or named vector specifying fixed aggregate benchmark target(s). If `NULL` (default), calibrates to the weighted aggregate of the direct estimates.

benchmark_group
: Optional vector, column name, or one-sided formula referencing the grouping / stratum column in `data` (e.g. time period / year) for group-specific self-benchmarking.

print_result
: print the estimated coefficients or not. Default `TRUE`.

compute_mse
: Deprecated (fastsae 1.0.0); use `mse_type`. `compute_mse = FALSE` is `mse_type = "none"`; `compute_mse = TRUE` is `mse_type = "bootstrap"` when `B` is given and `"analytical"` otherwise.

Details

Estimation method. The variance components are estimated by REML only, as in Marhuenda, Molina and Morales (2013) and `sae::eblupSTFH()`, which derive the Fisher-scoring algorithm, the analytical MSE and the bootstrap for REML; there is no `method` argument. The standard errors of the variance components are those of `sae::eblupSTFH()`: the inverse REML information of the last Fisher-scoring iteration.

Unsampled domains and missing periods. Cells whose response is `NA` (a domain without a sample in some period, or in every period) are not dropped. The model is defined on the full $D \times T$ grid: the SAR effect u_1 lives on all D domains of `W` and the AR(1) effect u_2 on all T periods. The variance components are estimated by REML from the sampled cells, i.e. from the marginal likelihood with the missing cells integrated out (missing at random), and every cell gets the best linear unbiased predictor of $x'_{dt}\beta + u_{1d} + u_{2dt}$ given the sampled data (Saei and Chambers, 2005; Benedetti et al., 2024 for the spatial effect):

$$\tilde{\theta}_{dt} = x'_{dt}\tilde{\beta} + a'_{dt}V^{-1}(y - X\tilde{\beta}),$$

with a_{dt} the covariance between the cell and the sampled direct estimates. A missing period of a sampled domain borrows from the other periods of the domain (AR(1)) and from the neighbours (SAR); a domain that is never sampled is predicted from its neighbours only (spatial kriging). With `model = "S"` the period effect of a missing cell is independent of the data, so only the spatial part is borrowed.

MSE. For sampled cells the analytical MSE is $g_1 + g_2 + 2g_3 - g_4$ (Marhuenda et al., 2013). For the cells predicted without a direct estimate it is $g_1 + g_2 + g_3$, where g_1 and g_2 are the MSE of the predictor with known variance components and g_3 comes from the exact derivative of the predictor in those components (`g4` is 0 for these cells). The further second-order correction of the plug-in g_1 is not applied: it is numerically unstable when ρ_1 approaches ± 1 or a variance component is weakly identified, while in simulations $g_1 + g_2 + g_3$ has a relative bias of a few percent and close to nominal coverage for these cells. The parametric bootstrap generates the random effects on the full grid and the sampling errors on the sampled cells, re-estimates the model on the sampled cells and predicts every cell, so it covers both kinds of cells. The auxiliary variables must be known for every cell, since they enter the prediction. Self-benchmarking calibrates the weighted aggregate of the sampled cells of each group.

Value

A list with the same structure as `seblup_area()`, with additional `estvarcomp` for spatio-temporal variance/autocorrelation components:

`estcoef` data frame with `beta`, `std.error`, `tvalue`, `pvalue`, as in `sae::eblupSTFH()`, plus `stderr_beta` and `zvalue` (copies, for the column names of the other fastsae models). The p-values use the normal distribution, so `tvalue` is a z-statistic.

`estvarcomp` data frame with estimate, std.error for sigma21, rho1, sigma22, rho2.

`goodness` named vector: `loglikelihood`, the full Gaussian log-likelihood at the REML estimates (as `loglike` in `sae::eblupSTFH()`), and `AIC` and `BIC` computed from it. The restricted (REML) likelihood is not reported: it depends on the covariates through X, so it cannot compare models with different covariates.

`df_eblup` data frame with domain, time, y, eblup, vardir, mse, rse, random_effect_u1, random_effect_u2 and `sampled` (`FALSE` for the cells predicted without a direct estimate). When `mse_type = "none"`, mse and rse are `NA`.

`model` model type ("ST" or "S").

`convergence` logical, whether the algorithm converged.

`n_iter` number of iterations.

`B` number of bootstrap replicates (only when mse_type = "bootstrap").

References

1. Marhuenda, Y., Molina, I., & Morales, D. (2013). Small area estimation with spatio-temporal Fay-Herriot models. Computational Statistics & Data Analysis, 58, 308-325.
2. Rao, J. N., & Molina, I. (2015). Small area estimation. John Wiley & Sons.
3. Saei, A., & Chambers, R. (2005). Out of sample estimation for small areas using area level data. Southampton Statistical Sciences Research Institute, Methodology Working Paper M05/11.
4. Benedetti, R., Piersimoni, F., Pratesi, M., Salvati, N., & Suesse, T. (2024). Handling out-of-sample areas to estimate the unemployment rate at local labour market areas in Italy. International Statistical Review. doi:10.1111/insr.12596

Examples

```
library(fastsae)

# mys_panel: 42 domains x 11 years; domains 21 and 25 are never sampled (y = NA).
# They stay in the data and in W, and are predicted from their neighbours.
m1 <- eblup_stfh(
  y ~ x1 + x2 + x3,
  data = mys_panel,
  vardir = ~vardir,
  domain = ~area,
  time = ~year,
  W = mys_proxmat,
  model = "ST"
)
```

```
# Analytical MSE for every cell, sampled or not
m2 <- eblup_stfh(
  y ~ x1 + x2 + x3,
  data = mys_panel,
  vardir = ~vardir,
  domain = ~area,
  time = ~year,
  W = mys_proxmat,
  model = "ST",
  mse_type = "analytical"
)
head(m2$df_eblup[!m2$df_eblup$sampled, ])
```

`eblup_twofold` *Empirical Best Linear Unbiased Prediction under a Two-fold Fay-Herriot Model.*

Description

This function gives the Empirical Best Linear Unbiased Prediction (EBLUP) under a two-fold Fay-Herriot model (Torabi & Rao, 2014): subareas nested within areas with an area random effect v_i and a subarea random effect u_{ij}.

Usage

```
eblup_twofold(
  formula,
  vardir,
  domain = NULL,
  subarea = NULL,
  data,
  method = c("REML", "ML"),
  mse_type = c("analytical", "bootstrap", "none"),
  B = 100,
  seed = -1,
  maxiter = 100,
  precision = 1e-04,
  self_benchmark = FALSE,
  benchmark_weight = NULL,
  benchmark_target = NULL,
  benchmark_group = NULL,
  print_result = TRUE
)
```

Arguments

- `formula` — an object of class formula that contains a description of the model to be fitted.
- `vardir` — vector or column names from data that contain variance sampling from the direct estimator.
- `domain` — vector, column name or one-sided formula referencing the area identifier column in `data`. Must be supplied.
- `subarea` — vector, column name or one-sided formula referencing the subarea identifier column in `data`. If NULL, rows are numbered consecutively.
- `data` — a data frame or a data frame extension (e.g. a tibble), one row per subarea.
- `method` — Fitting method can be chosen between 'REML' and 'ML'.
- `mse_type` — MSE estimator: 'analytical' (default; Prasad-Rao g1+g2+2*g3, fast; ML adds the bias correction), 'bootstrap' (parametric bootstrap, slower but robust) or 'none' (no MSE; `mse` and `rse` are `NA`).
- `B` — number of parametric bootstrap replicates (only used if mse_type = 'bootstrap').
- `seed` — integer seed for the bootstrap; `-1` (default) or `NULL` sets no seed. A fixed seed does not change the random number stream of the session.
- `maxiter` — maximum number of iterations allowed in the Fisher-scoring algorithm.
- `precision` — convergence tolerance limit for the Fisher-scoring algorithm.
- `self_benchmark` — Logical. If `TRUE`, fits a self-benchmarking model via the augmented model approach (Wang, Fuller, and Qu, 2008; Rao and Molina, 2015, Sec 10.5.1), ensuring that the weighted aggregate of small area estimates automatically matches the direct survey aggregate or specified benchmark targets without post-hoc adjustment.
- `benchmark_weight` — Optional vector, column name, or one-sided formula referencing the domain benchmarking weights in `data` (e.g. population sizes or shares). Defaults to equal weights.
- `benchmark_target` — Optional numeric scalar or named vector specifying fixed aggregate benchmark target(s). If `NULL` (default), calibrates to the weighted aggregate of the direct estimates.
- `benchmark_group` — Optional vector, column name, or one-sided formula referencing the grouping / stratum column in `data` (e.g. area identifier for subarea-to-area benchmarking) for group-specific self-benchmarking.
- `print_result` — print coefficient or not, default value is TRUE.

Details

The model has a form that is response ~ auxiliary variables, with one row per subarea. When the response contains NA the subarea is treated as non-sampled and predicted with the synthetic estimator. MSE is estimated either analytically (Prasad-Rao g1+g2+2*g3 for REML, minus the ML bias term b'grad(g1) for ML, with an exact matrix derivation for g3) or by parametric bootstrap.

Value

The function returns a list with the following objects: `estcoef` a data frame with the estimated model coefficients (`beta`), their standard errors (`std.error`, repeated as `stderr_beta` for compatibility with fastsae 0.1.0), z-statistics (`zvalue`) and normal p-values (`pvalue`), `random_effect_var` named vector with the estimated area (`sigma2_v`) and subarea (`sigma2_u`) variances (paper notation), `goodness` named vector: `loglikelihood` (the full Gaussian log-likelihood at the estimates, for ML and REML fits alike), `AIC` and `BIC` computed from it, following the convention of **sae** for the Fay-Herriot models; BIC uses the number of sampled subareas. The restricted (REML) likelihood is not reported: it depends on the covariates through X, so it cannot compare models with different covariates., `df_eblup` a data frame that contains domain, subarea, y, eblup, vardir, random_effect_area, random_effect_subarea, mse, and rse.

References

1. Torabi, M., & Rao, J. N. K. (2014). On small area estimation under a sub-area level model. *Journal of Multivariate Analysis*, 127, 36–55.
2. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons.

Examples

```
library(fastsae)

# Two-fold Fay-Herriot model (simulated illustration)
set.seed(1)
m <- 20
dat <- do.call(rbind, lapply(1:m, function(d) {
  nd <- sample(2:5, 1)
  data.frame(
    area = d, subarea = paste0(d, "-", seq_len(nd)),
    x1 = rnorm(nd), y = NA_real_, vardir = runif(nd, 0.2, 1)
  )
}))
dat$y <- 1 + dat$x1 + rnorm(m)[dat$area] + rnorm(nrow(dat), sd = 0.5) +
  rnorm(nrow(dat), sd = sqrt(dat$vardir))
m1 <- eblup_twofold(y ~ x1, vardir = "vardir", domain = "area",
                    subarea = "subarea", data = dat)
```

`export_sae` *Export Small Area Estimation Results to Excel or CSV*

Description

Exports comprehensive estimation tables and model summaries to multi-sheet Excel workbooks (`.xlsx`) or comma-separated files (`.csv`). Formatted to meet official statistics dissemination standards.

The generated Excel workbook contains up to four sheets:

- **Estimates**: Domain IDs, Direct survey estimates, SAE model predictions, Standard Errors (SE), MSE, RSE (%), 95% confidence / credible intervals, and official Reliability Flags.
- **Model_Summary**: Model formula, method, convergence status, regression coefficients ($\hat{\beta}$, SE, p-values), variance components (σ_u^2, spatial ρ, temporal ρ_t, ϕ), and goodness-of-fit metrics.
- **Benchmarked** (Optional): Pre- vs post-calibration values, adjustments, and percentage shifts when benchmarking calibration is applied.
- **Benchmarked_Groups** (Optional): the group-level (stage 1) calibration of a hierarchical benchmark.

With a `.csv` file, the estimates are written to `file` and the model summary to `<file>_summary.csv`; the benchmark sheets are only written to Excel workbooks.

Usage

```
export_sae(
  object,
  file,
  benchmark = NULL,
  thresholds = c(25, 50),
  overwrite = TRUE,
  ...
)
```

Arguments

`object`	A fitted `fastsae` object or a `fastsae_benchmark` object.
`file`	Character string specifying the target file path (must end with `.xlsx` or `.csv`).
`benchmark`	Optional `fastsae_benchmark` object to include calibration results alongside the model.
`thresholds`	Numeric vector of length 2 with the RSE (%) cut-offs of the reliability flags (see `add_reliability_flags`). Default is `c(25, 50)`.
`overwrite`	Logical indicating whether to overwrite an existing file. Default is `TRUE`.
`...`	Additional arguments.

Value

Invisibly returns a named list of data frames corresponding to the exported sheets.

Examples

```
library(fastsae)
data(mys)

fit_fh <- eblup_fh(y ~ x1 + x2, vardir = "vardir", data = mys)
tmp_file <- tempfile(fileext = ".xlsx")
export_sae(fit_fh, file = tmp_file)
unlink(tmp_file)
```

fitted.fastsae — *Extract fitted values (EBLUP or HB) from a fastsae object*

Description

Extract fitted values (EBLUP or HB) from a fastsae object

Usage

```
## S3 method for class 'fastsae'
fitted(object, ...)
```

Arguments

`object`	An object of class `fastsae`.
`...`	Additional arguments.

Value

Vector of fitted estimates.

gvf_smooth — *Generalized Variance Function (GVF) Smoothing for Sampling Variances*

Description

Fits Generalized Variance Function (GVF) models to smooth unstable direct sampling variance estimates (D_i) in Small Area Estimation. Supports log-linear regression (Wolter, 2007), Gamma generalized linear models (GLM with log link), and CV power models.

Usage

```
gvf_smooth(
  vardir,
  y = NULL,
  n = NULL,
  method = c("log_linear", "gamma_glm", "cv_power"),
  data = NULL,
  formula = NULL,
  bias_correction = TRUE
)
```

Arguments

`vardir` vector, column name, or one-sided formula referencing the direct sampling variances in `data`.

`y` optional vector, column name, or one-sided formula referencing the direct survey estimates.

`n` optional vector, column name, or one-sided formula referencing domain sample sizes.

`method` character string specifying the GVF smoothing model:

- `"log_linear"` Log-linear model: $\log(D_i) = \alpha_0 + \alpha_1 \log(y_i) + \alpha_2 \log(n_i) + \varepsilon_i$. Includes second-order bias correction upon exponentiation.
- `"gamma_glm"` Gamma GLM with log link: $\mathrm{E}[D_i] = \mu_i, \log(\mu_i) = \alpha_0 + \ldots$. Naturally suited for strictly positive variances without back-transformation bias.
- `"cv_power"` Power model on direct coefficients of variation: $\log(CV_i^2) = \alpha_0 + \ldots$.

Default is `"log_linear"`.

`data` optional data frame containing the variables.

`formula` optional user-supplied formula overriding the default model specification.

`bias_correction` logical. For `"log_linear"`, whether to apply log-normal bias correction $\tilde{D}_i = \exp(\hat{\mu}_i + \hat{\sigma}^2/2)$. Default is `TRUE`.

Value

An object of class `"gvf_smooth"` and `"fastsae_gvf"`, containing:

`smooth_vardir` numeric vector of smoothed sampling variances $\tilde{D}_i$.

`vardir` original raw sampling variances.

`fitted_model` the fitted `lm` or `glm` model object.

`method` method used.

`r_squared` coefficient of determination R^2 (or pseudo-R^2 for GLM).

`df` data frame with raw `vardir`, `smooth_vardir`, and covariates.

`call` the matched function call.

References

1. Wolter, K. M. (2007). *Introduction to Variance Estimation* (2nd ed.). Springer.
2. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation* (2nd ed.). John Wiley & Sons.
3. Rivest, L. P., & Belmonte, E. (2000). A conditional mean squared error of small area estimators. *Survey Methodology*, 26(1), 67-78.

Examples

```
library(fastsae)

# Smooth direct variances using direct estimates y and sample size n
gvf_res <- gvf_smooth(
  vardir = "vardir",
  y = "y",
  n = "n",
  data = mys,
  method = "log_linear"
)

# Inspect summary
summary(gvf_res)

# Use smoothed variances directly in Fay-Herriot model
fit_fh <- eblup_fh(
  y ~ x1 + x2 + x3,
  data = mys,
  vardir = gvf_res$smooth_vardir
)
```

---

hb_area *Hierarchical Bayes for Area-Level Small Area Estimation*

Description

Estimates small area parameters using area-level Hierarchical Bayesian models under various distributions (Gaussian, Binomial, Poisson, Negative Binomial, Beta, Gamma) and spatial random effect structures (non-spatial, BYM2, BYM, Besag/ICAR, Leroux). Two estimation engines are available through `engine`:

- `inla_engine()` (default, or `"inla"`): Integrated Nested Laplace Approximations (**INLA**; Rue et al., 2009).
- `jax_engine()` (or `"jax"`): full MCMC with the No-U-Turn sampler of 'NumPyro' (Phan et al., 2019) on 'JAX', on the CPU or an NVIDIA GPU. Needs Python with `jax` and `numpyro` through **reticulate**; see `setup_numpyro_env`.

The engine constructors hold all engine settings (priors, approximation strategy and further `INLA::inla()` arguments for INLA; MCMC length, device and coefficient prior for JAX), so `hb_area()` itself only describes the model.

Usage

```
hb_area(
  formula,
  data,
```

```
  domain = NULL,
  time = NULL,
  family = c("gaussian", "binomial", "poisson", "nbinomial", "beta", "gamma"),
  spatial = c("none", "bym2", "bym", "besag", "generic1", "leroux", "slm"),
  temporal = c("none", "rw1", "rw2", "ar1", "iid"),
 st_interaction = c("none", "domain-specific", "separable", "type1", "type2", "type3",
    "type4"),
  W = NULL,
  vardir = NULL,
  trials = NULL,
  exposure = NULL,
  engine = "inla",
  self_benchmark = FALSE,
  benchmark_weight = NULL,
  benchmark_target = NULL,
  benchmark_group = NULL,
  benchmark_method = c("posterior", "wfq"),
  print_result = TRUE,
  method = NULL,
  ...
)
```

Arguments

`formula` An object of class `formula` specifying the fixed-effects model (e.g., `y ~ x1 + x2`).

`data` A `data.frame` containing the area-level data (one row per area, or per area and period). Areas without a direct estimate (unsampled areas) are kept as rows with `NA` response and their covariates; they are predicted from the model. An area with no row in `data` has no covariates and is not predicted.

`domain` Vector, column name, or one-sided formula referencing the area/domain identifier in `data`. If `NULL`, domains are numbered consecutively.

`time` Vector, column name, or one-sided formula referencing the time period identifier in `data` (e.g. `time = "year"` or `~year`). Required when `temporal != "none"`.

`family` Character string specifying the response distribution / likelihood. Options:

- `"gaussian"`: Continuous response (Fay-Herriot model). If `vardir` is provided, sampling variances are treated as known. If `vardir = NULL`, residual variance is estimated.
- `"binomial"`: Binary / proportion response. Requires `trials`.
- `"poisson"`: Count response / disease rate. Supports `exposure`.
- `"nbinomial"`: Negative Binomial for overdispersed counts.
- `"beta"`: Continuous proportions strictly in (0, 1). If `vardir` is provided, area-specific precision is calculated via Janicki (2020) formula $\phi_i = y_i(1-y_i)/V_i - 1$ and injected via INLA's `scale` argument. If `trials` is provided, precision is scaled as $n_i - 1$.
- `"gamma"`: Skewed positive continuous response.

`spatial` Character string specifying the spatial random effect structure:

- `"none"`: Non-spatial IID area random intercept (default).
- `"bym2"`: Scaled Besag-York-Mollié 2 model (Riebler et al., 2016), decomposing area variance into spatial (ϕ) and unstructured components.
- `"bym"`: Classic Besag-York-Mollié (1991) model.
- `"besag"`: Intrinsic Conditional Autoregressive (ICAR) model.
- `"generic1"`: Leroux spatial autoregressive model.
- `"slm"`: Spatial lag model, $\eta = (I - \rho W)^{-1}(X\beta + \epsilon)$ (INLA `"slm"`). Note that the covariates are spatially filtered too, so this is *not* the Bayesian counterpart of `eblup_sfh`, whose SAR structure applies to the random effect only: $X\beta + (I - \rho W)^{-1}\epsilon$.

`temporal` Character string specifying the temporal random effect structure:

- `"none"`: No temporal random effect (default cross-sectional model).
- `"rw1"`: First-order random walk across time periods.
- `"rw2"`: Second-order random walk across time periods.
- `"ar1"`: First-order autoregressive process across time periods.
- `"iid"`: Unstructured independent time effects.

`st_interaction` Character string specifying the spatio-temporal structure:

- `"none"`: Additive main spatial and temporal effects (default).
- `"domain-specific"`: Domain-specific temporal random walk / AR(1) dynamics with shared variance parameter (directly corresponds to the **tipsae** spatio-temporal model). Not available with `temporal = "rw2"`: without a common trend, the linear trend of a domain without direct estimates has a flat prior; use `"type2"`.
- `"separable"`: Grouped dynamic spatial field evolving over time via AR(1) (corresponds to classical Spatio-Temporal Fay-Herriot models, Marhuenda et al. 2013). Only `temporal = "ar1"` or `"iid"`: with a random walk in time every domain's level in the field has a flat prior, as the field has no spatial main effect; use `"type4"`.
- `"type1"`: Knorr-Held (2000) Type I interaction (unstructured space $\times$ unstructured time).
- `"type2"`: Knorr-Held Type II interaction (unstructured space $\times$ structured time): an independent `temporal` process for each domain.
- `"type3"`: Knorr-Held Type III interaction (structured space $\times$ unstructured time): an independent ICAR field for each period.
- `"type4"`: Knorr-Held Type IV interaction (structured space $\times$ structured time): an ICAR field evolving over time with the `temporal` structure.

The `"type*"` options add the interaction to the main spatial and temporal effects. `"type3"` and `"type4"` need a structured `spatial` effect (`"bym2"`, `"bym"`, `"besag"` or `"leroux"`); with INLA the interaction is an ICAR (`"besag"`) field on the graph of `W`. With `temporal = "rw1"` or `"rw2"`, the type IV interaction (and type II with `"rw2"`) uses the scaled Kronecker structure with all the identifiability constraints of Knorr-Held (2000): sums over the domains are 0 in every period, and sums over the periods (and, for `"rw2"`, linear trends) are 0 in every

domain. INLA's default constraints leave the latter out; with them alone, a domain without direct estimates had a posterior SD of 36-58 (other domains about 0.4).

`W` Proximity or spatial adjacency matrix. Can be a square `matrix`, `Matrix`, or spdep nb or `listw` object. Dimensions must match the total number of domains in `data`. Required when `spatial != "none"`. The CAR models (`"besag"`, `"bym"`, `"bym2"`, `"leroux"`) use only the neighbourhood pattern `W > 0`, not the weights: give a neighbourhood graph (for example `create_weights(method = "queen")` or `"knn"`). A dense matrix such as distance-decay weights becomes a nearly complete graph and gives a warning. `"slm"` uses the row-standardised weights.

`vardir` Vector, column name, or formula specifying the sampling variances of the direct estimator (for `family = "gaussian"` or `family = "beta"`).

`trials` Vector, column name, or formula specifying sample sizes / total trials per area (for `family = "binomial"`).

`exposure` Vector, column name, or formula specifying expected counts or population exposure offsets (for `family = "poisson"` or `"nbinomial"`).

`engine` Estimation engine: `inla_engine()` or `jax_engine()`, or the strings `"inla"` (default) and `"jax"` for their default settings. See the Engines section.

`self_benchmark` Logical. If `TRUE`, the weighted aggregate of the area estimates is made to match the direct survey aggregate or `benchmark_target`; see `benchmark_method`.

`benchmark_weight`
Optional vector, column name, or one-sided formula referencing the domain benchmarking weights in `data` (e.g. population sizes or shares). Defaults to equal weights.

`benchmark_target`
Optional numeric scalar or named vector specifying fixed aggregate benchmark target(s). If `NULL` (default), calibrates to the weighted aggregate of the direct estimates.

`benchmark_group`
Optional vector, column name, or one-sided formula referencing the grouping / stratum column in `data` (e.g. province) for group-specific self-benchmarking.

`benchmark_method`
How self-benchmarking is done when `self_benchmark = TRUE`: `"posterior"` (default) benchmarks every posterior draw of the area parameters (an additive shift for `"gaussian"`, a common shift on the logit scale for `"binomial"` and `"beta"`, a common ratio for `"poisson"`, `"nbinomial"` and `"gamma"`), so the constraint holds exactly for the posterior mean and the reported SD / credible intervals refer to the benchmarked estimator (Bayesian benchmarking, cf. Datta et al., 2011). `"wfq"` uses the augmented-covariate approach of Wang, Fuller and Qu (2008), which is exact only for the linear Gaussian EBLUP; for the HB models it holds only approximately (non-flat prior on the coefficients, integration over the hyperparameters) and not at all for non-Gaussian likelihoods. With INLA, posterior draws need the package **sn** (used by `INLA::inla.posterior.sample()`); without it `"wfq"` is used with a warning.

`print_result` Logical. If `TRUE` (default), prints a summary of results.

method	Deprecated; use engine. method = "laplace" becomes engine = inla_engine(strategy = "laplace"); the former **lme4** GLMM route for binomial and Poisson models has been removed.
...	Deprecated. strategy, link, scale_model, prior_*, benchmark_draws, benchmark_seed and arguments of INLA::inla() are now given to inla_engine(); passed here they are forwarded to it with a warning.

Value

An object of class c("fastsae_hb_area", "fastsae_hb", "fastsae") containing:

- df_hb: Data frame with domain estimates, including domain, time (if specified), observed y, predicted hb, linear predictor linear_pred, posterior standard error sd, mse, relative error rse (%), 95% credible interval (ci_lower, ci_upper), and random_effect.
- estcoef: Data frame of estimated regression coefficients.
- hyperpar: Data frame of hyperparameter posterior estimates.
- random_effect_var: Posterior mean of the area random effect variance (see the Engines section on its accuracy when the variance is small).
- random_effect_var_time: Estimated temporal random effect variance (if temporal).
- phi: Estimated spatial variance proportion (for BYM2) or spatial autocorrelation.
- rho_time: Estimated temporal autocorrelation parameter (for AR1).
- goodness: Model fit metrics (DIC, WAIC, Marginal Log-Likelihood).
- family: Response family used.
- spatial: Spatial model type used.
- temporal: Temporal model type used.
- st_interaction: Spatio-temporal interaction type used.
- self_benchmark: Logical indicating whether self-benchmarking was active.
- benchmark_summary: Summary table of benchmark calibration (if self-benchmarking).
- fit: Raw fitted model object (inla, or the list returned by the Python backend for the JAX engine).
- engine: Name of the estimation engine ("inla" or "jax"); engine_spec holds its settings (inla_engine() or jax_engine() object).
- draws, rhat_max, device, elapsed_seconds: With the JAX engine, the posterior draws of the area parameters (rows of df_hb by draws), the largest split R-hat (convergence is rhat_max < 1.05), the JAX device and the sampling time.
- call: Matched function call.

Engines

Both engines return the same structure, so print, summary, benchmark_sae, compare_sae and the other methods work with either. Feature support:

Feature	inla	jax

Families	all six	all six
`spatial`	all	all but `"slm"`
`temporal`	all	all but `"rw2"`
`st_interaction`	all	all
Self-benchmarking	posterior, wfq	posterior; wfq (gaussian)
Horseshoe prior, Student-t random effects	–	yes

The `"jax"` engine uses weakly informative priors (half-normal standard deviations, uniform spatial mixing / autocorrelation parameters) instead of the PC priors of INLA, so its estimates differ slightly. For `family = "gamma"` it estimates a common shape and does not use `vardir`.

With the `"inla"` engine, Gaussian models and the Leroux and SLM effects of any family are fitted in INLA's `"classic"` mode (Rue, Martino and Chopin, 2009); the other models use INLA's default `"compact"` mode (van Niekerk et al., 2023), which leaves the linear predictors out of the latent field and is designed for data-rich models. Gaussian models with a `"type1"` to `"type4"` interaction and three hyperparameters (a Besag or iid area effect and a random walk or iid in time) also use a grid over the hyperparameters (`int.strategy = "grid"`); with `temporal = "ar1"` or a BYM, BYM2, Leroux or SLM area effect the grid grows exponentially (13 minutes for 12 areas with AR(1)) and was not validated, so INLA's default integration is kept. These choices rest on comparisons with long Stan runs with the same model and priors (Gaussian, binomial, Poisson, beta, gamma and negative binomial; iid, Besag, BYM2, Leroux and SLM effects; RW1 in time and the four Knorr-Held interactions):

- Gaussian: compact overstated the posterior SDs by up to 18% when the area variance was small relative to `vardir`; classic matched Stan (means within 0.07 posterior SD, SDs within 0.91-1.03 times) at the same speed. With an interaction both modes overstated the SDs by 6-13%; classic with the grid matched Stan (means within 0.04 SD, SDs within 2%) and took 5-8 times longer.
- Binomial, Poisson, beta and gamma with iid, Besag or BYM2 effects or an interaction: compact matched Stan (means within 0.16 SD, SDs within 0.95-1.08 times), and classic was much slower for many areas (beta, 2000 areas: 32 s against 1.3 s).
- Leroux and SLM: compact moved the Poisson posterior means by 0.12-0.15 SD; classic was within 0.03 SD.
- Negative binomial: with one count per area the overdispersion and the area effect are only weakly identified, and both modes were off (means up to 0.64 SD); the model gives a warning.

`inla_engine(inla.mode = , control.inla = )` overrides these settings. For Gaussian models with a small area variance, the classic summary of that variance (`random_effect_var`) was biased downwards, while the area estimates were accurate. The validation scripts are in the package repository (`benchmarks/validation`).

References

1. Rao, J. N. K., and Molina, I. (2015). *Small Area Estimation*. John Wiley & Sons.
2. Wang, J., Fuller, W. A., and Qu, Y. (2008). Small area estimation under a restriction. *Survey Methodology*, 34(1), 29-36.

3. Riebler, A., Sørbye, S. H., Simpson, D., and Rue, H. (2016). An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4), 1145-1165.
4. Marhuenda, Y., Molina, I., and Morales, D. (2013). Small area estimation with spatio-temporal Fay-Herriot models. *Computational Statistics & Data Analysis*, 58, 308-325.
5. De Nicolò, S., and Gardini, A. (2024). The R Package tipsae: Tools for Mapping Proportions and Indicators on the Unit Interval. *Journal of Statistical Software*, 108(1), 1-36.
6. Phan, D., Pradhan, N., and Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv:1912.11554*.
7. Piironen, J., and Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2), 5018-5051.
8. Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2), 319-392.
9. Knorr-Held, L. (2000). Bayesian modelling of inseparable space-time variation in disease risk. *Statistics in Medicine*, 19(17-18), 2555-2567.
10. van Niekerk, J., Krainski, E., Rustand, D., and Rue, H. (2023). A new avenue for Bayesian inference with INLA. *Computational Statistics & Data Analysis*, 181, 107692.

Examples

```
if (requireNamespace("INLA", quietly = TRUE)) {
  library(fastsae)

  # 1. Non-spatial Gaussian Fay-Herriot with INLA
  m_norm <- hb_area(
    y ~ x1 + x2 + x3,
    data = mys,
    vardir = "vardir",
    family = "gaussian"
  )

  # 2. Self-benchmarking Fay-Herriot with INLA (benchmarked posterior draws)
  m_bench <- hb_area(
    y ~ x1 + x2 + x3,
    data = mys,
    vardir = "vardir",
    self_benchmark = TRUE,
    benchmark_weight = "n"
  )

  # 3. Spatial BYM2 Gaussian Fay-Herriot with INLA, faster approximation;
  #    the CAR model needs a neighbourhood graph (4 nearest centroids)
 W_knn <- create_weights(cbind(sim_area$x_coord, sim_area$y_coord), method = "knn", k = 4)
  m_bym2 <- hb_area(
    y ~ x1 + x2 + x3,
    data = mys,
    vardir = "vardir",
    family = "gaussian",
```

```
    spatial = "bym2",
    W = W_knn,
    engine = inla_engine(strategy = "simplified.laplace")
  )
}

# 4. The same Fay-Herriot model by MCMC (NumPyro NUTS on JAX)
if (check_numpyro_available()) {
  m_jax <- hb_area(
    y ~ x1 + x2 + x3,
    data = mys,
    vardir = "vardir",
    engine = jax_engine(samples = 500, chains = 2)
  )
}
```

---

hb_engine *Estimation engines for the Hierarchical Bayes models*

Description

Constructors for the engine argument of hb_area, hb_unit and hb_twofold. Each collects the settings of one estimation engine, so that the model functions themselves only describe the model. Passing the string "inla" or "jax" is the same as passing inla_engine() or jax_engine() with their defaults.

Usage

```
inla_engine(
  strategy = c("laplace", "simplified.laplace", "gaussian"),
  link = NULL,
  scale_model = TRUE,
  prior_prec = NULL,
  prior_phi = NULL,
  prior_rho = NULL,
  prior_prec_time = NULL,
  prior_rho_time = NULL,
  prior_prec_subarea = NULL,
  draws = NULL,
  seed = 1L,
  ...
)

jax_engine(
  warmup = 500,
  samples = 1000,
  chains = 2,
```

```
  device = c("auto", "cpu", "cuda", "metal"),
  seed = 42,
  prior_beta = c("normal", "horseshoe"),
  robust = FALSE,
  keep_draws = TRUE,
  progress = interactive()
)

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

Arguments

`strategy` INLA approximation strategy: `"laplace"` (default, full Laplace approximation, highest accuracy), `"simplified.laplace"` (faster) or `"gaussian"` (fastest).

`link` Link function for the likelihood (`hb_area()` only). `NULL` (default) uses the canonical link of the family: identity for gaussian, logit for binomial and beta, log for poisson, nbinomial and gamma. Binomial and beta also accept `"probit"`, `"cloglog"` and `"loglog"`. The estimates (`hb`) are on the response scale for any link.

`scale_model` Logical; scale the intrinsic GMRF models (besag, bym, rw1, rw2) to a generalized variance of 1. Default `TRUE`.

`prior_prec` Prior for the precision of the area (domain) random effect. `NULL` (default): the PC prior `list(prior = "pc.prec", param = c(1, 0.01))` in `hb_area()` and `hb_twofold()`, `list(prior = "loggamma", param = c(0.01, 0.01))` in `hb_unit()`. The PC prior puts probability 0.01 on a standard deviation above 1, which suits area-level responses such as proportions, rates and log scales; unit-level responses are often on a raw scale (e.g. hectares), where it would over-shrink the area effects, hence the vague log-gamma prior. For an area-level response with a large scale, set `prior_prec = list(prior = "pc.prec", param = c(U, 0.01))` with `U` a plausible upper value of the area standard deviation.

`prior_phi` Prior for the mixing parameter of `spatial = "bym2"`. `NULL` (default): the PC prior `list(prior = "pc", param = c(0.5, 0.5))`.

`prior_rho` Prior for the spatial correlation of `spatial = "generic1"` or `"slm"` (`hb_area()` only); `NULL` uses the INLA default.

`prior_prec_time` Prior for the precision of the temporal random effect (`hb_area()` only); `NULL` uses the INLA default.

`prior_rho_time` Prior for the autocorrelation of `temporal = "ar1"` (`hb_area()` only); `NULL` uses the INLA default.

`prior_prec_subarea` Prior for the precision of the subarea random effect (`hb_twofold()` only). `NULL` (default): `list(prior = "pc.prec", param = c(1, 0.01))`.

`draws` Number of posterior draws from `INLA::inla.posterior.sample()`: for posterior self-benchmarking in `hb_area()` (`NULL`: 1000) and for the area aggregates

of hb_twofold() (NULL: 200; 0 skips the sampling and approximates the aggregate uncertainty from independent sub-area effects).

seed — Random seed of the sampler. Default 42.

... — Further arguments passed to INLA::inla(), such as control.compute or num.threads.

warmup — Number of warmup (adaptation) iterations per chain. Default 500.

samples — Number of posterior draws kept per chain. Default 1000.

chains — Number of MCMC chains. Default 2. With more than one chain, the fit reports the largest split R-hat and convergence is rhat_max < 1.05.

device — "auto" (default: CUDA when available, otherwise CPU), "cpu", "cuda" or "metal" (Apple silicon GPU, through jax-metal). JAX fixes its device at the first fit of an R session. "metal" has a fixed cost of about two minutes per fit, so on Apple silicon it pays off only for large numbers of domains (thousands); for that reason "auto" uses the CPU on macOS.

prior_beta — Prior for the regression coefficients: "normal" (default) or "horseshoe" (regularized horseshoe, for many candidate covariates).

robust — Logical; Student-t instead of normal random effects (the area effects, and the sub-area effects of hb_twofold()), with the degrees of freedom estimated. Default FALSE.

keep_draws — Logical; keep the posterior draws of the area parameters in $draws of the fit (needed by benchmark_sae() to propagate the posterior covariance). Default TRUE.

progress — Logical; show NumPyro's progress bar. Default interactive().

x — A "fastsae_engine" object.

Value

An object of class "fastsae_engine".

INLA

inla_engine() fits the model by integrated nested Laplace approximation (**INLA**); it is available for all HB models. Priors are given in **INLA** syntax. A prior or setting left at NULL takes the default of the model function; a setting the model does not use (such as prior_rho_time in hb_unit()) is ignored with a warning.

JAX

jax_engine() fits the model by MCMC (NumPyro's No-U-Turn sampler on JAX, Phan et al. 2019) on CPU, CUDA GPU or Apple Metal, through **reticulate**. It needs Python with jax and numpyro; see setup_numpyro_env and check_numpyro_available. The fixed effects get a normal or a regularized horseshoe prior (Piironen and Vehtari 2017), and the variance components half-normal priors. spatial = "slm" and temporal = "rw2" are not available. jax_engine() is available for hb_area() and hb_twofold().

References

Phan, D., Pradhan, N. and Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv:1912.11554*.

Piironen, J. and Vehtari, A. (2017). Sparsity information and regularization in the horseshoe and other shrinkage priors. *Electronic Journal of Statistics*, 11(2), 5018–5051.

See Also

hb_area, hb_unit, hb_twofold

Examples

```
inla_engine(strategy = "simplified.laplace")
jax_engine(samples = 2000, chains = 4, device = "cuda")
```

---

hb_twofold — *Two-Fold Hierarchical Bayes for Sub-Area Level Small Area Estimation*

Description

Estimates small area parameters under a two-fold sub-area level model (Torabi & Rao, 2014 Bayesian hierarchical framework) using Integrated Nested Laplace Approximations (**INLA**) or, with `engine = jax_engine()`, MCMC (NumPyro's No-U-Turn sampler on JAX). The model accounts for nested random effects at both the primary area level (v_i) and the nested sub-area level (u_{ij}), supporting Gaussian, Binomial (logistic), and Poisson responses, with optional spatial correlation structures (BYM2, Besag) across areas and aggregate area-level estimation.

Usage

```
hb_twofold(
  formula,
  vardir = NULL,
  domain,
  subarea = NULL,
  data,
  family = c("gaussian", "binomial", "poisson"),
  spatial = c("none", "bym2", "besag"),
  W = NULL,
  weight = NULL,
  trials = NULL,
  exposure = NULL,
  engine = "inla",
  compute_area = TRUE,
  print_result = TRUE,
  ...
)
```

Arguments

`formula` An object of class `formula` specifying the fixed-effects model (e.g., `y ~ x1 + x2`).

`vardir` Character string or vector specifying direct sampling variances (ψ_{ij}) for continuous (Gaussian) responses. Required when `family = "gaussian"`.

`domain` Character string, column name, or vector referencing the primary area identifier in `data`. Must be supplied.

`subarea` Optional character string, column name, or vector referencing the sub-area identifier in `data`. If `NULL`, sub-areas are numbered consecutively per domain.

`data` A `data.frame` or tibble containing one row per sub-area.

`family` Character string specifying the response likelihood. Options:

- `"gaussian"`: Continuous response with known sampling variance (default).
- `"binomial"`: Binary / count proportion with sample sizes specified by `trials`.
- `"poisson"`: Count response with expected exposures specified by `exposure`.

`spatial` Character string specifying the spatial structure across primary areas:

- `"none"`: Non-spatial independent and identically distributed (IID) area effects (default).
- `"bym2"`: Scaled Besag-York-Mollié 2 spatial model (Riebler et al., 2016).
- `"besag"`: Intrinsic Conditional Autoregressive (ICAR) spatial model.

`W` Proximity or spatial adjacency matrix across unique primary areas. Can be a square `matrix`, `Matrix`, or `spdep` `nb` or `listw` object. Dimensions must match the total number of unique primary areas in `domain`. Required when `spatial != "none"`.

`weight` Optional character string or numeric vector specifying sub-area population weights (w_{ij}) used to aggregate sub-area predictions to primary area estimates $\hat{\theta}_i = \sum_j w_{ij}\hat{\theta}_{ij}$. If `NULL`, equal weights per area are used.

`trials` Optional character string or vector specifying total number of trials per sub-area. Required when `family = "binomial"`.

`exposure` Optional character string or vector specifying expected baseline exposures per sub-area. Required when `family = "poisson"`.

`engine` Estimation engine: `"inla"` (default) or an `inla_engine()` object, or `"jax"` or a `jax_engine()` object. `inla_engine()` holds the approximation strategy, the priors (`prior_prec` for the area effect τ_v and `prior_prec_subarea` for the sub-area effect τ_u, both by default `list(prior = "pc.prec", param = c(1, 0.01))`; `prior_phi` for BYM2), `scale_model`, the number of posterior `draws` for the area aggregates (default 200) and their `seed`, and further `INLA::inla()` arguments. The Gaussian model is fitted in INLA's `"classic"` mode, the binomial and Poisson models in INLA's default `"compact"` mode, as in `hb_area`; `inla_engine(inla.mode = )` selects a mode. `jax_engine()` samples the same model by MCMC (see the Engines section).

`compute_area` Logical. If `TRUE` (default), computes aggregate primary area-level estimates and posterior standard errors from joint posterior draws (sampled from the INLA fit, or the MCMC draws with `jax_engine()`).

print_result	Logical. If TRUE (default), prints a summary of results.
...	Deprecated. strategy, scale_model, prior_prec_area (now prior_prec), prior_prec_subarea, prior_phi, n_samples (now draws) and arguments of INLA::inla() are now given to inla_engine(); passed here they are forwarded to it with a warning.

Value

An object of class c("fastsae_hb_twofold", "fastsae_hb", "fastsae") containing:

- df_hb: Data frame with sub-area level estimates:
  - domain: Primary area identifier.
  - subarea: Sub-area identifier.
  - y: Direct estimate / observed response (NA for unsampled).
  - hb: Posterior mean estimate of sub-area mean.
  - linear_pred: Posterior mean of linear predictor.
  - vardir: Direct sampling variance (for Gaussian).
  - sd: Posterior standard deviation (standard error).
  - mse: Posterior Mean Squared Error (sd^2).
  - rse: Relative Standard Error (%).
  - ci_lower: 2.5% quantile of posterior credible interval.
  - ci_upper: 97.5% quantile of posterior credible interval.
  - random_effect_area: Posterior mean of area random effect ($\hat{v}_i$).
  - random_effect_subarea: Posterior mean of sub-area random effect ($\hat{u}_{ij}$).
- df_area: Data frame with aggregate primary area-level estimates:
  - domain: Primary area identifier.
  - hb_area: Aggregate area mean estimate.
  - sd_area: Posterior standard error of area estimate.
  - mse_area: Posterior Mean Squared Error of area estimate.
  - rse_area: Relative Standard Error of area estimate (%).
  - ci_lower_area: 2.5% quantile of area credible interval.
  - ci_upper_area: 97.5% quantile of area credible interval.
  - n_subareas: Number of sub-areas in the area.
- df_subarea: Alias pointing to df_hb.
- estcoef: Data frame of estimated regression coefficients (β, standard error, z-value, p-value, credible intervals).
- hyperpar: Data frame of hyperparameter posterior estimates.
- random_effect_var: Named vector with the posterior means of the area (sigma2_v) and sub-area (sigma2_u) variances. When a variance is close to 0 its posterior is skewed and INLA's summary can be biased (on simulated data sigma2_u was 0.047 against an exact posterior mean of 0.072); the sub-area and area estimates are the reliable output.
- phi: Estimated spatial variance proportion (for BYM2).
- goodness: Model fit metrics (DIC, WAIC, Marginal Log-Likelihood).

- family: Response family used.
- spatial: Spatial model type used.
- model: Label of model ("HB-TWOFOLD").
- method: Description of method.
- convergence: Logical indicating convergence.
- level: "subarea".
- fit: Raw fitted INLA model object, or the list returned by the Python backend for the JAX engine.
- engine, engine_spec: Engine name ("inla" or "jax") and its settings.
- draws, rhat_max, device, elapsed_seconds: With the JAX engine, the posterior draws of the sub-area parameters (rows of df_hb by draws, kept when jax_engine(keep_draws = TRUE)), the largest split R-hat (convergence is rhat_max < 1.05), the JAX device and the sampling time.
- call: Matched function call.

Engines

Both engines fit the same model and return the same structure. With jax_engine() the area effect (iid, Besag or BYM2) and the iid sub-area effect have half-normal priors on their standard deviations (on the standardised response scale for family = "gaussian") instead of the PC priors of inla_engine(), so the estimates differ slightly. On simulated data (20 areas, 56 sub-areas; Gaussian, binomial, Poisson and Gaussian with BYM2) the area means of the two engines correlated above 0.999 and differed by a median of 0.02-0.07 posterior SD, with area SDs 0.85-1.14 times those of INLA. The posterior means of the variances were larger with MCMC (area variance 0.93 against 0.68 for the Gaussian model): their posteriors are skewed, and INLA's summary is biased downwards for small variances. jax_engine(robust = TRUE) gives Student-t area and sub-area effects and prior_beta = "horseshoe" a regularized horseshoe prior on the coefficients.

References

1. Torabi, M., & Rao, J. N. K. (2014). On small area estimation under a sub-area level model. *Journal of Multivariate Analysis*, 127, 36-55.
2. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation* (2nd ed.). John Wiley & Sons.
3. Riebler, A., Sørbye, S. H., Simpson, D., & Rue, H. (2016). An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4), 1145-1165.

Examples

```
if (requireNamespace("INLA", quietly = TRUE)) {
  library(fastsae)
  set.seed(42)
  m <- 10
  dat <- do.call(rbind, lapply(1:m, function(d) {
    nd <- sample(2:4, 1)
    data.frame(
```

```
      area = d,
      subarea = paste0(d, "-", seq_len(nd)),
      x1 = rnorm(nd),
      vardir = runif(nd, 0.2, 0.8)
    )
  }))
  dat$y <- 1 + dat$x1 + rnorm(m)[dat$area] + rnorm(nrow(dat), sd = 0.5) +
    rnorm(nrow(dat), sd = sqrt(dat$vardir))

  fit <- hb_twofold(y ~ x1, vardir = "vardir", domain = "area",
                    subarea = "subarea", data = dat)
  summary(fit)
}
```

---

hb_unit *Hierarchical Bayes for Unit-Level Small Area Estimation*

Description

Estimates small area parameters using unit-level Hierarchical Bayesian models (Battese-Harter-Fuller model and generalized linear mixed models) with Integrated Nested Laplace Approximations (**INLA**). Supports Gaussian, Binomial (logistic regression for binary unit responses), and Poisson likelihoods, with optional spatial random effect structures (BYM2, Besag) and finite population adjustments.

Usage

```
hb_unit(
  formula,
  unit_data,
  Xpop,
  domain_var,
  popsize_var = NULL,
  family = c("gaussian", "binomial", "poisson"),
  spatial = c("none", "bym2", "besag"),
  W = NULL,
  popnmean_xpop = NULL,
  engine = "inla",
  print_result = TRUE,
  ...
)
```

Arguments

formula	An object of class `formula` specifying the unit-level fixed-effects model (e.g., `y ~ x1 + x2`).
unit_data	A `data.frame` containing the unit-level survey sample data.

`Xpop` A `data.frame` containing auxiliary population information for all domains (either domain population means or unit-level population records). Must contain the domain identifier specified by `domain_var`.

`domain_var` Character string specifying the column name for the domain identifier in both `unit_data` and `Xpop`.

`popsize_var` Optional character string specifying the column name for domain population sizes (N_d) in `Xpop`. If provided, finite population adjustment $\hat{\bar{Y}}_d = f_d \bar{y}_{d,s} + (1 - f_d)\hat{\mu}_{d,r}$ is computed, where $f_d = n_d/N_d$. If `NULL`, the superpopulation expectation $\hat{\bar{Y}}_d = \bar{X}_d^\top \hat{\beta} + \hat{u}_d$ is reported.

`family` Character string specifying the response likelihood. Options:

- `"gaussian"`: Continuous response (Battese-Harter-Fuller model, default).
- `"binomial"`: Binary / Bernoulli response (0 or 1, e.g. poverty indicator).
- `"poisson"`: Count response (non-negative integer).

The domain means of `"binomial"` and `"poisson"` are approximate and biased; see Details.

`spatial` Character string specifying the spatial random effect structure across domains:

- `"none"`: Non-spatial independent and identically distributed (IID) domain effects (default).
- `"bym2"`: Scaled Besag-York-Mollié 2 spatial model (Riebler et al., 2016).
- `"besag"`: Intrinsic Conditional Autoregressive (ICAR) spatial model.

`W` Proximity or spatial adjacency matrix. Can be a square `matrix`, `Matrix`, or spdep `nb` or `listw` object. Dimensions must match the total number of unique domains in `Xpop`. Required when `spatial != "none"`.

`popnmean_xpop` Optional matrix or data frame of auxiliary population means per domain. If `NULL` (default), means are extracted directly from `Xpop`.

`engine` Estimation engine: `"inla"` (default) or an `inla_engine()` object holding the approximation strategy, the priors (`prior_prec`, default `list(prior = "loggamma", param = c(0.01, 0.01))`, vague on any response scale; `prior_phi` for BYM2), `scale_model` and further `INLA::inla()` arguments. The Gaussian model is fitted in INLA's `"classic"` mode, the binomial and Poisson models in INLA's default `"compact"` mode (for them classic was much slower: 72 s against 2.6 s for 4000 binary units); `inla_engine(inla.mode = )` selects a mode.

`print_result` Logical. If `TRUE` (default), prints a summary of results.

`...` Deprecated. `strategy`, `scale_model`, `prior_prec`, `prior_phi` and arguments of `INLA::inla()` are now given to `inla_engine()`; passed here they are forwarded to it with a warning.

Details

Non-Gaussian families. The non-sampled part of a domain mean is predicted at the mean covariates of the non-sampled units, $g^{-1}(\bar{x}_{d,r}^\top \beta + u_d)$. For the identity link of `"gaussian"` this is the mean of $x_j^\top \beta + u_d$ over the units. For the logit and log links of `"binomial"` and `"poisson"` it is not: the mean of $g^{-1}(x_j^\top \beta + u_d)$ over the units differs from it whenever the covariates vary within the domain (for the log link it is larger by the factor $E[\exp((x_j - \bar{x})^\top \beta)] > 1$). In a simulation with

one normal covariate of within-domain variance 1, the domain means of the sampled domains were biased by -8% (binomial, slope 1.2) and -16% (Poisson, slope 0.6), and the 95% credible intervals covered the true means in 90% and 78% of the cases. Unbiased estimation needs the covariates of every population unit, as in the empirical best predictor of Molina and Rao (2010); a unit-level `Xpop` is currently averaged per domain. A warning is given for these families.

Value

An object of class `c("fastsae_hb_unit", "fastsae_hb", "fastsae")` containing:

- `df_hb`: Data frame with domain estimates, including:
  - `domain`: Domain identifier.
  - `hb`: Estimated domain mean or proportion.
  - `linear_pred`: Posterior mean of domain linear predictor.
  - `sd`: Posterior standard deviation (standard error).
  - `mse`: Posterior Mean Squared Error (sd^2).
  - `rse`: Relative Standard Error (%).
  - `ci_lower`: 2.5% quantile of posterior credible interval.
  - `ci_upper`: 97.5% quantile of posterior credible interval.
  - `samp_size`: Sample size (n_d) in the domain (0 for unsampled).
  - `pop_size`: Population size (N_d), if `popsize_var` is provided.
  - `estimated_total`: Estimated domain population total (if `popsize_var` is provided).
  - `sample_mean`: Direct sample mean ($\bar{y}_{d,s}$).
  - `random_effect`: Posterior mean of domain random effect ($\hat{u}_d$).
- `estcoef`: Data frame of estimated regression coefficients (posterior mean, sd, z-value, p-value, credible intervals).
- `hyperpar`: Data frame of hyperparameter posterior estimates.
- `random_effect_var`: Posterior mean of the domain random effect variance (σ_u^2). With few domains and the vague default prior its posterior is very skewed and INLA's summary of it is not accurate (on `cornsoybean`, 12 counties: about 70 against an exact posterior mean of 41 and median of 7, while the county means and SDs agree with the exact posterior). Do not read it as a REML estimate; the domain estimates in `df_hb` are the reliable output.
- `residual_var`: Estimated residual variance (σ_e^2, for Gaussian).
- `phi`: Estimated spatial variance proportion (for BYM2).
- `goodness`: Model fit metrics (DIC, WAIC, Marginal Log-Likelihood).
- `family`: Response family used.
- `spatial`: Spatial model type used.
- `unsampled_domains`: Character vector of unsampled domain identifiers.
- `fit`: Raw fitted INLA model object.
- `engine`, `engine_spec`: Engine name (`"inla"`) and its settings.
- `call`: Matched function call.

References

1. Molina, I., & Rao, J. N. K. (2010). Small area estimation of poverty indicators. *Canadian Journal of Statistics*, 38(3), 369-385.
2. Battese, G. E., Harter, R. M., & Fuller, W. A. (1988). An error-components model for prediction of county crop areas using survey and satellite data. *Journal of the American Statistical Association*, 83(401), 28-36.
3. Rao, J. N. K., & Molina, I. (2015). *Small Area Estimation* (2nd ed.). John Wiley & Sons.
4. Riebler, A., Sørbye, S. H., Simpson, D., & Rue, H. (2016). An intuitive Bayesian spatial model for disease mapping that accounts for scaling. *Statistical Methods in Medical Research*, 25(4), 1145-1165.
5. Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B*, 71(2), 319-392.

Examples

```
if (requireNamespace("INLA", quietly = TRUE)) {
  library(fastsae)
  data(cornsoybean)
  data(cornsoybeanmeans)

  # Prepare auxiliary population means
  df_pop <- cornsoybeanmeans
  names(df_pop)[names(df_pop) == "MeanCornPixPerSeg"] <- "CornPix"
  names(df_pop)[names(df_pop) == "MeanSoyBeansPixPerSeg"] <- "SoyBeansPix"

  # Unit survey data
  df_sample <- cornsoybean
  names(df_sample)[names(df_sample) == "County"] <- "CountyIndex"

  # Fit unit-level Hierarchical Bayes model
  fit_hb_u <- hb_unit(
    formula = CornHec ~ CornPix + SoyBeansPix,
    unit_data = df_sample,
    Xpop = df_pop,
    domain_var = "CountyIndex",
    popsize_var = "PopnSegments"
  )
  print(fit_hb_u)
}
```

`map_sae` *Unified Spatial Choropleth Mapping for Small Area Estimates*

Description

Provides a comprehensive, single-gateway ("satu pintu") spatial choropleth mapping interface for Small Area Estimation models fitted with **fastsae** (e.g. `eblup_fh`, `eblup_sfh`, `eblup_stfh`, `eblup_bhf`, `hb_area`, and `benchmark_sae`).

Intelligently handles:

- **Single Model Visualizations**: Point estimates (`"estimate"`), Relative Standard Error (`"rse"`), and 3-color official statistics reliability classification (`"reliability"`: green for reliable, amber for use with caution, red for unreliable; cut-offs from `thresholds`, see `add_reliability_flags`).
- **Direct vs Model Comparisons**: Side-by-side facet maps contrasting erratic direct survey estimates against smoothed small area model estimates (`"comparison"`).
- **Two-Model Comparisons**: Side-by-side maps or spatial difference maps (`"difference"`) contrasting two competing models (e.g. Classical FH vs Spatial FH, or FastSAE vs Stan).
- **Benchmarked Calibrations**: Visualizing pre- vs post-benchmarked calibrations.
- **Smart Key Matching**: Automatically detects the common domain identifier between the model and `sf_geom`, with optional manual override via `key`.

Usage

```
map_sae(object, ...)

## S3 method for class 'fastsae'
map_sae(
  object,
  model2 = NULL,
  sf_geom = NULL,
  key = NULL,
  type = c("estimate", "rse", "reliability", "comparison", "difference"),
  indicator = NULL,
  palette = NULL,
  thresholds = c(25, 50),
  facet_scales = "fixed",
  title = NULL,
  subtitle = NULL,
  ...
)

## S3 method for class 'fastsae_benchmark'
map_sae(
  object,
  sf_geom = NULL,
  key = NULL,
  type = c("comparison", "estimate", "difference"),
  indicator = NULL,
  palette = NULL,
  title = NULL,
  subtitle = NULL,
```

```
  ...
)

## S3 method for class 'list'
map_sae(
  object,
  sf_geom = NULL,
  key = NULL,
  type = c("comparison", "difference"),
  indicator = NULL,
  palette = NULL,
  title = NULL,
  subtitle = NULL,
  ...
)

## Default S3 method:
map_sae(
  object,
  sf_geom = NULL,
  key = NULL,
  type = c("estimate", "rse", "reliability"),
  indicator = NULL,
  palette = NULL,
  thresholds = c(25, 50),
  title = NULL,
  subtitle = NULL,
  ...
)
```

Arguments

`object` A fitted `fastsae` model object, a `fastsae_benchmark` object, a named list of `fastsae` objects, or a data frame containing domain estimates.

`...` Additional arguments passed to `ggplot2` layers.

`model2` Optional second `fastsae` model object for direct model-to-model comparison.

`sf_geom` Spatial polygon geometry of class `sf`. Optional if the model was fitted on data that already inherits from `sf`.

`key` Optional character string or named vector specifying the column(s) used to match model domain identifiers with the spatial geometry in `sf_geom`.

- If `NULL` (default), automatically detects the matching key by evaluating common column names or the highest set intersection overlap between domain IDs and `sf_geom` columns.
- If a single string (e.g. `key = "kd_kab"`), specifies the column in `sf_geom` to match with model domain IDs.
- If a named string (e.g. `key = c("domain" = "kd_kab")`), explicitly pairs the model's domain column with the spatial column in `sf_geom`.

type	Character string specifying the visualization type: • "estimate": Choropleth map of small area point estimates (HB or EBLUP). • "rse": Choropleth map of Relative Standard Errors / CV (%). • "reliability": 3-color traffic light map of the reliability tiers (by default RSE $\leq 25\%$ reliable, 25–50% use with caution, $> 50\%$ unreliable). • "comparison": Side-by-side facet choropleth map (Direct vs Model, or Model 1 vs Model 2). • "difference": Spatial difference map ($\hat{\theta}^{(2)} - \hat{\theta}^{(1)}$ or $\hat{\theta}^{\text{Model}} - \hat{\theta}^{\text{Direct}}$) with divergent color palette.
indicator	Optional alias for type. If supplied, overrides type.
palette	Character string specifying a color palette. Default is automatically selected based on type: "viridis" for estimates, "magma" for RSE, official traffic-light colors for reliability, and "RdBu" / "PuOr" for differences.
thresholds	Numeric vector of length 2 defining the RSE (%) thresholds for reliability classification (see add_reliability_flags). Default is c(25, 50), the BPS-Statistics Indonesia criteria.
facet_scales	Character string passed to ggplot2::facet_wrap(scales = ...). Default is "fixed".
title	Optional title for the plot. If NULL, a descriptive title is generated automatically.
subtitle	Optional subtitle for the plot.

Value

A ggplot object containing the thematic choropleth map.

Examples

```
library(fastsae)
data(mys)

# Create a synthetic spatial grid for demonstration
if (requireNamespace("sf", quietly = TRUE)) {
  grid_sf <- sf::st_make_grid(
    sf::st_polygon(list(matrix(c(0,0, 6,0, 6,7, 0,7, 0,0), ncol = 2, byrow = TRUE))),
    cellsize = c(1, 1),
    what = "polygons"
  )[seq_len(nrow(mys))]
  mys_sf <- sf::st_sf(area = mys$area, geometry = grid_sf)

  # 1. Fit Fay-Herriot model
  fit_fh <- eblup_fh(y ~ x1 + x2, vardir = "vardir", data = mys)

  # 2. Single model point estimate map (auto-detected key)
  p1 <- map_sae(fit_fh, sf_geom = mys_sf)

  # 3. Explicit key matching
  p2 <- map_sae(fit_fh, sf_geom = mys_sf, key = "area")
```

```
  # 4. Reliability classification map (BPS criteria: RSE <= 25%, 25-50%, > 50%)
  p3 <- map_sae(fit_fh, sf_geom = mys_sf, type = "reliability")

  # 5. Side-by-side comparison map: Direct Survey vs Model SAE
  p4 <- map_sae(fit_fh, sf_geom = mys_sf, type = "comparison")
}
```

`mys` *mys: mean years of schooling people with disabilities.*

Description

A synthetic dataset containing the mean years of schooling for people with disabilities across 42 regencies/municipalities.

Usage

```
mys
```

Format

A data frame with 42 rows and 9 variables with 10 domains as non-sampled areas.

area regency municipality identifier

y mean years of schooling people with disabilities (NA for unsampled areas)

vardir variance sampling from the direct estimator for each area

rse relative standard error (%)

x1 Number of Elementary Schools

x2 Number of Junior High Schools

x3 Number of Senior High Schools

n Number of eligible samples

weight Weight

Source

Simulated based on empirical characteristics of the National Socio-Economic Survey (Susenas), BPS-Statistics Indonesia.

`mys_panel` *mys_panel: mean years of schooling people with disabilities 2016 - 2026.*

Description

A synthetic panel dataset containing the mean years of schooling for people with disabilities across 42 regencies/municipalities over 11 time periods (2016 - 2026).

Usage

```
mys_panel
```

Format

A data frame with 462 rows and 10 variables.

area regency municipality identifier

year year of observation (2016 to 2026)

y mean years of schooling people with disabilities (NA for unsampled areas)

vardir variance sampling from the direct estimator for each area

rse relative standard error (%)

x1 Number of Elementary Schools

x2 Number of Junior High Schools

x3 Number of Senior High Schools

n Number of eligible samples

weight Weight

Source

Simulated panel data based on empirical characteristics of the National Socio-Economic Survey (Susenas), BPS-Statistics Indonesia.

mys_proxmat — *Example proximity matrix*

Description

A 42 by 42 row-standardized spatial proximity matrix for the 42 areas in `mys`. Every pair of areas has a positive weight, larger for areas whose centroids (`sim_area$x_coord`, `sim_area$y_coord`) are close, so it suits the SAR models (`eblup_sfh`, `eblup_stfh`, `hb_area(spatial = "slm")`). The CAR models of `hb_area` use only the pattern `W > 0`, which for this matrix is a complete graph; build a neighbourhood graph for them, for example `create_weights(cbind(sim_area$x_coord, sim_area$y_coord), method = "knn", k = 4)`.

Usage

```
mys_proxmat
```

Format

A square numeric matrix with 42 rows and 42 columns with values between 0 and 1.

Source

Simulated proximity weights between the 42 areas.

plot.fastsae — *Plot method for fastsae objects*

Description

Provides standard R `plot()` dispatch for `fastsae` models. Automatically routes to `map_sae` if spatial geometry is supplied, to two-model comparison if a second `fastsae` model is provided, or to `autoplot` for diagnostic and estimate plots.

Usage

```
## S3 method for class 'fastsae'
plot(x, y = NULL, ...)
```

Arguments

`x`	An object of class `fastsae`.
`y`	Optional second object (e.g. another `fastsae` model for comparison).
`...`	Additional arguments passed to `map_sae` or `autoplot`.

Value

A `ggplot` object.

print.fastsae *Print a fastsae object*

Description

Print a fastsae object

Usage

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

Arguments

x	An object of class fastsae.
...	Additional arguments passed to print methods.

Value

The original object invisibly.

print.summary.fastsae *Print summary of a fastsae object*

Description

Print summary of a fastsae object

Usage

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

Arguments

x	An object of class summary.fastsae.
...	Additional arguments.

Value

The original object invisibly.

residuals.fastsae — *Extract residuals from a fastsae object*

Description

Extract residuals from a fastsae object

Usage

```
## S3 method for class 'fastsae'
residuals(object, ...)
```

Arguments

object	An object of class fastsae.
...	Additional arguments.

Value

Vector of residuals for sampled areas.

setup_numpyro_env — *Set up the Python environment for the NumPyro / JAX engine*

Description

Creates (or updates) a Python environment with jax and numpyro for hb_area(engine = "jax") and activates it for **reticulate**.

Usage

```
setup_numpyro_env(envname = "r-numpyro-gpu", method = "auto", device = "auto")
```

Arguments

envname	Name of the conda or virtualenv environment. Default "r-numpyro-gpu".
method	"auto" (virtualenv), "virtualenv", or "conda".
device	"auto" (CUDA if nvidia-smi is found, otherwise CPU), "metal" (Apple Silicon), "cuda" (NVIDIA), or "cpu". Versions are pinned to tested combinations: jax==0.5.0 and numpyro==0.19.0 for CPU/CUDA; jax==0.4.34, jaxlib==0.4.34, jax-metal==0.1.1 for Metal.

Value

Invisible logical TRUE upon successful setup.

See Also

check_numpyro_available, hb_area

Examples

```
## Not run:
setup_numpyro_env(device = "auto")

## End(Not run)
```

---

sim_area — *Multi-Distribution Synthetic Area-Level Dataset*

Description

A synthetic dataset containing 42 domains with multiple response variables representing various probability distributions (Gaussian, Poisson, Binomial, Beta, Negative Binomial, and Gamma). Designed for testing and benchmarking generalized linear and spatial small area estimation models (HB / INLA). It shares the 42-domain spatial structure of mys_proxmat.

Usage

```
data(sim_area)
```

Format

A data frame with 42 rows and 14 variables:

area Integer domain identifier (1 to 42).

x1 Continuous explanatory covariate.

x2 Uniform explanatory covariate.

y_gaussian Gaussian direct estimator response with unsampled domains as NA.

vardir Direct sampling variance for Gaussian Fay-Herriot model.

y_poisson Count response (Poisson) with unsampled domains as NA.

exposure Expected population count / exposure offset for Poisson model.

y_binomial Number of successes (Binomial) with unsampled domains as NA.

trials Sample size / number of trials for Binomial model.

y_beta Continuous proportion response in (0, 1) for Beta regression.

y_nbinomial Overdispersed count response for Negative Binomial model.

y_gamma Skewed positive continuous response for Gamma regression.

x_coord Centroid x coordinate.

y_coord Centroid y coordinate.

Source

Simulated using sim_area_data based on the spatial proximity structure of mys_proxmat.

Examples

```
data(sim_area)
head(sim_area)
```

---

sim_area_data *Simulate Multi-Distribution Small Area Data with Spatial Autocorrelation*

Description

Generates synthetic area-level datasets for small area estimation, including continuous (Gaussian, Gamma), count (Poisson, Negative Binomial), and proportion/rate responses (Binomial, Beta), driven by shared covariates and structured spatial random effects (BYM2 / SAR).

Usage

```
sim_area_data(
  D = 42,
  W = NULL,
  spatial_type = c("knn", "grid", "ring"),
  rho = 0.5,
  phi = 0.6,
  sigma_u = 0.5,
  beta = c(1, 0.8, -0.5),
  n_unsampled = 6,
  seed = NULL
)
```

Arguments

D	Integer. Number of domains (areas). Default is 42.
W	Optional spatial proximity or adjacency matrix of dimension $D \times D$. If NULL, a spatial matrix is automatically simulated using sim_spatial_weights.
spatial_type	Character. Topology for simulated W if W = NULL: "knn", "grid", or "ring". Default is "knn".
rho	Numeric. Spatial autocorrelation autoregressive parameter ($\|\rho\| < 1$). Default is 0.5.
phi	Numeric. Proportion of spatial marginal variance relative to total area variance (BYM2 parameter, $0 \leq \phi \leq 1$). Default is 0.6.
sigma_u	Numeric. Overall standard deviation of domain random effects. Default is 0.5.
beta	Numeric vector of length 3 giving true regression coefficients $(\beta_0, \beta_1, \beta_2)$. Default is c(1.0, 0.8, -0.5).

n_unsampled	Integer. Number of domains to mark as unsampled (where response variables are set to NA). Default is 6.
seed	Optional integer. Random seed for reproducibility.

Value

An object of class c("fastsae_sim_data", "list") containing:

data A data frame with columns: domain, x1, x2, y_gaussian, vardir, y_poisson, exposure, y_binomial, trials, y_beta, y_nbinomial, y_gamma, x_coord, y_coord.

W Binary adjacency matrix $D \times D$.

W_std Row-standardized proximity matrix $D \times D$.

coords Coordinates of domain centroids.

u_spatial True structured spatial random effect vector.

u_iid True unstructured random effect vector.

u_total True total random effect vector.

phi True spatial variance fraction.

rho True spatial autoregressive parameter.

Examples

```
# 1. Simulate dataset with default KNN spatial structure
sim <- sim_area_data(D = 40, spatial_type = "knn", seed = 123)
print(sim)
head(sim$data)

# 2. Using an existing spatial matrix
my_W <- sim_spatial_weights(D = 30, type = "grid")
sim2 <- sim_area_data(W = my_W, seed = 456)
```

---

sim_panel *Multi-Distribution Spatio-Temporal Panel Dataset*

Description

A synthetic panel dataset containing 42 domains observed over 5 time periods (2022 to 2026, total 210 observations). Features multiple response variables representing various probability distributions (Gaussian, Poisson, Binomial, Beta, Negative Binomial, and Gamma) driven by domain-level spatial autocorrelation and first-order autoregressive AR(1) temporal dynamics. It shares the 42-domain spatial structure of mys_proxmat and is formatted in domain-major order for direct use in eblup_stfh.

Usage

```
data(sim_panel)
```

Format

A data frame with 210 rows and 15 variables in domain-major order:

area Integer domain identifier (1 to 42).

year Year of observation (2022 to 2026).

x1 Dynamic explanatory covariate with spatial & temporal variation.

x2 Dynamic uniform explanatory covariate.

y_gaussian Gaussian direct estimator response with unsampled domains as NA.

vardir Direct sampling variance for Gaussian Fay-Herriot model.

y_poisson Count response (Poisson) with unsampled domains as NA.

exposure Expected population count / exposure offset for Poisson model.

y_binomial Number of successes (Binomial) with unsampled domains as NA.

trials Sample size / number of trials for Binomial model.

y_beta Continuous proportion response in (0, 1) for Beta regression.

y_nbinomial Overdispersed count response for Negative Binomial model.

y_gamma Skewed positive continuous response for Gamma regression.

x_coord Centroid x coordinate.

y_coord Centroid y coordinate.

Source

Simulated using `sim_series_data` based on the spatial proximity structure of `mys_proxmat`.

Examples

```
data(sim_panel)
head(sim_panel)
```

---

sim_series_data *Simulate Spatio-Temporal Multi-Distribution Small Area Data*

Description

Generates synthetic spatio-temporal panel datasets for small area estimation. Combines domain-level spatial autocorrelation (SAR) with domain-specific first-order autoregressive AR(1) temporal dynamics across T time periods, producing multi-distribution responses (Gaussian Fay-Herriot, Poisson, Binomial, Beta, Negative Binomial, Gamma).

Usage

```
sim_series_data(
  D = 42,
  T = 5,
  time_start = 2022,
  W = NULL,
  spatial_type = c("knn", "grid", "ring"),
  rho_s = 0.5,
  rho_t = 0.6,
  sigma_s = 0.4,
  sigma_t = 0.3,
  trend = c("linear", "random_walk", "ar1", "none"),
  trend_slope = 0.05,
  beta = c(1, 0.8, -0.5),
  n_unsampled = 4,
  prop_intermittent = 0.05,
  sort_order = c("domain-major", "time-major"),
  seed = NULL
)
```

Arguments

`D`	Integer. Number of domains (areas). Default is 42.
`T`	Integer. Number of time periods ($T \geq 2$). Default is 5.
`time_start`	Integer. Starting year / time label. Default is 2022.
`W`	Optional spatial proximity or adjacency matrix of dimension $D \times D$. If `NULL`, a spatial matrix is automatically simulated using `sim_spatial_weights`.
`spatial_type`	Character. Topology for simulated `W` if `W = NULL`: `"knn"`, `"grid"`, or `"ring"`. Default is `"knn"`.
`rho_s`	Numeric. Domain spatial autoregressive autocorrelation parameter ($\lvert\rho_s\rvert < 1$, corresponding to `rho1` in `eblup_stfh`). Default is 0.5.
`rho_t`	Numeric. Temporal AR(1) autocorrelation parameter ($\lvert\rho_t\rvert < 1$, corresponding to `rho2` in `eblup_stfh`). Default is 0.6.
`sigma_s`	Numeric. Standard deviation of domain spatial random effects (u_1). Default is 0.4.
`sigma_t`	Numeric. Standard deviation of stationary spatio-temporal effects (u_2). Default is 0.3.
`trend`	Character. Deterministic temporal trend type: `"linear"`, `"random_walk"`, `"ar1"`, or `"none"`. Default is `"linear"`.
`trend_slope`	Numeric. Slope / scale of the temporal trend. Default is 0.05.
`beta`	Numeric vector of length 3 giving fixed regression coefficients ($\beta_0, \beta_1, \beta_2$). Default is `c(1.0, 0.8, -0.5)`.
`n_unsampled`	Integer. Number of domains that are completely unsampled across all time periods. Default is 4.

`prop_intermittent`
: Numeric. Proportion of domain-by-time observations to mark as intermittently missing (excluding persistent unsampled domains). Default is 0.05.

`sort_order`
: Character. Row sorting order: `"domain-major"` (all time periods for domain 1, then domain 2, required by `eblup_stfh`) or `"time-major"`. Default is `"domain-major"`.

`seed`
: Optional integer. Random seed for reproducibility.

Value

An object of class `c("fastsae_sim_series", "list")` containing:

data A data frame of dimension $(D \cdot T) \times 15$ with columns: `area`, `year`, `x1`, `x2`, `y_gaussian`, `vardir`, `y_poisson`, `exposure`, `y_binomial`, `trials`, `y_beta`, `y_nbinomial`, `y_gamma`, `x_coord`, `y_coord`.

W Binary adjacency matrix $D \times D$.

W_std Row-standardized spatial weights matrix $D \times D$.

coords Centroid coordinates of the D domains.

u_spatial True domain spatial random effect vector of length D.

u_temporal True spatio-temporal random effect matrix of dimension $D \times T$.

parameters List of true simulation parameters (`rho_s`, `rho_t`, `sigma_s`, `sigma_t`, `beta`, `trend`).

Examples

```
# 1. Simulate a 30-domain panel over 4 years
sim_panel <- sim_series_data(D = 30, T = 4, time_start = 2021, seed = 123)
print(sim_panel)
head(sim_panel$data)

# 2. Directly estimate Spatio-Temporal Fay-Herriot model (eblup_stfh)
## Not run:
fit_stfh <- eblup_stfh(
  y_gaussian ~ x1 + x2,
  data = sim_panel$data,
  domain = ~area,
  time = ~year,
  vardir = ~vardir,
  W = sim_panel$W_std
)
summary(fit_stfh)

## End(Not run)
```

sim_spatial_weights *Simulate Spatial Proximity and Adjacency Matrices*

Description

Generates synthetic spatial proximity or adjacency matrices for small area estimation models. Supports k-nearest neighbors (KNN) on random 2D coordinates, regular grid contiguity (lattice), and 1D circular rings.

Usage

```
sim_spatial_weights(
  D = 40,
  type = c("knn", "grid", "ring"),
  style = c("B", "W"),
  k = 4,
  coords = NULL,
  seed = NULL
)
```

Arguments

`D`	Integer. Number of small areas (domains). Default is 40.
`type`	Character. Topology type: `"knn"` (k-nearest neighbors on 2D coordinates), `"grid"` (Rook contiguity on a regular 2D lattice), or `"ring"` (1D circular chain). Default is `"knn"`.
`style`	Character. Weight style: `"B"` for symmetric binary adjacency (0/1) with zero diagonal (standard for INLA graph models such as BYM2, BYM, Besag), or `"W"` for row-standardized weights (sum of each row equals 1, standard for SAR/SLM models). Default is `"B"`.
`k`	Integer. Number of nearest neighbors per domain when `type = "knn"`. Default is 4. Must be between 1 and `D - 1`.
`coords`	Optional numeric matrix of dimension `c(D, 2)` containing 2D spatial coordinates. If provided and `type = "knn"`, distances are computed from these coordinates.
`seed`	Optional integer. Random seed for reproducible coordinate generation.

Details

When `type = "knn"`, coordinates are sampled from Uniform$(0,1)^2$ if not provided. Each domain is connected to its k nearest neighbors. To ensure a symmetric adjacency graph required by GMRF spatial priors (INLA), edges are made mutual: domain i and j are connected if j is among the k nearest neighbors of i or vice versa.

When `type = "grid"`, domains are placed on a regular $r \times c$ lattice where $r = \lfloor \sqrt{D} \rfloor$ and $c = \lceil D/r \rceil$. If $r \times c > D$, the first D lattice points are retained. Rook contiguity (sharing a common edge) is used.

When `type = "ring"`, domain i is connected to $i-1$ and $i+1$ in a closed ring.

Value

A square numeric matrix of dimension $D \times D$ with row and column names set to domain identifiers (`"1"`, `"2"`, ..., `"D"`). Attributes include `"coords"` (2D coordinates) and `"type"`.

Examples

```
# 1. Binary adjacency matrix for 20 domains using KNN
W_bin <- sim_spatial_weights(D = 20, type = "knn", k = 3, seed = 123)
dim(W_bin)
table(W_bin)

# 2. Row-standardized matrix on a regular grid
W_std <- sim_spatial_weights(D = 25, type = "grid", style = "W")
rowSums(W_std)
```

`spatial_test` *Pre-Modeling Spatial Diagnostic and Autocorrelation Tests*

Description

Evaluates whether spatial autocorrelation is present in direct domain estimates and regression residuals, conducts Lagrange Multiplier (LM) tests (Anselin, 1988), and performs comparative model fit tests between the standard Fay-Herriot (`eblup_fh`) and Spatial Fay-Herriot (`eblup_sfh`) models. Provides an explicit, automated recommendation on whether a spatial model is statistically necessary.

Usage

```
spatial_test(
  formula,
  vardir = NULL,
  data,
  W = NULL,
  alpha = 0.05,
  alternative = c("greater", "two.sided", "less")
)
```

Arguments

`formula`	Regression formula describing the model (e.g. `y ~ x1 + x2`).
`vardir`	Optional vector, column name, or formula referencing sampling variances of the direct estimator. Required for full model comparison (Standard FH vs Spatial FH).
`data`	A `data.frame` or `sf` polygon object containing the variables in `formula`.

W	Proximity or spatial adjacency matrix. If NULL and data is an sf polygon object, it is automatically constructed via create_weights.
alpha	Significance level for hypothesis tests (default is 0.05).
alternative	Alternative hypothesis for Moran's I ("greater", "two.sided", "less").

Details

Moran's I uses the randomisation variance of Cliff and Ord (1981); the LM-error and LM-lag statistics follow Anselin (1988). These tests assume homoscedastic OLS errors, whereas direct estimates have domain-specific sampling variances, so they are descriptive. The recommendation therefore rests on one test, so that its false positive rate is alpha: with vardir, the likelihood ratio test of $\rho = 0$ between eblup_fh and eblup_sfh fitted by ML (which accounts for the sampling variances); without it, the LM-error test.

Value

An object of class c("fastsae_spatial_test", "list") containing:

response_test Global Moran's I test results on direct response variable y.

residual_test Moran's I, Anselin LM-Error, and Anselin LM-Lag test results on regression residuals.

model_comparison ML log-likelihoods, AIC, BIC, and the likelihood ratio test (LRT) of $\rho = 0$ comparing standard vs spatial FH, both fitted by ML (if vardir provided).

spatial_recommended Logical recommendation, from a single test at level alpha (see Details).

decision_test The test behind spatial_recommended: "lrt" or "lm_error".

reason Character string explaining the recommendation.

References

1. Anselin, L. (1988). *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers.
2. Cliff, A. D., & Ord, J. K. (1981). *Spatial Processes: Models & Applications*. Pion London.
3. Petrucci, A., & Salvati, N. (2006). Small area estimation for spatial data in R. *Statistical Methods & Applications*, 14(3), 391-407.

Examples

```
library(fastsae)
data(mys)
data(mys_proxmat)

# Run spatial diagnostics on simulated survey data
st <- spatial_test(
  formula = y ~ x1 + x2,
  vardir = "vardir",
  data = mys,
  W = mys_proxmat
)
```

```
print(st)
```

---

summary.fastsae *Summarize a fastsae object*

Description

Summarize a fastsae object

Usage

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

Arguments

object	An object of class fastsae.
...	Additional arguments.

Value

An object of class summary.fastsae.

Index

∗ **datasets**
cornsoybean, 13
cornsoybeanmeans, 13
mys, 58
mys_panel, 59
mys_proxmat, 60
sim_area, 63
sim_panel, 65

add_reliability_flags, 3, *12*, *34*, *55*, *57*
autoplot, *4*, *60*
autoplot.fastsae, 4
autoplot.fastsae_benchmark (benchmark_sae), 6
autoplot.fastsae_comparison, *12*
autoplot.fastsae_comparison (compare_sae), 11
autoplot.fastsae_diagnose, 5
autoplot.list (autoplot.fastsae), 4

benchmark_sae, 6, *41*

check_numpyro_available, 9, *46*, *63*
coef.fastsae, 10
compare_sae, 11, *41*
cornsoybean, 13, *13*
cornsoybeanmeans, 13
create_weights, 14, *71*

diagnose, *5*, *6*, 16

eblup_bhf, 18
eblup_fh, *16*, 20, *70*, *71*
eblup_sfh, *16*, 23, *39*, *60*, *70*, *71*
eblup_stfh, *16*, 26, *60*, *65*, *67*, *68*
eblup_twofold, 31
export_sae, 33

fitted.fastsae, 35

gvf_smooth, 35

hb_area, *10*, *16*, 37, *44*, *47*, *48*, *60*, *63*
hb_engine, 44
hb_twofold, *44*, 47, *47*
hb_unit, *44*, *47*, 51

inla_engine, *37*, *40*, *41*, *48*, *49*, *52*
inla_engine (hb_engine), 44

jax_engine, *37*, *40*, *41*, *48*
jax_engine (hb_engine), 44

map_sae, *12*, 54, *60*
mys, 58
mys_panel, 59
mys_proxmat, 60, *63–66*

plot.fastsae, 60
plot.fastsae_benchmark (benchmark_sae), 6
plot.fastsae_comparison (compare_sae), 11
print.fastsae, 61
print.fastsae_benchmark (benchmark_sae), 6
print.fastsae_comparison (compare_sae), 11
print.fastsae_engine (hb_engine), 44
print.summary.fastsae, 61

residuals.fastsae, 62

sae_area_multi (sim_area), 63
sae_panel_multi (sim_panel), 65
setup_numpyro_env, *10*, *37*, *46*, 62
sim_area, 63
sim_area_data, 64, *64*
sim_panel, 65
sim_series_data, 66, *66*
sim_spatial_weights, *64*, *67*, 69
spatial_test, 70
summary.fastsae, 72

summary.fastsae_benchmark
(benchmark_sae), 6
summary.fastsae_comparison
(compare_sae), 11