Package 'spatialAtomizeR'

March 8, 2026

Type Package

Title Spatial Analysis with Misaligned Data Using Atom-Based
Regression Models

Version 0.2.7

Date 2026-03-08

Description Implements atom-based regression models (ABRM) for analyzing spatially
misaligned data. Provides functions for simulating misaligned spatial data,
preparing NIMBLE model inputs, running MCMC diagnostics, and providing re-
sults. All main functions return
S3 objects with print(), summary(), and plot() methods for intuitive result exploration.
Methods originally described in Mugglin et al. (2000) <doi:10.1080/01621459.2000.10474279>,
further investigated in Trevisani & Gelfand (2013), and applied in Nethery et al. (2023)
<doi:10.1101/2023.01.10.23284410>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports nimble, sp, sf, spdep, MASS, dplyr, tidyr, ggplot2, reshape2,
coda, BiasedUrn, stats, utils, grDevices, methods, raster

Suggests testthat (>= 3.0.0), knitr, rmarkdown, tigris

VignetteBuilder knitr

URL https://github.com/bellayqian/spatialAtomizeR

BugReports https://github.com/bellayqian/spatialAtomizeR/issues

RoxygenNote 7.3.3

NeedsCompilation no

Author Yunzhe Qian [aut, cre],
Rachel Nethery [aut],
Nancy Krieger [ctb] (Contributed to the project conceptualization and
manuscript),
Nykesha Johnson [ctb] (Contributed to the project conceptualization and
manuscript)

Contents

---

coef.abrm   *Extract Coefficients from ABRM Objects*

Description

Returns the posterior mean estimates for all regression coefficients as a named numeric vector.

Usage

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

Arguments

object	An object of class "abrm" returned by run_abrm.
...	Additional arguments (currently unused).

Value

A named numeric vector of posterior mean estimates. Names correspond to parameter labels (e.g., "beta_x[1]", "beta_y[1]", "beta_0_y").

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "normal", dist_covariates_y = "normal",
  dist_y = "normal", x_intercepts = 0, y_intercepts = 0,
  beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(), norm_idx_x = 1, norm_idx_y = 1,
  dist_y = 1, niter = 1000, nburnin = 500, nchains = 2, seed = 1,
  save_plots = FALSE
)
coef(results)
```

---

confint.abrm   *Credible Intervals for ABRM Objects*

Description

Returns the 95% posterior credible intervals for all estimated parameters. Note: these are Bayesian credible intervals from MCMC quantiles, not frequentist confidence intervals.

Usage

```
## S3 method for class 'abrm'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

object	An object of class "abrm" returned by run_abrm.
parm	Optional character vector of parameter names to subset. If NULL (default), all parameters are returned.
level	Confidence level for the credible interval (default 0.95). When level = 0.95 (the default), the pre-computed 95% intervals stored during model fitting are returned directly. For other levels, intervals are recomputed from the raw MCMC posterior samples.
...	Additional arguments (currently unused).

Value

A matrix with two columns, CI.Lower and CI.Upper, and one row per parameter.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "normal", dist_covariates_y = "normal",
  dist_y = "normal", x_intercepts = 0, y_intercepts = 0,
  beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(), norm_idx_x = 1, norm_idx_y = 1,
  dist_y = 1, niter = 1000, nburnin = 500, nchains = 2, seed = 1,
  save_plots = FALSE
)
confint(results)
confint(results, parm = "beta_x[1]")
```

---

fitted.abrm   *Fitted Values for ABRM Objects*

Description

Computes posterior-mean fitted values at the Y-grid level by applying the estimated regression coefficients to the supplied covariate data.

Usage

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

Arguments

object	An object of class "abrm" returned by [run_abrm](#).
...	Additional arguments (currently unused).

Value

A data frame with one row per Y-grid cell and four columns:

y_grid_id The original Y-grid cell ID.

observed The observed outcome value y for each Y-grid cell. Note: the outcome is observed at the Y-grid level for all cells. This is distinct from covariates, which may be *latent* (unobserved at the Y-grid level) for cells that span multiple atoms.

fitted The model-predicted outcome on the original outcome scale (counts for Poisson/binomial; continuous for normal).

residual Observed minus fitted.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "normal", dist_covariates_y = "normal",
  dist_y = "normal", x_intercepts = 0, y_intercepts = 0,
  beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(), norm_idx_x = 1, norm_idx_y = 1,
  dist_y = 1, niter = 1000, nburnin = 500, nchains = 2, seed = 1,
  save_plots = FALSE
)
fitted(results)
```

---

get_abrm_model   *Get ABRM Model Code for NIMBLE*

Description

Returns the NIMBLE code for the Atom-Based Regression Model with mixed-type variables. Automatically registers custom distributions if not already registered.

Usage

```
get_abrm_model()
```

Value

A `nimbleCode` object containing the NIMBLE model specification for the ABRM. Pass this directly to run_abrm.

Examples

```
model_code <- get_abrm_model()
print(model_code)
```

---

plot.abrm   *Plot Method for ABRM Objects*

Description

Displays MCMC diagnostic plots — trace plots and posterior density plots — for the parameters monitored during model fitting. If no diagnostic plots are stored in the object, a message is issued instead.

Usage

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

Arguments

x   An object of class "abrm" returned by run_abrm.

...   Additional arguments (currently unused).

Value

Invisibly returns x. Called for its side effect of rendering the MCMC diagnostic plots.

Examples

```
## Step 1: Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed            = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y          = "poisson",
  x_intercepts    = c(0, -1),
  y_intercepts    = c(0, -1),
  beta0_y         = -1,
  beta_x          = c(0.1, -0.05),
  beta_y          = c(-0.1, 0.05)
)

## Step 2: Retrieve the NIMBLE model code
model_code <- get_abrm_model()

## Step 3: Fit the ABRM
results <- run_abrm(
  gridx       = sim_data$gridx,
  gridy       = sim_data$gridy,
  atoms       = sim_data$atoms,
  model_code  = model_code,
  true_params = sim_data$true_params,
  norm_idx_x  = 1,
```

```
    pois_idx_x  = 2,
    norm_idx_y  = 1,
    pois_idx_y  = 2,
    dist_y      = 2,
    niter       = 1000,
    nburnin     = 500,
    nchains     = 2,
    seed        = 1,
    save_plots  = FALSE
)

## Step 4: Display MCMC trace and posterior density plots
plot(results)
```

---

plot.misaligned_data   *Plot Misaligned Data Object*

Description

Visualizes the spatial layout of a `misaligned_data` object. By default, both grids are overlaid to illustrate the misalignment between them.

Usage

```
## S3 method for class 'misaligned_data'
plot(
  x,
  which = c("both", "gridy", "gridx", "atoms"),
  color_y = "blue",
  color_x = "orange",
  title = NULL,
  ...
)
```

Arguments

x	A `misaligned_data` object.
which	Character string specifying what to plot. One of `"both"` (default), `"gridy"`, `"gridx"`, or `"atoms"`.
color_y	Color for the Y-grid boundaries. Default `"blue"`.
color_x	Color for the X-grid boundaries. Default `"orange"`.
title	Optional character string for the plot title. If `NULL`, a default title is used.
...	Additional arguments passed to `ggplot2::geom_sf()` when `which` is not `"both"`.

Value

The input x, invisibly.

Examples

```
## Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed            = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y          = "poisson",
  x_intercepts    = c(0, -1),
  y_intercepts    = c(0, -1),
  beta0_y         = -1,
  beta_x          = c(0.1, -0.05),
  beta_y          = c(-0.1, 0.05)
)
# Default: overlay both grids to visualise misalignment
plot(sim_data)

# Plot a single component
plot(sim_data, which = "gridy")
plot(sim_data, which = "gridx")
plot(sim_data, which = "atoms")

# Custom colours
plot(sim_data, color_y = "steelblue", color_x = "firebrick")

# Custom title
plot(sim_data, title = "Utah Spatial Misalignment")
```

---

  print.abrm   *Print Method for ABRM Objects*

Description

Prints a concise summary of a fitted ABRM, including the number of parameters estimated and, when true parameter values are available, the mean absolute bias and 95% credible-interval coverage rate.

Usage

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

Arguments

x	An abrm object
...	Additional arguments (unused)

Value

Invisibly returns x. Called for its side effect of printing a concise ABRM model summary including number of parameters, mean absolute bias, and credible-interval coverage rate (when true parameter values are available).

Examples

```
## Step 1: Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed            = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y          = "poisson",
  x_intercepts    = c(0, -1),
  y_intercepts    = c(0, -1),
  beta0_y         = -1,
  beta_x          = c(0.1, -0.05),
  beta_y          = c(-0.1, 0.05)
)

## Step 2: Retrieve the NIMBLE model code
model_code <- get_abrm_model()

## Step 3: Fit the ABRM
## (niter and nburnin are intentionally small for illustration only;
##  use larger values, e.g. niter = 50000, nburnin = 30000, in practice)
results <- run_abrm(
  gridx       = sim_data$gridx,
  gridy       = sim_data$gridy,
  atoms       = sim_data$atoms,
  model_code  = model_code,
  true_params = sim_data$true_params,
  norm_idx_x  = 1,
  pois_idx_x  = 2,
  norm_idx_y  = 1,
  pois_idx_y  = 2,
  dist_y      = 2,
  niter       = 1000,
  nburnin     = 500,
  nchains     = 2,
  seed        = 1,
  save_plots  = FALSE
)

## Step 4: Print a concise model summary
print(results)
# Reports: number of parameters, mean absolute bias, coverage rate
```

---

print.misaligned_data *Print Method for Misaligned Data Objects*

Description

Print Method for Misaligned Data Objects

Usage

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

Arguments

x   A misaligned_data object

...   Additional arguments (unused)

Value

Invisibly returns the input object x. The function is called for its side effect of printing a summary
of the simulated misaligned spatial data including grid dimensions and number of atoms.

Examples

```
## Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed             = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y           = "poisson",
  x_intercepts     = c(0, -1),
  y_intercepts     = c(0, -1),
  beta0_y          = -1,
  beta_x           = c(0.1, -0.05),
  beta_y           = c(-0.1, 0.05)
)
print(sim_data)
```

---

print.summary.abrm *Print Method for summary.abrm Objects*

Description

Prints the full parameter table from a `summary.abrm` object, including posterior means, standard
deviations, credible intervals, and (when available) bias and coverage.

Usage

```
## S3 method for class 'summary.abrm'
print(x, digits = 4, ...)
```

Arguments

x	An object of class ″summary.abrm″ returned by summary.abrm.
digits	Integer; number of significant digits. Default 4.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = ″normal″, dist_covariates_y = ″normal″,
  dist_y = ″normal″, x_intercepts = 0, y_intercepts = 0,
  beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(), norm_idx_x = 1, norm_idx_y = 1,
  dist_y = 1, niter = 1000, nburnin = 500, nchains = 2, seed = 1,
  save_plots = FALSE
)
s <- summary(results)
print(s)              # same as typing summary(results) interactively
print(s, digits = 2)  # fewer decimal places
```

---

print.vcov.abrm	*Print Method for vcov.abrm Objects*

Description

Prints all variance-covariance matrices stored in a ″vcov.abrm″ object returned by vcov.abrm.

Usage

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

Arguments

x	An object of class ″vcov.abrm″.
...	Additional arguments (currently unused).

Value

Invisibly returns x. Called for its side effect of printing the intercept variance and the X-grid and Y-grid covariance matrices.

Examples

```
## Step 1: Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed             = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y           = "poisson",
  x_intercepts     = c(0, -1),
  y_intercepts     = c(0, -1),
  beta0_y          = -1,
  beta_x           = c(0.1, -0.05),
  beta_y           = c(-0.1, 0.05)
)

## Step 2: Retrieve the NIMBLE model code
model_code <- get_abrm_model()

## Step 3: Fit the ABRM
results <- run_abrm(
  gridx       = sim_data$gridx,
  gridy       = sim_data$gridy,
  atoms       = sim_data$atoms,
  model_code  = model_code,
  true_params = sim_data$true_params,
  norm_idx_x  = 1,
  pois_idx_x  = 2,
  norm_idx_y  = 1,
  pois_idx_y  = 2,
  dist_y      = 2,
  niter       = 1000,
  nburnin     = 500,
  nchains     = 2,
  seed        = 1,
  save_plots  = FALSE
)

## Step 4: Extract and print the variance-covariance matrices
vcov_mats <- vcov(results)
print(vcov_mats)
# Prints: intercept variance, X-grid and Y-grid covariance matrices
```

---

print.waic_abrm   *Print Method for waic_abrm Objects*

Description

Displays a formatted WAIC summary for a fitted ABRM.

Usage

```
## S3 method for class 'waic_abrm'
print(x, digits = 3, ...)
```

Arguments

x	An object of class "waic_abrm" returned by [waic.abrm](#).
digits	Integer; number of decimal places. Default 3.
...	Additional arguments (currently unused).

Value

Invisibly returns x.

---

run_abrm   *Run ABRM Analysis*

Description

Fits an Atom-Based Regression Model (ABRM) to spatially misaligned data using Bayesian MCMC via NIMBLE. Works with both simulated and real-world data.

Usage

```
run_abrm(
  gridx,
  gridy,
  atoms,
  model_code,
  true_params = NULL,
  norm_idx_x = NULL,
  pois_idx_x = NULL,
  binom_idx_x = NULL,
  norm_idx_y = NULL,
  pois_idx_y = NULL,
  binom_idx_y = NULL,
  dist_y = 2,
  niter = 50000,
  nburnin = 30000,
  nchains = 2,
  thin = 10,
  seed = NULL,
```

```
    sim_metadata = NULL,
    save_plots = FALSE,
    output_dir = NULL,
    compute_waic = FALSE
)
```

Arguments

`gridx`	The X-grid sf dataframe, containing a numeric area ID variable named 'ID' and covariates named 'covariate_x_1','covariate_x_2',...
`gridy`	The Y-grid sf dataframe, containing a numeric area ID variable named 'ID', covariates named 'covariate_y_1','covariate_y_2',...., and an outcome named 'y'.
`atoms`	The atom sf dataframe, which should contain numeric variables named 'ID_x' and 'ID_y' holding the X-grid and Y-grid cell IDs for each atom, as well as an atom-level population count named 'population'.
`model_code`	NIMBLE model code from get_abrm_model()
`true_params`	The true outcome model regression coefficient parameters, if known (e.g., from simulate_misaligned_data())
`norm_idx_x`	Vector of numeric indices of X-grid covariates (ordered as 'covariate_x_1','covariate_x_2',...) that should be treated as normally-distributed
`pois_idx_x`	Vector of numeric indices of X-grid covariates (ordered as 'covariate_x_1','covariate_x_2',...) that should be treated as Poisson-distributed
`binom_idx_x`	Vector of numeric indices of X-grid covariates (ordered as 'covariate_x_1','covariate_x_2',...) that should be treated as binomial-distributed
`norm_idx_y`	Vector of numeric indices of Y-grid covariates (ordered as 'covariate_y_1','covariate_y_2',...) that should be treated as normally-distributed
`pois_idx_y`	Vector of numeric indices of Y-grid covariates (ordered as 'covariate_y_1','covariate_y_2',...) that should be treated as Poisson-distributed
`binom_idx_y`	Vector of numeric indices of Y-grid covariates (ordered as 'covariate_y_1','covariate_y_2',...) that should be treated as binomial-distributed
`dist_y`	Distribution type for outcome (1=normal, 2=poisson, 3=binomial)
`niter`	Number of MCMC iterations (default: 50000)
`nburnin`	Number of burn-in iterations (default: 30000)
`nchains`	Number of MCMC chains (default: 2)
`thin`	Thinning interval (default: 10)
`seed`	Integer seed for reproducibility. Each chain uses seed+(chain_number-1) (default: NULL)
`sim_metadata`	Optional named list of simulation metadata (e.g., sim_number, x_correlation, y_correlation) passed through to run_nimble_model and used to label diagnostic plot file names when save_plots = TRUE. For non-simulation use, leave as NULL.
`save_plots`	Logical, whether to save diagnostic plots (default: FALSE)

| output_dir | Directory for saving outputs (default: NULL) |
| compute_waic | Logical; if TRUE, NIMBLE computes the Widely Applicable Information Criterion (WAIC) during MCMC sampling and stores it in the returned object. Retrieve it afterwards with [waic](). Default FALSE. |

Value

An object of class "abrm": a named list with components `mcmc_results`, `parameter_estimates`, `all_parameters`, `fitted_values` (numeric vector of Y-grid-level predicted outcome values on the original outcome scale), `y_grid_ids` (integer vector of Y-grid cell IDs in model order), and `y_observed` (numeric vector of observed outcome values for directly observed Y-grid cells). Use `fitted()` to extract a formatted comparison table of observed vs. fitted values, and `waic()` to retrieve model fit criteria when `compute_waic = TRUE`.

Examples

```
# Simulate misaligned spatial data with one normal covariate per grid
sim_data <- simulate_misaligned_data(
  seed = 1,
  dist_covariates_x = "normal",
  dist_covariates_y = "normal",
  dist_y = "normal",
  x_intercepts = 0,
  y_intercepts = 0,
  beta0_y = 0,
  beta_x = 0.1,
  beta_y = -0.1
)

# Retrieve the pre-compiled NIMBLE model code
model_code <- get_abrm_model()

# Fit the ABRM (use small niter/nburnin for illustration only)
results <- run_abrm(
  gridx      = sim_data$gridx,
  gridy      = sim_data$gridy,
  atoms      = sim_data$atoms,
  model_code = model_code,
  true_params = sim_data$true_params,
  norm_idx_x = 1,
  norm_idx_y = 1,
  dist_y     = 1,
  niter      = 1000,
  nburnin    = 500,
  nchains    = 2,
  seed       = 1,
  save_plots = FALSE
)

print(results)    # concise model summary
summary(results)  # full parameter table
plot(results)     # MCMC trace and density plots
```

```
    vcov(results)      # posterior variance-covariance matrices
```

---

run_nimble_model   *Run NIMBLE Model with Diagnostics*

Description

Run NIMBLE Model with Diagnostics

Usage

```
run_nimble_model(
  constants,
  data,
  inits,
  sim_metadata = NULL,
  model_code,
  niter = 50000,
  nburnin = 30000,
  nchains = 2,
  thin = 10,
  seed = NULL,
  save_plots = FALSE,
  output_dir = NULL,
  compute_waic = FALSE
)
```

Arguments

constants	List of model constants
data	List of data
inits	List of initial values
sim_metadata	List with simulation metadata (optional)
model_code	NIMBLE code object
niter	Number of MCMC iterations (default: 50000)
nburnin	Number of burn-in iterations (default: 30000)
nchains	Number of MCMC chains (default: 2)
thin	Thinning interval (default: 10)
seed	Integer seed for reproducibility. Each chain uses seed+(chain_number-1) (default: NULL)
save_plots	Logical, whether to save diagnostic plots (default: FALSE)
output_dir	Directory for saving plots (default: NULL)
compute_waic	Logical; if TRUE, WAIC is computed by NIMBLE and stored in the result, enabling waic() for model comparison. Default FALSE to keep fitting fast.

Value

A named list with components `samples` (per-chain MCMC sample matrices), `summary` (posterior summary statistics), and `convergence` (Gelman-Rubin statistics, effective sample sizes, and optional diagnostic plots).

Examples

```
# The recommended workflow is to call run_abrm(), which calls
# run_nimble_model() internally after preparing all inputs.
sim_data <- simulate_misaligned_data(
  seed = 1,
  dist_covariates_x = "normal",
  dist_covariates_y = "normal",
  dist_y = "normal",
  x_intercepts = 0,
  y_intercepts = 0,
  beta0_y = 0,
  beta_x = 0.1,
  beta_y = -0.1
)
model_code <- get_abrm_model()
results <- run_abrm(
  gridx = sim_data$gridx,
  gridy = sim_data$gridy,
  atoms = sim_data$atoms,
  model_code = model_code,
  true_params = sim_data$true_params,
  norm_idx_x = 1,
  norm_idx_y = 1,
  dist_y = 1,
  niter = 1000, nburnin = 500, nchains = 2,
  seed = 1, save_plots = FALSE
)
summary(results)
```

```
simulate_misaligned_data
```
Simulate Misaligned Spatial Data

Description

Simulate Misaligned Spatial Data

Usage

```
simulate_misaligned_data(
  seed = 2,
```

```
    dist_covariates_x = c("normal", "poisson", "binomial"),
    dist_covariates_y = c("normal", "poisson", "binomial"),
    dist_y = "poisson",
    x_intercepts = rep(0, 3),
    y_intercepts = rep(0, 3),
    rho_x = 0.6,
    rho_y = 0.6,
    x_correlation = 0.5,
    y_correlation = 0.5,
    beta0_y = NULL,
    beta_x = NULL,
    beta_y = NULL,
    diff_pops = TRUE,
    xy_cov_cor = FALSE
)
```

Arguments

seed   Random seed (default = 2)

dist_covariates_x

               Vector specifying distribution type for each synthetic X-grid covariate ('poisson', 'binomial', or 'normal')

dist_covariates_y

               Vector specifying distribution type for each synthetic Y-grid covariate ('poisson', 'binomial', or 'normal')

dist_y   Distribution type for synthetic outcome variable (one of 'poisson', 'binomial', or 'normal')

x_intercepts   Intercepts for X covariates

y_intercepts   Intercepts for Y covariates

rho_x   Spatial correlation parameter for X-grid covariates (0 to 1 with higher values yielding more spatial correlation, default = 0.6)

rho_y   Spatial correlation parameter for Y-grid covariates and outcome (0 to 1 with higher values yielding more spatial correlation, default = 0.6)

x_correlation Between-variable correlation for all pairs of X-grid covariates (default = 0.5)

y_correlation Between-variable correlation for all pairs of Y-grid covariates (default = 0.5)

beta0_y   Intercept for outcome model

beta_x   Outcome model coefficients for X-grid covariates

beta_y   Outcome model coefficients for Y-grid covariates

diff_pops   Logical, indicating whether the atoms should be generated with different population sizes (diff_pops = TRUE) or a common population size (diff_pops = FALSE)

xy_cov_cor   Logical, indicating whether the atom-level spatial random effects for X-grid and Y-grid covariates should be correlated (xy_cov_cor = TRUE) or not. When set to TRUE, the x_correlation and rho_x parameters are used to generate all covariates (separate correlation parameters are not allowed for X-grid and Y-grid covariates).

Value

An object of class `"misaligned_data"`: a named list with components `gridy` (Y-grid sf data frame with outcome and Y covariates), `gridx` (X-grid sf data frame with X covariates), `atoms` (atom-level sf data frame, the spatial intersection of the two grids), and `true_params` (list of the true regression coefficients used for data generation).

Examples

```
sim_data <- simulate_misaligned_data(
seed = 1,
dist_covariates_x = c('normal', 'poisson', 'binomial'),
dist_covariates_y = c('normal', 'poisson', 'binomial'),
dist_y = "poisson",
x_intercepts = c(4, -1, -1),
y_intercepts = c(4, -1, -1),
beta0_y = -1,
x_correlation = 0.5,
y_correlation = 0.5,
beta_x = c(-0.03, 0.1, -0.2),
beta_y = c(0.03, -0.1, 0.2)
)
class(sim_data)            # "misaligned_data"
print(sim_data)            # grid dimensions and atom count
summary(sim_data)          # includes true beta values
names(sim_data$atoms)      # atom-level variables
sim_data$true_params$beta_x  # true X-grid coefficients
```

---

summary.abrm   *Summary Method for ABRM Objects*

Description

Returns a structured summary of a fitted ABRM, including posterior means, standard deviations, and 95 coefficients. When true parameter values are available (i.e., the model was fitted on simulated data via [simulate_misaligned_data](#)), mean absolute bias and credible-interval coverage are also reported.

Usage

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

Arguments

object	An object of class `"abrm"` returned by [run_abrm](#).
...	Additional arguments (currently unused).

Value

An object of class `"summary.abrm"` (a list) with components:

`n_params` Number of regression parameters estimated.

`mean_abs_bias` Mean absolute bias across parameters, or NA if true values were not supplied.

`coverage` Percentage of parameters whose true value falls within its 95% credible interval, or NA
      if unavailable.

`estimates` Data frame of posterior summaries.

The object is printed via `print.summary.abrm`.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "normal", dist_covariates_y = "normal",
  dist_y = "normal", x_intercepts = 0, y_intercepts = 0,
  beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(), norm_idx_x = 1, norm_idx_y = 1,
  dist_y = 1, niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
summary(results)     # full parameter table with bias and coverage
```

---

summary.misaligned_data
                        Summary Method for Misaligned Data Objects

---

Description

Prints grid dimensions, atom counts, and the true regression coefficients (`beta_x` and `beta_y`) used
to generate the data.

Usage

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

Arguments

object   An object of class `"misaligned_data"` returned by `simulate_misaligned_data`.

...   Additional arguments (currently unused).

Value

Invisibly returns `object`. Called for its side effect of printing the data summary including the true parameter values.

Examples

```
## Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed            = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y          = "poisson",
  x_intercepts    = c(0, -1),
  y_intercepts    = c(0, -1),
  beta0_y         = -1,
  beta_x          = c(0.1, -0.05),
  beta_y          = c(-0.1, 0.05)
)
summary(sim_data)
```

---

vcov.abrm   *Variance-Covariance Method for ABRM Objects*

Description

Extracts variance-covariance matrices for regression coefficients from MCMC posterior samples. Returns separate matrices for X-grid and Y-grid coefficients.

Usage

```
## S3 method for class 'abrm'
vcov(object, ...)
```

Arguments

object   An object of class "abrm" returned by run_abrm.

...   Additional arguments (unused)

Details

The variance-covariance matrices are computed from the posterior samples of the MCMC chains. If multiple chains were run, samples are combined across chains before computing covariances.

Value

A list with class "vcov.abrm" containing:

vcov_beta_x	Variance-covariance matrix for X-grid coefficients
vcov_beta_y	Variance-covariance matrix for Y-grid coefficients
vcov_beta_0	Variance of the intercept (scalar)
vcov_all	Combined variance-covariance matrix for all parameters

Examples

```
## Step 1: Simulate misaligned spatial data
sim_data <- simulate_misaligned_data(
  seed             = 1,
  dist_covariates_x = c("normal", "poisson"),
  dist_covariates_y = c("normal", "poisson"),
  dist_y           = "poisson",
  x_intercepts     = c(0, -1),
  y_intercepts     = c(0, -1),
  beta0_y          = -1,
  beta_x           = c(0.1, -0.05),
  beta_y           = c(-0.1, 0.05)
)

## Step 2: Retrieve the NIMBLE model code
model_code <- get_abrm_model()

## Step 3: Fit the ABRM
results <- run_abrm(
  gridx       = sim_data$gridx,
  gridy       = sim_data$gridy,
  atoms       = sim_data$atoms,
  model_code  = model_code,
  true_params = sim_data$true_params,
  norm_idx_x  = 1,
  pois_idx_x  = 2,
  norm_idx_y  = 1,
  pois_idx_y  = 2,
  dist_y      = 2,
  niter       = 1000,
  nburnin     = 500,
  nchains     = 2,
  seed        = 1,
  save_plots  = FALSE
)

## Step 4: Extract posterior variance-covariance matrices
vcov_mats <- vcov(results)

# Posterior standard errors for X-grid and Y-grid coefficients
sqrt(diag(vcov_mats$vcov_beta_x))
sqrt(diag(vcov_mats$vcov_beta_y))
```

```
# Posterior correlation matrix across all beta parameters
cov2cor(vcov_mats$vcov_all)
```

---

waic   *Generic Function for WAIC*

Description

Extracts the Widely Applicable Information Criterion (WAIC) from a fitted model object.

Usage

```
waic(object, ...)
```

Arguments

object   A fitted model object.

...   Additional arguments passed to methods.

Value

A "waic_abrm" object for abrm models. See [waic.abrm](#) for details.

---

waic.abrm   *WAIC for ABRM Objects*

Description

Returns the Widely Applicable Information Criterion (WAIC) for a fitted ABRM. WAIC is the recommended information criterion for Bayesian models and is computed by NIMBLE when compute_waic = TRUE is passed to [run_abrm](#).

Usage

```
## S3 method for class 'abrm'
waic(object, ...)
```

Arguments

object   An object of class "abrm" returned by [run_abrm](#).

...   Additional arguments (currently unused).

Details

WAIC (Watanabe 2010) is defined as:

$$\text{WAIC} = -2 \cdot \text{lppd} + 2 \cdot p_{\text{WAIC}}$$

where lppd is the log pointwise predictive density and p_{WAIC} is the effective number of parameters. Unlike AIC, WAIC is fully Bayesian and averages over the posterior distribution rather than evaluating at a point estimate.

WAIC values are only meaningful when compared between models fitted on *identical* data with the same outcome distribution. A true marginal log-likelihood is not available for ABRM, so `AIC` and `logLik` are not supported; use `waic()` instead.

Value

An object of class `"waic_abrm"`, which is a named list with components:

waic The scalar WAIC value. Lower values indicate better predictive accuracy penalised for model complexity.

lppd Log pointwise predictive density — the goodness-of-fit component of WAIC.

penalty The effective number of parameters (p_{WAIC}) — the complexity penalty component.

n_params The number of regression parameters estimated.

Returns an error if the model was not fitted with `compute_waic = TRUE`.

References

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11, 3571–3594.

See Also

`waic`, `print.waic_abrm`, `run_abrm`

Examples

```
sim_data <- simulate_misaligned_data(
  seed            = 1,
  dist_covariates_x = "normal",
  dist_covariates_y = "normal",
  dist_y          = "normal",
  x_intercepts    = 0,
  y_intercepts    = 0,
  beta0_y         = 0,
  beta_x          = 0.1,
  beta_y          = -0.1
)
results <- run_abrm(
  gridx         = sim_data$gridx,
  gridy         = sim_data$gridy,
```

```
      atoms        = sim_data$atoms,
      model_code   = get_abrm_model(),
      norm_idx_x   = 1,
      norm_idx_y   = 1,
      dist_y       = 1,
      niter        = 1000,
      nburnin      = 500,
      nchains      = 2,
      seed         = 1,
      compute_waic = TRUE    # required; re-fit if this was FALSE
    )

    w <- waic(results)
    print(w)          # formatted table: WAIC, lppd, pWAIC, n_params
    w$waic            # raw scalar for comparing models
    w$penalty         # effective number of parameters (pWAIC)
```

Index