Package: spatialAtomizeR (via r-universe)

November 23, 2026

Type Package

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

Version 0.3.0

Date 2026-11-17

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 results. 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

LazyData true

Depends R (>= 4.1.0)

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

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

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libicu-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-11-23 03:12:35 UTC

RemoteUrl https://github.com/bellayqian/spatialatomizer

RemoteRef HEAD

RemoteSha 9299333ee5fe7e12323b79dbd1c65d8f85a1e671

Contents

abrm_params . . . 2
abrm_plots . . . 4
coef.abrm . . . 5
confint.abrm . . . 6
create_atoms . . . 7
fitted.abrm . . . 8
get_abrm_model . . . 9
mcmc_diagnostics . . . 9
plot.misaligned_data . . . 10
print.abrm . . . 11
print.misaligned_data . . . 12
print.summary.abrm . . . 13
print.vcov.abrm . . . 14
print.waic_abrm . . . 15
run_abrm . . . 15
run_nimble_model . . . 19
simulate_misaligned_data . . . 21
summary.abrm . . . 23
summary.misaligned_data . . . 24
vcov.abrm . . . 25
waic . . . 26
waic.abrm . . . 27

Index **29**

`abrm_params` *Construct True Parameter Values for* `run_abrm()`

Description

Builds the `true_params` argument of `run_abrm`: the true regression coefficients of the outcome model. When `true_params` is supplied, `run_abrm()` reports the bias, relative bias and credible-interval coverage of each estimated coefficient. This is mainly useful with simulated data, where the truth is known; `simulate_misaligned_data` already returns its truth in this format.

Usage

```
abrm_params(beta0_y = NULL, beta_x = NULL, beta_y = NULL, beta_a = NULL, ...)

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

Arguments

beta0_y	Numeric scalar, the intercept of the outcome model.
beta_x	Numeric vector of coefficients for the X-grid covariates.
beta_y	Numeric vector of coefficients for the Y-grid covariates.
beta_a	Numeric vector of coefficients for atom-level covariates (only needed if the model uses atom_vars).
...	Unused.
x	An "abrm_params" object.

Details

The outcome model is

$$g(\mu) = \beta_0 + \sum_j \beta_{x,j} x_j + \sum_k \beta_{y,k} y_k + \sum_l \beta_{a,l} a_l + \dots$$

for X-grid covariates x_j, Y-grid covariates y_k and (optional) atom-level covariates a_l. Each of beta_x, beta_y and beta_a is either

- an **unnamed** vector, matched to the covariates by *position* (the order of x_vars, y_vars and atom_vars in run_abrm()), or
- a **named** vector, matched by covariate *name*, in any order. The names must be exactly the covariate names.

Components left as NULL are treated as unknown: the corresponding rows of the results have NA true values and are excluded from the bias and coverage summaries.

Lengths and names are checked against the data when the object is passed to run_abrm().

Value

An object of class "abrm_params": a named list with the components beta0_y, beta_x, beta_y and beta_a (NULL when not supplied) plus anything passed through .... Elements are accessed as for any list, e.g. p$beta_x.

print() returns x invisibly.

Examples

```
# Matched by position
p <- abrm_params(beta0_y = -1, beta_x = c(0.1, -0.05), beta_y = 0.2)
p

# Matched by covariate name (any order)
abrm_params(beta0_y = -1,
            beta_x = c(clinics = 0.1, income = -0.05),
            beta_y = c(age = 0.2))


  sim <- simulate_misaligned_data(
```

```
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
fit <- run_abrm(
  sim$gridx, sim$gridy, sim$atoms, get_abrm_model(),
  x_types = "norm", y_types = "norm", dist_y = "norm",
  true_params = abrm_params(beta0_y = 0, beta_x = 0.1, beta_y = -0.1),
  niter = 1000, nburnin = 500, seed = 1
)
fit$parameter_estimates
```

---

abrm_plots *MCMC Trace and Density Plots for ABRM Objects*

Description

plot_trace() draws MCMC trace plots and plot_density() draws posterior density plots of the regression coefficients of a fitted ABRM, one panel per parameter and one colour per chain. Panels are labelled with the covariate names. plot() on an abrm object calls one of these, chosen by type.

Usage

```
plot_trace(x, parm = NULL, title = "MCMC trace plots")

plot_density(x, parm = NULL, title = "Posterior density plots")

## S3 method for class 'abrm'
plot(x, type = c("trace", "density"), ...)
```

Arguments

x	An object of class "abrm" returned by run_abrm.
parm	Optional character vector of parameters to plot, given either as covariate names (e.g. "intercept", "covariate_x_1") or as internal names (e.g. "beta_x[1]"). Default: all coefficients.
title	Plot title.
type	For plot(): "trace" (default) or "density".
...	For plot(): further arguments passed to plot_trace() or plot_density().

Value

A ggplot object, drawn when printed. It can be modified with additional ggplot2 layers or saved with ggplot2::ggsave().

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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(), x_types = "norm", y_types = "norm",
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
plot_trace(results)
plot_density(results, parm = c("covariate_x_1", "covariate_y_1"))
plot(results, type = "density")   # same as plot_density(results)
```

`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, named `"intercept"` followed by the covariate names.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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(), x_types = "norm", y_types = "norm",
```

```
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
coef(results)
```

`confint.abrm` *Credible Intervals for ABRM Objects*

Description

Returns equal-tailed posterior credible intervals for the regression coefficients, computed from the MCMC samples pooled across chains. 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 parameters, given as covariate names (e.g. `"covariate_x_1"`, `"intercept"`) or internal names (e.g. `"beta_x[1]"`). If `NULL` (default), all coefficients are returned.
`level`	Credible level (default `0.95`).
`...`	Additional arguments (currently unused).

Value

A matrix with two columns, `CI.Lower` and `CI.Upper`, and one row per parameter, labelled with covariate names.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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(), x_types = "norm", y_types = "norm",
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
confint(results)
confint(results, parm = "covariate_x_1", level = 0.9)
```

`create_atoms` *Create Atoms by Intersecting Two Grids*

Description

Computes the atoms, i.e. the non-empty polygonal pieces of the spatial intersection of the Y-grid and X-grid, using `st_intersection`. Lower-dimensional pieces where cell boundaries merely touch (points and line segments) are dropped, as are pieces whose area does not exceed `min_area`.

Usage

```
create_atoms(gridy, gridx, id_y = "ID", id_x = "ID", min_area = 0)
```

Arguments

`gridy`	Y-grid `sf` data frame.
`gridx`	X-grid `sf` data frame.
`id_y`, `id_x`	Names of the ID columns in `gridy` and `gridx`. IDs may be integer, numeric or character, but must be unique within each grid. Whole-number numeric IDs (for example GEOIDs converted with `as.numeric()`) are stored as integers in the result when they fit; character IDs are kept as they are. Default `"ID"` for both.
`min_area`	Atoms with area less than or equal to this value (in the units of `sf::st_area()`) are dropped. Useful for removing slivers caused by slightly mismatched boundaries in real shapefiles. Default 0.

Value

An `sf` data frame with one row per atom and columns `ID_y` and `ID_x` holding the IDs (as in `gridy` and `gridx`) of the cells each atom belongs to.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", beta0_y = 0, beta_x = 0.1, beta_y = -0.1
)
atoms <- create_atoms(sim_data$gridy, sim_data$gridx)
nrow(atoms)
```

`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 Y-grid cell ID, as supplied in `gridy`.

`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 = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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(), x_types = "norm", y_types = "norm",
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
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)
```

mcmc_diagnostics *MCMC Convergence Diagnostics for ABRM Objects*

Description

Returns the convergence diagnostics computed after fitting: posterior mean and standard deviation, Gelman-Rubin potential scale reduction factor (R-hat) and effective sample size (ESS) for each regression coefficient.

Usage

```
mcmc_diagnostics(
  object,
  parm = NULL,
  rhat_threshold = 1.1,
  ess_threshold = 400
)
```

Arguments

object	An object of class "abrm" returned by run_abrm.
parm	Optional character vector of parameters, given as covariate names or internal names (as for plot_trace). Default: all.
rhat_threshold, ess_threshold	A parameter is flagged as converged when rhat < rhat_threshold and ess > ess_threshold. Defaults 1.1 and 400.

Value

A data frame with one row per parameter and columns variable (covariate name or "intercept"), parameter (internal name), mean, sd, rhat, ess and converged. rhat (and therefore converged) is NA when only one chain was run. Use all(mcmc_diagnostics(fit)$converged) for an overall check.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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$atoms,
  model_code = get_abrm_model(), x_types = "norm", y_types = "norm",
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
mcmc_diagnostics(results)
mcmc_diagnostics(results, ess_threshold = 100)
all(mcmc_diagnostics(results)$converged)
```

---

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". which = "atoms" is an error if x$atoms has not been computed (see create_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

A ggplot object. It is drawn when printed (for example at the console), and can be modified with further ggplot2 layers or saved with ggplot2::ggsave().

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = c("norm", "pois"),
  dist_covariates_y = c("norm", "pois"), dist_y = "pois",
  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 = "atoms")

# The result is a ggplot object that can be extended
p <- plot(sim_data, color_y = "steelblue", color_x = "firebrick")
p + ggplot2::labs(caption = "Simulated data")
```

---

print.abrm *Print Method for ABRM Objects*

Description

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

Usage

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

Arguments

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

Value

Invisibly returns x.

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = c("norm", "pois"),
  dist_covariates_y = c("norm", "pois"), dist_y = "pois",
  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)
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(),
  x_types = c("norm", "pois"), y_types = c("norm", "pois"), dist_y = "pois",
  true_params = sim_data$true_params,
  niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
print(results)
```

---

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("norm", "pois"),
  dist_covariates_y = c("norm", "pois"),
  dist_y            = "pois",
  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 = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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(), x_types = "norm", y_types = "norm",
  dist_y = "norm", niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
s <- summary(results)
print(s, digits = 2)
```

---

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`.

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. Variables can have any column names: tell run_abrm() which columns to use through x_vars, y_vars, outcome, atom_vars, the ID arguments and population.

Usage

```
run_abrm(
  gridx,
  gridy,
  atoms,
  model_code,
  x_types,
  y_types,
  dist_y = "pois",
  x_vars = NULL,
  y_vars = NULL,
  outcome = "y",
```

```
  atom_vars = NULL,
  id_x = "ID",
  id_y = "ID",
  atom_id_x = "ID_x",
  atom_id_y = "ID_y",
  population = NULL,
  true_params = NULL,
  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 data frame, containing an area ID column (see id_x) and the X-grid covariates (see x_vars).
gridy	The Y-grid sf data frame, containing an area ID column (see id_y), the Y-grid covariates (see y_vars) and the outcome (see outcome).
atoms	The atom sf data frame (for example from create_atoms). Must contain the X-grid and Y-grid ID of each atom (see atom_id_x, atom_id_y), the population denominator column(s) (see population) and any atom-level covariates (see atom_vars).
model_code	NIMBLE model code from get_abrm_model.
x_types	Character vector with one element per X-grid covariate (in the order of x_vars) giving its distribution: "norm", "pois" or "binom". The long forms "normal", "poisson" and "binomial" are also accepted.
y_types	As x_types, for the Y-grid covariates (in the order of y_vars).
dist_y	Distribution of the outcome: "norm", "pois" (default) or "binom" (long forms also accepted).
x_vars	Character vector of X-grid covariate column names in gridx. If NULL (default), all columns named covariate_x_1, covariate_x_2, ... are used.
y_vars	Character vector of Y-grid covariate column names in gridy. If NULL (default), all columns named covariate_y_1, covariate_y_2, ... are used.
outcome	Name of the outcome column in gridy. Default "y".
atom_vars	Optional character vector of atom-level covariate column names in atoms. If NULL (default), any columns named covariate_a_1, covariate_a_2, ... are used; use character(0) to fit without atom-level covariates.
id_x, id_y	Names of the area ID columns in gridx and gridy. IDs may be integer, numeric or character but must be unique within each grid. Default "ID".

`atom_id_x`, `atom_id_y`
: Names of the columns in `atoms` holding each atom's X-grid and Y-grid ID. Defaults `"ID_x"` and `"ID_y"` (the names produced by `create_atoms`).

`population`
: Population denominator(s) in `atoms`. Either
  - a single column name used for every variable, or
  - a named list `list(x = , y = , outcome = )` giving one column name per X-grid covariate, one per Y-grid covariate (each may be a single name to recycle) and one for the outcome, or
  - `NULL` (default): use a column named `population` if present; otherwise per-variable columns named `population_covariate_x_1`, ..., `population_covariate_y_1`, ..., and `population_y` (as produced by `simulate_misaligned_data`).

`true_params`
: Optional true values of the outcome-model coefficients, used to report bias and credible-interval coverage. Build it with `abrm_params` (recommended), or pass the `true_params` component returned by `simulate_misaligned_data`, which has the same format. A plain named list with the same elements is also accepted. See **Format of** `true_params` below.

`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 to a PDF (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`.

`...`
: Only used to support the deprecated arguments `norm_idx_x`, `pois_idx_x`, `binom_idx_x`, `norm_idx_y`, `pois_idx_y` and `binom_idx_y`, which are converted to `x_types` / `y_types` with a warning. Any other argument is an error.

Value

An object of class `"abrm"`: a named list with components

`mcmc_results` Output of `run_nimble_model`: MCMC samples, posterior summary, convergence diagnostics (see `mcmc_diagnostics`) and, optionally, WAIC.

`parameter_estimates` Data frame with one row per regression coefficient: `variable` (the covariate name, or `"intercept"`), `parameter` (the internal parameter name, e.g. `"beta_x[1]"`), `estimated_beta`, `std_error`, `ci_lower`, `ci_upper`, and, when `true_params` is supplied, `true_beta`, `bias`, `relative_bias` and `within_ci`.

`all_parameters` Posterior summaries of all monitored parameters.

`fitted_values` Numeric vector of Y-grid-level predicted outcome values on the original outcome scale.

`y_grid_ids` Y-grid cell IDs (as in `gridy[[id_y]]`) in model order.

`y_observed` Observed outcome values in model order.

`param_labels` Named character vector mapping internal parameter names to covariate names.

`model_info` List recording `x_vars`, `y_vars`, `atom_vars`, `outcome`, `x_types`, `y_types` and `dist_y`.

Use `fitted()` for a table of observed vs. fitted values, `plot_trace` / `plot_density` for MCMC plots, `mcmc_diagnostics` for R-hat and effective sample sizes, and `waic()` for model fit criteria when `compute_waic = TRUE`.

Format of `true_params`

`true_params` is a list, best created with `abrm_params()`, with the following elements. Any of them may be omitted; coefficients without a true value get `NA` in the results and are left out of the bias and coverage summaries.

`beta0_y` Numeric scalar: the outcome-model intercept. (`beta_0_y` is accepted as an alias.)

`beta_x` Numeric vector of length `length(x_vars)`: the coefficients of the X-grid covariates. Unnamed vectors are matched by position (the order of `x_vars`); named vectors are matched by covariate name.

`beta_y` Numeric vector of length `length(y_vars)`: the coefficients of the Y-grid covariates, matched as for `beta_x`.

`beta_a` Numeric vector of length `length(atom_vars)`: the coefficients of the atom-level covariates, matched as for `beta_x`.

Other elements (such as `x_correlation` from `simulate_misaligned_data()`) are ignored. For example:

```
true_params = abrm_params(beta0_y = -1,
                          beta_x  = c(0.10, -0.05),
                          beta_y  = c(income = -0.10, pct_old = 0.05))
```

Examples

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

```
# Retrieve the 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,
  x_types     = "norm",
  y_types     = "norm",
  dist_y      = "norm",
  true_params = abrm_params(beta0_y = 0, beta_x = 0.1, beta_y = -0.1),
  niter       = 1000,
  nburnin     = 500,
  nchains     = 2,
  seed        = 1
)

print(results)             # concise model summary
summary(results)           # full parameter table
plot_trace(results)        # MCMC trace plots (a ggplot object)
plot_density(results)      # posterior density plots (a ggplot object)
mcmc_diagnostics(results)  # R-hat and effective sample sizes

# Covariates can have any names
gx <- sim_data$gridx; names(gx)[names(gx) == "covariate_x_1"] <- "income"
gy <- sim_data$gridy; names(gy)[names(gy) == "covariate_y_1"] <- "age"
results2 <- run_abrm(
  gridx = gx, gridy = gy, atoms = sim_data$atoms, model_code = model_code,
  x_vars = "income", y_vars = "age", outcome = "y",
  x_types = "norm", y_types = "norm", dist_y = "norm",
  niter = 1000, nburnin = 500, seed = 1
)
coef(results2)
```

`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 trace and density plots to a PDF file (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, with the outcome coefficients named beta_0_y, beta_x[], beta_y[] and beta_a[]), summary (posterior summary statistics), convergence (Gelman-Rubin statistics and effective sample sizes; see mcmc_diagnostics) and, when compute_waic = TRUE, WAIC.

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 = "norm",
  dist_covariates_y = "norm",
```

```
  dist_y = "norm",
  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$atoms,
  model_code = model_code,
  x_types = "norm",
  y_types = "norm",
  dist_y = "norm",
  true_params = sim_data$true_params,
  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("norm", "pois", "binom"),
  dist_covariates_y = c("norm", "pois", "binom"),
  dist_y = "pois",
  x_intercepts = rep(0, length(dist_covariates_x)),
  y_intercepts = rep(0, length(dist_covariates_y)),
  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,
```

```
  diff_pops_per_var = FALSE,
  xy_cov_cor = FALSE
)
```

Arguments

seed	Random seed (default = 2)
dist_covariates_x	Character vector giving the distribution of each synthetic X-grid covariate: "norm", "pois" or "binom" (the long forms "normal", "poisson", "binomial" are also accepted).
dist_covariates_y	As dist_covariates_x, for the Y-grid covariates.
dist_y	Distribution of the synthetic outcome: one of "norm", "pois" or "binom" (long forms also accepted).
x_intercepts	Intercepts for X covariates (one per covariate).
y_intercepts	Intercepts for Y covariates (one per covariate).
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 (required).
beta_x	Outcome model coefficients for X-grid covariates (required; one per X covariate).
beta_y	Outcome model coefficients for Y-grid covariates (required; one per Y covariate).
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). Ignored when diff_pops_per_var = TRUE.
diff_pops_per_var	Logical. When TRUE, each covariate and the outcome receive their own population denominator column in the atoms object (named population_covariate_x_1, population_covariate_y_1, ..., and population_y). When FALSE (default), a single population column is used for all variables.
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 integer ID, outcome y and Y covariates covariate_y_1, ...), gridx (X-grid sf data frame with integer ID and X covariates covariate_x_1, ...), atoms (atom-level sf data frame, the spatial intersection of the two grids, with integer ID_y, ID_x and ID_atomorder), and true_params (an abrm_params object holding the true regression coefficients used for data generation, ready to pass to run_abrm).

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1,
  dist_covariates_x = c("norm", "pois", "binom"),
  dist_covariates_y = c("norm", "pois", "binom"),
  dist_y = "pois",
  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 with true_params), 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 = "norm", dist_covariates_y = "norm",
  dist_y = "norm", 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$atoms,
  model_code = get_abrm_model(), x_types = "norm", y_types = "norm",
  dist_y = "norm", true_params = sim_data$true_params,
  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

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = c("norm", "pois"),
  dist_covariates_y = c("norm", "pois"), dist_y = "pois",
  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, Y-grid and (if present) atom-level coefficients, plus the combined matrix. Rows and columns are labelled with covariate names.

Usage

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

Arguments

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

Details

The matrices are computed from the posterior samples pooled across chains.

Value

A list with class "vcov.abrm" containing:

`vcov_beta_0`	Posterior variance of the intercept (named scalar)
`vcov_beta_x`	Variance-covariance matrix for X-grid coefficients
`vcov_beta_y`	Variance-covariance matrix for Y-grid coefficients
`vcov_beta_a`	Variance-covariance matrix for atom-level coefficients (only when atom-level covariates were used)
`vcov_all`	Combined variance-covariance matrix for all coefficients

Examples

```
sim_data <- simulate_misaligned_data(
  seed = 1, dist_covariates_x = c("norm", "pois"),
  dist_covariates_y = c("norm", "pois"), dist_y = "pois",
  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)
)
results <- run_abrm(
  gridx = sim_data$gridx, gridy = sim_data$gridy, atoms = sim_data$atoms,
  model_code = get_abrm_model(),
  x_types = c("norm", "pois"), y_types = c("norm", "pois"), dist_y = "pois",
  niter = 1000, nburnin = 500, nchains = 2, seed = 1
)
vcov_mats <- vcov(results)
sqrt(diag(vcov_mats$vcov_beta_x))
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 = "norm",
  dist_covariates_y = "norm",
  dist_y            = "norm",
  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(),
  x_types      = "norm",
  y_types      = "norm",
  dist_y       = "norm",
  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

abrm_params, 2, *17, 18*, *23*
abrm_plots, 4
AIC, *27*

coef.abrm, 5
confint.abrm, 6
create_atoms, 7, *11*, *16, 17*

fitted.abrm, 8

get_abrm_model, 9, *16*

logLik, *27*

mcmc_diagnostics, 9, *17, 18*, *20*

plot.abrm *(*abrm_plots*)*, 4
plot.misaligned_data, 10
plot_density, *18*
plot_density *(*abrm_plots*)*, 4
plot_trace, *10*, *18*
plot_trace *(*abrm_plots*)*, 4
print.abrm, 11
print.abrm_params *(*abrm_params*)*, 2
print.misaligned_data, 12
print.summary.abrm, 13, *24*
print.vcov.abrm, 14
print.waic_abrm, 15, *28*

run_abrm, *2*, *4–6*, *8–10*, 15, *23*, *25*, *27, 28*
run_nimble_model, *17*, 19

simulate_misaligned_data, *2*, *17*, 21, *24*
st_intersection, *7*
summary.abrm, *13*, 23
summary.misaligned_data, 24

vcov.abrm, *14*, 25

waic, *17*, *20*, 26, *28*
waic.abrm, *15*, *26*, 27