Package: inlaops (via r-universe)

November 5, 2026

Title Deployment, Versioning and Scoring Tools for 'inlabru' Models

Version 0.1.2

Description Serves Bayesian spatial and latent Gaussian models fit with 'inlabru' as versioned Representational State Transfer (REST) application programming interfaces (APIs). Includes a 'vetiver' bridge (versioning, REST API serving with posterior uncertainty, 'Docker' packaging) for 'bru' objects, a 'parsnip' model constructor scoped to fixed-effects models, and predictive scoring helpers (continuous ranked probability, log, squared-error and Dawid-Sebastiani scores) computed from inlabru::generate() posterior samples via 'scoringRules'.

License GPL (>= 3)

URL https://github.com/novica/inlaops, https://novica.github.io/inlaops/

BugReports https://github.com/novica/inlaops/issues

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports dplyr, inlabru (>= 2.15.0), rlang, tibble

Suggests broom, callr, fmesher, ggplot2, hardhat, httpuv, httr2, INLA, jsonlite, knitr, parsnip, pins, plumber, purrr, recipes, rsample, scoringRules, sf, sn, testthat (>= 3.0.0), vetiver, yardstick, quarto

Enhances sp

VignetteBuilder quarto

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

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

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

Date/Publication 2026-11-05 08:15:28 UTC

RemoteUrl https://github.com/novica/inlaops

RemoteRef HEAD

RemoteSha 778d4d18d47fa3991704aa955c1974524e710eff

Contents

bru_cv_predict 2
bru_reg 4
bru_score 5

Index **7**

`bru_cv_predict` *Cross-validation predictions for fixed-effects inlabru models*

Description

Wraps `rsample::vfold_cv()` and re-fits the model on each training fold, returning a tidy tibble of holdout predictions. Scoped to models with fixed effects and IID random intercepts only.

Usage

```
bru_cv_predict(
  formula,
  family,
  data,
  v = 5L,
  pred_formula,
  n_samples = 500L,
  seed = NULL,
  strata = NULL,
  ...
)
```

Arguments

`formula`	Model formula passed to `inlabru::bru()`.
`family`	INLA family string.
`data`	Training data frame.
`v`	Number of folds (default 5).
`pred_formula`	Formula passed to `inlabru::predict()` for the response scale (e.g. `~ exp(Intercept + x)`).
`n_samples`	Number of posterior samples per fold (default 500).

seed	Integer seed passed to set.seed() before fold assignment so that two identical calls produce identical fold splits. The session RNG state is restored after fold assignment via on.exit. NULL (default) leaves the session RNG state unchanged.
strata	Optional name of a column in data to use for stratified sampling. Passed through to rsample::vfold_cv(). Useful for sparse counts or imbalanced binomial outcomes where random folds can leave a fold with zero events.
...	Extra named arguments forwarded to every per-fold inlabru::bru() call (e.g. options = inlabru::bru_options(...), alternative control.compute settings). config = TRUE is injected unconditionally so per-fold posterior sampling always works.

Details

Structured random effects (SPDE, BYM2, AR1, RW1/RW2) cannot be cross-validated by simply subsetting the data. Each requires per-fold reconstruction of the index structure: SPDE models need a new A-matrix (mesh projector) built from the fold's observation coordinates; BYM2 areal models need the adjacency graph to account for areas with no fold observations; AR1/RW models assume a contiguous integer time index that breaks when folds remove time points. None of this can be done generically. For those models use spatial block cross-validation and evaluate with bru_score().

Folds run sequentially. Callers who want parallel folds should compose with their own parallel backend (e.g. future.apply::future_lapply) around a per-fold inlabru::bru() call — this package stays out of the daemon lifecycle to keep the scope focused on bridging, not execution.

Value

A tibble with columns .truth, .pred, .pred_low, .pred_high, .pred_sd.

Examples

```
set.seed(1)
dat <- data.frame(x = rnorm(60))
dat$y <- 1 + 0.5 * dat$x + rnorm(60, sd = 0.5)
bru_cv_predict(
  formula = y ~ Intercept(1) + x,
  family = "gaussian",
  data = dat,
  v = 3,
  pred_formula = ~ Intercept + x,
  n_samples = 100,
  seed = 42
)
```

bru_reg *parsnip model constructor for inlabru fixed-effects models*

Description

Scoped to models with fixed effects and IID random intercepts only. Spatial models with BYM2, SPDE, AR1, or LGCP likelihoods should use inlabru's native interface.

Usage

```
bru_reg(mode = "regression", family = "gaussian")
```

Arguments

mode	Model mode. Currently only "regression" is supported.
family	INLA family string, e.g. "gaussian", "poisson".

Details

The formula passed to `parsnip::fit()` must use inlabru component syntax. An explicit intercept is required: `y ~ Intercept(1) + x1 + x2`. Standard `y ~ x1 + x2` formulas are not supported because inlabru does not add an intercept automatically.

IID random intercepts are supported in the fit, but `predict()` via the parsnip interface returns the **marginal** mean — only the fixed effects are included in the response-scale linear predictor. The IID random intercept is integrated over rather than conditioned on any specific group level, which matches the "new group" use case but silently ignores the random effect for held-out observations from a seen group. If you need conditional predictions, call `inlabru::predict()` directly with a formula that references the IID component.

Value

A `bru_reg` model spec.

Examples

```
bru_reg(family = "poisson")
```

bru_score *Predictive scores for inlabru models*

Description

Thin wrapper over scoringRules that takes posterior samples from inlabru::generate() and returns four predictive scores per observation.

Usage

```
bru_score(fit, newdata, pred_formula, y_col, n_samples = 1000L, seed = NULL)
```

Arguments

fit	A fitted bru object. Must have been fit with control.compute = list(config = TRUE).
newdata	Data frame of covariate values for held-out observations.
pred_formula	Formula passed to inlabru::generate(). Must produce draws of Y on the outcome scale; see Details.
y_col	Name of the response column in newdata.
n_samples	Number of posterior samples (default 1000).
seed	Optional non-zero integer. If non-NULL, the scores are reproducible: the seed is used for R's RNG (the r*() draws in pred_formula) and for INLA's posterior sampler, which then runs single-threaded. The caller's RNG stream is restored afterwards.

Details

pred_formula must produce samples of Y — including observation-model stochasticity for discrete families. The caller writes the full distributional expression. Inside an inlabru generate() formula the predictor is evaluated once per posterior sample: latent-field symbols (Intercept, SPDE field values, ...) come in as scalars, and covariates come in as vectors over observations. So length(<covariate>) gives the observation count for the per-sample r*() call:

- Poisson: ~ rpois(length(x), exp(Intercept + x))
- Binomial: ~ rbinom(length(x), size = n_trials, prob = plogis(Intercept + x))
- Gaussian: ~ rnorm(length(x), mean = Intercept + x, sd = 1 / sqrt(Precision_for_the_Gaussian_observatio

Hyperparameters are available in the formula under their standardised names (see inlabru::bru_standardise_names()), so the Gaussian observation noise is sampled jointly with the latent field.

This is the same contract inlabru::generate() already exposes — the package does not inject observation noise or dispatch on family. Users who want response-scale samples for a family not listed above write the appropriate random draw in the formula.

CRPS and the kernel log-score are strictly proper for continuous Y. For count / binomial Y the empirical CRPS on integer draws is still a reasonable proper score; scoringRules::logs_sample

uses a kernel-density estimate which can be unstable on discrete supports — treat `logs` as approximate in that case.

Dawid-Sebastiani variance is floored at `.Machine$double.eps` to avoid division by zero when the sample variance collapses.

Value

A tibble with one row per observation and columns `crps`, `logs`, `se`, `ds`.

References

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477), 359–378. doi:10.1198/016214506000001437

Examples

```
set.seed(1)
dat <- data.frame(x = rnorm(40))
dat$y <- rpois(40, exp(0.5 + 0.3 * dat$x))
fit <- bru_inlaops_fit(
  y ~ Intercept(1) + x,
  data = dat[1:30, ],
  family = "poisson"
)
bru_score(
  fit,
  newdata = dat[31:40, ],
  pred_formula = ~ rpois(length(x), exp(Intercept + x)),
  y_col = "y",
  n_samples = 200,
  seed = 1
)
```

Index

bru_cv_predict, 2
bru_reg, 4
bru_score, 5
bru_score(), *3*

inlabru::bru_standardise_names(), *5*

rsample::vfold_cv(), *3*