Package 'FusionForests'

August 9, 2026

Type Package

Title Bayesian Tree Ensembles for Data Fusion and Causal Inference

Version 1.0.1

Date 2026-08-01

Maintainer Tijn Jacobs <t.jacobs@vu.nl>

Description Bayesian tree ensemble models for data fusion and causal inference. The main model FusionForest() combines data from a randomised controlled trial and an observational study using separate tree forests, allowing for unmeasured confounding in the observational data. Continuous and (interval-)censored survival outcomes are supported. Posterior summaries of treatment effect estimates and interpretable linear projections are provided.

URL https://github.com/tijn-jacobs/FusionForests

BugReports https://github.com/tijn-jacobs/FusionForests/issues

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports Rcpp, ShrinkageTrees

LinkingTo Rcpp (>= 1.0.11)

Suggests survival, testthat (>= 3.0.0)

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation yes

Author Tijn Jacobs [aut, cre] (ORCID: <https://orcid.org/0009-0003-6188-9296>)

Repository CRAN

Date/Publication 2026-08-09 06:40:10 UTC

Contents

`FusionForest` *FusionForest*

Description

Bayesian data-fusion model combining an RCT and an observational study to estimate heterogeneous treatment effects.

Usage

```
FusionForest(
  y,
  status = NULL,
  observed_left_time = NULL,
  observed_right_time = NULL,
  interval_censoring_indicator = NULL,
  X_train_control,
  X_train_treat,
  treatment_indicator_train,
  source_indicator_train,
  X_test_control = NULL,
  X_test_treat = NULL,
  X_test_deconf = NULL,
  treatment_indicator_test = NULL,
  source_indicator_test = NULL,
  outcome_type = "continuous",
  timescale = "time",
  decomposition = "four-forest",
  number_of_trees_control = 200,
  number_of_trees_treat = 100,
  number_of_trees_deconf = 50,
  number_of_trees_deviation = 50,
  k_control = 0.5,
  k_treat = 0.5,
  k_deconf = 0.5,
```

```
  k_deviation = 0.5,
  power_control = 2,
  base_control = 0.95,
  power_deviation = 2,
  base_deviation = 0.95,
  power_treat = 3,
  base_treat = 0.95,
  power_deconf = 3,
  base_deconf = 0.25,
  p_grow = 0.4,
  p_prune = 0.4,
  nu = 3,
  q = 0.9,
  sigma = NULL,
  N_post = 5000,
  N_burn = 5000,
  treatment_coding = c("centered", "binary", "adaptive"),
  propensity_train = NULL,
  propensity_test = NULL,
 error_dist = c("gaussian", "shared_dp", "source_dp", "source_dp_scale", "source_hdp",
    "source_hdp_scale"),
  error_truncation_K = 50L,
  error_atom_scale = 0.5,
  error_mass_init = 1,
  store_posterior_sample = FALSE,
  verbose = TRUE
)
```

Arguments

`y`	Numeric vector of outcomes (survival times or continuous responses).
`status`	Integer vector of event indicators (`1` = event observed, `0` = censored). Required when `outcome_type = "right-censored"`. When `interval_censoring_indicator[i] == 1`, `status[i]` should be `0` (the event is known to lie in an interval, not at a point).
`observed_left_time`, `observed_right_time`, `interval_censoring_indicator`	Optional numeric vectors of length `n`, used only for right-censored outcomes to allow interval-censored events. For each observation i: `status[i] = 1` Event observed at `y[i]`; no augmentation. `observed_left_time[i]` and `observed_right_time[i]` are ignored. `status[i] = 0` **and** `interval_censoring_indicator[i] = 0` Standard right-censoring at `observed_right_time[i]` (event time unknown but greater than `observed_right_time[i]`). `status[i] = 0` **and** `interval_censoring_indicator[i] = 1` Interval-censored: event lies in `(observed_left_time[i], observed_right_time[i]]`. At each sweep the event time is augmented by a truncated-normal draw on that interval.

When all three are NULL (the default) the wrapper falls back to right-censoring with observed_right_time = y and interval_censoring_indicator = 0, exactly reproducing the historical right-censored behaviour. Bounds are supplied on the same scale as y (e.g. raw survival time when timescale = "time"; log-time when timescale = "log"). Ignored when outcome_type != "right-censored".

X_train_control
: Numeric matrix of covariates for the prognostic (m_0 or μ) and deconfounding (c) forests. One row per training observation.

X_train_treat
: Numeric matrix of covariates for the treatment-effect (τ) forest. Must have the same number of rows as X_train_control.

treatment_indicator_train
: Integer vector of treatment assignments (1 = treated, 0 = control) for training observations.

source_indicator_train
: Integer vector of data-source labels (1 = RCT, 0 = observational study) for training observations.

X_test_control, X_test_treat, X_test_deconf
: Optional test matrices. If omitted, predictions are returned for a single row at the column means of the training data.

treatment_indicator_test, source_indicator_test
: Optional integer vectors for the test set (same coding as the training-set equivalents).

outcome_type
: Character; either "continuous" or "right-censored".

timescale
: Character; "time" (raw survival times, will be log-transformed internally) or "log" (already on log scale).

decomposition
: Character; must be "four-forest" (the default and currently the only option).

number_of_trees_control, number_of_trees_treat, number_of_trees_deconf
: Number of trees in each BART ensemble. Defaults: 200 (control), 100 (treat), 50 (deconf).

number_of_trees_deviation
: Number of trees for the g forest. Default 50.

k_control, k_treat, k_deconf, k_deviation
: Leaf-prior scales for the four forests, used as $\omega_X = k_X/\sqrt{m_X}$ where m_X is the corresponding number of trees. Larger values give a more diffuse leaf prior (weaker shrinkage); smaller values give a more concentrated prior (stronger shrinkage). The defaults k_control = k_treat = k_deconf = k_deviation = 0.5 reproduce the standard $0.5/\sqrt{m}$ scaling used in BART/BCF. k_deviation is the deviation-forest scale (formerly k_g); setting k_deviation = 0 collapses the deviation prior to a point mass at zero and disables g entirely (full pooling).

power_control, base_control, power_deviation, base_deviation, power_treat, base_treat, power_deconf, base_deconf
: Tree-topology prior parameters, set separately for each of the four forests. The probability that a node at depth d is non-terminal is base$/(1+d)^{\text{power}}$ (larger power or smaller base gives shallower trees). There is no shared global power/base;

each forest carries its own pair and is set independently. Defaults follow METHODOLOGY.tex: the baseline μ (control) and deviation g use `(2, 0.95)`; the treatment effect τ uses `(3, 0.95)`; the confounding function c (deconf) uses `(3, 0.25)`.

`p_grow`, `p_prune` Probabilities of proposing a grow or prune move at each MCMC step.

`nu` Degrees of freedom for the inverse-chi-squared prior on σ^2.

`q` Quantile used to set the scale parameter λ of the error-variance prior.

`sigma` Optional fixed value for σ. If `NULL` (default), σ is estimated from the data and updated each MCMC iteration.

`N_post`, `N_burn` Number of posterior and burn-in MCMC iterations.

`treatment_coding`

Character; how the binary treatment indicator $Z \in \{0, 1\}$ is mapped to the regression weight b_i in the BCF parameterisation $y = \mu(X) + b\,\tau(X) + \ldots$ (and similarly for the deconfounding term). One of:

`"centered"` **(default)** $b = 0.5$ if treated, $b = -0.5$ if control. Both arms inform τ; μ is the average-arm mean function.

`"binary"` $b = 1$ if treated, $b = 0$ if control. Only treated rows inform τ; μ is the control-arm mean function.

`"adaptive"` $b_i = z_i - \pi_i$, the propensity-score residual (Hahn et al., 2020). Requires `propensity_train` and `propensity_test`.

The τ and deconfounding (c) forests are fit with per-observation weights $w_i = b_i^2$ so that the weighted sufficient statistics match the full-data likelihood; rows with $|b_i|$ essentially zero are excluded from those updates.

`propensity_train`, `propensity_test`

Numeric vectors of estimated propensity scores in (0, 1) for the training and test rows. Required when `treatment_coding = "adaptive"`; ignored otherwise.

`error_dist` Character; residual-distribution prior. One of the options below. See '`inst/error_distributions.md`' for a full reference (model statements, Gibbs steps, posterior storage, and a selection guide).

`"gaussian"` **(default)** Single normal residual, $\varepsilon_i \sim N(0, \sigma^2)$.

`"shared_dp"` Centred Dirichlet-process mixture of Gaussians on the residual, pooled across data sources (AFTrees-style; Henderson et al., 2020). Atoms are recentred to weighted mean zero to identify the structural mean components.

`"source_dp"` Independent centred Dirichlet-process mixtures per source (Stage A of the HDP-CDP plan). RCT and RWD get their own atoms, weights, and concentration; σ is shared.

`"source_dp_scale"` Same as `"source_dp"` but with per-source error scales σ_s. Each source gets its own inverse-gamma conjugate update for σ_s, and the per-source precision is propagated to the forest backfitting via per-observation precision weights.

`"source_hdp"` Hierarchical centred Dirichlet process (Stage B): atoms θ_k^* are shared across sources via a top-level $DP(\gamma, H)$; per-source weights π_{sk}

and means μ_s restore identifiability of the structural components. Concentration parameters γ, M_0, M_1 are updated via Escobar-West auxiliary-variable steps; the top-level sticks β use the Antoniak-Teh table-count augmentation.

`"source_hdp_scale"` Same as `"source_hdp"` but with per-source error scales σ_s. Label sampling and the precision-weighted shared-atom posterior use σ_s, and each σ_s is refreshed via the inverse-gamma conjugate step from `"source_dp_scale"`. Useful when the two sources are believed to share residual *shape* (atoms) but differ in *spread*.

`error_truncation_K`
: Integer; truncation level for the stick-breaking representation. Default 50. Increase if the number of occupied components approaches the bound during MCMC.

`error_atom_scale`
: Numeric; prior standard deviation for each atom on the standardised response scale, so each atom has prior $N(0, \texttt{error\_atom\_scale}^2)$. Default 0.5 (so atoms are expected within roughly ± 1 on the standardised scale).

`error_mass_init`
: Numeric; starting value for the concentration parameter α (or each M_s under `"source_dp"`). Updated by the sampler via an AFTrees-style Gamma conjugate step with fixed hyperprior $\mathrm{Gamma}(2, 0.1)$. Default 1.

`store_posterior_sample`
: Logical; if `TRUE`, the full $N_{\text{post}} \times n$ posterior sample matrices are returned for all component forests.

`verbose` Logical; print a progress bar and summary statistics.

Details

The model uses a MAP-prior four-forest decomposition:

$$\log(T) = \mu(X) + (1 - S)\, g(X) + b\,[\tau(X) + (1 - S)\, c(X)] + \varepsilon,$$

where $\mu(X)$ is a shared baseline fit to all data, $g(X)$ captures the RWD-specific deviation, and ε is a mean-zero error term (Gaussian or a Dirichlet-process mixture; see `error_dist`). Each forest has a Gaussian leaf prior $N(0, \omega^2)$ parameterised uniformly as

$$\omega_X \;=\; k_X / \sqrt{m_X},$$

where m_X is the number of trees in forest X and the user-tunable scale k_X controls how informative the prior is (smaller $k_X \Rightarrow$ stronger shrinkage toward zero). The deviation forest's scale `k_deviation` sets the strength of the MAP-prior borrowing between RCT and RWD.

Value

A named list with components:

train_predictions, test_predictions Posterior mean of the total fitted values.

train_predictions_control, test_predictions_control Posterior mean of the shared baseline $\mu(X)$.

train_predictions_treat, test_predictions_treat Posterior mean of the CATE $\tau(X)$.

train_predictions_deconf, test_predictions_deconf Posterior mean of the confounding function $c(X)$ (RWD rows only for training).

train_predictions_deviation, test_predictions_deviation Posterior mean of the RWD deviation $g(X)$ (RWD rows only for training).

sigma Posterior sample of σ (or the fixed value if `sigma` was supplied).

acceptance_ratio_control, acceptance_ratio_treat, acceptance_ratio_deconf Tree-update acceptance rates.

acceptance_ratio_deviation Acceptance rate for g.

train_predictions_sample_control, ... Full posterior sample matrices (only present when `store_posterior_sample = TRUE`).

Examples

```
# Small simulated fusion example combining an RCT and RWD
set.seed(1)
n <- 100
X <- matrix(rnorm(n * 3), n, 3)
s <- rbinom(n, 1, 0.5)  # 1 = RCT, 0 = RWD
a <- rbinom(n, 1, 0.5)  # treatment
y <- X[, 1] + 0.5 * a + rnorm(n)

fit <- FusionForest(
  y = y,
  X_train_control = X,
  X_train_treat = X,
  treatment_indicator_train = a,
  source_indicator_train = s,
  N_post = 50, N_burn = 25,
  verbose = FALSE
)
print(fit)
summary(fit)
```

`fusion_estimand` *Causal survival estimands from a FusionForest fit*

Description

Compute posterior draws of the survival difference or the acceleration factor (AF) at the covariate values supplied to `FusionForest()` as the test set, using the closed-form expressions of the AFT decomposition with (H)DP error.

Usage

```
fusion_estimand(
  fit,
  estimand = c("SD", "AF"),
  time = NULL,
  target_source = c("rwd", "rct"),
  population_average = FALSE,
  bayesian_bootstrap = TRUE,
  seed = NULL
)
```

Arguments

`fit`	A fitted `FusionForest` object obtained with `store_posterior_sample = TRUE`. Must include posterior sample matrices for all relevant component forests. When `estimand = "SD"`, the fit must use one of the supported `error_dist` families.
`estimand`	Character, one of `"SD"`, `"AF"`.
`time`	Numeric scalar (positive). Required for `"SD"`; ignored for `"AF"`. Interpreted on the *time* (not log-time) scale.
`target_source`	Character, `"rwd"` (default) or `"rct"`. Selects the target population s_t appearing in the survival formula (eqs.\ 2.6–2.8 of `estimands.tex`). Ignored for `"AF"` since the causal AF is shared across sources.
`population_average`	Logical; if `TRUE` the function returns posterior draws of the population-averaged estimand (a length R vector). Otherwise returns the $R \times n_{ev}$ matrix of per-x draws. Default `FALSE`.
`bayesian_bootstrap`	Logical; if `TRUE` (default) the population average uses Dirichlet weights drawn fresh at each iteration. If `FALSE` the average uses equal weights $1/n_{ev}$. Ignored when `population_average = FALSE`.
`seed`	Integer or `NULL`. Sets the RNG seed used to draw the Bayesian-bootstrap weights, for reproducibility.

Details

Under

$$\log T = m_0(X, S) + A\tau(X) + (1 - S)Ac(X) + \varepsilon,$$

with mean-zero error ε following a (source-specific) Gaussian or Dirichlet-process mixture distribution, the survival function under treatment a in target population s_t is

$$S_a(t|x, s_t) = 1 - F_{s_t}\big(\log t - \eta_a(x, s_t)\big),$$

where F_{s_t} is the error distribution function in source s_t, $\eta_a(x, s_t) = m_0(x, s_t) + a\tau(x) + (1 - s_t)ac(x)$ and $m_0(x, s_t) = \mu(x) + (1 - s_t)g(x)$. The estimands implemented here are

`"SD"` Survival difference $\Delta_{SD}(t; x, s_t) = S_1(t|x, s_t) - S_0(t|x, s_t)$.

`"AF"` Causal acceleration factor $\exp(\tau(x))$. Does not depend on the error distribution or on `target_source`.

Population-averaged versions use Bayesian-bootstrap weights drawn fresh at each MCMC iteration ($w^{(r)} \sim Dirichlet(1, \ldots, 1)$), as in Rubin (1981); set `bayesian_bootstrap = FALSE` for equal weights.

Value

If `population_average = FALSE`, an $R \times n_{ev}$ matrix of posterior draws of the estimand at each evaluation point (test rows of the fit). Otherwise a length-R numeric vector of population-averaged draws. In both cases the result carries attributes `estimand`, `time`, `target_source`, and (when applicable) `bayesian_bootstrap`.

References

Rubin, D.~B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, 9, 130–134.

Examples

```
# Right-censored survival fusion fit with stored posterior samples
set.seed(1)
n <- 100
X <- matrix(rnorm(n * 3), n, 3)
s <- rbinom(n, 1, 0.5)
a <- rbinom(n, 1, 0.5)
true_time <- exp(1 + X[, 1] + 0.5 * a + 0.3 * rnorm(n))
cens_time <- rexp(n, rate = 1 / (2 * mean(true_time)))
time   <- pmin(true_time, cens_time)
status <- as.integer(true_time <= cens_time)

fit <- FusionForest(
  y = time, status = status,
  X_train_control = X, X_train_treat = X,
  treatment_indicator_train = a, source_indicator_train = s,
  X_test_control = X, X_test_treat = X,
  treatment_indicator_test = a, source_indicator_test = s,
  outcome_type = "right-censored",
  N_post = 50, N_burn = 25,
  store_posterior_sample = TRUE, verbose = FALSE
)

# Posterior draws of the acceleration factor at each test point
af <- fusion_estimand(fit, estimand = "AF")
quantile(colMeans(af), c(0.25, 0.5, 0.75))
```

`fusion_projection` *Posterior linear projection of the CATE or acceleration factor*

Description

Projects each posterior draw of the heterogeneous treatment effect onto a user-supplied linear basis via (weighted) least squares. The resulting $R \times k$ collection of coefficient vectors is a sample from the posterior of the projection, i.e.\ the pushforward of $\tau(\cdot)$ (or $\exp \tau(\cdot)$) under the projection map of Woody, Carvalho and Murray (2020). Uncertainty is inherited exactly from the original posterior without refitting.

Usage

```
fusion_projection(
  fit,
  basis,
  X_eval,
  target = c("cate", "af"),
  af_scale = c("multiplicative", "log"),
  which = c("test", "train"),
  weights = c("uniform", "bayesian_bootstrap"),
  center = TRUE,
  scale = FALSE,
  seed = NULL
)
```

Arguments

`fit`	A fitted `FusionForest` object with `store_posterior_sample = TRUE`.
`basis`	One-sided formula defining the projection basis, e.g.\ `~ x1 + x2 + x1:x2`. Built into a design matrix via `stats::model.matrix`.
`X_eval`	Data frame of evaluation covariates. Must have one row per posterior-sample column of the chosen `which` set (i.e.\ the same rows passed as `X_test_*` or `X_train_*` when fitting). Column names must match the variables referenced in `basis`.
`target`	Character; `"cate"` projects $\tau(x)$ on the AFT log-time scale, `"af"` projects the acceleration factor $\exp \tau(x)$ or its log (see `af_scale`).
`af_scale`	Only used when `target = "af"`. `"multiplicative"` (default) projects $\exp \tau$ directly, so γ_k is the change in the AF per unit of covariate k. `"log"` projects τ (equivalent to `target = "cate"`; kept as a labelled alias).
`which`	Which evaluation rows to use, `"test"` (default) or `"train"`. Determines the implicit target population of the summary.
`weights`	`"uniform"` (default) or `"bayesian_bootstrap"`. The latter draws fresh Dirichlet weights at each MCMC iteration.
`center, scale`	Logical; centre and/or scale the non-intercept columns of Φ before the projection. Defaults: `TRUE`, `FALSE`.
`seed`	Integer or `NULL`. Sets the RNG seed used for the Bayesian-bootstrap weights, for reproducibility.

Details

At iteration r the coefficient vector solves

$$\gamma^{(r)} = \arg\min_{\gamma} \sum_i w_i^{(r)} \big(\theta^{(r)}(x_i) - \phi(x_i)^\top \gamma\big)^2,$$

where $\theta^{(r)}$ is one of $\tau^{(r)}$ (`target = "cate"`), $\exp \tau^{(r)}$ (`target = "af"`, `af_scale = "multiplicative"`), or $\tau^{(r)}$ again (`target = "af"`, `af_scale = "log"`; kept as an alias for clarity).

With `weights = "uniform"` the QR factorisation of Φ is computed once and reused. With `weights = "bayesian_bootstrap"` fresh Dirichlet weights $w^{(r)} \sim \mathrm{Dirichlet}(1, \ldots, 1)$ are drawn at each iteration (Rubin, 1981), injecting covariate-distribution uncertainty into the projection.

Columns of Φ other than the intercept are centred (and optionally scaled) once, on `X_eval`, before the projection. The centring and scaling vectors are returned as attributes so coefficients can be mapped back to original units.

Value

An $R \times k$ numeric matrix of posterior draws of the projection coefficients, one row per MCMC iteration, with `colnames` from `model.matrix`. Attributes: `target`, `af_scale`, `which`, `weights`, `center` and `scale` vectors (`NULL` if not applied), and the original `formula`.

References

Woody, S., Carvalho, C.~M., and Murray, J.~S. (2020). Bayesian inference with posterior projection. *Journal of Computational and Graphical Statistics*, 29(4), 798–808.

Rubin, D.~B. (1981). The Bayesian bootstrap. *The Annals of Statistics*, 9, 130–134.

Examples

```
# Continuous fusion fit with stored posterior samples
set.seed(1)
n <- 100
X <- matrix(rnorm(n * 3), n, 3)
colnames(X) <- paste0("x", 1:3)
s <- rbinom(n, 1, 0.5)
a <- rbinom(n, 1, 0.5)
y <- X[, 1] + (0.5 + 0.3 * X[, 2]) * a + rnorm(n)

fit <- FusionForest(
  y = y,
  X_train_control = X, X_train_treat = X,
  treatment_indicator_train = a, source_indicator_train = s,
  X_test_control = X, X_test_treat = X,
  treatment_indicator_test = a, source_indicator_test = s,
  N_post = 50, N_burn = 25,
  store_posterior_sample = TRUE, verbose = FALSE
)

# Project the posterior CATE surface onto a linear basis
proj <- fusion_projection(fit, basis = ~ x1 + x2 + x3,
```

```
                          X_eval = as.data.frame(X))
colMeans(proj)
```

`print.FusionForest` *Print a FusionForest fit*

Description

Displays a compact overview of a fitted `FusionForest()` model: outcome type, forest decomposition, error model, sample sizes and MCMC settings.

Usage

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

Arguments

`x`	A `FusionForest` object.
`...`	Ignored.

Value

`x`, invisibly.

`print.summary.FusionForest`
Print a FusionForest summary

Description

Print a FusionForest summary

Usage

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

Arguments

`x`	A `summary.FusionForest` object.
`digits`	Number of significant digits to print.
`...`	Ignored.

Value

`x`, invisibly.

SimpleBART *SimpleBART*

Description

Standard single-forest BART model: $Y = f(X) + \varepsilon$. A clean entry point for benchmarking and development (e.g., testing IRS).

Usage

```
SimpleBART(
  y,
  X_train,
  X_test = NULL,
  number_of_trees = 200,
  power = 2,
  base = 0.95,
  p_grow = 0.4,
  p_prune = 0.4,
  nu = 3,
  q = 0.9,
  sigma = NULL,
  N_post = 1000,
  N_burn = 1000,
  verbose = TRUE,
  irs = 0L,
  store_posterior_sample = FALSE
)
```

Arguments

y	Numeric vector of outcomes (length n).
X_train	Numeric matrix of training covariates (n x p).
X_test	Optional numeric matrix of test covariates (n_test x p). If NULL, test predictions are returned at the column means.
number_of_trees	Number of trees in the BART ensemble (default 200).
power, base	Tree topology prior parameters.
p_grow, p_prune	Probabilities of grow / prune proposals.
nu	Degrees of freedom for the inverse-chi-squared prior on sigma^2.
q	Quantile for setting the scale parameter lambda.
sigma	Optional fixed sigma. If NULL, sigma is estimated.
N_post, N_burn	Number of posterior / burn-in MCMC iterations.
verbose	Logical; print progress bar.

`irs` Integer IRS mode: 0 = off (default), 1 = skip-then-draw (NaN obs excluded from MH ratio, routed after acceptance), 2 = draw-then-decide (routing drawn before MH, all obs in ratio), 3 = uniform random routing (P=0.5, ablation baseline). Modes 4-6 = same as 1-3 but NaN-routed observations are excluded from the leaf mean (mu) posterior draw.

`store_posterior_sample` Logical; if `TRUE`, return the full `N_post` x `n_test` matrix of posterior test predictions (element `test_predictions_sample`).

Value

A named list with:

train_predictions Posterior mean fitted values (length n).

test_predictions Posterior mean test predictions (length n_test).

sigma Posterior samples of sigma (after burn-in).

acceptance_ratio Mean tree-proposal acceptance rate.

train_predictions_sample (If `store_posterior_sample` = `TRUE`) Matrix of dimension `N_post` x `n` with per-iteration train predictions.

test_predictions_sample (If `store_posterior_sample` = `TRUE`) Matrix of dimension `N_post` x `n_test` with per-iteration test predictions.

Examples

```
set.seed(1)
n <- 100
X <- matrix(rnorm(n * 3), n, 3)
y <- X[, 1] + rnorm(n)
fit <- SimpleBART(y = y, X_train = X, N_post = 50, N_burn = 25,
                  verbose = FALSE)
cor(y, fit$train_predictions)
```

`SimpleBCF` *SimpleBCF*

Description

Two-forest Bayesian Causal Forest: $Y = \mu(X, e) + \tau(X) \cdot A + \varepsilon$, where μ is a prognostic forest (optionally including the propensity score e) and τ is a treatment effect forest. Both forests use a standard BART prior.

Usage

```
SimpleBCF(
  y,
  X_train,
  treatment_indicator,
  propensity_score = NULL,
  X_test = NULL,
  treatment_indicator_test = NULL,
  propensity_score_test = NULL,
  number_of_trees_prog = 200,
  number_of_trees_treat = 50,
  power = 2,
  base = 0.95,
  p_grow = 0.4,
  p_prune = 0.4,
  nu = 3,
  q = 0.9,
  sigma = NULL,
  N_post = 1000,
  N_burn = 1000,
  verbose = TRUE,
  irs = 0L,
  store_posterior_sample = FALSE
)
```

Arguments

`y`	Numeric vector of outcomes (length n).
`X_train`	Numeric matrix of covariates (n x p).
`treatment_indicator`	Integer vector of treatment assignments (0/1, length n).
`propensity_score`	Optional numeric vector of estimated propensity scores (length n). Appended as an extra covariate to the prognostic forest.
`X_test`	Optional test covariate matrix (n_test x p).
`treatment_indicator_test`	Integer vector (0/1) for test. Defaults to all-ones if `NULL`.
`propensity_score_test`	Optional propensity scores for the test set (length n_test).
`number_of_trees_prog`, `number_of_trees_treat`	Number of trees in the prognostic / treatment forest.
`power`, `base`	Tree topology prior parameters.
`p_grow`, `p_prune`	Probabilities of grow / prune proposals.
`nu`	Degrees of freedom for the inverse-chi-squared prior on sigma^2.
`q`	Quantile for setting the scale parameter lambda.

`sigma`	Optional fixed sigma. If `NULL`, sigma is estimated.
`N_post`, `N_burn`	Number of posterior / burn-in MCMC iterations.
`verbose`	Logical; print progress bar.
`irs`	Integer IRS mode (applied to both forests): 0 = off, 1 = skip-then-draw, 2 = draw-then-decide, 3 = uniform random routing.
`store_posterior_sample`	Logical; if `TRUE`, return the full posterior sample matrices for test predictions.

Value

A named list with:

train_predictions Posterior mean of mu(x,e) + b * tau(x) (length n).

test_predictions Posterior mean of mu(x,e) + b * tau(x) (length n_test).

train_predictions_prog Posterior mean of mu(x,e).

test_predictions_prog Posterior mean of mu(x,e).

train_predictions_treat Posterior mean of tau(x) (CATE estimates on training set).

test_predictions_treat Posterior mean of tau(x) (CATE estimates on test set).

sigma Posterior samples of sigma.

acceptance_ratio_prog Acceptance rate (prognostic).

acceptance_ratio_treat Acceptance rate (treatment).

test_predictions_treat_sample (If `store_posterior_sample`) N_post x n_test matrix of per-iteration tau(x) predictions.

test_predictions_sample (If `store_posterior_sample`) N_post x n_test matrix of per-iteration total predictions.

Examples

```
set.seed(1)
n <- 100
X <- matrix(rnorm(n * 3), n, 3)
a <- rbinom(n, 1, 0.5)
y <- X[, 1] + 0.5 * a + rnorm(n)
fit <- SimpleBCF(y = y, X_train = X, treatment_indicator = a,
                 N_post = 50, N_burn = 25, verbose = FALSE)
mean(fit$train_predictions_treat)  # average estimated CATE
```

summary.FusionForest *Summarise a FusionForest fit*

Description

Computes posterior summaries of a fitted `FusionForest()` model: the residual standard deviation and the distribution of the estimated individual treatment effects (posterior means of the treatment forest evaluated at the training covariates).

Usage

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

Arguments

`object`	A `FusionForest` object.
`...`	Ignored.

Value

An object of class `summary.FusionForest`: a list with elements `meta`, `sigma` (posterior draws of the residual SD on the standardised scale, `NULL` if `sigma` was fixed), `treatment_effects` (posterior-mean treatment forest predictions for the training data) and `acceptance_ratios`.

Index