Package: PRTree (via r-universe)

October 6, 2026

Type Package

Date 2026-07-13

Title Probabilistic Regression Trees

Version 1.1.0

Depends R (>= 4.3.0)

Suggests ggplot2

Imports tidyr (>= 1.3.0), gridExtra (>= 2.3), stats, grDevices, utils, tidyselect, graphics, rlang

Description Implementation of Probabilistic Regression Trees (PRTree), providing functions for model fitting and prediction, with specific adaptations to handle missing values. The main computations are implemented in 'Fortran' for high efficiency. The package is based on the PRTree methodology described in Alkhoury et al. (2020), ``Smooth and Consistent Probabilistic Regression Trees'' <https://proceedings.neurips.cc/paper_files/paper/2020/file/8289889263db4a40463e3f358bb7c7a1-Paper.pdf>. Details on the treatment of missing data and implementation aspects are presented in Prass, T.S.; Neimaier, A.S.; Pumi, G. (2025), ``Handling Missing Data in Probabilistic Regression Trees: Methods and Implementation in R'' <doi:10.48550/arXiv.2510.03634>.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

Collate 'prtree.R' 'prtree_control.R' 'prtree_control_utils.R' 'prtree_cv.R' 'prtree_cv_methods.R' 'prtree_rules.R' 'prtree_defaults.R' 'prtree_grid_expansion.R' 'prtree_main.R' 'prtree_main_methods.R' 'prtree_messages.R' 'prtree_predict.R' 'prtree_split.R' 'prtree_validation.R'

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Author Taiane Schaedler Prass [aut, ths, cre] (ORCID: <https://orcid.org/0000-0003-3136-909X>), Alisson Silva Neimaier [aut] (ORCID: <https://orcid.org/0000-0002-7524-0776>)

Maintainer Taiane Schaedler Prass <taianeprass@gmail.com>

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

Date/Publication 2026-07-13 20:30:10 UTC

RemoteUrl https://github.com/cran/PRTree

RemoteRef HEAD

RemoteSha f22feb5343606b54f56b456871a30d707ae322e3

Contents

expand_sigma_grid . . . 2
plot.prtree . . . 4
plot.prtree.cv . . . 5
plot_tree . . . 6
pr_tree . . . 7
pr_tree_control . . . 9
pr_tree_control_cv . . . 13
pr_tree_cv . . . 15
predict.prtree . . . 17
print.idx.split . . . 18
print.prtree . . . 18
print.prtree.control . . . 19
print.prtree.control_cv . . . 19
print.prtree.cv . . . 20
print.sigma_grid . . . 20
print.summary.idx.split . . . 21
print.summary.prtree . . . 21
print.summary.prtree.cv . . . 22
summary.idx.split . . . 22
summary.prtree . . . 23
summary.prtree.cv . . . 23
train_test_split . . . 24

Index **27**

`expand_sigma_grid` *Create a grid of sigma vectors for Probabilistic Regression Trees*

Description

Creates a grid of candidate $\boldsymbol{\sigma}$ vectors for use in pr_tree. The grid is generated from the feature standard deviations of `X`, or using unit scales when `X = NULL`

Usage

```
expand_sigma_grid(X = NULL, n_feat = NULL, idx_train = NULL,
  grid_size = 8, min_mult = 0, max_mult = 2, tiny_sigma = NULL,
  verbose = TRUE)
```

Arguments

X	A numeric vector, matrix or data frame of predictor variables. If NULL, the grid is generated using unit scales (all features = 1.0). In this case, n_feat must be provided.
n_feat	Optional integer. Number of features. Required when X = NULL. Ignored when X is provided.
idx_train	Optional integer vector. Indices of observations to use for computing feature standard deviations. If NULL (default), all observations in X are used. Ignored when X = NULL.
grid_size	Integer. Number of candidate vectors to generate when sigma_grid = NULL. Default is 8. See 'Details' for how multipliers are generated. Ignored when a custom grid is provided.
min_mult	Numeric. Minimum multiplier for automatic grid generation. Default is 0. See 'Details' for how this defines the multiplier interval. Ignored when a custom grid is provided.
max_mult	Numeric. Maximum multiplier for automatic grid generation. Default is 2. Ignored when a custom grid is provided.
tiny_sigma	A single numeric value or NULL. If provided, it is added to the grid as an additional candidate representing a very small σ value (same value for all features). This approximates the limiting case $\boldsymbol{\sigma} \to \mathbf{0}$, where the smooth assignment function Ψ converges to the hard indicator function, recovering deterministic splits. A typical value is 1e-20. See 'Details' for its role in grid generation. Default is NULL. Ignored when a custom grid is provided.
verbose	Logical. If TRUE (default), prints progress and informational messages. If FALSE, suppresses all non-error console output.

Details

When X is provided, the grid is based on the empirical standard deviations of the features. If a standard deviation for a feature cannot be computed (e.g., due to all missing values or a constant feature), it defaults to 1.0 to ensure the grid generation proceeds without errors.

Automatic grid generation

The automatic grid is generated as follows:

- For each feature j, the base values are the empirical standard deviations $\hat{\sigma}_j$ from the training data.
- If grid_size = 1, the grid consists solely of these base values $\hat{\boldsymbol{\sigma}}$.
- If grid_size > 1, the grid is constructed using grid_size multipliers equally spaced in the interval:

- `(min_mult, max_mult]` when `min_mult = 0` (excludes 0, includes `max_mult`).
- `[min_mult, max_mult]` when `min_mult > 0` (both inclusive).

Each multiplier m generates a candidate vector $m \times \hat{\boldsymbol{\sigma}}$.

- If `tiny_sigma` is provided, it is added as an extra candidate (first row) with $\boldsymbol{\sigma} = \text{tiny_sigma} \cdot \mathbf{1}_p$, where $\mathbf{1}_p$ is a vector of ones of length p (the number of features). This results in a final grid of size `grid_size + 1`.

See Also

pr_tree_control for setting control parameters.

pr_tree_control_cv for cross-validation settings.

pr_tree for fitting a single tree.

pr_tree_cv for cross-validation.

Examples

```
# With actual data, default range (0, 2]
X <- matrix(runif(100 * 3, 0, 10), ncol = 3)
grid1 <- expand_sigma_grid(X, grid_size = 8)

# Custom range with min_mult > 0 (includes min_mult)
grid2 <- expand_sigma_grid(X, grid_size = 8, min_mult = 0.5, max_mult = 3)

# Without data (unit scales)
grid3 <- expand_sigma_grid(n_feat = 3, grid_size = 8)

# Only tiny sigma
grid4 <- expand_sigma_grid(n_feat = 3, grid_size = 0, tiny_sigma = 1e-20)
```

---

plot.prtree *Plot method for PRTree objects*

Description

Provides visualization of a fitted Probabilistic Regression Tree model. Note that the original response vector `y` is only required for plots 1-4. Plot 5 (terminal node probabilities) can be generated without `y`.

Usage

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

Arguments

x	An object of class prtree, as returned by pr_tree.
y	Original response vector used to fit the model. Required only for plots 1-4 (Fitted vs Observed, Residuals, Histogram, Q-Q plot). Can be omitted if only plot 5 is requested.
which	Which plots to produce. Options are: • 1: **Ovserved vs Fitted** - Scatter plot comparing observe responses against fitted values. Requires y. • 2: **Residuals (index plot)** - Residuals plotted against observation index. Useful for detecting patterns or outliers. Requires y. • 3: **Histogram of residuals** - Distribution of residuals with Freedman-Diaconis bin selection. Requires y. • 4: **Q-Q plot of residuals** - Quantile-quantile plot to assess normality of residuals. Requires y. • 5: **Terminal node probabilities** - Boxplot of $P(R\|X)$ for each terminal node. Does not require y. If NULL (default), shows all plots (1-5).
ncol	Integer. Number of columns in the plot layout. If NULL (default), an appropriate number of columns is chosen automatically based on the number of plots.
...	Further arguments passed to ggplot2 (e.g., color, size, alpha).

Value

Invisibly returns the original object.

---

plot.prtree.cv *Plot method for PRTree Cross-Validation objects*

Description

Provides visualizations for cross-validation results of Probabilistic Regression Trees.

Usage

```
## S3 method for class 'prtree.cv'
plot(x, which = NULL, ncol = NULL, ...)
```

Arguments

x	An object of class prtree.cv, as returned by pr_tree_cv.
which	Which plots to produce. Options are: • 1: **RMSE across iterations/folds** - Shows validation and test RMSE for each iteration/fold.

\item `2`: \strong{Selected sigma values} - Values of selected sigma
across iterations/folds. Colors distinguish features.

\item `3`: \strong{Distribution of sigma} - For single feature: histogram;
for multiple features: boxplots colored by feature.

\item `4`: \strong{Sigma vs Validation RMSE} - Scatter plot with LOESS smooth
to show relationship between selected sigma and validation RMSE.

If NULL (default), shows all applicable plots.

ncol	Integer. Number of columns in the plot layout. If NULL (default), an appropriate number of columns is chosen automatically.
...	Further arguments passed to ggplot2 (e.g., size, alpha).

Value

Invisibly returns the original object.

---

plot_tree *Plot a PRTree as a dendrogram with gamma and P distributions*

Description

Draws a tree representation of a fitted PRTree model with root at top. Terminal nodes show gamma coefficients and a mini heatmap of the distribution of the corresponding column of the probability matrix P.

Usage

```
plot_tree(x, heatmap_width = 0.8, heatmap_height = 0.08, max_bins = 6,
  ...)
```

Arguments

x	An object of class prtree.
heatmap_width	Numeric. Width of heatmaps as fraction of node spacing. Default = 0.8 (80% of spacing).
heatmap_height	Numeric. Height of heatmaps. Default = 0.08.
max_bins	Integer. Maximum number of bins in the heatmap. Default = 6.
...	Further arguments passed to plot.

Value

Invisibly returns the original object.

pr_tree *Probabilistic Regression Trees (PRTrees)*

Description

Fits a Probabilistic Regression Tree (PRTree) model. This is the main user-facing function of the package.

Usage

```
pr_tree(y, X, control = list(), ...)
```

Arguments

y	A numeric vector for the dependent variable.
X	A strictly numeric matrix or data frame for the independent variables. Categorical variables or factors must be encoded (e.g., one-hot encoding) prior to model fitting.
control	A list of control parameters, typically created by pr_tree_control. Default values are taken from pr_tree_control for any parameters not specified. Alternatively, control parameters can be passed directly via the ... argument.
...	Control parameters to be passed to pr_tree_control. These will override any parameters specified in the control list.

Details

The tree is built using the training data specified by idx_train (or determined by prop_test). If validation data is available (the complement of idx_train), it is used to select the optimal σ values from the grid.

Grid generation:

- If sigma_grid is not provided, it is automatically generated using **only the training set** (the observations identified by idx_train, which can be passed via control or calculated internally from prop_test). The validation/test data is used only for evaluation.
- Alternatively, users can create a custom grid using the helper function expand_sigma_grid(X, ...) and pass it via the control parameter. This allows using the full dataset (or any desired subset) to define the grid of σ values to be tested.

Important: The returned model is trained **only on the training set**, not on the combined training + validation data. This matches standard practice where validation data is used only for hyperparameter selection, not for final parameter estimation.

If you want to retrain the model using the selected σ values on a larger dataset (e.g., combining training and validation), you can:

1. Extract the optimal σ from the fitted model (model$sigma)
2. Create a new control object with sigma_grid = matrix(model$sigma, nrow = 1) and set prop_test = 0 to use all data for training
3. Run pr_tree again

Value

An object of class `prtree` containing the fitted model. This is a list with the following components

`n_obs`	Number of observations used in the model.
`n_feat`	Number of features (predictor variables) in the model.
`features`	The features names: either `colnames(X)` or a vector with generic names `X1,...,Xn_feat`
`yhat`	The estimated values for `y`.
`XRegion`	A matrix with two columns indicating the terminal node (region) each observation belongs to. The first column (`TRUE`) may have `NA` for observations with missing values. The second column (`Internal`) shows the region assigned by the algorithm.
`dist`	A list with distribution information: • `dist_name`: Name of the distribution ("norm", "lnorm", "t", "gamma"). • `dist_code`: Integer code used by Fortran (1-4). • `par_name`: Name of the distribution parameter (e.g., "df", "shape"). • `<par_name>`: Value of the distribution parameter.
`fill_type`	Fortran code corresponding to the method used to fill the matrix P when missing values are present.
`P`	The matrix of probabilities for each terminal node.
`gamma`	The values of the γ_j weights estimated for the returned tree
`mse`	The mean squared error for the training, test/validation, and global datasets.
`sigma`	The optimal $\boldsymbol{\sigma}$ vector selected by the grid search.
`nodes_matrix_info`	A matrix with information for each node of the tree.
`regions`	A data frame with the bounds of each variable in each node of the returned tree.

See Also

pr_tree_cv for cross-validation options.

pr_tree_control for setting control parameters.

expand_sigma_grid for creating custom sigma grids.

Examples

```
set.seed(1234)
X <- matrix(runif(200, 0, 10), ncol = 1)
eps <- matrix(rnorm(200, 0, 0.05), ncol = 1)
y <- cos(X) + eps

# Fit model
reg <- pr_tree(y, X, max_terminal_nodes = 9, prop_hold = 0)

# Visualize fit (X is 1D, so we can plot the relationship)
ord <- order(X)
```

```
plot(X[ord], reg$yhat[ord],
  type = "l", col = "blue", lwd = 2,
  xlab = "x", ylab = "y",
  main = "PRTree Fit to cos(x)"
)
points(X[ord], y[ord], col = "red", cex = 0.5)
legend("topright",
  legend = c("Fitted", "Observed"),
  col = c("blue", "red"), lty = c(1, NA), pch = c(NA, 1)
)

# Diagnostic plots
par(mar = c(3, 4, 1.5, 1))
plot(reg, y, which = c(1, 4, 2, 5), ncol = 2)

# Plotting the final tree
plot_tree(reg, max_bins = 10)
```

`pr_tree_control` *Control parameters for PRTree*

Description

This function creates a list of control parameters for the `pr_tree` function, with validation for each parameter.

Usage

```
pr_tree_control(sigma_grid = NULL, grid_size = 8, min_mult = 0,
  max_mult = 2, tiny_sigma = NULL, max_terminal_nodes = 15L,
  max_depth = max_terminal_nodes - 1, cp = 0.01, n_min = 5L,
  prop_x = 0.1, p_min = 0.05, prop_hold = 0.2, idx_train = NULL,
  fill_type = 2L, proxy_crit = "both", n_candidates = 3L,
  by_node = FALSE, dist = "norm", iprint = -1, verbose = TRUE, ...)
```

Arguments

`sigma_grid`	Optional. Custom grid of candidate $\boldsymbol{\sigma}$ vectors. See 'Details' for accepted formats and behavior. Default is `NULL` (automatic generation). **Note:** The dimensions of `sigma_grid` are not validated inside pr_tree_control. Dimension checking against the number of features is performed internally during the execution of `pr_tree` or `pr_tree_cv`.
`grid_size`	Integer. Number of candidate vectors to generate when `sigma_grid = NULL`. Default is 8. See 'Details' for how multipliers are generated. Ignored when a custom grid is provided.
`min_mult`	Numeric. Minimum multiplier for automatic grid generation. Default is 0. See 'Details' for how this defines the multiplier interval. Ignored when a custom grid is provided.

`max_mult` Numeric. Maximum multiplier for automatic grid generation. Default is 2. Ignored when a custom grid is provided.

`tiny_sigma` A single numeric value or `NULL`. If provided, it is added to the grid as an additional candidate representing a very small σ value (same value for all features). This approximates the limiting case $\boldsymbol{\sigma} \to \mathbf{0}$, where the smooth assignment function Ψ converges to the hard indicator function, recovering deterministic splits. A typical value is `1e-20`. See 'Details' for its role in grid generation. Default is `NULL`. Ignored when a custom grid is provided.

`max_terminal_nodes` A non-negative integer. The maximum number of terminal nodes (regions/leaves) in the output tree. The default is 15.

`max_depth` A non-negative integer. The maximum depth of the decision tree. The depth is defined as the length of the longest path from the root to a leaf. The default is `max_terminal_nodes - 1`.

`cp` A positive numeric value. The complexity parameter. Any split that does not decrease the MSE (mean square error) by a factor of at least `cp` will be ignored. The default is 0.01.

`n_min` A positive integer. The minimum number of observations required in a terminal (final) node. While Probabilistic Regression Trees can theoretically handle empty regions, this restriction is enforced to ensure the stable and well-posed calculation of the region's cut points (thresholds) based on empirical data. The default is 5.

`prop_x` A numeric value between 0 and 1. The minimum proportion of observations in a region that must have a probability greater than `p_min` to attempt a split. Default is 0.1. See 'Details' for more information.

`p_min` A numeric value between 0 and 1. The threshold probability used in conjunction with `prop_x` to evaluate if a region contains enough probability mass to be split. Default is 0.05. See 'Details'.

`prop_hold` A numeric value between 0 (inclusive) and 1 (exclusive) that specifies the proportion of the data to be held out for model validation or testing. Default is `0.2`. Ignored if `idx_train` is provided.

`idx_train` Optional integer vector specifying which observations to use as the training sample. Default is `NULL`. Overrides `prop_hold`.

`fill_type` Integer indicating the method to be used to fill the probability matrix when `X` has NA's. Default is 2.

- `0`: uniform (same probability for both child nodes).
- `1`: attributes all probability to the child node that is compatible with the observed values.
- `2`: computes the probability restricted to the observed entries.

`proxy_crit` Character. Default is `"both"`. Criterion used to associate an observation with missing values to a region:

- `"mean"`: maximizes the difference in means after a split.
- `"var"`: maximizes the variability between nodes.
- `"both"`: combines the `"mean"` and `"var"` criteria.

`n_candidates` Integer. The number of competing candidates to consider when searching for the best split. To select the candidates, a proxy improvement measure is used. Then a full analysis is performed to choose the best among the `n_candidates` candidates. Higher values increase accuracy but significantly extend computation time due to the full analysis being performed on more splits. Default is 3.

`by_node` Logical. If `TRUE`, the algorithm selects `n_candidates` for each node and then makes a full analysis to choose the best among all nodes. Otherwise the `n_candidates` are selected globally. Default is `FALSE`.

`dist` Character. The distribution to be used in the model. One of `"norm"` (Gaussian), `"lnorm"` (log-normal), `"t"` (Student's t), or `"gamma"` (Gamma). Default is `"norm"`.

`iprint` Integer. Controls the verbosity of the Fortran backend. Default is -1 (silent).

- `iprint < 0`: No printing.
- `iprint = 0`: Prints basic information.
- `iprint > 0`: As for `iprint = 0` plus progress reports.

`verbose` Logical. If `TRUE` (default), prints progress and informational messages. If `FALSE`, suppresses all non-error console output.

`...` Extra parameters to be passed to the chosen distribution.

- `"norm"`: Uses the standard Gaussian distribution. No extra parameters required.
- `"lnorm"`: Uses the log-normal distribution with `meanlog = 0`. Requires `sdlog`.
- `"t"`: Uses the t distribution. Requires `df`.
- `"gamma"`: Uses the gamma distribution with `scale = 1`. Requires `shape`.

Details

Custom grid specification

When a custom grid is provided via `sigma_grid`, it must follow these rules:

- A **single number** is expanded to a constant vector (same value for all features).
- A **numeric vector** of length p (the number of features) is used directly as one candidate vector.
- A $k \times p$ **numeric matrix** provides k candidates, one per row.

In all custom grid cases, the automatic grid parameters (`grid_size`, `min_mult`, `max_mult`, `tiny_sigma`) are ignored.

Automatic grid generation

The automatic grid is generated as follows:

- For each feature j, the base values are the empirical standard deviations $\hat{\sigma}_j$ from the training data.
- If `grid_size = 1`, the grid consists solely of these base values $\hat{\boldsymbol{\sigma}}$.
- If `grid_size > 1`, the grid is constructed using `grid_size` multipliers equally spaced in the interval:

  - `(min_mult, max_mult]` when `min_mult = 0` (excludes 0, includes `max_mult`).
  - `[min_mult, max_mult]` when `min_mult > 0` (both inclusive).

  Each multiplier m generates a candidate vector $m \times \hat{\boldsymbol{\sigma}}$.
- If `tiny_sigma` is provided, it is added as an extra candidate (first row) with $\boldsymbol{\sigma} = \text{tiny_sigma} \cdot \mathbf{1}_p$, where $\mathbf{1}_p$ is a vector of ones of length p (the number of features). This results in a final grid of size `grid_size + 1`.

Splitting Constraints and Numerical Stability

The parameters `prop_x` and `p_min` work together to control the splitting process. A splitting attempt is made in a given region only when at least a proportion `prop_x` of the training rows have a probability greater than `p_min` in that region's column of the probability matrix P. The split will be ignored if any of the resulting child regions do not meet this same criterion.

Warning: Setting either parameter to zero may lead to near-zero columns in P, making the matrix computationally singular or ill-conditioned for inversion.

Data Splitting Strategy The package uses a flexible data splitting strategy controlled by the `prop_hold` and `idx_train` arguments:

- **Automatic Split (`idx_train = NULL`):** The training indices are computed automatically based on `prop_hold`. The split uses stratified sampling to ensure that the proportion of observations with missing values is preserved across the training and validation/test sets.
- **Manual Split (`idx_train` provided):** The supplied indices are used directly, and `prop_hold` is ignored. The vector must be a valid subset of `1:nrow(X)`. This approach is useful for reusing specific splits from previous analyses, performing custom cross-validation, or ensuring identical training sets across different models.

Role of the Hold-out Set When `prop_hold > 0` and `idx_train` is `NULL`, the exact role of the held-out data depends on the grid search parameters (`sigma_grid` or `grid_size`):

- **Multiple candidates (`grid_size > 1`):** The hold-out data acts as a **validation set** to select the best $\boldsymbol{\sigma}$. If `prop_hold = 0`, the selection is based on the in-sample MSE of the training data.
- **Single candidate (`grid_size = 1`):** The hold-out data acts as a **test set** to evaluate the final model. If `prop_hold = 0`, the entire dataset is used for training without an independent test evaluation.

Value

A list of class `prtree.control` containing the validated control parameters.

Examples

```
# Get default control parameters
controls <- pr_tree_control()

# Customize some parameters
ctrl1 <- pr_tree_control(max_depth = 5, n_candidates = 5)

# equivalent calls
ctrl2.v1 <- pr_tree_control(dist = "t", df = 4)
ctrl2.v2 <- pr_tree_control(dist = "t", dist_pars = list(df = 4))
```

`pr_tree_control_cv` *Control parameters for PRTree with optional cross-validation*

Description

Creates a list of control parameters for Probabilistic Regression Trees. The base function pr_tree_control returns parameters for tree construction. The extended function pr_tree_control_cv returns both tree parameters and cross-validation settings.

Usage

```
pr_tree_control_cv(method = "montecarlo", n_rep = 10, only_sigma = FALSE,
  prop_test = 0.2, prop_valid = 0.2, stratify = FALSE,
  update_final = TRUE, fold_idx = NULL, ...)
```

Arguments

`method`	Character. Cross-validation method: either `"kfold"` or `"montecarlo"`. Default is `"montecarlo"`.
`n_rep`	Integer. Number of folds for k-fold CV or number of iterations for Monte Carlo CV. Default is 10.
`only_sigma`	Logical. If `TRUE`, performs only $\boldsymbol{\sigma}$ estimation without estimating test error. Default is `FALSE`. • When `sigma_grid` has multiple rows (or is `NULL`): `only_sigma = TRUE` selects $\boldsymbol{\sigma}$ but does not use a test set. `only_sigma = FALSE` performs both $\boldsymbol{\sigma}$ selection and test error estimation. • When `sigma_grid` has a single row (fixed sigma): `only_sigma` is automatically set to `FALSE`, as only test error estimation is meaningful.
`prop_test`	Numeric between 0 and 1. Proportion of data for the test set in Monte Carlo CV. Default is 0.2. • For `method = "montecarlo"`: Defines the proportion for the test set. If `0`, no test set is used (useful for $\boldsymbol{\sigma}$ estimation only). Ignored if `only_sigma = TRUE`. • For `method = "kfold"`: This parameter is ignored. The test set is defined by the folds. To perform only $\boldsymbol{\sigma}$ estimation with k-fold, set `only_sigma = TRUE`.
`prop_valid`	Numeric between 0 and 1. Proportion of the training set to use for validation ($\boldsymbol{\sigma}$ selection). Default is 0.2. • If `> 0`: the training set is split into training (`1 - prop_valid`) and validation (`prop_valid`) sets. The optimal $\boldsymbol{\sigma}$ is chosen based on validation MSE. • If `0`: No validation split is created. The algorithm still evaluates all $\boldsymbol{\sigma}$ candidates in the grid, but selects the optimal one based on the in-sample (training) MSE. This can lead to overfitting.

`stratify`	Logical. If `TRUE`, performs stratified sampling based on missing values in `X`. Default is `FALSE`.
`update_final`	Logical. If `TRUE`, updates the final model with the optimal σ after cross-validation. Default is `TRUE`. When `TRUE` (and `prop_valid > 0`), after selecting the optimal σ on a validation set, the model is **re-fitted** using the entire training set of the current fold/iteration before evaluating on the test set. If `FALSE`, the model fitted on the smaller training subset (used for sigma selection) is used for evaluation, which can be faster but may yield higher test error estimates.
`fold_idx`	Integer. Optional. A numeric vector with fold numbers (from 1 to `n_rep`) for each observation, allowing for pre-defined folds. If provided, `method` is ignored and k-fold CV is used.
`...`	Control parameters for PRTree. See pr_tree_control

Details

Choosing the right configuration:

Goal	Method	Parameters
Estimate sigma + test error	any	`prop_valid > 0`, `only_sigma = FALSE`
Test error only (sigma known)	any	`prop_valid = 0`, `only_sigma = FALSE`
Sigma selection only	montecarlo	`prop_test = 0`, `only_sigma = TRUE`
Sigma selection only	kfold	`only_sigma = TRUE`

Value

A list of class `prtree.control_cv` containing merged with cross-validation settings

See Also

pr_tree for fitting a single tree,

pr_tree_cv for cross-validation.

Examples

```
# Default k-fold CV
cv1 <- pr_tree_control_cv()

# Monte Carlo CV with custom parameters
cv2 <- pr_tree_control_cv(
  method = "montecarlo",
  n_iter = 20,
  prop_test = 0.3,
  prop_valid = 0.2,
  stratify = TRUE
)
```

pr_tree_cv *Cross-Validation for Probabilistic Regression Trees*

Description

Performs cross-validation for Probabilistic Regression Trees. Supports both k-fold and Monte Carlo (repeated random subsampling) methods.

Usage

```
pr_tree_cv(y, X, control_cv = list(), verbose = TRUE, ...)
```

Arguments

y	A numeric vector of response values.
X	A numeric matrix or data frame of predictor variables.
control_cv	A list of cross-validation control parameters, typically created by pr_tree_control_cv. Default values are taken from pr_tree_control_cv for any parameters not specified. Alternatively, control parameters can be passed directly via the ... argument.
verbose	Logical. If TRUE (default), prints progress and informational messages. If FALSE, suppresses all non-error console output.
...	Control parameters to be passed to pr_tree_control.

Details

The pr_tree_cv function provides flexible cross-validation for Probabilistic Regression Trees, supporting three common scenarios:

1. Sigma + test error estimation (default)

When both prop_valid > 0 and a grid of σ values is provided (either via control$sigma_grid or automatically generated), the function performs a two-step procedure in each iteration/fold:

- The training data is further split into training and validation sets according to prop_valid.
- The validation set is used to select the optimal σ from the grid.
- The model is then refitted using the **full training set** (training + validation) with the selected σ.
- Finally, the refitted model is evaluated on the test set.

This yields both the selected σ values for each iteration/fold and estimates of the test error.

2. Test error estimation only (sigma vector fixed)

When prop_valid = 0 or NULL, no internal validation is performed. This is appropriate when σ is already known (e.g., from a previous analysis) and you only need to estimate the test error. In this case:

- The model is trained directly on the full training set using the provided σ values (a vector or 1 by n_feat matrix).
- The test error is computed on the test set.

This mode is computationally lighter and focuses solely on error estimation.

3. Sigma selection only (no test error)

When prop_test = 0 (for Monte Carlo) or when you're interested only in the selected σ values and not in test error estimation, the function can be used without a test set. In this case:

- All data is used for training and validation.
- The optimal σ is selected via internal validation (if prop_valid > 0) or using the full training set (if prop_valid = 0).
- No test error is computed (rmse_by_rep will contain NULL).

This mode is useful when you want to estimate σ from the entire dataset before refitting a final model, or when you're only interested in the stability of σ estimates across different splits.

Cross-validation methods

Two resampling methods are available:

- **Monte Carlo CV** (method = "montecarlo"): In each of n_iter iterations, the data is randomly split into training (1 - prop_test) and testing (prop_test) sets. Observations may appear in multiple test sets across iterations.
- **k-fold CV** (method = "kfold"): The data is partitioned into n_folds disjoint subsets. Each fold serves as the test set once, while the remaining n_folds - 1 folds are used for training and validation. This ensures every observation is tested exactly once.

Value

A list of class prtree.cv with components:

sigma_matrix — Matrix of selected σ values, with one row per iteration/fold and one column per feature.

rmse_by_rep — A list with components:

- validation: Numeric vector of validation RMSE values (NULL if no validation set used).
- test: Numeric vector of test RMSE values (NULL if only_sigma = TRUE or no test set).

track_sigma — Logical indicating whether sigma varies across iterations/folds (TRUE) or is fixed (FALSE). When TRUE, different sigma values may be produced for each iteration/fold, either because they are estimated from the data of that iteration (sigma_grid = NULL) or selected from a grid of candidates (grid_size > 1). When FALSE, the same sigma value is used for all iterations/folds.

Examples

```
# Generate example data
set.seed(123)
X <- matrix(runif(1000, 0, 10), ncol = 2)
y <- 2 * sin(X[, 1]) + 0.5 * X[, 2] + rnorm(500, 0, 0.2)

# Default k-fold CV
cv1 <- pr_tree_cv(y, X)
plot(cv1)

# Monte Carlo CV with custom parameters
control_cv <- pr_tree_control_cv(
  method = "montecarlo",
  n_iter = 20,
  prop_test = 0.3,
  prop_valid = 0.2
)
cv2 <- pr_tree_cv(y, X, control_cv = control_cv)
plot(cv2)
```

---

predict.prtree *Predict from a Probabilistic Regression Tree Model*

Description

Obtains predictions from a fitted `prtree` object.

Usage

```
## S3 method for class 'prtree'
predict(object, newdata, complete = FALSE, ...)
```

Arguments

`object`	An object of class `prtree`, as returned by pr_tree.
`newdata`	A numeric matrix or data frame containing new data for which to generate predictions. Must contain the exact same number and order of predictor variables as the data used to fit the model. It can contain missing values (`NA`).
`complete`	Logical. If `FALSE` (default), only the vector of predicted values is returned. If `TRUE`, a list containing both the predicted values and the probability matrix `P` is returned.
`...`	further arguments passed to or from other methods.

Value

If `complete = FALSE`, a numeric vector of predicted values (`yhat`). If `complete = TRUE`, a list containing:

`yhat`	The numeric vector of predicted values.
`P`	The probability matrix for the new data.

`print.idx.split` *Print method for idx.split objects*

Description

Prints a brief summary of a train/test split result.

Usage

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

Arguments

`x`	An object of class `idx.split`.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

`print.prtree` *Print method for PRTree objects*

Description

Prints a brief summary of a fitted Probabilistic Regression Tree model.

Usage

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

Arguments

`x`	An object of class `prtree`, as returned by pr_tree.
`...`	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.prtree.control *Print method for PRTree control objects*

Description

Prints a summary of the control parameters for PRTree model fitting.

Usage

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

Arguments

x	An object of class `prtree.control`, as returned by pr_tree_control.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.prtree.control_cv *Print method for PRTree control objects*

Description

Prints a summary of the control parameters for PRTree model fitting.

Usage

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

Arguments

x	An object of class `prtree.control_cv`, as returned by pr_tree_control_cv.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.prtree.cv *Print method for PRTree Cross-Validation objects*

Description

Prints a brief summary of cross-validation results for Probabilistic Regression Trees.

Usage

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

Arguments

x	An object of class prtree.cv, as returned by pr_tree_cv.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.sigma_grid *Print method for sigma grid objects*

Description

Prints a summary of a sigma grid generated by expand_sigma_grid.

Usage

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

Arguments

x	An object of class sigma_grid, as returned by expand_sigma_grid.
digits	minimal number of significant digits.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.summary.idx.split *Print method for summary of* idx.split *objects*

Description

Prints a detailed summary of a train/test split result.

Usage

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

Arguments

x	An object of class summary.idx.split.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.summary.prtree *Print method for summary of PRTree objects*

Description

Prints a summary of a fitted Probabilistic Regression Tree model.

Usage

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

Arguments

x	An object of class summary.prtree.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

print.summary.prtree.cv *Print method for summary of PRTree Cross-Validation objects*

Description

Prints a detailed summary of cross-validation results.

Usage

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

Arguments

x	An object of class summary.prtree.cv.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the original object.

summary.idx.split *Summary method for* idx.split *objects*

Description

Provides a detailed summary of a train/test split result, including statistics about missing values if available.

Usage

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

Arguments

object	An object of class idx.split.
...	Further arguments passed to or from other methods.

Value

A list of class summary.idx.split containing detailed statistics.

summary.prtree *Summary method for PRTree objects*

Description

Provides a detailed summary of a fitted Probabilistic Regression Tree model.

Usage

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

Arguments

object	An object of class prtree, as returned by pr_tree.
...	Further arguments passed to or from other methods.

Value

A list of class summary.prtree containing summary statistics.

summary.prtree.cv *Summary method for PRTree Cross-Validation objects*

Description

Provides a detailed summary of cross-validation results for Probabilistic Regression Trees, including statistics for RMSE and selected sigma values.

Usage

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

Arguments

object	An object of class prtree.cv, as returned by [pr_tree_cv].
...	Further arguments passed to or from other methods.

Value

A list of class `summary.prtree.cv` containing:

`method`	Cross-validation method used.
`config`	Complete configuration list.
`n_features`	Number of features.
`sigma_summary`	Matrix with summary statistics for selected sigma values (Min, Mean, Median, Max, SD) for each feature.
`rmse_summary`	List with summary statistics for validation and/or test RMSE (Min, Q1, Median, Mean, Q3, Max, SD).

`train_test_split` *Split data into training and testing sets*

Description

Creates indices for training and testing sets. Can use simple random sampling or stratified sampling based on missing values.

Usage

```
train_test_split(n_obs = NULL, prop_test = 0.2, stratify = FALSE,
  is_NA = NULL, n_rep = NULL)
```

Arguments

`n_obs`	Integer. Total number of observations. Required if `stratify = FALSE`. Ignored when `stratify = TRUE` (in which case `n_obs` is determined from `is_NA`).
`prop_test`	Numeric between 0 and 1. Proportion of data to use for testing. Default is 0.2.
`stratify`	Logical. If `TRUE`, performs stratified sampling based on missing values. In this case, `is_NA` must be provided. Default is `FALSE`.
`is_NA`	Optional logical vector. `TRUE` indicates rows with missing values. Required if `stratify = TRUE`. If provided when `stratify = FALSE`, its length must equal `n_obs` and missing statistics will be computed (see details).
`n_rep`	Optional integer. Number of folds for creating a stratified or simple random k-fold partition. If `NULL` (default), the function generates a single training/test split according to `prop_test`. When specified, `prop_test` is ignored and the function returns a vector of fold assignments (`fold_idx`) instead of training and testing indices.

Details

The function creates indices for training and testing sets using either simple random sampling or stratified sampling based on missing values.

Stratified sampling: When `stratify = TRUE`, the sampling preserves the proportion of rows with missing values in both training and testing sets. In this case:

- `is_NA` must be provided
- `n_obs` is ignored (determined from `is_NA`)

Missing values statistics: If `is_NA` is provided but `stratify = FALSE`, the function will still compute missing values statistics for the resulting splits. The length of `is_NA` is against `n_obs` to ensure consistency. These statistics will be available in the summary output, with a note indicating that they were not used for stratification.

Reproducibility: To ensure reproducible splits, call set.seed before using this function.

Value

An object of class `idx.split`.

If `n_rep = NULL` (default), the returned object contains:

- `idx_train`: Integer vector of training indices.
- `idx_test`: Integer vector of testing indices (or `NULL` if `prop_test = 0`).
- `n_train`: Integer. Number of training observations.
- `n_test`: Integer. Number of testing observations.
- `stratified`: Logical indicating whether stratified sampling was used.
- `missing_stats`: List with missing-value counts (if `is_NA` is provided), containing:
  - `train_na`: Number of rows with missing values in the training set.
  - `test_na`: Number of rows with missing values in the testing set.
  - `total_na`: Total number of rows with missing values.

If `n_rep` is specified, the returned object contains:

- `fold_idx`: Integer vector indicating the fold assignment of each observation.
- `n_rep`: Integer. Number of folds.
- `stratified`: Logical indicating whether stratified sampling was used.
- `fold_stats`: Data frame summarizing each fold, including the number of observations, the number of rows with missing values, and the corresponding proportion of missing values.

Examples

```
# Simple random split
set.seed(123)
split <- train_test_split(n_obs = 100, prop_test = 0.3)
str(split)

# Stratified split by missing values
```

```
missing <- sample(c(TRUE, FALSE), 100, replace = TRUE, prob = c(0.2, 0.8))
split <- train_test_split(
  prop_test = 0.3,
  stratify = TRUE,
  is_NA = missing
)
summary(split)

# Compute missing stats without stratification
missing2 <- sample(c(TRUE, FALSE), 100, replace = TRUE, prob = c(0.2, 0.8))
split <- train_test_split(
  n_obs = 100,
  prop_test = 0.3,
  stratify = FALSE,
  is_NA = missing2
)
summary(split) # Note about missing stats will appear


# Stratified 10-fold partition
folds <- train_test_split(
  n_obs = 1000,
  is_NA = sample(c(TRUE, FALSE), prob = c(0.1, 0.9), replace = TRUE, size = 1000),
  stratify = TRUE,
  n_rep = 5
)
summary(folds)
```

Index

expand_sigma_grid, 2, *8*, *20*

plot.prtree, 4
plot.prtree.cv, 5
plot_tree, 6
pr_tree, *2*, *4, 5*, 7, *7*, *14*, *17, 18*, *23*
pr_tree_control, *4*, *7, 8*, 9, *9*, *13–15*, *19*
pr_tree_control_cv, *4*, 13, *13*, *15*, *19*
pr_tree_cv, *4, 5*, *8*, *14*, 15, *15*, *20*
predict.prtree, 17
print.idx.split, 18
print.prtree, 18
print.prtree.control, 19
print.prtree.control_cv, 19
print.prtree.cv, 20
print.sigma_grid, 20
print.summary.idx.split, 21
print.summary.prtree, 21
print.summary.prtree.cv, 22

set.seed, *25*
summary, *25*
summary.idx.split, 22
summary.prtree, 23
summary.prtree.cv, 23

train_test_split, 24