Package 'ROOT'

January 22, 2026

Title ROOT (Rashomon Set of Optimal Trees)

Version 0.0.0.9000

Description ROOT (Rashomon set of Optimal Trees) is a general framework for globally optimizing user-specified functionals over interpretable binary weight functions represented as sparse decision trees. It searches over many candidate trees to construct a Rashomon set of near-optimal solutions and derives a characteristic summary tree that highlights stable patterns in the optimized weights.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests mlbench,
testthat (>= 3.0.0),
knitr,
rmarkdown,
ragg

Config/testthat/edition 3

Imports MASS,
rpart,
gbm,
stats,
withr,
rpart.plot

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

Contents

`characterize_tree` *Fit a shallow decision tree to characterize learned weights* `w`

Description

Trains a classification tree on the covariates `X` to predict the binary membership `w`. This provides an interpretable summary of how the weighted subgroup can be distinguished by `X`.

Usage

```
characterize_tree(X, w, max_depth = 3)
```

Arguments

`X`	A `data.frame` of covariates.
`w`	A `vector` of length `nrow(X)` that is binary. Accepts `0` and `1` or a `factor` with two levels.
`max_depth`	An `integer(1)` giving the maximum tree depth. Default is `3`.

Details

The tree uses the Gini index for classification and no pruning with complexity parameter `cp = 0`. Depth control is through `max_depth`. If `w` is not a factor it is converted internally. The resulting rules indicate which covariates and splits separate the two classes defined by `w`.

Value

An `rpart` object that represents the fitted classification tree.

characterizing_underrep *Characterize under-represented subgroups (wrapper around ROOT)*

Description

A convenience wrapper around `ROOT()` for under-representation analyses. It takes a single `data` set and calls `ROOT()` either in generalizability_path mode (when `generalizability_path = TRUE`) or in general optimization mode (`generalizability_path = FALSE`).

Usage

```
characterizing_underrep(
  data,
  global_objective_fn = NULL,
  generalizability_path = FALSE,
  leaf_proba = 0.25,
  seed = 123,
  num_trees = 10,
  vote_threshold = 2/3,
  explore_proba = 0.05,
  feature_est = "Ridge",
  feature_est_args = list(),
  top_k_trees = FALSE,
  k = 10,
  cutoff = "baseline",
  verbose = FALSE
)
```

Arguments

`data`	A `data.frame` containing covariates and, in generalizability_path mode, also columns `Y`, `Tr`, and `S`.
`global_objective_fn`	function with signature `function(D) -> numeric` scoring the entire state and minimized by ROOT. If NULL, a default variance-based objective is used (see `objective_default()`).
`generalizability_path`	`logical(1)`. If `TRUE`, calls `ROOT()` with `generalizability_path = TRUE` and expects columns `Y`, `Tr`, and `S` in `data`. If `FALSE`, calls `ROOT()` in general optimization mode. Default `FALSE`.
`leaf_proba`	numeric(1) tuning parameter that increases the chance a node stops splitting by selecting a synthetic `"leaf"` feature. Internally, the probability of choosing `"leaf"` is `leaf_proba / (1 + leaf_proba)` (assuming the covariate probabilities sum to 1). Default `0.25`.
`seed`	Random seed for reproducibility.
`num_trees`	Number of trees to grow.
`vote_threshold`	Majority vote threshold used for `w_opt`.
`explore_proba`	Exploration probability in tree growth.

feature_est	Either "Ridge", "GBM", or a custom feature importance function.
feature_est_args	List of extra arguments passed to feature_est when it is a function.
top_k_trees	Logical; if TRUE, uses top k trees by objective, otherwise a cutoff rule.
k	Number of trees when top_k_trees = TRUE.
cutoff	Numeric or "baseline" Rashomon cutoff.
verbose	Logical; if TRUE, prints progress/estimands from ROOT().

Details

When generalizability_path = TRUE, data must contain standardized columns:

- Y: outcome,
- Tr: treatment indicator (0/1),
- S: sample indicator (1 = trial, 0 = target).

Value

A characterizing_underrep S3 object (a list) with:

root	The ROOT object returned by ROOT().
combined	The input data (for continuity with prior API).
leaf_summary	Data frame of terminal node rules and labels, or NULL.

Examples

```
## Not run:
char.output = characterizing_underrep(diabetes_data,generalizability_path = TRUE, seed = 123)

## End(Not run)
```

---

choose_feature *Randomly choose a split feature based on provided probabilities*

Description

Selects one feature name at random according to a probability vector that may include a special "leaf" entry.

Usage

```
choose_feature(split_feature, depth)
```

Arguments

split_feature	A named numeric vector of feature selection probabilities. Names correspond to feature identifiers and may include "leaf".
depth	An integer(1) giving the current tree depth. Present for parity with the Python version and does not change probabilities here.

Value

A `character(1)` which is the chosen feature name or `"leaf"`.

Note

The factor $2^{(0 \cdot \text{depth}/4)}$ present in the code equals 1 and does not change the first element weight. All probabilities are normalized to sum to 1 before sampling.

`compute_transport_scores`
Compute transport influence scores for generalization mode

Description

Internal helper used in the `generalization` path to construct glm-based, IPW-style scores for transporting trial effects to a target population without double machine learning.

Usage

```
compute_transport_scores(data, outcome, treatment, sample)
```

Arguments

`data`	A `data.frame` containing the outcome, treatment indicator, sample indicator, and covariates. All columns except `outcome`, `treatment`, and `sample` are treated as covariates X and must be suitable for use in `stats::glm()` with `family = binomial()`.
`outcome`	A length-1 character string giving the name of the outcome column in `data`. Must be numeric (e.g., a continuous outcome or a 0/1 indicator).
`treatment`	A length-1 character string giving the name of the treatment indicator column in `data`. Must be coded 0/1.
`sample`	A length-1 character string giving the name of the sample indicator column in `data`, with 1 for trial rows and 0 for target rows.

Details

The function treats `data` as a stacked dataset with a sample indicator S (`sample`) taking value 1 in the randomized trial and 0 in the target sample. It proceeds in three steps:

1. Fit a sampling model $P(S = 1|X)$ using logistic regression on all rows of `data`.
2. Within the trial subset $S = 1$, fit a treatment model $P(T = 1|X, S = 1)$ using logistic regression.
3. For each row, form the density ratio $r(X) = P(S = 0|X)/P(S = 1|X)$ and compute a Horvitz-Thompson-style transported score

$$v(X, T, Y) = r(X) \left[\frac{TY}{e(X)} - \frac{(1 - T)Y}{1 - e(X)} \right],$$

where $e(X) = P(T = 1|X, S = 1)$.

The resulting score vector `v` and its squared version `vsq` can be used as pseudo-outcomes for tree-based search over transported treatment effects.

Value

A list with two numeric vectors of length `nrow(data)`:

`v` The transported influence-style score.

`vsq` The element-wise square of `v`, i.e., `v^2`.

`diabetes_data` *Simulated Diabetes Dataset for Examples*

Description

A toy dataset for illustrating `ROOT` examples and tests.

Usage

```
data(diabetes_data)
```

Format

A `data.frame` with one row per individual and the columns:

Age45 Indicator in `0/1`: age >= 45.

DietYes Indicator in `0/1`: on a diet program.

Race_Black Indicator in `0/1`: race is Black.

S Sample indicator in `0/1`: `1` means RCT or source, `0` means target.

Sex_Male Indicator in `0/1`: male.

Tr Treatment assignment in `0/1`.

Y Observed outcome (`numeric` or `0/1`).

Abbreviations

RCT means randomized clinical trial. ATE means Average Treatment Effect.

`gen_S` *Generate sample indicator S drawn from a Bernoulli distribution*

Description

Generates a binary sample inclusion indicator `S` for each observation, using a logistic model influenced by a rectangular region in `X0` and `X1`. In the ROOT generalizability path, `S = 1` is interpreted as belonging to the trial sample and `S = 0` as belonging to the target sample.

Usage

```
gen_S(X, seed = NULL)
```

Arguments

X	A data.frame of covariates that contains at least X0 and X1.
seed	Optional numeric(1) seed. If NULL no seed is set.

Details

The inclusion probability is $p = plogis(a)$ where $a = 0.25 - 2 * I(X0in(0.5, 1)andX1in(0.5, 1))$.

Value

A data.frame with one column S of values in 0 or 1.

---

gen_T *Generate treatment indicator Tr drawn from a Bernoulli distribution*

Description

Assigns treatment indicators combining a randomized design for S == 1 (trial sample) and an observational assignment driven by X0 for S == 0 (target sample).

Usage

```
gen_T(X, S, seed = NULL)
```

Arguments

X	A data.frame of covariates.
S	A data.frame with column S in 0 or 1. This should be the output of gen_S().
seed	Optional numeric(1) seed. If NULL no seed is set.

Details

For S == 1 the treatment probability is 0.5. For S == 0 the treatment probability is plogis(X0). The combined probability is $pi_i = S_i * 0.5 + (1 - S_i) * plogis(X0_i)$.

Value

A list with:

Tr	data.frame with one column Tr in 0 or 1.
pi	numeric vector of assignment probabilities per observation.

gen_XY *Generate covariates X and potential outcomes (Y0, Y1)*

Description

Simulates a regression problem based on Friedman number one and defines a heterogeneous treatment effect. Uses `mlbench.friedman1` to generate feature matrix `X` and a baseline outcome `Y0`. The treatment potential outcome `Y1` is defined as `Y1 = Y0 + log(Y0 + 1)`.

Usage

```
gen_XY(n = 1000, seed = NULL)
```

Arguments

`n`	A `numeric(1)` or `integer(1)` giving the number of observations to simulate. Must be positive.
`seed`	Optional `numeric(1)` for the random number generator seed. If `NULL` no seed is set.

Details

The `mlbench.friedman1` generator creates ten continuous features and a baseline outcome `Y0` with additive noise. The potential outcome `Y1` adds a nonlinear term `log(Y0 + 1)` to `Y0`.

Value

A `list` with two components:

`X`	`data.frame` of simulated covariates with columns `X0, X1, ...`.
`Y`	`data.frame` with columns `Y0` and `Y1`.

get_data *Convenience wrapper to generate a full simulated dataset*

Description

Generates covariates, sample inclusion, treatment assignments, and observed outcomes for a given sample size by calling `gen_XY()`, `gen_S()`, and `gen_T()`. The returned `data` element is ready for use with `ROOT(data = ..., generalizability_path = TRUE)`, since it contains columns `Y`, `Tr`, and `S` in the expected format.

Usage

```
get_data(n = 1000, seed = NULL)
```

Arguments

`n`	A `numeric(1)` or `integer(1)` sample size.
`seed`	Optional `numeric(1)` base seed. If provided, internal generators use simple offsets for reproducibility.

Value

A list with:

data	data.frame of length n with covariates X0, X1, ..., observed outcome Y, sample indicator S, and treatment indicator Tr. This is directly usable by ROOT in generalizability_path = TRUE mode.
Y	data.frame of potential outcomes with columns Y0 and Y1.

Examples

```
sim <- get_data(n = 100, seed = 599)
dim(sim$data)
head(sim$data$Y)
head(sim$Y)
```

---

loss_from_objective *Backward and fast path micro evaluator adaptor*

Description

Wraps a global objective global_objective_fn(D) into a splitter compatible loss function loss_fn(val, indices, D) by calling objective_if on a temporary copy of D.

Usage

```
loss_from_objective(global_objective_fn)
```

Arguments

global_objective_fn	A function with signature function(D) -> numeric that returns a scalar to be minimized. For example objective_default.

Value

A function loss_fn(val, indices, D) suitable for use in ROOT and split_node. It sets w = val on indices without mutation of the original D and then returns global_objective_fn(D).

---

midpoint *Compute the midpoint of a numeric vector*

Description

Calculates the midpoint defined as $(\max(X) + \min(X))/2$ while ignoring any NA values in X.

Usage

```
midpoint(X)
```

Arguments

X	A numeric vector.

Value

A numeric(1) giving the midpoint of the finite values in X. Returns NA_real_ when X is empty or has no finite values.

---

objective_default *Generic objective interface*

Description

Default objective that serves as a proxy for Standard Error in Weighted Transported Average Treatment Effect and Population Average Treatment Effect.

Usage

```
objective_default(D)
```

Arguments

D	A data.frame with at least numeric columns vsq and w.

Details

Computes sqrt(sum(vsq_i * w_i)) / (sum(w_i))^2. Requires columns vsq and w in D. The goal is to minimize the value. Supply your own function(D) -> numeric to use a different objective.

Value

A numeric(1) objective value. Returns Inf when undefined.

Abbreviations

ATE means Average Treatment Effect. SE means Standard Error. TATE means Transported ATE. WTATE means Weighted TATE. WATE means Weighted ATE. PATE means Population ATE.

objective_if *Helper to evaluate the objective after a hypothetical local change*

Description

Evaluates `global_objective_fn` on a temporary copy of `D` after setting `w = val` for the rows selected by `indices`.

Usage

```
objective_if(val, indices, D, global_objective_fn)
```

Arguments

`val`	A `numeric(1)` that must be either `0` or `1`.
`indices`	An `integer` vector of row indices or a `character` vector of row names that receive `val`.
`D`	A `data.frame` used by `global_objective_fn`. Must contain columns `w` and `vsq`.
`global_objective_fn`	A `function` with signature `function(D) -> numeric`.

Value

A `numeric(1)` objective value after the hypothetical change.

plot.characterizing_underrep *Plot Under represented Population Characterization*

Description

Visualizes the decision tree derived from the `ROOT` analysis. Highlights which subgroups are represented where `w = 1` versus underrepresented where `w = 0` in generalization mode, or simply `w(x)` in `{0,1}` in general optimization mode.

Usage

```
## S3 method for class 'characterizing_underrep'
plot(
  x,
  main = "Underrepresented Population Characterization",
  cex.main = 1.2,
  ...
)
```

Arguments

x	A characterizing_underrep S3 object with x$root$f present as an rpart object for the summary or characterization tree.
main	Character string for the plot title. Default is "Underrepresented Population Characterization".
cex.main	Numeric scaling factor for the title text size. Default is 1.2.
...	Additional arguments passed to rpart.plot::prp().

Value

NULL. The plot is drawn to the active graphics device.

Examples

```
## Not run:
char.output = characterizing_underrep(diabetes_data, generalizability_path = TRUE, seed = 123)
plot(char.output)
plot(char.output, main = "My Custom Title", cex.main = 1.5)

## End(Not run)
```

---

plot.ROOT *Plot the ROOT summary tree*

Description

Visualizes the decision tree that characterizes the weighted subgroup (the weight function $w(d)$ in {0,1}) identified by ROOT(), using rpart.plot::prp().

Usage

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

Arguments

x	A "ROOT" S3 object returned by ROOT() with x$f an rpart object representing the summary / characterization tree.
...	Additional arguments passed to rpart.plot::prp().

Value

No return value; the plot is drawn to the active graphics device.

Examples

```
## Not run:
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)
plot(ROOT.output)

## End(Not run)
```

reduce_weight *Reduce a feature selection weight by one half and renormalize*

Description

Lowers the probability weight of a given feature by one half and then renormalizes the full vector to sum to one.

Usage

```
reduce_weight(fj, split_feature)
```

Arguments

fj	A character(1) feature name that must be present in names(split_feature).
split_feature	A named numeric vector of probabilities for features as used in splitting.

Details

This is used when a feature split was rejected. The feature probability is halved to reduce the chance of immediate reselection which encourages exploration of other features. If fj equals "leaf" its weight is also halved.

Value

A numeric vector of the same length as split_feature that sums to 1.

ROOT *Ensemble of weighted trees for general optimization and Rashomon selection*

Description

Builds multiple weighted trees, then identifies a "Rashomon set" of top-performing trees and aggregates their weight assignments by majority vote.

Usage

```
ROOT(
  data,
  global_objective_fn = NULL,
  generalizability_path = FALSE,
  leaf_proba = 0.25,
  seed = NULL,
  num_trees = 10,
  vote_threshold = 2/3,
  explore_proba = 0.05,
  feature_est = "Ridge",
  feature_est_args = list(),
  top_k_trees = FALSE,
```

```
  k = 10,
  cutoff = "baseline",
  verbose = FALSE
)
```

Arguments

data	A data.frame containing the dataset. In *general optimization* mode (generalizability_path = FALSE), data can be any set of covariates and auxiliary columns. The user supplies a global_objective_fn that takes a data frame with a column w and returns a scalar loss. In *generalizability_path* mode (generalizability_path = TRUE), data must contain columns "Y" (outcome), "Tr" (treatment indicator, 0/1), and "S" (sample indicator, 1 = trial, 0 = target). ROOT internally constructs transportability scores and, if no custom objective is given, uses a default variance-based loss.
global_objective_fn	function with signature function(D) -> numeric scoring the entire state and minimized by ROOT. If NULL, a default variance-based objective is used (see objective_default()).
generalizability_path	logical(1). If TRUE, use the built-in transportability objective based on (Y, Tr, S). If FALSE, treat data as arbitrary and rely on global_objective_fn. Default FALSE.
leaf_proba	numeric(1) tuning parameter that increases the chance a node stops splitting by selecting a synthetic "leaf" feature. Internally, the probability of choosing "leaf" is leaf_proba / (1 + leaf_proba) (assuming the covariate probabilities sum to 1). Default 0.25.
seed	optional numeric(1) seed for reproducibility.
num_trees	integer(1) number of trees to grow. Default 10.
vote_threshold	numeric(1) in (0.5, 1] giving the majority vote threshold for final w = 1. Default 2/3.
explore_proba	numeric(1) giving the exploration probability at leaves. Default 0.05.
feature_est	either a character(1) in c("Ridge", "GBM") or a function(X, y, ...) returning a named nonnegative numeric vector of importances with names matching columns of X. Used only to bias which covariates are chosen for splitting. If it fails, ROOT falls back to uniform feature sampling with a warning.
feature_est_args	list of additional arguments passed to a user supplied feature_est function.
top_k_trees	logical(1). If TRUE, select top k trees by objective; otherwise use cutoff. Default FALSE.
k	integer(1) giving the number of top trees when top_k_trees = TRUE. Default 10.
cutoff	numeric(1) or "baseline". Used as the Rashomon cutoff when top_k_trees = FALSE. "baseline" uses the objective at w = 1 (all weights equal to 1).
verbose	logical(1). If TRUE, prints unweighted and (when available) weighted estimates and their standard errors in generalizability_path mode.

Details

The function is framed as a general functional optimization routine: given data D_n and a loss $L(w, D_n)$, ROOT searches over interpretable tree-based weight functions $w(d)$ in {0,1}.

Value

An object of class "ROOT" (a list) with elements:

- D_rash: data frame with Rashomon-set votes and w_opt.
- D_forest: data frame with forest-level working columns.
- w_forest: list of per-tree results from split_node().
- rashomon_set: indices of selected trees.
- global_objective_fn: the objective function used.
- f: summary classifier (e.g., rpart tree) or NULL.
- testing_data: data frame aligned to rows used to compute scores.
- estimate: (only if generalizability_path = TRUE) list with unweighted and weighted estimands, SEs, and a note about the SE.
- generalizability_path: logical flag.

Examples

```
## Not run:
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)

## End(Not run)
```

`split_node` *Recursive split builder for weighted tree*

Description

Recursively builds a weighted decision tree to optimize a global objective, using an exploration versus exploitation choice at leaves. Internal and used by `ROOT()`.

Usage

```
split_node(
  split_feature,
  X,
  D,
  parent_loss,
  depth,
  explore_proba = 0.05,
  choose_feature_fn = choose_feature,
  reduce_weight_fn = reduce_weight,
  global_objective_fn = objective_default,
  max_depth = 8,
  min_leaf_n = 5,
  log_fn = function(...) {
 },
  max_rejects_per_node = 1000
)
```

Arguments

split_feature	A named numeric vector of feature selection probabilities. Must include the name "leaf".
X	A data.frame of current observations. Includes candidate split feature columns and may include a working copy of weights w.
D	A data.frame representing the global state. Must include columns w and vsq. Row names must align to X.
parent_loss	A numeric(1) giving the loss of the parent node. Used to decide if a split improves the objective.
depth	An integer(1) giving the current tree depth.
explore_proba	A numeric(1) between 0 and 1 for the probability of flipping the exploit choice at a leaf.
choose_feature_fn	A function to choose the next feature. Default is choose_feature.
reduce_weight_fn	A function to penalize the last tried feature on a rejected split. Default is reduce_weight.
global_objective_fn	A function with signature function(D) -> numeric that scores the entire state.
max_depth	An integer(1) giving the maximum depth. A node becomes a leaf at this depth.
min_leaf_n	An integer(1) giving the minimum number of rows to attempt a split. Otherwise make a leaf.
log_fn	A function for logging. Default is a function that performs no operation.
max_rejects_per_node	An integer(1) giving the safety budget of rejected splits before forcing a leaf.

Value

A list representing the subtree. Includes updated D and a field named "local objective".

---

summary.characterizing_underrep

Summarize a characterizing_underrep fit

Description

Summarizes the ROOT summary which includes unweighted and (when in generalization mode) weighted estimates with standard errors, as reported by summary.ROOT(). Provides a brief overview of terminal rules from the annotated summary tree when available.

Usage

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

Arguments

object	A characterizing_underrep S3 object. Expected components include root which is a ROOT object (summarized by summary.ROOT()) and may contain f which is an rpart object for the summary tree, and leaf_summary which is a data.frame with one row per terminal node and may include a rule column of type character.
...	Currently unused. Included for S3 compatibility.

Details

Delegates core statistics and estimands to summary(object$root). Previews up to ten terminal rules when a summary tree exists.

Value

object returned invisibly. Printed output is a human readable summary.

Abbreviations

ATE means Average Treatment Effect. RCT means Randomized Controlled Trial. SE means Standard Error. TATE means Transported ATE. WTATE means Weighted TATE. WATE means Weighted ATE. PATE means Population ATE.

Examples

```
## Not run:
char.output = characterizing_underrep(diabetes_data,generalizability_path = TRUE, seed = 123)
summary(char.output)

## End(Not run)
```

---

summary.ROOT *Summarize a ROOT fit*

Description

Provides a human-readable summary of a ROOT object, including:

1. the summary characterization tree f,
2. the first few rows of testing_data,
3. the global_objective_fn used during optimization, and
4. in generalization mode (generalization = TRUE), the unweighted and weighted estimands with their standard errors and an explanatory note for the weighted SE.

Usage

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

Arguments

object	A "ROOT" S3 object returned by ROOT().
...	Currently unused and included for S3 compatibility.

Details

When generalization = TRUE, the unweighted estimand corresponds to a SATE-type quantity and the weighted estimand to a WTATE-type quantity for the transported target population. When generalization = FALSE, ROOT is used for general functional optimization and no causal labels are imposed; the summary focuses on the tree and diagnostics.

Value

object returned invisibly. Printed output is for inspection.

Diagnostics

The summary also reports:

- the number of trees grown,
- the size of the Rashomon set,
- the percentage of observations with ensemble vote w_opt == 1.

Examples

```
## Not run:
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)
summary(ROOT.output)

## End(Not run)
```

Index