Package 'ROOT'

March 10, 2026

Title Rashomon Set of Optimal Trees

Version 0.1.1

Description Implements a general framework for globally optimizing
user-specified objective functionals over interpretable binary weight functions
represented as sparse decision trees, called ROOT (Rashomon Set of Optimal Trees).
It searches over candidate trees to construct a
Rashomon set of near-optimal solutions and derives a summary tree highlighting stable
patterns in the optimized weights. ROOT includes a built-in generalizability mode for
identifying subgroups in trial settings for transportability analyses
(Parikh et al. (2025) <doi:10.1080/01621459.2025.2495319>).

License MIT + file LICENSE

Encoding UTF-8

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

URL https://github.com/peterliu599/ROOT

BugReports https://github.com/peterliu599/ROOT/issues

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Yiren Hou [aut] (ORCID: <https://orcid.org/0009-0005-0422-4268>, Equal
contribution),
Peter Liu [aut, cre] (0009-0000-2691-5637, Equal contribution),
Sean McGrath [aut] (ORCID: <https://orcid.org/0000-0002-7281-3516>),
Harsh Parikh [aut] (ORCID: <https://orcid.org/0000-0003-1927-8646>)

Maintainer Peter Liu <bliu68@jh.edu>

Repository CRAN

Date/Publication 2026-03-10 11:40:12 UTC

Contents

---

characterizing_underrep

Characterize Underrepresented Subgroups

Description

A high-level wrapper around [ROOT](#)() for identifying and characterizing subgroups that are *insufficiently represented* in a randomized controlled trial (RCT) relative to a target population. The function returns an interpretable decision tree describing which subgroups should be included ($w(X) = 1$) or excluded ($w(X) = 0$) from the analysis, along with the corresponding target average treatment effect estimates.

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",
  max_depth = 8L,
  min_leaf_n = 2L,
  max_rejects_per_node = 10L,
  verbose = FALSE
)
```

Arguments

`data`	A `data.frame` containing covariates and, in generalizability mode, also columns `Y`, `Tr`, and `S`.
`global_objective_fn`	A function with signature `function(D) -> numeric` scoring the entire state and minimized by ROOT. If `NULL` (the default), a variance-based objective is used that targets the precision of the WTATE estimator.
`generalizability_path`	Logical. If `TRUE`, runs the full generalizability analysis (design + analysis stages) and expects columns `Y`, `Tr`, and `S` in `data`. If `FALSE`, runs ROOT in general optimization mode. Default `FALSE`.
`leaf_proba`	A `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). Higher values produce shallower, more interpretable trees. Default `0.25`.
`seed`	Random seed for reproducibility.
`num_trees`	Number of trees to grow in the ROOT forest. More trees explore the tree space more thoroughly but increase computation time.
`vote_threshold`	Controls how Rashomon-set tree votes are aggregated into `w_opt`. Accepts a numeric threshold in `(0, 1]`, one of `"majority"` / `"mean"` / `"median"`, or a custom `function(votes) -> integer` vector. See [ROOT](#) for full details. Default 2/3 (majority vote).
`explore_proba`	Exploration probability in tree growth. Controls the explore-exploit trade-off: with probability `explore_proba`, a leaf is assigned a random weight; otherwise, it receives the greedy optimal weight. Default `0.05`.
`feature_est`	Either `"Ridge"`, `"GBM"`, or a custom feature importance function. Used to bias which covariates are selected for splitting. Default `"Ridge"`.
`feature_est_args`	List of extra arguments passed to `feature_est` when it is a function.
`top_k_trees`	Logical; if `TRUE`, selects the top `k` trees by objective value for the Rashomon set. If `FALSE`, uses `cutoff` instead. Default `FALSE`.
`k`	Number of trees to retain when `top_k_trees = TRUE`. Default `10`.
`cutoff`	Numeric or `"baseline"`. When `top_k_trees = FALSE`, trees with objective values below this cutoff are included in the Rashomon set. `"baseline"` uses the objective value at $w \equiv 1$ (no subgroups excluded). Default `"baseline"`.
`max_depth`	Maximum depth of each tree grown during the forest construction stage. A node at `depth == max_depth` is forced to be a leaf. Shallower trees are more interpretable but less flexible. Default 8.
`min_leaf_n`	Minimum number of observations required in a node for splitting to be attempted. If a node contains fewer than `min_leaf_n` observations it becomes a leaf. Default 2.

max_rejects_per_node

> Maximum number of consecutive rejected splits (splits that do not improve the objective) allowed at a single node before the node is forced to become a leaf. This prevents infinite recursion in pathological cases. Default `10`.

verbose   Logical; if `TRUE`, prints the unweighted and weighted estimands with standard errors. Default `FALSE`.

Value

A `characterizing_underrep` S3 object (a `list`) with:

root   The [ROOT] object returned by `ROOT()`. Contains the Rashomon set, weight assignments (`root$D_rash$w_opt`), summary tree (`root$f`), and (in generalizability mode) treatment effect estimates (`root$estimate`).

combined   The input `data`.

leaf_summary   A `data.frame` with one row per terminal node of the summary (characteristic) tree, giving the decision rule, predicted weight, sample size, and a label indicating whether the subgroup is "Represented (keep, w = 1)" or "Under-represented (drop, w = 0)". `NULL` if no summary tree was produced.

What does "underrepresented" mean?

In the context of generalizing treatment effects from a trial to a target population, a subgroup is considered **underrepresented** (or *insufficiently represented*) when it occupies a region of the covariate space that both (a) has limited overlap between the trial and the target population, and (b) exhibits heterogeneous treatment effects.

Formally, the contribution of a unit with covariates $X = x$ to the variance of the target average treatment effect (TATE) estimator depends on both the selection ratio $\ell(x) = P(S = 1 \mid X = x)/P(S = 0 \mid X = x)$ and the conditional average treatment effect. Subgroups where $\ell(x)$ is small, and conditional average treatment effect deviates from the overall TATE, contribute disproportionately to estimator variance. These are the subgroups that `characterizing_underrep()` identifies and characterizes. The sample average treatment effect (SATE) is a finite sample equivalent version of the TATE.

The generalizability workflow

When `generalizability_path = TRUE`, this function implements the two-stage approach of Parikh et al. (2025):

1. **Design stage:** ROOT learns binary weights $w(X)$ that minimize the variance of the weighted target average treatment effect (WTATE) estimator, subject to interpretability constraints (tree structure). The resulting decision tree characterizes which subgroups are well-represented ($w = 1$) and which are underrepresented ($w = 0$).

2. **Analysis stage:** The WTATE is estimated on the refined target population that excludes the underrepresented subgroups. This estimand trades some generality for greater precision and credibility.

The key estimands are:

- **SATE** (Sample Average Treatment Effect): the treatment effect for the full target population based on the trial sample, which may be imprecise if certain subgroups are underrepresented. It is a finite sample equivalent version of the TATE.
- **WTATE** (Weighted Target Average Treatment Effect): the treatment effect restricted to the sufficiently represented subpopulation, estimated with lower variance.

General optimization mode

When `generalizability_path = FALSE`, this function behaves as a convenience wrapper around `ROOT()` for arbitrary binary weight optimization. The user can supply a custom objective function via `global_objective_fn`; ROOT will learn an interpretable tree-based weight function minimizing that objective. See `vignette("optimization_path_example")` for an example.

Data requirements

When `generalizability_path = TRUE`, `data` must contain the following standardized columns:

- `Y`: numeric outcome variable.
- `Tr`: binary treatment indicator (0 = control, 1 = treated).
- `S`: binary sample indicator (1 = trial/RCT, 0 = target population).

All remaining columns are treated as pretreatment covariates X available for splitting.

References

Parikh H, Ross RK, Stuart E, Rudolph KE (2025). "Who Are We Missing?: A Principled Approach to Characterizing the Underrepresented Population." *Journal of the American Statistical Association*. doi:10.1080/01621459.2025.2495319

See Also

`ROOT` for the underlying optimization engine; `vignette("generalizability_path_example")` for a detailed worked example of the generalizability workflow; `vignette("optimization_path_example")` for general optimization mode.

Examples

```
# --- Generalizability analysis ---
# diabetes_data has columns Y, Tr, S, and covariates
data(diabetes_data, package = "ROOT")

char_fit <- characterizing_underrep(
  data                 = diabetes_data,
  generalizability_path = TRUE,
  num_trees            = 20,
  top_k_trees          = TRUE,
  k                    = 10,
  seed                 = 123
)
```

```
# View the characterization tree
plot(char_fit)

# Inspect which subgroups are underrepresented
char_fit$leaf_summary

# Treatment effect estimates (SATE and WTATE)
char_fit$root$estimate
```

---

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).

---

plot.characterizing_underrep
                    Plot Underrepresented 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 generalizability mode, or simply w(x) in {0,1} in general optimization mode.

Usage

```
## S3 method for class 'characterizing_underrep'
plot(
  x,
  main = "Final Characterized Tree from Rashomon Set",
  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 `"Final Characterized Tree from Rashomon Set"`.
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

```
char.output = characterizing_underrep(diabetes_data,generalizability_path = TRUE, seed = 123)
plot(char.output)
```

    `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

```
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)
plot(ROOT.output)
```

---

print.characterizing_underrep

Print a characterizing_underrep fit

Description

Print the ROOT summary which includes unweighted and (when in generalizability mode) weighted estimates with standard errors, as reported by summary.ROOT().

Usage

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

Arguments

x   A characterizing_underrep S3 object. Expected components include root
                    which is a ROOT object (summarized by print.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 print(x$root).

Value

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

Abbreviations

ATE Average treatment effect.

RCT Randomized controlled trial.

SE Standard error.

TATE Transported average treatment effect.

WTATE Weighted transported average treatment effect.

SATE Sample average treatment effect.

Examples

```
char.output = characterizing_underrep(diabetes_data,generalizability_path = TRUE, seed = 123)
print(char.output)
```

---

print.ROOT   *Print a ROOT fit*

Description

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

1. the summary characterization tree f,

2. in generalizability mode (generalizability_path = TRUE), the unweighted and weighted estimands with their standard errors and an explanatory note for the weighted standard error (SE).

Usage

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

Arguments

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

Value

object returned invisibly. Printed output is for inspection.

Abbreviations

ATE Average treatment effect.

RCT Randomized controlled trial.

SE Standard error.

TATE Transported average treatment effect.

WTATE Weighted transported average treatment effect.

SATE Sample average treatment effect.

When generalizability_path = 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 generalizability_path = FALSE, ROOT is used for general functional optimization and no causal labels are imposed.

Examples

```
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)
print(ROOT.output)
```

ROOT   *Rashomon Set of Optimal Trees (ROOT) for Functional Optimization*

Description

ROOT (Rashomon Set of Optimal Trees) is a general-purpose functional optimization algorithm that learns interpretable, tree-structured binary weight functions $w(X) \in \{0, 1\}$. Given a dataset D_n and a global objective function $L(w, D_n)$, ROOT searches over the space of decision trees to find weight assignments that minimize the objective function.

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",
  max_depth = 8L,
  min_leaf_n = 2L,
  max_rejects_per_node = 10L,
  verbose = FALSE
)
```

Arguments

data   A data.frame containing the dataset.

In **general optimization mode** (generalizability_path = FALSE), data can contain any 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 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

A function with signature `function(D) -> numeric` scoring the entire state and minimized by ROOT. `D` is the working data frame containing at least a column `w` with the current weight assignments. If `NULL` (default), a variance-based objective is used.

generalizability_path

`Logical(1)`. If `TRUE`, use the built-in transportability objective for trial-to-target generalizability. If `FALSE`, treat `data` as arbitrary and rely on `global_objective_fn`. Default `FALSE`.

leaf_proba

A numeric 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). Higher values produce shallower, sparser trees. Default `0.25`.

seed

An optional numeric seed for reproducibility.

num_trees

An integer number of trees to grow. More trees explore the solution space more thoroughly. Default 10.

vote_threshold

Controls how per-observation votes from the Rashomon set trees are aggregated into the final binary weight `w_opt`. Accepts one of:

- A **numeric** in `(0, 1)`: `w_opt = 1` when the mean fraction of Rashomon-set trees voting 1 strictly exceeds this value. An error is raised if any observation's mean vote equals the threshold exactly (tie is undefined). Default 2/3.
- `"majority"`: equivalent to the numeric threshold 2/3.
- `"mean"`: `w_opt = 1` when the mean vote strictly exceeds `0.5`. Errors on exact `0.5` ties.
- `"median"`: `w_opt = 1` when the per-row median vote strictly exceeds `0.5`. Errors on exact `0.5` ties.
- A **function** with signature `function(votes) -> integer` vector, where `votes` is a numeric matrix with one row per observation and one column per Rashomon-set tree (each cell 0 or 1). Must return an integer vector of 0s and 1s of length `nrow(votes)`.

explore_proba

A numeric giving the exploration probability at leaves. With probability `explore_proba`, a leaf receives a random weight; otherwise the greedy-optimal weight is used. Default `0.05`.

feature_est

Either `"Ridge"`, `"GBM"`, or a custom `function(X, y, ...) -> named numeric` vector returning nonnegative importance values with names matching columns of `X`. Used to bias which covariates are chosen for splitting. If the method fails, ROOT falls back to uniform feature sampling with a warning.

feature_est_args

A list of additional arguments passed to a user-supplied `feature_est` function.

top_k_trees

`Logical(1)`. If `TRUE`, select the top k trees by objective value for the Rashomon set. If `FALSE`, use the `cutoff` threshold. Default `FALSE`.

k

An integer giving the number of top trees when `top_k_trees = TRUE`. Default 10.

cutoff	A numeric or "baseline". Used as the Rashomon cutoff when `top_k_trees = FALSE`. "baseline" uses the objective at $w \equiv 1$ (all weights equal to 1, i.e., no exclusions).
max_depth	Maximum depth of each tree grown during the forest construction stage. A node at `depth == max_depth` is forced to be a leaf. Shallower trees are more interpretable but less flexible. Default 8.
min_leaf_n	Minimum number of observations required in a node for splitting to be attempted. If a node contains fewer than `min_leaf_n` observations it becomes a leaf. Default 2.
max_rejects_per_node	
	Maximum number of consecutive rejected splits (splits that do not improve the objective) allowed at a single node before the node is forced to become a leaf. This prevents infinite recursion in pathological cases. Default `10`.
verbose	`Logical(1)`. If TRUE, prints the unweighted and (when available) weighted treatment effect estimates and standard errors in generalizability mode.

Value

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

D_rash	Data frame containing the Rashomon-set tree votes and the final aggregated weight w_opt for each observation.
D_forest	Data frame with all forest-level working columns.
w_forest	List of per-tree results from the tree-building routine.
rashomon_set	Integer vector of indices identifying which trees were selected into the Rashomon set.
global_objective_fn	
	The objective function used.
f	The characteristic (summary) tree fitted to w_opt, as an rpart object. NULL if all observations received the same weight or no covariates were available.
testing_data	Data frame of observations used for optimization (trial units when `generalizability_path = TRUE`).
estimate	(Only when `generalizability_path = TRUE`) A list with the unweighted (SATE) and weighted (WTATE) point estimates, and standard errors
generalizability_path	
	Logical flag echoing the input.

The optimization problem

ROOT solves the functional optimization problem:

$$w^* \in \arg\min_w L(D_n, w)$$

where $w : \mathbb{R}^p \to \{0, 1\}$ maps a p-dimensional covariate vector to a binary include/exclude decision. The key challenge is that, unlike standard tree algorithms, the global loss $L(D_n, w)$ is not decomposable as a sum of losses over independent subsets of the data. This means conventional greedy, divide-and-conquer tree-building strategies do not apply. ROOT addresses this through a randomization-based tree construction with an explore-exploit strategy.

How ROOT works

The algorithm proceeds in several stages:

1. **Feature importance estimation:** Split probabilities are estimated using Ridge regression, Gradient Boosting Machine (GBM), or a user-supplied function, biasing the search toward covariates likely to be informative.

2. **Stochastic tree construction:** `num_trees` trees are grown. At each internal node, a feature is drawn according to the estimated split probabilities (or a "leaf" token is drawn, terminating the branch). Splits are made at the midpoint of the selected feature's empirical distribution. An explore-exploit strategy assigns leaf weights: with probability `explore_proba` a random weight is chosen; otherwise the greedy optimal weight (reducing the global objective) is used.

3. **Rashomon set selection:** Trees are ranked by their global objective values. The top-k trees (or all trees below a cutoff) form the Rashomon set: a collection of near-optimal but potentially different models, each providing a characterization of the optimal weight function.

4. **Aggregation:** Per-observation votes from the Rashomon set are combined (by default, majority vote) to produce the final weight vector `w_opt`.

5. **Characteristic tree:** A single summary decision tree is fitted to the aggregated `w_opt` assignments, providing a concise, interpretable description of the weight function.

Generalizability mode

When `generalizability_path = TRUE`, ROOT implements the methodology of Parikh et al. (2025) for characterizing underrepresented subgroups in trial-to-target generalizability analyses. In this mode:

- `data` must contain columns `Y` (outcome), `Tr` (treatment, 0/1), and `S` (sample indicator, 1 = trial, 0 = target).

- ROOT internally computes transportability scores based on inverse-probability weighting (IPW), estimates the selection model $P(S = 1 \mid X)$, and constructs Horvitz-Thompson-style influence scores.

- The default objective minimizes the variance of the weighted target average treatment effect (WTATE) estimator. This objective accounts for both the selection odds (trial participation probability) and treatment effect heterogeneity, so that subgroups are flagged as underrepresented only when they both lack trial representation and exhibit effect modification.

- The output includes the unweighted sample average treatment effect (SATE) and the WTATE with standard errors.

See `characterizing_underrep` for a higher-level wrapper that additionally produces a leaf-level summary table, and `vignette("generalizability_path_example")` for a worked example.

General optimization mode

When `generalizability_path = FALSE`, ROOT operates as a general functional optimizer. The user supplies any `data.frame` and (optionally) a custom `global_objective_fn`. If no objective is supplied, ROOT uses a default variance-based loss operating on the `vsq` column (per-unit variance proxy). See `vignette("optimization_path_example")` for an example.

References

Parikh H, Ross RK, Stuart E, Rudolph KE (2025). "Who Are We Missing?: A Principled Approach to Characterizing the Underrepresented Population." *Journal of the American Statistical Association*. doi:10.1080/01621459.2025.2495319

See Also

characterizing_underrep for a higher-level wrapper with leaf-summary output; vignette("generalizability_path_e) for the generalizability workflow; vignette("optimization_path_example") for general optimization.

Examples

```
# --- Generalizability mode ---
data(diabetes_data, package = "ROOT")
root_fit <- ROOT(
  data                 = diabetes_data,
  generalizability_path = TRUE,
  num_trees            = 20,
  top_k_trees          = TRUE,
  k                    = 10,
  seed                 = 123
)
# --- General optimization mode (custom objective) ---
my_objective <- function(D) {
  w <- D$w
  if (sum(w) == 0) return(Inf)
  sqrt(sum(w * D$vsq) / sum(w)^2)
}
set.seed(123)
n_assets <- 100

# Asset features
volatility <- runif(n_assets, 0.05, 0.40)  # annualised volatility
beta       <- runif(n_assets, 0.5,  1.8)   # market beta
sector     <- sample(c("Tech", "Finance", "Energy", "Health"),
                      n_assets, replace = TRUE)

# Simulate returns: r_i = beta_i * r_market + epsilon_i
market      <- rnorm(1000, 0.0005, 0.01)
returns_mat <- sapply(seq_len(n_assets), function(i)
  beta[i] * market + rnorm(1000, 0, volatility[i] / sqrt(252))
)

# Per-asset return variance (the objective proxy ROOT will minimize)
vsq <- apply(returns_mat, 2, var)

my_data <- data.frame(
  vsq    = vsq,
  vol    = volatility,
 beta   = beta,
 sector = as.integer(factor(sector))
```

```
)

opt_fit <- ROOT(
  data               = my_data,
  global_objective_fn = my_objective,
  num_trees          = 20,
  seed               = 42
)
```

---

summary.characterizing_underrep

Summarize a characterizing_underrep fit

Description

Summarizes the ROOT summary which includes unweighted and (when in generalizability 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 readable summary.

Abbreviations

ATE Average treatment effect.

RCT Randomized controlled trial.

SE Standard error.

TATE Transported average treatment effect.

WTATE Weighted transported average treatment effect.

SATE Sample average treatment effect.

Examples

```
char.output = characterizing_underrep(diabetes_data,generalizability_path = TRUE, seed = 123)
summary(char.output)
```

---

summary.ROOT   *Summarize a ROOT fit*

Description

Provides a readable summary of a ROOT object, including:

1. the summary characterization tree f,

2. whether the user supplied a custom global_objective_fn (Yes/No), and

3. in generalizability mode (generalizability_path = TRUE), the unweighted and weighted estimands with their standard errors.

Usage

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

Arguments

object   A "ROOT" S3 object returned by ROOT().

...   Currently unused and included for S3 compatibility.

Value

object returned invisibly. Printed output is for inspection.

Abbreviations

ATE Average treatment effect.

RCT Randomized controlled trial.

SE Standard error.

TATE Transported average treatment effect.

WTATE Weighted transported average treatment effect.

SATE Sample average treatment effect.

When `generalizability_path = 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 `generalizability_path = FALSE`, ROOT is used for general functional optimization and no causal labels are imposed; the summary focuses on the tree and diagnostics.

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

```
ROOT.output = ROOT(diabetes_data,generalizability_path = TRUE, seed = 123)
summary(ROOT.output)
```

Index