Package ‘ROOT’

December 4, 2025

Title Identifying Underrepresented Subpopulations With Interpretable 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. The current implementation focuses on generalizability and transportability problems. Given trial and target data, ROOT learns weighting rules that optimize target treatment effect estimators and helps identify subpopulations that are underrepresented or contribute disproportionately to the variance of the target treatment effect estimate.

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 (wraps* ROOT*)*

Description

Combines an RCT (S = 1) and a target dataset (S = 0), then calls ROOT() to learn a weighted tree that identifies subgroups with different representation in the target population compared to the trial.

Usage

```
characterizing_underrep(
  DataRCT,
  covariateColName_RCT,
  trtColName_RCT,
  outcomeColName_RCT,
  DataTarget,
  covariateColName_TargetData,
  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,
  global_objective_fn = objective_default,
  keep_threshold = 0.5,
  lX_threshold = NULL
)
```

Arguments

DataRCT	A data.frame containing the randomized clinical trial data. Must include treatment, outcome, and covariate columns.
covariateColName_RCT	A character vector of covariate column names in DataRCT.
trtColName_RCT	A character(1) naming the treatment column in DataRCT with values 0 or 1.
outcomeColName_RCT	A character(1) naming the outcome column in DataRCT.
DataTarget	A data.frame containing the target population covariates only.
covariateColName_TargetData	A character vector of covariate column names in DataTarget.

leaf_proba	A numeric(1) giving the probability for the "leaf" option in ROOT tree growth. Default 0.25.
seed	An integer(1) seed for reproducibility. Default 123.
num_trees	An integer(1) number of trees to grow. Default 10.
vote_threshold	A numeric(1) in (0.5, 1] for the majority vote threshold. Default 2/3.
explore_proba	A numeric(1) exploration probability. Default 0.05.
feature_est	Either a character(1) in c("Ridge","GBM") or a function(X, y, ...) that returns a named nonnegative numeric vector of importances.
feature_est_args	A named list of extra arguments passed to the user supplied feature_est function.
top_k_trees	A logical(1). If TRUE, selects top k trees by objective; otherwise uses cutoff. Default FALSE.
k	An integer(1) number of trees used when top_k_trees = TRUE. Default 10.
cutoff	A numeric(1) or the value "baseline" used as the Rashomon set cutoff when top_k_trees = FALSE.
verbose	A logical(1). If TRUE, prints progress and estimand summaries. Default FALSE.
global_objective_fn	A function with signature function(D) -> numeric to minimize. Default objective_default.
keep_threshold	Unused; kept for backward compatibility. A numeric(1) if provided.
lX_threshold	Unused; kept for backward compatibility. A numeric(1) if provided.

Value

A characterizing_underrep S3 object (a list) with components:

root	The resulting ROOT object returned by ROOT().
combined	A data.frame with the stacked RCT and target data used for analysis.
leaf_summary	A data.frame of terminal node summaries with rules, counts, percentages, and labels when a summary tree exists; otherwise NULL.

Abbreviations

RCT means randomized clinical trial. ATE means Average Treatment Effect. WTATE means Weighted Transported ATE. SE means Standard Error.

References

Parikh, H., Ross, R. K., Stuart, E., and Rudolph, K. E. (2025). Who Are We Missing?: A Principled Approach to Characterizing the Underrepresented Population. *Journal of the American Statistical Association*, 1–32.

Examples

```
## Not run:
# Load example data
data(diabetes_data)

# Split into Trial (S=1) and Target (S=0)
trial  <- subset(diabetes_data, S == 1)
target <- subset(diabetes_data, S == 0)

# Run characterization
res <- characterizing_underrep(
  DataRCT = trial,
  covariateColName_RCT = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  trtColName_RCT = "Tr",
  outcomeColName_RCT = "Y",
  DataTarget = target,
  covariateColName_TargetData = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  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 $2^{(0 \cdot \mathrm{depth}/4)}$ present in the code equals 1 and does not change the first element weight. All probabilities are normalized to sum to 1 before sampling.

`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.

`estimate` *Compute pseudo outcome components* `a` *and* `b` *and their product* `v`

Description

Uses nuisance model outputs to compute inverse probability weighted quantities for ATE in the trial sample.

Usage

```
estimate(testing_data, outcome, treatment, sample, pi, pi_m, e_m)
```

Arguments

`testing_data`	A `data.frame` that contains at least outcome, treatment, and sample indicator columns.
`outcome`	A `character(1)` name of the outcome column in `testing_data`.
`treatment`	A `character(1)` name of the treatment column in `testing_data` with values `0` or `1`.
`sample`	A `character(1)` name of the sample indicator column in `testing_data` with values `0` or `1`.

`pi`	A `numeric(1)` giving the estimated prevalence $P(S = 1)$ from the training data.
`pi_m`	A fitted `glm` model for $P(S = 1 \mid X)$.
`e_m`	A fitted `glm` model for $P(Tr = 1 \mid X, S = 1)$.

Details

Predictions from `pi_m` and `e_m` are clamped to `[1e-8, 1 - 1e-8]` for stability.

Value

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

`v`	Pseudo outcome values.
`a`	IPW adjusted outcome contrast.
`b`	Overlap weight factors.

`estimate_dml` *Cross fitted estimation of pseudo outcomes for two sample Double ML*

Description

Trains nuisance models on each training fold and computes pseudo outcomes on the corresponding test fold, then aggregates results. Returns the pseudo outcome table and aligned evaluation data.

Usage

```
estimate_dml(data, outcome, treatment, sample, crossfit = 5)
```

Arguments

`data`	A `data.frame` containing at least the outcome, treatment, and sample indicator columns.
`outcome`	A `character(1)` name of the outcome column.
`treatment`	A `character(1)` name of the treatment column with values `0` or `1`.
`sample`	A `character(1)` name of the sample indicator column with values `0` or `1`.
`crossfit`	An `integer(1)` number of folds for cross fitting where the value is at least `2`. Default `5`.

Value

A `list` with:

`df_v`	`data.frame` with one row per kept observation indexed by `primary_index`. Contains `te`, `a`, `b`, `te_sq`, `a_sq`. Only `S == 1` rows with finite values are kept.
`data2`	`data.frame` subset of the original `data` corresponding to `df_v$primary_index`.

Note

Rows with infinite or undefined weights are removed. Squared deviation columns are centered in the `S == 1` group.

`estimate_dml_single` *Cross fitted Double ML for single sample mode*

Description

Runs K fold cross fitting to produce pseudo outcomes for ATE estimation when no sample membership indicator is available or has no variation.

Usage

```
estimate_dml_single(data, outcome, treatment, crossfit = 5)
```

Arguments

`data`	A `data.frame` containing `outcome`, `treatment`, and covariates.
`outcome`	A `character(1)` name of the outcome column.
`treatment`	A `character(1)` name of the binary treatment indicator column with values `0` or `1`.
`crossfit`	An `integer(1)` number of folds for cross fitting where the value is at least `2`. Default `5`.

Value

A `list` with:

`df_v`	`data.frame` with one row per kept observation that contains `te`, `a`, `b`, `te_sq`, `a_sq`, and `primary_index`.
`data2`	`data.frame` subset of `data` that corresponds to `df_v$primary_index`.

`estimate_single` *Compute single sample pseudo outcomes*

Description

Computes single sample pseudo outcome components for ATE style estimation: $a_i = T_i Y_i / e_i - (1 - T_i) Y_i / (1 - e_i)$ with $v_i = a_i$ and $b_i \equiv 1$.

Usage

```
estimate_single(testing_data, outcome, treatment, e_m)
```

Arguments

`testing_data`	A `data.frame` containing at least `outcome`, `treatment`, and covariates.
`outcome`	A `character(1)` name of the outcome column.
`treatment`	A `character(1)` name of the binary treatment indicator column with values `0` or `1`.
`e_m`	A fitted `glm` model for $P(T = 1 \mid X)$ such as the result of `train_single`.

Value

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

`v`	Pseudo outcome values which equal `a` in single sample mode.
`a`	IPW adjusted outcome contrast.
`b`	Vector of ones because there is no sample overlap weighting in single sample mode.

`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`.

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 $p = \text{plogis}(a)$ where $a = 0.25 - 2 \cdot I\{X0 \in (0.5, 1) \wedge X1 \in (0.5, 1)\}$.

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` and an observational assignment driven by `X0` for `S == 0`.

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`.
`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 $\pi_i = S_i \cdot 0.5 + (1 - S_i) \cdot \text{plogis}(X0_i)$.

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 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)` which introduces heterogeneous treatment effect.

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, ...` up to `X(p-1)`.
`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()`.

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.

Value

A `list` with:

`data`	`data.frame` of length `n` with covariates `X0,...`, sample indicator `S`, treatment indicator `Tr`, and observed outcome `Yobs`.
`Y`	`data.frame` of length `n` with potential outcomes `Y0` and `Y1`.

Examples

```
sim <- get_data(n = 100, seed = 599)
dim(sim$data)
head(sim$data$Yobs)
head(sim$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 $\sqrt{\sum_i \mathrm{vsq}_i \cdot w_i \big/ \left(\sum_i w_i\right)^2}$. 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`.

Usage

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

Arguments

`x`	A `characterizing_underrep` S3 object with `x$root$f` present as an `rpart` object for the summary or characterization tree. The `rpart` model should have classification levels that allow identification of the represented class.
`...`	Additional arguments passed to `rpart.plot::prp()`.

Value

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

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:
# Load example data
data(diabetes_data)

# Split into Trial (S=1) and Target (S=0)
trial  <- subset(diabetes_data, S == 1)
target <- subset(diabetes_data, S == 0)

# Run characterization
res <- characterizing_underrep(
  DataRCT = trial,
  covariateColName_RCT = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  trtColName_RCT = "Tr",
  outcomeColName_RCT = "Y",
  DataTarget = target,
  covariateColName_TargetData = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  seed = 123
)

# Plot the annotated tree
plot(res)

## End(Not run)
```

`plot.ROOT` *Plot the ROOT summary tree*

Description

Visualizes the decision tree that characterizes the weighted subgroup 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 which is the summary or 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:
# Load example data
data(diabetes_data)

# Run ROOT
res <- ROOT(data = diabetes_data, outcome = "Y", treatment = "Tr", sample = "S")

# Plot characterization tree
plot(res)

## End(Not run)
```

`print.characterizing_underrep` *Print a characterizing_underrep fit*

Description

Prints the `ROOT` summary which includes unweighted and weighted estimates with standard errors. The weighted SE is omitted when a custom `global_objective_fn` was used in `ROOT()`. Provides a shorter overview of results.

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. The `ROOT` object may contain `D_forest` which is data frame like, `D_rash` which may include `w_opt`, and `rashomon_set` which is an `integer` vector of selected tree indices.
`...`	Currently unused. Included for S3 compatibility.

Details

Delegates core statistics, number of trees grown and Rashomon size, and the percentage of observations with ensemble vote `w_opt == 1` to `print(x$root)`.

Value

`x` 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:
# Load example data
data(diabetes_data)

# Split into Trial (S=1) and Target (S=0)
trial  <- subset(diabetes_data, S == 1)
target <- subset(diabetes_data, S == 0)

# Run characterization
res <- characterizing_underrep(
  DataRCT = trial,
  covariateColName_RCT = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  trtColName_RCT = "Tr",
  outcomeColName_RCT = "Y",
  DataTarget = target,
  covariateColName_TargetData = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  seed = 123
)

# Print core results
print(res)

## End(Not run)
```

---

print.ROOT *Print a ROOT fit*

Description

Prints a ROOT object by reporting the primary estimands and core diagnostics. The first lines report:

1. the unweighted estimate which is ATE in an RCT for single sample or TATE for two sample and its SE
2. the weighted estimate which is WATE in RCT or WTATE using w_opt and its SE whenever w_opt is effectively binary which means subset mean SE. If w_opt is nonbinary, the SE is omitted with a note.

Subsequent lines include number of trees grown, Rashomon size, and the percentage of observations with ensemble vote w_opt == 1.

Usage

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

Arguments

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

Details

When `x$estimate` is absent, calculations mirror those described in `summary.ROOT`.

Value

`x` 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:
# Load example data
data(diabetes_data)

# Run ROOT
res <- ROOT(data = diabetes_data, outcome = "Y", treatment = "Tr", sample = "S")

# Print core results
print(res)

## 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 (loss/objective agnostic) 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,
  outcome,
  treatment,
  sample,
  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,
  global_objective_fn = objective_default
)
```

Arguments

`data`	A `data.frame` containing the dataset. Must include outcome, treatment, and sample indicator columns.
`outcome`	A `character(1)` specifying the name of the outcome column in `data`.
`treatment`	A `character(1)` specifying the name of the treatment indicator column in `data`. Values should be `0` or `1`.
`sample`	A `character(1)` specifying the name of the sample indicator column in `data` with values `0` or `1`. Use `NULL` for single sample SATE mode.
`leaf_proba`	A `numeric(1)` giving the probability mass for the `"leaf"` option in each tree. Default `0.25`.

seed	An optional `numeric(1)` seed for reproducibility. Default `NULL`.
num_trees	An `integer(1)` giving the number of trees to grow in the forest. Default `10`.
vote_threshold	A `numeric(1)` in `(0.5, 1]` giving the majority vote threshold for final `weight = 1`. Default `2/3`.
explore_proba	A `numeric(1)` giving the probability of exploration at leaves in each tree. 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 where names match columns of `X`.
feature_est_args	A `list` of extra arguments passed to a user supplied `feature_est` function.
top_k_trees	A `logical(1)`. If `TRUE`, select top `k` trees by objective; otherwise use `cutoff`. Default `FALSE`.
k	An `integer(1)` giving the number of top trees used when `top_k_trees = TRUE`. Default `10`.
cutoff	A `numeric(1)` or the `character(1)` value `"baseline"`. Used as the Rashomon set cutoff when `top_k_trees = FALSE`.
verbose	A `logical(1)`. If `TRUE`, prints two lines with unweighted and weighted estimates and their SE. Default `FALSE`.
global_objective_fn	A `function` with signature `function(D) -> numeric` that scores the entire state and is minimized. Default `objective_default`.

Value

An S3 object of class `"ROOT"` which is a `list` with elements including:

D_rash	`data.frame` that contains the Rashomon set votes and `w_opt`.
D_forest	`data.frame` with forest level working columns including `v`, `vsq`, `S`, and the `w_tree_*` columns.
w_forest	`list` of per tree results returned by `split_node()`.
rashomon_set	`integer` vector of selected tree indices.
global_objective_fn	The `function` used for the global objective.
f	An `rpart` object for the summary tree or `NULL` when no covariates exist.
testing_data	`data.frame` aligned to the rows used to compute scores.
estimate	`list` with elements `estimand_unweighted`, `value_unweighted`, `se_unweighted`, `estimand_weighted`, `value_weighted`, `se_weighted`, `se_weighted_note`, `n_analysis`, `sum_w`, `n_A`.

Abbreviations

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

References

Parikh, H., Ross, R. K., Stuart, E., and Rudolph, K. E. (2025). Who Are We Missing?: A Principled Approach to Characterizing the Underrepresented Population. *Journal of the American Statistical Association*, 1–32.

Examples

```
## Not run:
# Load example data
data(diabetes_data)

# Run ROOT
res <- ROOT(data = diabetes_data, outcome = "Y", treatment = "Tr", sample = "S")

## 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"`.

`stratified_kfold` *Stratified K fold index generator*

Description

Splits indices into `K` folds while preserving the class distribution of a binary factor. This mimics the behavior of stratified K fold allocation to keep the ratio of classes in each fold.

Usage

```
stratified_kfold(S, K = 5)
```

Arguments

`S`	A `vector` or `factor` indicating class membership for stratification. Typical values are `0` or `1`.
`K`	An `integer(1)` number of folds. If `K` exceeds the number of observations it is reduced to that number.

Value

A `list` of length `K` where each element is an `integer` vector of row indices assigned to that fold. The union of all folds equals `seq_along(S)` and folds are close in size.

summary.characterizing_underrep

Summarize a characterizing_underrep fit

Description

Summarizes the ROOT summary which includes unweighted and weighted estimates with standard errors. The weighted SE is omitted when a custom `global_objective_fn` was used in `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 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 to `summary(object$root)`. Previews up to ten terminal rules when a summary tree exists and reports plot availability.

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:
# Load example data
data(diabetes_data)

# Split into Trial (S=1) and Target (S=0)
trial  <- subset(diabetes_data, S == 1)
target <- subset(diabetes_data, S == 0)

# Run characterization
res <- characterizing_underrep(
  DataRCT = trial,
  covariateColName_RCT = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  trtColName_RCT = "Tr",
  outcomeColName_RCT = "Y",
  DataTarget = target,
  covariateColName_TargetData = c("Race_Black", "Sex_Male", "DietYes", "Age45"),
  seed = 123
)

# View Summary
summary(res)

## End(Not run)
```

---

summary.ROOT *Summarize a ROOT fit*

Description

Summarizes a ROOT object by reporting the primary estimands and key model diagnostics. The first lines report:

1. the unweighted estimate which is ATE in an RCT for single sample or TATE for two sample and its standard error *SE*
2. the weighted estimate which is WATE in RCT or WTATE using w_opt and its SE whenever w_opt is effectively binary which means subset mean SE. If w_opt is nonbinary, the SE is omitted with a note.

Subsequent lines describe the estimand type, number of trees, size of the Rashomon set, presence of a summary tree, covariate count, observation count, baseline loss, selected tree losses, and the proportion kept by w_opt.

Usage

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

Arguments

object	A ROOT S3 object returned by ROOT(). Expected components include D_forest which is data frame like with columns v, vsq, S, lX, and w_tree_*, D_rash which may contain w_opt, rashomon_set which is integer(), optional estimate which is a list with fields estimand_unweighted, value_unweighted, se_unweighted, estimand_weighted, value_weighted, se_weighted, se_weighted_note, and optional f which is an rpart object.
...	Currently unused and included for S3 compatibility.

Details

This method prefers precomputed estimates in object$estimate. If unavailable, it recomputes:

- Unweighted effect as $\bar{v}$ over the analysis set where the analysis set is all rows in single sample and S == 1 in two sample

- Unweighted SE as $\sqrt{\frac{1}{n(n-1)}\sum(v_i - \bar{v})^2}$ which is `sqrt( sum((v - vbar)^2) / ( n * (n - 1) ) )`
- Weighted effect when `w_opt` is binary with $A = \{i : w_i = 1\}$ which is `A <- which(w == 1)`, that is $\bar{v}_A = \frac{1}{n_A}\sum_{i\in A} v_i$ which is `mean(v[w == 1])`
- Weighted SE which is WTATE or WATE for binary `w_opt` as $\sqrt{\frac{1}{n_A(n_A-1)}\sum_{i\in A}(v_i - \bar{v}_A)^2}$ which is `sqrt( sum( (v[w == 1] - vbarA)^2 ) / ( nA * (nA - 1) ) )`

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:
# Load example data
data(diabetes_data)

# Run ROOT
res <- ROOT(data = diabetes_data, outcome = "Y", treatment = "Tr", sample = "S")

# Summary of results
summary(res)

## End(Not run)
```

`train` *Train nuisance models for weighting*

Description

Fits models to estimate sampling and treatment propensities on training data by logistic regression.

Usage

```
train(training_data, outcome, treatment, sample)
```

Arguments

`training_data`	A `data.frame` that contains the training dataset.
`outcome`	A `character(1)` name of the outcome column. Typically this is the observed outcome such as `"Yobs"`.
`treatment`	A `character(1)` name of the treatment indicator column such as `"Tr"`.
`sample`	A `character(1)` name of the sample inclusion indicator column such as `"S"`.

Value

A `list` with:

`pi`	`numeric(1)` prevalence of `S == 1` in the training data.
`pi_m`	`glm` model with binomial family for $P(S = 1 \mid X)$.
`e_m`	`glm` model with binomial family for $P(Tr = 1 \mid X, S = 1)$.

`train_single` *Train treatment propensity model for single sample mode*

Description

Fits a logistic regression model for $P(T = 1 \mid X)$ on the provided training data. Used by the single sample Double ML path where no sample selection model is required.

Usage

```
train_single(training_data, outcome, treatment)
```

Arguments

`training_data`	A `data.frame` containing the outcome, treatment, and covariates. Only `treatment` and covariates are used for fitting.
`outcome`	A `character(1)` name of the outcome column. Present for a consistent signature and not used here.
`treatment`	A `character(1)` name of the binary treatment indicator column with values `0` or `1`.

Value

A `list` with:

`e_m`	A `glm` model with binomial family for the treatment propensity.

Index