Package: aggTrees (via r-universe)

August 27, 2024

Type Package

Title Aggregation Trees

Version 2.1.0

Description Nonparametric data-driven approach to discovering heterogeneous subgroups in a selection-on-observables framework. 'aggTrees' allows researchers to assess whether there exists relevant heterogeneity in treatment effects by generating a sequence of optimal groupings, one for each level of granularity. For each grouping, we obtain point estimation and inference about the group average treatment effects. Please reference the use as Di Francesco (2022) <doi:10.2139/ssrn.4304256>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

RoxygenNote 7.2.3

Imports boot, broom, car, caret, estimatr, grf, rpart, rpart.plot, stats, stringr

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://riccardo-df.github.io/aggTrees/

BugReports https://github.com/riccardo-df/aggTrees/issues

Repository https://riccardo-df.r-universe.dev

RemoteUrl https://github.com/riccardo-df/aggtrees

RemoteRef HEAD

RemoteSha f59504d50a497e5b7363aa078508ab1c551673e0

Contents

`avg_characteristics_rpart` *Leaves Average Characteristics*

Description

Computes the average characteristics of units in each leaf of an `rpart` object.

Usage

```
avg_characteristics_rpart(tree, X)
```

Arguments

`tree`	An `rpart` object.
`X`	Covariate matrix (no intercept).

Details

`avg_characteristics_rpart` regresses each covariate on a set of dummies denoting leaf membership. This way, we get the average characteristics of units in each leaf, together with a standard error.

Leaves are ordered in increasing order of their predictions (from most negative to most positive).

Standard errors are estimated via the Eicker-Huber-White estimator.

Value

A list storing each regression as an `lm_robust` object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

`causal_ols_rpart`, `estimate_rpart`

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Construct a tree.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame("Y" = Y, X), maxdepth = 2)

## Compute average characteristics in each leaf.
results <- avg_characteristics_rpart(tree, X)
results
```

---

balance_measures *Balance Measures*

Description

Compute several balance measures to check whether the covariate distributions are balanced across treatment arms.

Usage

```
balance_measures(X, D)
```

Arguments

X	Covariate matrix (no intercept).
D	Treatment assignment vector.

Details

For each covariate in X, balance_measures computes sample averages and standard deviations for both treatment arms. Additionally, two balance measures are computed:

Norm. Diff. Normalized differences, computed as the differences in the means of each covariate across treatment arms, normalized by the sum of the within-arm variances. They provide a measure of the discrepancy between locations of the covariate distributions across treatment arms.

Log S.D. Log ratio of standard deviations are computed as the logarithm of the ratio of the within-arm standard deviations. They provide a measure of the discrepancy in the dispersion of the covariate distributions across treatment arms.

Compilation of the LATEX code requires the following packages: booktabs, float, adjustbox.

Value

Prints LATEX code in the console.

Author(s)

Elena Dal Torrione, Riccardo Di Francesco

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Print table.
balance_measures(X, D)
```

build_aggtree *Aggregation Trees*

Description

Nonparametric data-driven approach to discovering heterogeneous subgroups in a selection-on-observables framework. The approach constructs a sequence of groupings, one for each level of granularity. Groupings are nested and feature an optimality property. For each grouping, we obtain point estimation and standard errors for the group average treatment effects (GATEs) using debiased machine learning procedures. Additionally, we assess whether systematic heterogeneity is found by testing the hypotheses that the differences in the GATEs across all pairs of groups are zero. Finally, we investigate the driving mechanisms of effect heterogeneity by computing the average characteristics of units in each group.

Usage

```
build_aggtree(
  Y_tr,
  D_tr,
  X_tr,
  Y_hon = NULL,
  D_hon = NULL,
  X_hon = NULL,
  cates_tr = NULL,
  cates_hon = NULL,
  method = "aipw",
  scores = NULL,
  ...
)

inference_aggtree(object, n_groups, boot_ci = FALSE, boot_R = 2000)
```

Arguments

Y_tr	Outcome vector for training sample.
D_tr	Treatment vector for training sample.
X_tr	Covariate matrix (no intercept) for training sample.
Y_hon	Outcome vector for honest sample.
D_hon	Treatment vector for honest sample.
X_hon	Covariate matrix (no intercept) for honest sample.
cates_tr	Optional, predicted CATEs for training sample. If not provided by the user, CATEs are estimated internally via a causal_forest.
cates_hon	Optional, predicted CATEs for honest sample. If not provided by the user, CATEs are estimated internally via a causal_forest.
method	Either "raw" or "aipw", controls how node predictions are computed.

scores	Optional, vector of scores to be used in computing node predictions. Useful to save computational time if scores have already been estimated. Ignored if method == "raw".
...	Further arguments from rpart.control.
object	An aggTrees object.
n_groups	Number of desired groups.
boot_ci	Logical, whether to compute bootstrap confidence intervals.
boot_R	Number of bootstrap replications. Ignored if boot_ci == FALSE.

Details

Aggregation trees are a three-step procedure. First, the conditional average treatment effects (CATEs) are estimated using any estimator. Second, a tree is grown to approximate the CATEs. Third, the tree is pruned to derive a nested sequence of optimal groupings, one for each granularity level. For each level of granularity, we can obtain point estimation and inference about the GATEs.

To implement this methodology, the user can rely on two core functions that handle the various steps.

Constructing the Sequence of Groupings:

build_aggtree constructs the sequence of groupings (i.e., the tree) and estimate the GATEs in each node. The GATEs can be estimated in several ways. This is controlled by the method argument. If method == "raw", we compute the difference in mean outcomes between treated and control observations in each node. This is an unbiased estimator in randomized experiment. If method == "aipw", we construct doubly-robust scores and average them in each node. This is unbiased also in observational studies. Honest regression forests and 5-fold cross fitting are used to estimate the propensity score and the conditional mean function of the outcome (unless the user specifies the argument scores).

The user can provide a vector of the estimated CATEs via the cates_tr and cates_hon arguments. If no CATEs are provided, these are estimated internally via a causal_forest using only the training sample, that is, Y_tr, D_tr, and X_tr.

GATEs Estimation and Inference:

inference_aggtree takes as input an aggTrees object constructed by build_aggtree. Then, for the desired granularity level, chosen via the n_groups argument, it provides point estimation and standard errors for the GATEs. Additionally, it performs some hypothesis testing to assess whether we find systematic heterogeneity and computes the average characteristics of the units in each group to investigate the driving mechanisms.

Point estimates and standard errors for the GATEs:
GATEs and their standard errors are obtained by fitting an appropriate linear model. If method == "raw", we estimate via OLS the following:

$$Y_i = \sum_{l=1}^{|T|} L_{i,l}\gamma_l + \sum_{l=1}^{|T|} L_{i,l}D_i\beta_l + \epsilon_i$$

with L_{i, l} a dummy variable equal to one if the i-th unit falls in the l-th group, and |T| the number of groups. If the treatment is randomly assigned, one can show that the betas identify the GATE of each group. However, this is not true in observational studies due to selection into treatment. In this case, the user is expected to use method == "aipw" when calling build_aggtree. In this case, inference_aggtree uses the scores in the following regression:

$$score_i = \sum_{l=1}^{|T|} L_{i,l}\beta_l + \epsilon_i$$

This way, betas again identify the GATEs.

Regardless of method, standard errors are estimated via the Eicker-Huber-White estimator.

If boot_ci == TRUE, the routine also computes asymmetric bias-corrected and accelerated 95% confidence intervals using 2000 bootstrap samples. Particularly useful when the honest sample is small-ish.

Hypothesis testing:
inference_aggtree uses the standard errors obtained by fitting the linear models above to test the hypotheses that the GATEs are different across all pairs of leaves. Here, we adjust p-values to account for multiple hypotheses testing using Holm's procedure.

Average Characteristics:
inference_aggtree regresses each covariate on a set of dummies denoting group membership. This way, we get the average characteristics of units in each leaf, together with a standard error. Leaves are ordered in increasing order of their predictions (from most negative to most positive). Standard errors are estimated via the Eicker-Huber-White estimator.

Caution on Inference:

Regardless of the chosen method, both functions estimate the GATEs, the linear models, and the average characteristics of units in each group using only observations in the honest sample. If the honest sample is empty (this happens when the user either does not provide Y_hon, D_hon, and X_hon or sets them to NULL), the same data used to construct the tree are used to estimate the above quantities. This is fine for prediction but invalidates inference.

Value

build_aggtree returns an aggTrees object.

inference_aggtree returns an aggTrees.inference object, which in turn contains the aggTrees object used in the call.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

plot.aggTrees print.aggTrees.inference

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Training-honest sample split.
honest_frac <- 0.5
splits <- sample_split(length(Y), training_frac = (1 - honest_frac))
training_idx <- splits$training_idx
honest_idx <- splits$honest_idx

Y_tr <- Y[training_idx]
D_tr <- D[training_idx]
X_tr <- X[training_idx, ]

Y_hon <- Y[honest_idx]
D_hon <- D[honest_idx]
X_hon <- X[honest_idx, ]

## Construct sequence of groupings. CATEs estimated internally.
groupings <- build_aggtree(Y_tr, D_tr, X_tr, # Training sample.
                           Y_hon, D_hon, X_hon) # Honest sample.

## Alternatively, we can estimate the CATEs and pass them.
library(grf)
forest <- causal_forest(X_tr, Y_tr, D_tr) # Use training sample.
cates_tr <- predict(forest, X_tr)$predictions
cates_hon <- predict(forest, X_hon)$predictions

groupings <- build_aggtree(Y_tr, D_tr, X_tr, # Training sample.
                           Y_hon, D_hon, X_hon, # Honest sample.
                           cates_tr, cates_hon) # Predicted CATEs.

## We have compatibility with generic S3-methods.
summary(groupings)
print(groupings)
plot(groupings) # Try also setting 'sequence = TRUE'.

## To predict, do the following.
```

```
tree <- subtree(groupings$tree, cv = TRUE) # Select by cross-validation.
head(predict(tree, data.frame(X_hon)))

## Inference with 4 groups.
results <- inference_aggtree(groupings, n_groups = 4)

summary(results$model) # Coefficient of leafk is GATE in k-th leaf.

results$gates_diff_pairs$gates_diff # GATEs differences.
results$gates_diff_pairs$holm_pvalues # leaves 1-2 not statistically different.

## LATEX.
print(results, table = "diff")
print(results, table = "avg_char")
```

`causal_ols_rpart` *Estimation and Inference about the GATEs with rpart Objects*

Description

Obtains point estimates and standard errors for the group average treatment effects (GATEs), where groups correspond to the leaves of an `rpart` object. Additionally, performs some hypothesis testing.

Usage

```
causal_ols_rpart(
  tree,
  Y,
  D,
  X,
  method = "aipw",
  scores = NULL,
  boot_ci = FALSE,
  boot_R = 2000
)
```

Arguments

`tree`	An `rpart` object.
`Y`	Outcome vector.
`D`	Treatment assignment vector
`X`	Covariate matrix (no intercept).
`method`	Either `"raw"` or `"aipw"`, defines the outcome used in the regression.
`scores`	Optional, vector of scores to be used in the regression. Useful to save computational time if scores have already been estimated. Ignored if `method == "raw"`.
`boot_ci`	Logical, whether to compute bootstrap confidence intervals.
`boot_R`	Number of bootstrap replications. Ignored if `boot_ci == FALSE`.

Details

Point estimates and standard errors for the GATEs:

The GATEs and their standard errors are obtained by fitting an appropriate linear model. If `method == "raw"`, we estimate via OLS the following:

$$Y_i = \sum_{l=1}^{|T|} L_{i,l}\gamma_l + \sum_{l=1}^{|T|} L_{i,l}D_i\beta_l + \epsilon_i$$

with `L_{i, l}` a dummy variable equal to one if the i-th unit falls in the l-th leaf of `tree`, and |T| the number of groups. If the treatment is randomly assigned, one can show that the betas identify the GATE in each leaf. However, this is not true in observational studies due to selection into treatment. In this case, the user is expected to use `method == "aipw"` to run the following regression:

$$score_i = \sum_{l=1}^{|T|} L_{i,l}\beta_l + \epsilon_i$$

where score_i are doubly-robust scores constructed via honest regression forests and 5-fold cross fitting (unless the user specifies the argument `scores`). This way, betas again identify the GATEs.

Regardless of `method`, standard errors are estimated via the Eicker-Huber-White estimator.

If `boot_ci == TRUE`, the routine also computes asymmetric bias-corrected and accelerated 95% confidence intervals using 2000 bootstrap samples.

If `tree` consists of a root only, `causal_ols_rpart` regresses `y` on a constant and `D` if `method == "raw"`, or regresses the doubly-robust scores on a constant if `method == "aipw"`. This way, we get an estimate of the overall average treatment effect.

Hypothesis testing:

`causal_ols_rpart` uses the standard errors obtained by fitting the linear models above to test the hypotheses that the GATEs are different across all pairs of leaves. Here, we adjust p-values to account for multiple hypotheses testing using Holm's procedure.

Caution on Inference:

"honesty" is a necessary requirement to get valid inference. Thus, observations in `Y`, `D`, and `X` must not have been used to construct the `tree` and the `scores`.

Value

A list storing:

`model` The model fitted to get point estimates and standard errors for the GATEs, as an `lm_robust` object.

gates_diff_pairs
: Results of testing whether GATEs differ across all pairs of leaves. This is a list storing GATEs differences and p-values adjusted using Holm's procedure (check p.adjust). NULL if the tree consists of a root only.

boot_ci
: Bootstrap confidence intervals (this is an empty list if boot_ci == FALSE.

scores
: Vector of doubly robust scores. NULL if method == 'raw'.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

estimate_rpart avg_characteristics_rpart

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Split the sample.
splits <- sample_split(length(Y), training_frac = 0.5)
training_idx <- splits$training_idx
honest_idx <- splits$honest_idx

Y_tr <- Y[training_idx]
D_tr <- D[training_idx]
X_tr <- X[training_idx, ]

Y_hon <- Y[honest_idx]
D_hon <- D[honest_idx]
X_hon <- X[honest_idx, ]

## Construct a tree using training sample.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame("Y" = Y_tr, X_tr), maxdepth = 2)
```

```
## Estimate GATEs in each node (internal and terminal) using honest sample.
results <- causal_ols_rpart(tree, Y_hon, D_hon, X_hon, method = "raw")

summary(results$model) # Coefficient of leafk:D is GATE in k-th leaf.

results$gates_diff_pair$gates_diff # GATEs differences.
results$gates_diff_pair$holm_pvalues # leaves 1-2 and 3-4 not statistically different.
```

`dr_scores` *Doubly-Robust Scores*

Description

Constructs doubly-robust scores via K-fold cross-fitting.

Usage

```
dr_scores(Y, D, X, k = 5)
```

Arguments

`Y`	Outcome vector.
`D`	Treatment assignment vector.
`X`	Covariate matrix (no intercept).
`k`	Number of folds.

Details

Honest regression forests are used to estimate the propensity score and the conditional mean function of the outcome.

Value

A vector of scores.

Author(s)

Riccardo Di Francesco

estimate_rpart *GATE Estimation with rpart Objects*

Description

Replaces node predictions of an rpart object using external data to estimate the group average treatment effects (GATEs).

Usage

```
estimate_rpart(tree, Y, D, X, method = "aipw", scores = NULL)
```

Arguments

tree	An rpart object.
Y	Outcome vector.
D	Treatment assignment vector.
X	Covariate matrix (no intercept).
method	Either "raw" or "aipw", controls how node predictions are replaced.
scores	Optional, vector of scores to be used in replacing node predictions. Useful to save computational time if scores have already been estimated. Ignored if method == "raw".

Details

If method == "raw", estimate_rpart replaces node predictions with the differences between the sample average of the observed outcomes of treated units and the sample average of the observed outcomes of control units in each node, which is an unbiased estimator of the GATEs if the assignment to treatment is randomized.

If method == "aipw", estimate_rpart replaces node predictions with sample averages of doubly-robust scores in each node. This is a valid estimator of the GATEs in observational studies. Honest regression forests and 5-fold cross fitting are used to estimate the propensity score and the conditional mean function of the outcome (unless the user specifies the argument scores).

estimate_rpart allows the user to implement "honest" estimation. If observations in y, D and X have not been used to construct the tree, then the new predictions are honest in the sense of Athey and Imbens (2016). To get standard errors for the tree's estimates, please use causal_ols_rpart.

Value

A tree with node predictions replaced, as an rpart object, and the scores (if method == "raw", this is NULL).

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

causal_ols_rpart avg_characteristics_rpart

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Split the sample.
splits <- sample_split(length(Y), training_frac = 0.5)
training_idx <- splits$training_idx
honest_idx <- splits$honest_idx

Y_tr <- Y[training_idx]
D_tr <- D[training_idx]
X_tr <- X[training_idx, ]

Y_hon <- Y[honest_idx]
D_hon <- D[honest_idx]
X_hon <- X[honest_idx, ]

## Construct a tree using training sample.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame("Y" = Y_tr, X_tr), maxdepth = 2)

## Estimate GATEs in each node (internal and terminal) using honest sample.
new_tree <- estimate_rpart(tree, Y_hon, D_hon, X_hon, method = "raw")
new_tree$tree
```

expand_df *Covariate Matrix Expansion*

Description

Expands the covariate matrix, adding interactions and polynomials. This is particularly useful for penalized regressions.

Usage

```
expand_df(X, int_order = 2, poly_order = 4, threshold = 0)
```

Arguments

X	Covariate matrix (no intercept).
int_order	Order of interactions to be added. Set equal to one if no interactions are desired.
poly_order	Order of the polynomials to be added. Set equal to one if no polynomials are desired.
threshold	Drop binary variables representing less than threshold% of the population. Useful to speed up computation.

Details

expand_df assumes that categorical variables are coded as factors. Also, no missing values are allowed.

expand_df uses model.matrix to expand factors to a set of dummy variables. Then, it identifies continuous covariates as those not having 0 and 1 as unique values.

expand_df first introduces all the int_order-way interactions between the variables (using the expanded set of dummies), and then adds poly_order-order polynomials for continuous covariates.

Value

The expanded covariate matrix, as a data frame.

Author(s)

Riccardo Di Francesco

get_leaves *Number of Leaves*

Description

Extracts the number of leaves of an rpart object.

Usage

```
get_leaves(tree)
```

Arguments

tree	An rpart object.

Value

The number of leaves.

Author(s)

Riccardo Di Francesco

See Also

subtree node_membership leaf_membership

Examples

```
## Generate data.
set.seed(1986)

n <- 3000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))

Y <- exp(X[, 1]) + 2 * X[, 2] * X[, 2] > 0 + rnorm(n)

## Construct tree.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame(Y, X))

## Extract number of leaves.
n_leaves <- get_leaves(tree)
n_leaves
```

leaf_membership *Leaf Membership*

Description

Constructs a variable that encodes in which leaf of an rpart object the units in a given data frame fall.

Usage

```
leaf_membership(tree, X)
```

Arguments

tree	An rpart object.
X	Covariate matrix (no intercept).

Value

A factor whose levels denote in which leaf each unit falls. Leaves are ordered in increasing order of their predictions (from most negative to most positive).

Author(s)

Riccardo Di Francesco

See Also

subtree node_membership get_leaves

Examples

```
## Generate data.
set.seed(1986)

n <- 3000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))

Y <- exp(X[, 1]) + 2 * X[, 2] * X[, 2] > 0 + rnorm(n)

## Construct tree.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame(Y, X))

## Extract number of leaves.
leaves_factor <- leaf_membership(tree, X)
```

```
head(leaves_factor)
```

---

node_membership *Node Membership*

Description

Constructs a binary variable that encodes whether each observation falls into a particular node of an rpart object.

Usage

```
node_membership(tree, X, node)
```

Arguments

tree	An rpart object.
X	Covariate matrix (no intercept).
node	Number of node.

Value

Logical vector denoting whether each observation in X falls into node.

Author(s)

Riccardo Di Francesco

See Also

subtree leaf_membership get_leaves

Examples

```
## Generate data.
set.seed(1986)

n <- 3000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))

Y <- exp(X[, 1]) + 2 * X[, 2] * X[, 2] > 0 + rnorm(n)

## Construct tree.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame(Y, X))
```

```
## Extract number of leaves.
is_in_third_node <- node_membership(tree, X, 3)
head(is_in_third_node)
```

`plot.aggTrees` *Plot Method for aggTrees Objects*

Description

Plots an `aggTrees` object.

Usage

```
## S3 method for class 'aggTrees'
plot(x, leaves = get_leaves(x$tree), sequence = FALSE, ...)
```

Arguments

`x`	An `aggTrees` object.
`leaves`	Number of leaves of the desired tree. This can be used to plot subtrees.
`sequence`	If `TRUE`, the whole sequence of optimal groupings is displayed in a short animation.
`...`	Further arguments from `prp`.

Details

Nodes are colored using a diverging palette. Nodes with predictions smaller than the ATE (i.e., the root prediction) are colored in blue shades, and nodes with predictions larger than the ATE are colored in red shades. Moreover, predictions that are more distant in absolute value from the ATE get darker shades. This way, we have an immediate understanding of the groups with extreme GATEs.

Value

Plots an `aggTrees` object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

build_aggtree, inference_aggtree

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Training-honest sample split.
honest_frac <- 0.5
splits <- sample_split(length(Y), training_frac = (1 - honest_frac))
training_idx <- splits$training_idx
honest_idx <- splits$honest_idx

Y_tr <- Y[training_idx]
D_tr <- D[training_idx]
X_tr <- X[training_idx, ]

Y_hon <- Y[honest_idx]
D_hon <- D[honest_idx]
X_hon <- X[honest_idx, ]

## Construct sequence of groupings. CATEs estimated internally.
groupings <- build_aggtree(Y_tr, D_tr, X_tr,
                           Y_hon, D_hon, X_hon)

## Plot.
plot(groupings)
plot(groupings, leaves = 3)
plot(groupings, sequence = TRUE)
```

---

print.aggTrees *Print Method for aggTrees Objects*

Description

Prints an aggTrees object.

Usage

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

Arguments

x	aggTrees object.
...	Further arguments passed to or from other methods.

Value

Prints an aggTrees object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

build_aggtree, inference_aggtree

---

print.aggTrees.inference *Print Method for aggTrees.inference Objects*

Description

Prints an aggTrees.inference object.

Usage

```
## S3 method for class 'aggTrees.inference'
print(x, table = "avg_char", ...)
```

Arguments

x	aggTrees.inference object.
table	Either "avg_char" or "diff", controls which table must be produced.
...	Further arguments passed to or from other methods.

Details

A description of each table is provided in its caption.

Some covariates may feature zero variation in some leaf. This generally happens to dummy variables used to split some nodes. In this case, when `table == "avg_char"` a warning message is produced displaying the names of the covariates with zero variation in one or more leaves. The user should correct the table by removing the associated standard errors.

Compilation of the LATEX code requires the following packages: `booktabs`, `float`, `adjustbox`, `multirow`.

Value

Prints LATEX code.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

`build_aggtree`, `inference_aggtree`

Examples

```
## Generate data.
set.seed(1986)

n <- 1000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))
D <- rbinom(n, size = 1, prob = 0.5)
mu0 <- 0.5 * X[, 1]
mu1 <- 0.5 * X[, 1] + X[, 2]
Y <- mu0 + D * (mu1 - mu0) + rnorm(n)

## Training-honest sample split.
honest_frac <- 0.5
splits <- sample_split(length(Y), training_frac = (1 - honest_frac))
training_idx <- splits$training_idx
honest_idx <- splits$honest_idx

Y_tr <- Y[training_idx]
```

```
D_tr <- D[training_idx]
X_tr <- X[training_idx, ]

Y_hon <- Y[honest_idx]
D_hon <- D[honest_idx]
X_hon <- X[honest_idx, ]

## Construct sequence of groupings. CATEs estimated internally.
groupings <- build_aggtree(Y_tr, D_tr, X_tr,
                           Y_hon, D_hon, X_hon)

## Analyze results with 4 groups.
results <- inference_aggtree(groupings, n_groups = 4)

## Print results.
print(results, table = "diff")
print(results, table = "avg_char")
```

`sample_split` *Sample Splitting*

Description

Splits the sample into training and honest subsamples.

Usage

```
sample_split(n, training_frac = 0.5)
```

Arguments

`n`	Size of the sample to be split.
`training_frac`	Fraction of units for the training sample.

Value

A list storing the indexes for the two different subsamples.

Author(s)

Riccardo Di Francesco

subtree *Subtree*

Description

Extracts a subtree with a user-specified number of leaves from an rpart object.

Usage

```
subtree(tree, leaves = NULL, cv = FALSE)
```

Arguments

tree	An rpart object.
leaves	Number of leaves of the desired subtree.
cv	If TRUE, leaves is ignored and a cross-validation criterion is used to select a partition.

Value

The subtree, as an rpart object.

Author(s)

Riccardo Di Francesco

See Also

get_leaves node_membership leaf_membership

Examples

```
## Generate data.
set.seed(1986)

n <- 3000
k <- 3

X <- matrix(rnorm(n * k), ncol = k)
colnames(X) <- paste0("x", seq_len(k))

Y <- exp(X[, 1]) + 2 * X[, 2] * X[, 2] > 0 + rnorm(n)

## Construct tree.
library(rpart)
tree <- rpart(Y ~ ., data = data.frame(Y, X), cp = 0)

## Extract subtree.
sub_tree <- subtree(tree, leaves = 4)
```

```
sub_tree_cv <- subtree(tree, cv = TRUE)
```

---

summary.aggTrees *Summary Method for aggTrees Objects*

Description

Summarizes an aggTrees object.

Usage

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

Arguments

object	aggTrees object.
...	Further arguments passed to or from other methods.

Value

Prints the summary of an aggTrees object.

Author(s)

Riccardo Di Francesco

References

- Di Francesco, R. (2022). Aggregation Trees. CEIS Research Paper, 546. doi:10.2139/ssrn.4304256.

See Also

build_aggtree, inference_aggtree

Index