# Package ‘semtree’

November 26, 2023

**Type** Package  

**Title** Recursive Partitioning for Structural Equation Models  

**Author** Andreas M. Brandmaier [aut, cre],  
John J. Prindle [aut],  
Manuel Arnold [aut],  
Caspar J. Van Lissa [aut]  

**Maintainer** Andreas M. Brandmaier &lt;andy@brandmaier.de&gt;  

**Depends** R (&gt;= 2.10), OpenMx (&gt;= 2.6.9),  

**Imports** bitops, rpart, rpart.plot (&gt;= 3.0.6), cluster, stringr,  
lavaan, ggplot2, tidyr, methods, strucchange, sandwich, zoo,  
crayon, clisymbols, future.apply, data.table, ctsemOMX, expm,  
gridBase  

**Suggests** knitr, rmarkdown, viridis, MASS, psychTools, testthat, future  

**Description** SEM Trees and SEM Forests -- an extension of model-based decision  
trees and forests to Structural Equation Models (SEM). SEM trees hierarchically  
split empirical data into homogeneous groups each sharing similar data patterns  
with respect to a SEM by recursively selecting optimal predictors of these  
differences. SEM forests are an extension of SEM trees. They are ensembles of  
SEM trees each built on a random sample of the original data. By aggregating  
over a forest, we obtain measures of variable importance that are more robust  
than measures from single trees. A description of the method was published by  
Brandmaier, von Oertzen, McArdle, & Lindenberger (2013) &lt;doi:10.1037/a0030001&gt;  

**License** GPL-3  

**Encoding** UTF-8  

**LazyLoad** yes  

**Version** 0.9.19  

**Date** 2023-11-24  

**RoxygenNote** 7.2.3  

**VignetteBuilder** knitr  

**BugReports** https://github.com/brandmaier/semtree/issues
URL: https://github.com/brandmaier/semtree
Language: en-US
NeedsCompilation: no
Repository: CRAN
Date/Publication: 2023-11-26 14:20:02 UTC

R topics documented:

- semtree-package .................................................. 3
- biodiversity ......................................................... 3
- coef.semtree .......................................................... 4
- computePval_maxLR .................................................. 4
- diversityMatrix ...................................................... 5
- evaluate ............................................................... 5
- evaluateDataLikelihood ............................................. 6
- evaluateTree ......................................................... 7
- findOtherSplits ...................................................... 8
- fitSubmodels ......................................................... 8
- getDepth ............................................................... 9
- getHeight ............................................................ 10
- getLeaves ............................................................ 10
- getNodeById ......................................................... 11
- getNumNodes ......................................................... 11
- getParDiffForest .................................................... 12
- getParDiffTree ...................................................... 13
- getTerminalNodes ................................................... 13
- isLeaf ............................................................... 14
- kl ................................................................. 14
- lgcm ............................................................... 15
- merge.semforest ..................................................... 15
- modelEstimates ..................................................... 16
- outliers ............................................................. 16
- parameters ......................................................... 17
- partialDependence ................................................ 18
- partialDependence_data .......................................... 19
- partialDependence_growth ....................................... 20
- plotParDiffForest .................................................. 21
- plotParDiffTree ..................................................... 22
- plotTreeStructure ................................................. 22
- predict.semforest ................................................ 23
- proximity ........................................................... 24
- prune ............................................................... 25
- se ................................................................. 26
- semforest ........................................................... 27
- semforest.control .................................................. 28
- semtree ............................................................. 28
**Description**

SEM Tree Package

**Usage**

```r
.SCALE_METRIC
```

**Format**

An object of class numeric of length 1.

---

**biodiversity**

*Quantify bio diversity of a SEM Forest*

**Description**

A function to calculate biodiversity of a `semforest` object.

**Usage**

```r
biodiversity(x, aggregate.fun = median)
```

**Arguments**

- `x` A `semforest` object
- `aggregate.fun` Takes a function to apply to the vector of pairwise diversities. By default, this is the median.

**Author(s)**

Andreas M. Brandmaier
computePval_maxLR

---

**coef.semtree**

Return the parameter estimates of a given leaf of a SEM tree

**Description**

Return the parameter estimates of a given leaf of a SEM tree

**Usage**

```r
## S3 method for class 'semtree'
coef(object, ...)
```

**Arguments**

- `object`: semtree. A SEM tree node.
- `...`: Extra arguments. Currently unused.

@exportS3Method coef semtree

---

**computePval_maxLR**

Wrapper function for computing the maxLR corrected p value from strucchange

**Description**

Wrapper function for computing the maxLR corrected p value from strucchange

**Usage**

```r
computePval_maxLR(maxLR, q, covariate, from, to, nrep)
```

**Arguments**

- `maxLR`: maximum of the LR test statistics
- `q`: number of free SEM parameters / degrees of freedom
- `covariate`: covariate under evaluation. This is important to get the level of measurement from the covariate and the bin size for ordinal and categorical covariates.
- `from`: numeric from interval (0, 1) specifying start of trimmed sample period. With the default from = 0.15 the first and last 15 percent of observations are trimmed. This is only needed for continuous covariates.
- `to`: numeric from interval (0, 1) specifying end of trimmed sample period. By default, to is 1.
- `nrep`: numeric. Number of replications used for simulating from the asymptotic distribution (passed to efpFunctional). Only needed for ordinal covariates.
**diversityMatrix**

**Value**

Numeric. p value for maximally selected LR statistic

**Author(s)**

Manuel Arnold

---

**diversityMatrix**

* Diversity Matrix

**Description**

Computes a diversity matrix using a distance function between trees

**Usage**

```
diversityMatrix(forest, divergence = klsym, showProgressBar = TRUE)
```

**Arguments**

- **forest**
  - A SEM forest
- **divergence**
  - A divergence function such as hellinger or klsym
- **showProgressBar**
  - Boolean. Show a progress bar.

---

**evaluate**

* Average Deviance of a Dataset given a Forest

**Description**

Evaluates the average deviance (-2LL) of a dataset given a forest.

**Usage**

```
evaluate(x, data = NULL, ...)
```

**Arguments**

- **x**
  - A fitted `semforest` object
- **data**
  - A data.frame
- **...**
  - No extra parameters yet.

**Value**

Average deviance
Author(s)
Andreas M. Brandmaier

References

See Also
evaluateDataLikelihood, semtree, semforest

evaluateDataLikelihood

*Compute the Negative Two-Loglikelihood of some data given a model (either OpenMx or lavaan)*

Description
This helper function is used in the semforest varimp and proximity aggregate functions.

Usage
```r
evaluateDataLikelihood(model, data, data_type = "raw")
```

Arguments
- `model`: A OpenMx model as used in semtree and semforest.
- `data`: Data set to apply to a fitted model.
- `data_type`: Type of data ("raw", "cov", "cor")

Value
Returns a -2LL model fit for the model

Author(s)
Andreas M. Brandmaier, John J. Prindle

References

See Also
semtree, semforest
evaluateTree

**Evaluate Tree -2LL**

**Description**

A helper function to evaluate the negative two log-likelihood (-2LL) of leaf (terminal) nodes for a dataset. When given a `semtree` and a unique dataset, the model estimates -2LL for the tree parameters and data subsets that fit the tree branching criteria.

**Usage**

```r
evaluateTree(tree, test_set, data_type = "raw", leaf_ids = NULL)
```

**Arguments**

- `tree`: A fitted `semtree` object
- `test_set`: Dataset to fit to a fitted `semtree` object
- `data_type`: type of data ("raw", "cov", "cor")
- `leaf_ids`: Identifies which nodes are leaf nodes. Default is NULL, which checks model for leaf nodes and fills this information in automatically.

**Value**

A list with two elements:

- `deviance`: Combined -2LL for leaf node models of the tree.
- `num_models`: Number of leaf nodes used for the deviance calculations.

**Author(s)**

Andreas M. Brandmaier, John J. Prindle

**References**


**See Also**

- `evaluateDataLikelihood`, `semtree`, `semforest`
Find Other Node Split Values

Description

Search tool to search nodes for alternative splitting values found during the semtree process. Given a particular node, competing split values are listed assuming they also meet the criteria for a significant splitting value as set by semtree.control.

Usage

findOtherSplits(node, tree)

Arguments

node A node from a semtree object.
tree A semtree object which the node is part of.

Value

A data.frame() with rows corresponding to the variable names and split values for alternative splits found in the node of interest. ...

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


Fit multigroup model for evaluating a candidate split

Description

Fit multigroup model for evaluating a candidate split
getDepth

Usage

```r
fitSubmodels(
  model,
  subset1,
  subset2,
  control,
  invariance = NULL,
  return.models = FALSE
)
```

Arguments

- `model`: A model specification that is used as template for each of the two groups
- `subset1`: Dataset for the first group model
- `subset2`: Dataset for the second group model
- `control`: a `semtree.control` object
- `invariance`: fit models with invariant parameters if given. NULL otherwise (default).
- `return.models`: boolean. Return the fitted models returns NA if fit fails

getDepth

*Get the depth (or, height) a tree.*

Description

Returns the length of the longest path from a root node to a leaf node.

Usage

```r
getDepth(tree)
```

Arguments

- `tree`: A `semtree` object

Author(s)

Andreas M. Brandmaier

References

**getHeight**  
*Determine Height of a Tree*

**Description**
Returns height of a SEM Tree, which equals to the length of the longest path from root to a terminal node.

**Usage**
```r
getHeight(tree)
```

**Arguments**
- **tree**
  A SEM tree.

**Author(s)**
Andreas M. Brandmaier, John J. Prindle

**References**

---

**getLeafs**  
*Get a list of all leaves in a tree*

**Description**
Get a list of all leaves in a tree by recursively searching the tree starting at the given node (if not data object is given. If data is given, the function returns the leaves that are predicted for each row of the given data.

**Usage**
```r
getLeafs(tree, data = NULL)
```

**Arguments**
- **tree**
  A `semtree` object
- **data**
  A `data.frame`

**Author(s)**
Andreas M. Brandmaier
**getNodeById**

**Description**

Return a node matching a given node ID

**Usage**

```r
getNodeById(tree, id)
```

**Arguments**

- `tree` A SEM Tree object.
- `id` Numeric. A Node id.

**Author(s)**

Andreas M. Brandmaier, John J. Prindle

**References**


---

**getNumNodes**

**Description**

Counts the number of nodes in a tree.

**Usage**

```r
getNumNodes(tree)
```

**Arguments**

- `tree` A SEM tree object.
getParDiffForest

Description

Returns a list of tables with some measure of parameter differences between post-split nodes.

Usage

getParDiffForest(forest, measure = "wald", normalize = FALSE)

Arguments

- **forest**: a semforest object.
- **measure**: a character. "wald" (default) gives the squared parameter differences divided by their pooled standard errors. "test" gives the contributions of the parameters to the test statistic. "raw" gives the absolute values of the parameter differences.
- **normalize**: logical value; if TRUE parameter differences of each split are divided by sum of all differences the corresponding split. Set to FALSE by default.

Value

A list with data.frames containing parameter differences for each tree of the forest. The rows of the data.frames correspond to the non-leaf nodes of the respective trees. The first column contains the name of the predictor variables and the remaining columns contain the parameter differences. The rows of the data.frames are named by the node IDs as given by getNodeById and the columns are named as in coef.

Author(s)

Manuel Arnold
getParDiffTree  

Return table with parameter differences of a tree

Description

Returns a table with some measure of parameter differences between post-split nodes.

Usage

getParDiffTree(tree, measure = "wald", normalize = FALSE)

Arguments

tree  a semtree object.
measure a character. "wald" (default) gives the squared parameter differences divided by their pooled standard errors. "test" gives the contributions of the parameters to the test statistic."raw" gives the absolute values of the parameter differences.
normalize logical value; if TRUE parameter differences of each split are divided by sum of all differences the corresponding split. Set to FALSE by default.

Value

A matrix containing parameter differences. The matrix has \( n \) rows and \( k \) columns, where \( n \) is the number of non-leaf nodes of the tree and \( k \) is the number of model parameters. The rows are named by the node IDs as given getNodeById and the columns are named as in coef.

Author(s)

Manuel Arnold

getTerminalNodes  

Returns all leafs of a tree

Description

Returns all leafs (=terminal nodes) of a tree.

Usage

getTerminalNodes(tree)

Arguments

tree  A semtree object.
Author(s)

Andreas M. Brandmaier, John J. Prindle

References


isLeaf

Test whether a semtree object is a leaf.

Description

Tests whether a semtree object is a leaf. Returns TRUE or FALSE.

Usage

isLeaf(tree)

Arguments

tree A semtree object

Author(s)

Andreas M. Brandmaier

References


kl

Distances

Description

Divergence measures for multivariate normal distributions as used in the diversityMatrix function.

Usage

kl(mu1, cov1, mu2, cov2)
lgcm

Arguments
mu1      Mean vector
cov1     Covariance matrix
mu2      Mean vector
cov2     Covariance matrix

Description
This data set provides simple data to fit with a LGCM.

Format
lgcm is a matrix containing 400 rows and 8 columns of simulated data. Longitudinal observations are o1-o5. Covariates are agegroup, training, and noise.

Author(s)
Andreas M. Brandmaier <brandmaier@mpib-berlin.mpg.de>

merge.semforest

Description
This overrides generic base::merge() to merge two forests into one.

Usage
## S3 method for class 'semforest'
merge(x, y, ...)

Arguments
x      A SEM Forest
y      A second SEM Forest
...    Extra arguments. Currently unused.

Author(s)
Andreas M. Brandmaier, John J. Prindle
References


See Also

semtree

modelEstimates

Description

Return model estimates of the tree.

Usage

modelEstimates(tree, ...)

Arguments

- `tree`: A semtree object.
- `...`: Optional arguments.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


outliers

Description

Find outliers based on case proximity.

Usage

outliers(prox)
parameters

Arguments

prox A proximity matrix.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

proximity

descriptions SEMtrees Parameter Estimates Table

Description

Returns a table of parameters with columns corresponding to freely estimated parameters and rows corresponding to nodes in the tree.

Usage

parameters(tree, leafs.only = TRUE)

Arguments

tree A SEMtree object obtained from semtree
leafs.only Default = TRUE. Only the terminal nodes (leafs) are printed. If set to FALSE, all node parameters are written to the data.frame.

Details

The row names of the resulting data frame correspond to internal node ids and the column names correspond to parameters in the SEM. Standard errors of the estimates can be obtained from parameters.

Value

Returns a data.frame with rows for parameters and columns for terminal nodes.

Author(s)

Andreas M. Brandmaier, John J. Prindle
References


See Also

`semtree`, `semtree.control`, `se`

partialDependence  
Compute partial dependence

Description

Compute the partial dependence of a predictor, or set of predictors, on a model parameter.

Usage

partialDependence(
  x,
  data,
  reference.var,
  support = 20,
  points = NULL,
  mc = NULL,
  FUN = "median",
  ...
)

Arguments

x  
An object for which a method exists

data  
Optional data.frame that was used to train the model.

reference.var  
Character vector, referring to the (independent) reference variable or variables for which partial dependence is calculated. Providing two (or more) variables allows for probing interactions, but note that this is computationally expensive.

support  
Integer. Number of grid points for interpolating the reference.var. Alternatively, use points for one or more variables named in reference.var.

points  
Named list, with elements corresponding to reference.var. Use this argument to provide specific points for which to obtain marginal dependence values; for example, the mean and +/- 1SD of reference.var.

mc  
Integer. If mc is not NULL, the function will sample mc number of rows from data with replacement, to estimate marginal dependency using Monte Carlo integration. This is less computationally expensive.

FUN  
Character string with function used to integrate predictions across all elements of x.

...  
Extra arguments passed to FUN.
partialDependence_data

Create dataset to compute partial dependence

Description

Create a dataset with fixed values for reference.var for all other values of data, or using mc random samples from data (Monte Carlo integration).

Usage

```r
partialDependence_data(
  data,
  reference.var,
  support = 20,
  points = NULL,
  mc = NULL,
  keep_id = FALSE
)
```

Arguments

- **data**: The data.frame that was used to train the model.
- **reference.var**: Character vector, referring to the (independent) reference variable or variables for which partial dependence is calculated. Providing two (or more) variables allows for probing interactions, but note that this is computationally expensive.
- **support**: Integer. Number of grid points for interpolating the reference.var. Alternatively, use points for one or more variables named in reference.var.
- **points**: Named list, with elements corresponding to reference.var. Use this argument to provide specific points for which to obtain marginal dependence values; for example, the mean and +/- 1SD of reference.var.
- **mc**: Integer. If mc is not NULL, the function will sample mc number of rows from data with replacement, to estimate marginal dependency using Monte Carlo integration. This is less computationally expensive.
- **keep_id**: Boolean. Default is false. Should output contain a row id column?

Author(s)

Caspar J. Van Lissa
partialDependence_growth

*Compute partial dependence for latent growth models*

**Description**

Compute the partial dependence of a predictor, or set of predictors, on the predicted trajectory of a latent growth model.

**Usage**

```r
partialDependence_growth(
  x, data, reference.var, support = 20, points = NULL, mc = NULL, 
  FUN = "median", times = NULL, parameters = NULL, ...
)
```

**Arguments**

- `x`: An object for which a method exists
- `data`: Optional data.frame that was used to train the model.
- `reference.var`: Character vector, referring to the (independent) reference variable or variables for which partial dependence is calculated. Providing two (or more) variables allows for probing interactions, but note that this is computationally expensive.
- `support`: Integer. Number of grid points for interpolating the `reference.var`. Alternatively, use points for one or more variables named in `reference.var`.
- `points`: Named list, with elements corresponding to `reference.var`. Use this argument to provide specific points for which to obtain marginal dependence values; for example, the mean and +/- 1SD of `reference.var`.
- `mc`: Integer. If `mc` is not NULL, the function will sample `mc` number of rows from `data` with replacement, to estimate marginal dependency using Monte Carlo integration. This is less computationally expensive.
- `FUN`: Character string with function used to integrate predictions across all elements of `x`.
- `times`: Numeric matrix, representing the factor loadings of a latent growth model, with columns equal to the number of growth parameters, and rows equal to the number of measurement occasions.
parameters  Character vector of the names of the growth parameters; defaults to NULL, which assumes that the growth parameters are the only parameters and are in the correct order.

Extra arguments passed to FUN.

Author(s)

Caspar J. Van Lissa

---

**Description**

Visualizes parameter differences between post-split nodes in a forest with boxplots.

**Usage**

```r
plotParDiffForest(
  forest,
  plot = "boxplot",
  measure = "wald",
  normalize = FALSE,
  predictors = NULL,
  title = TRUE
)
```

**Arguments**

- **forest**
  - a semforest object.
- **plot**
  - a character that specifies the plot type. Available plot types are "boxplot" (default) and "jitter" for a jittered strip plot with mean and standard deviation.
- **measure**
  - a character. "wald" (default) gives the squared parameter differences divided by their pooled standard errors. "test" gives the contributions of the parameters to the test statistic. "raw" gives the absolute values of the parameter differences.
- **normalize**
  - logical value; if TRUE parameter differences of each split are divided by sum of all differences the corresponding split. Set to FALSE by default.
- **predictors**
  - a character. Select predictors that are to be plotted.
- **title**
  - logical value; if TRUE a title is added to the plot.

**Author(s)**

Manuel Arnold
### plotTreeStructure

**Description**

Plots the structure of a semtree object. This function is similar to `plot.semtree`, but it does not print the parameter values in the leaf nodes and labels the leaf nodes instead.

### plotParDiffTree

**Description**

Visualizes parameter differences between post-split nodes with different plot types.

**Usage**

```r
plotParDiffTree(
  tree,  # a semtree object.
  plot = "ballon",  # a character that specifies the plot typ. Available plot types are "ballon" (default), "heatmap", and "bar".
  measure = "wald",  # a character. "wald" (default) gives the squared parameter differences divided by their pooled standard errors. "test" gives the contributions of the parameters to the test statistic. "raw" gives the absolute values of the parameter differences.
  normalize = FALSE,  # logical value; if TRUE parameter differences of each split are divided by sum of all differences the corresponding split. Set to FALSE by default.
  title = TRUE,  # logical value; if TRUE a title is added to the plot.
  structure = FALSE  # logical value; if TRUE the structure of the tree is plotted on the right side.
)
```

**Arguments**

- `tree`: a semtree object.
- `plot`: a character that specifies the plot typ. Available plot types are "ballon" (default), "heatmap", and "bar".
- `measure`: a character. "wald" (default) gives the squared parameter differences divided by their pooled standard errors. "test" gives the contributions of the parameters to the test statistic. "raw" gives the absolute values of the parameter differences.
- `normalize`: logical value; if TRUE parameter differences of each split are divided by sum of all differences the corresponding split. Set to FALSE by default.
- `title`: logical value; if TRUE a title is added to the plot.
- `structure`: logical value; if TRUE the structure of the tree is plotted on the right side.

### Author(s)

Manuel Arnold
predict.semforest

Usage

plotTreeStructure(tree, type = 2, no.plot = FALSE, ...)

Arguments

tree a semtree object.
type Type of plot. See prp from rpart.plot.
no.plot logical value; if TRUE structure of the tree is printed to the console.
... additional arguments passed to prp from rpart.plot.

Author(s)

Manuel Arnold

predict.semforest Predict method for semtree and semforest

Description

Predict method for semtree and semforest

Usage

## S3 method for class 'semforest'
predict(object, data, type = "node_id", ...)

Arguments

object Object of class semtree or 'semforest'.
data New test data of class data.frame. If no data is provided, attempts to extract the data from the object.
type Type of prediction. One of 'c('node_id')'. See Details.
... further arguments passed to or from other methods.

Value

Object of class matrix.

Author(s)

Caspar J. van Lissa, Andreas Brandmaier
proximity

Compute proximity matrix

Description

Compute a n by n matrix across all trees in a forest, where n is the number of rows in the data, reflecting the proportion of times two cases ended up in the same terminal node of a tree.

Usage

proximity(x, data, ...)

Arguments

x An object for which a method exists.
data A data.frame on which proximity is computed
... Parameters passed to other functions.

Details

SEM Forest Case Proximity

Value

A matrix with dimensions [i, j] whose elements reflect the proportion of times case i and j were in the same terminal node of a tree.

Author(s)

Caspar J. Van Lissa, Andreas M. Brandmaier, John J. Prindle

References


See Also

`semforest`, `semtree`

Examples

```
nodeids <- structure(c(9, 3, 5, 7, 10, 4, 6, 8, 9, 3, 5, 7, 10, 4, 6, 8),
  .Dim = c(4L, 4L))
class(nodeids) <- "semforest_node_id"
sims <- proximity(nodeids)
sims <- as.dist(1-sims)
hc <- hclust(dd)
groups <- cutree(hc, 2)
```
Prune a SEM Tree or SEM Forest

Description

Returns a new tree with a maximum depth selected by the user. can be used in conjunction with plot commands to view various pruning levels.

Usage

prune(object, ...)

Arguments

object A semtree or semforest object.

... Optional parameters, such as max.depth the maximum depth of each tree, or also num.trees when pruning a forest.

Details

The returned tree is only modified by the number of levels for the tree. This function does not reevaluate the data, but provides alternatives to reduce tree complexity. If the user would like to alter the tree by increasing depth, then max.depth option must be adjusted in the semtree.control object (provided further splits are able to be computed).

Value

Returns a semtree object.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

semtree, semtree.control
se

SEMtrees Parameter Estimates Standard Error Table

Description

Returns a table of standard errors with columns corresponding to freely estimated standard errors and rows corresponding to nodes in the tree.

Usage

se(tree, leafs.only = TRUE)

Arguments

- **tree**: A SEMtree object obtained from `semtree`
- **leafs.only**: Default = TRUE. Only the terminal nodes (leafs) are printed. If set to FALSE, all node standard errors are written to the data.frame.

Details

The row names of the resulting data frame correspond to internal node ids and the column names correspond to standard errors in the SEM. Parameter estimates can be obtained from `parameters`.

Value

Returns a data.frame with rows for parameters and columns for terminal nodes.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

`semtree, semtree.control, parameters`
Create a SEM Forest

Description

Grows a SEM Forest from a template model and a dataset. This may take some time.

Usage

```r
semforest(
  model,  
  data,  
  control = NULL,  
  predictors = NULL,  
  constraints = NULL,  
  ...
)
```

Arguments

- `model` A template SEM. Same as in `semtree`.
- `data` A dataframe to create a forest from. Same as in `semtree`.
- `control` A semforest control object to set forest parameters.
- `predictors` An optional list of covariates. See semtree code example.
- `constraints` An optional list of covariates. See semtree code example.
- `...` Optional parameters.

Value

A semforest object.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

`semtree`
semforest.control  SEM Forest Control Object

Description
A SEM Forest control object to tune parameters of the forest learning algorithm.

Usage
semforest.control(
  num.trees = 5,
  sampling = "subsample",
  control = NA,
  mtry = 2,
  remove_dead_trees = TRUE
)

Arguments
num.trees  Number of trees.
sampling  Sampling procedure. Can be subsample or bootstrap.
control  A SEM Tree control object. Will be generated by default.
mtry  Number of subsampled covariates at each node.
remove_dead_trees  Remove trees from forest that had runtime errors

Author(s)
Andreas M. Brandmaier, John J. Prindle

References

semtree  SEM Tree: Recursive Partitioning for Structural Equation Models

Description
Structural equation model (SEM) trees are a combination of SEM and decision trees (also known as classification and regression trees or recursive partitioning). SEM trees hierarchically split empirical data into homogeneous groups sharing similar data patterns with respect to a SEM by recursively selecting optimal predictors of these differences from a potentially large set of predictors.
Usage

semtree(
  model,
  data = NULL,
  control = NULL,
  constraints = NULL,
  predictors = NULL,
  ...
)

Arguments

model A template model specification from OpenMx using the mxModel function (or a lavaan model using the lavaan function with option fit=FALSE). Model must be syntactically correct within the framework chosen, and converge to a solution.
data Data.frame used in the model creation using mxModel or lavaan are input here. Order of modeled variables and predictors is not important when providing a dataset to semtree.
control semtree model specifications from semtree.control are input here. Any changes from the default setting can be specified here.
constraints A semtree.constraints object setting model parameters as constrained from the beginning of the semtree computation. This includes options to globally or locally set equality constraints and to specify focus parameters (i.e., parameter subsets that exclusively go into the function evaluating splits). Also, options for measurement invariance testing in trees are included.
predictors A vector of variable names matching variable names in dataset. If NULL (default) all variables that are in dataset and not part of the model are potential predictors. Optional function input to select a subset of the unmodeled variables to use as predictors in the semtree function.

... Optional arguments passed to the tree growing function.

Details

Calling semtree with an OpenMx or lavaan model creates a tree that recursively partitions a dataset such that the partitions maximally differ with respect to the model-predicted distributions. Each resulting subgroup (represented as a leaf in the tree) is represented by a SEM with a distinct set of parameter estimates.

Predictors (yet unmodeled variables) can take on any form for the splitting algorithm to function (categorical, ordered categories, continuous). Care must be taken in choosing how many predictors to include in analyses because as the number of categories grows for unordered categorical variables, the number of multigroup comparisons increases exponentially for unordered categories.

Currently available evaluation methods for assessing partitions:
1. "naive" selection method compares all possible split values to one another over all predictors included in the dataset.
2. "fair" selection uses a two step procedure for analyzing split values on predictors at each node of the tree. The first phase uses half of the sample to examine the model improvement for each split
value on each predictor, and retains the value that presents the largest improvement for each
predictor. The second phase then evaluates these best split points for each predictor on the second
half of the sample. The best improvement for the c splits tested on c predictors is selected for the
node and the dataset is split from this node for further testing.
3. "score" uses score-based test statistics. These statistics are much faster than the classic SEM tree
approach while having favorable statistical properties.
All other parameters controlling the tree growing process are available through a separate `semtree.control`
object.

Value

A semtree object is created which can be examined with `summary`, `plot`, and `print`.

Author(s)

Andreas M. Brandmaier, John J. Prindle, Manuel Arnold

References

model trees. Psychological Methods, 18(1), 71-86.

See Also

`semtree.control`, `summary.semtree`, `parameters`, `se`, `prune.semtree`, `subtree`, `OpenMx`, `lavaan`

Description

A SEM Tree constraints object holds information regarding specifics on how the tree is grown
(similar to the control object). The SEM tree control object holds all information that is independent
of a specific model whereas the constraints object holds information that is specific to a certain
model (e.g., specifies differential treatment of certain parameters, e.g., by holding them constant
across the forest).

Usage

```r
semtree.constraints(
  local.invariance = NULL,
  global.invariance = NULL,
  focus.parameters = NULL
)
```
Arguments

local.invariance
Vector of parameter names that are locally equal, that is, they are assumed to be equal when assessing a local split but allowed to differ subsequently.

global.invariance
Vector of parameter names that are globally equal, that is, estimated only once and then fixed in the tree.

focus.parameters
Vector of parameter names that exclusively are evaluated for between-group differences when assessing split candidates. If NULL all parameters add to the difference.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

semtree

Description

A semtree.control object contains parameters that determine the tree growing process. These parameters include choices of different split candidate selection procedures and hyperparameters of those. Calling the constructor without parameters creates a default control object. A number of tree growing methods are included in with this package: 1. 'naive' splitting takes the best split value of all possible splits on each covariate. 2. 'fair' selection is so called because it tests all splits on half of the data, then tests the best split value for each covariate on the other half of the data. The equal footing of each covariate in this two phase test removes bias from testing variables with many possible splits compared to those with few. 3. "fair3" does the phases described above, with an additional step of retesting all of the split values on the best covariate found in the second phase. Variations in the sample from subsetting are removed and bias in split selection further reduced. 4. 'score' implements modern score-based statistics.
Usage

```r
semtree.control(
    method = "naive",
    min.N = 20,
    max.depth = NA,
    alpha = 0.05,
    alpha.invariance = NA,
    folds = 5,
    exclude.heywood = TRUE,
    progress.bar = TRUE,
    verbose = FALSE,
    bonferroni = FALSE,
    use.all = FALSE,
    seed = NA,
    custom.stopping.rule = NA,
    mtry = NA,
    report.level = 0,
    exclude.code = NA,
    linear = TRUE,
    min.bucket = 10,
    naive.bonferroni.type = 0,
    missing = "ignore",
    use.maxlm = FALSE,
    strucchange.from = 0.15,
    strucchange.to = NULL,
    strucchange.nrep = 50000,
    refit = TRUE,
    ctsem_sd = FALSE
)
```

Arguments

- `method` Default: 'naive'. One out of c("score","fair","naive") for either an unbiased two-step selection algorithm, a naive take-the-best, or a score-based testing scheme.
- `min.N` Default: 10. Minimum sample size per a node, used to determine whether to continue splitting a tree or establish a terminal node.
- `max.depth` Default: NA. Maximum levels per a branch. Parameter for limiting tree growth.
- `alpha` Default: 0.05. Significance level for splitting at a given node.
- `alpha.invariance` Default: NA. Significance level for invariance tests. If NA, the value of alpha is used.
- `folds` Default: 5. Defines the number of folds for the "cv" method.
- `exclude.heywood` Default: TRUE. Reports whether there is an identification problem in the covariance structure of an SEM tested.
- `progress.bar` Default: NA. Option to disable the progress bar for tree growth.
### semtree.control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>FALSE</td>
<td>Option to turn on or off all model messages during tree growth.</td>
</tr>
<tr>
<td>bonferroni</td>
<td>FALSE</td>
<td>Correct for multiple tests with Bonferroni type correction.</td>
</tr>
<tr>
<td>use.all</td>
<td></td>
<td>Treatment of missing variables. By default, missing values stay in a decision node. If TRUE, cases are distributed according to a maximum likelihood principle to the child nodes.</td>
</tr>
<tr>
<td>seed</td>
<td>NA</td>
<td>Set a random number seed for repeating random fold generation in tree analysis.</td>
</tr>
<tr>
<td>custom.stopping.rule</td>
<td>NA</td>
<td>Otherwise, this can be a boolean function with a custom stopping rule for tree growing.</td>
</tr>
<tr>
<td>mtry</td>
<td>NA</td>
<td>Number of sample columns to use in SEMforest analysis.</td>
</tr>
<tr>
<td>report.level</td>
<td>0</td>
<td>Values up to 99 can be used to increase the number of onscreen reports for semtree analysis.</td>
</tr>
<tr>
<td>exclude.code</td>
<td>NA</td>
<td>NPSOL error code for exclusion from model fit evaluations when finding best split. Models with errors during fitting are retained.</td>
</tr>
<tr>
<td>linear</td>
<td></td>
<td>If TRUE (default), the structural equation model is assumed to not contain any nonlinear parameter constraints and scores are computed analytically, resulting in a shorter runtime. Only relevant for models fitted with OpenMx.</td>
</tr>
<tr>
<td>min.bucket</td>
<td></td>
<td>Minimum bucket size. This is the minimum size any node must have, such that a given split is considered valid. Minimum bucket size is a lower bound to the sample size in the terminal nodes of a tree.</td>
</tr>
<tr>
<td>naive.bonferroni.type</td>
<td>0</td>
<td>When set to zero, bonferroni correction for the naive test counts the number of dichotomous tests. When set to one, bonferroni correction counts the number of variables tested.</td>
</tr>
<tr>
<td>missing</td>
<td></td>
<td>Missing value treatment. Default is ignore</td>
</tr>
<tr>
<td>use.maxlm</td>
<td></td>
<td>Use MaxLR statistic for split point selection (as proposed by Arnold et al., 2021)</td>
</tr>
<tr>
<td>strucchange.from</td>
<td></td>
<td>Strucchange argument. See their package documentation.</td>
</tr>
<tr>
<td>strucchange.to</td>
<td></td>
<td>Strucchange argument. See their package documentation.</td>
</tr>
<tr>
<td>strucchange.nrep</td>
<td></td>
<td>Strucchange argument. See their package documentation.</td>
</tr>
<tr>
<td>refit</td>
<td></td>
<td>If TRUE (default) the initial model is fitted on the data provided to semtree.</td>
</tr>
<tr>
<td>ctsem_sd</td>
<td></td>
<td>If FALSE (default) no standard errors of CT model parameters are computed. Requesting standard errors increases runtime.</td>
</tr>
</tbody>
</table>

#### Value

A control object containing a list of the above parameters.

#### Author(s)

Andreas M. Brandmaier, John J. Prindle, Manuel Arnold
References


See Also

`semtree`

Examples

```r
# create a control object with an alpha level of 1%
my.control <- semtree.control(alpha=0.01)

# set the minimum number of cases per node to ten
my.control$min.N <- 10

# print contents of the control object
print(my.control)
```

strip

<table>
<thead>
<tr>
<th>Retain only basic tree structure</th>
</tr>
</thead>
</table>

Description

Removes all elements of a semforest or semtree except for the tree structure and terminal node parameters. This is to reduce the heavy memory footprint of sem trees and forests.

Usage

`strip(x, parameters = NULL)`

Arguments

- `x` An object for which a method exists.
- `parameters` Character vector, referencing parameters in the SEM model. Defaults to `NULL`, in which case all free model parameters are returned.
subforest

Details

Objects of class semforest and semtree are very large, which complicates downstream operations such as making partial dependence plots, or using the model in interactive contexts (like Shiny apps). Running strip removes all elements of the model except for the tree structure and terminal node parameters. Note that some methods are no longer available for the resulting object - e.g., varimp requires the terminal node SEM models to compute the likelihood ratio.

Value

List

Examples

```r
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

### subforest

*Creates subsets of trees from forests*

Description

Creates subsets of a forest. This can be used to subset a number of trees, e.g. from:(from+num), or to remove all null (type="nonnull") trees that were due to errors, or to randomly select a sub forest (type=random).

Usage

```
subforest(forest, num = NULL, type = "nonnull", from = 1)
```

Arguments

- **forest**: A SEM Forest object.
- **num**: Number of trees to select.
- **type**: Either 'random' or 'nonnull' or NULL. First selects a random subset, second selects all non-null trees, third allows subsetting trees.
- **from**: Starting index if type=NULL.

Author(s)

Andreas M. Brandmaier, John J. Prindle
References

---

subtree

**SEMtree Partitioning Tool**

Description
The subtree function returns a tree from a selected node of the semtree returned tree.

Usage
```r
subtree(tree, startNode = NULL, level = 0, foundNode = FALSE)
```

Arguments
- `tree`: A SEMtree object obtained from *semtree*
- `startNode`: Node id, which will be future root node (0 to max node number of tree)
- `level`: Ignore. Only used internally.
- `foundNode`: Ignore. Only used internally.

Details
The row names of the resulting data frame correspond to internal node ids and the column names correspond to standard errors in the SEM. Standard errors of the estimates can be obtained from *se*.

Value
Returns a *semtree* object which is a partitioned tree from the input semtree.

Author(s)
Andreas M. Brandmaier, John J. Prindle

References

See Also
*semtree, semtree.control*
### toTable

**Tabular Representation of a SEM Tree**

**Description**

Converts a tree into a tabular representation. This may be useful as a textual representation for use in manuscripts.

**Usage**

```r
toTable(tree, added.param.cols = NULL, round.param = NULL)
```

**Arguments**

- `tree`: A SEM Tree object.
- `added.param.cols`: String. Add extra columns with parameter estimates. Pass a vector with the names of the parameters that should be rendered in the table.
- `round.param`: Integer. Number of digits to round parameter estimates. Default is no rounding (NULL).

**Author(s)**

Andreas M. Brandmaier

**References**


### varimp

**SEM Forest Variable Importance**

**Description**

A function to calculate relative variable importance for selecting node splits over a `semforest` object.
Usage

```r
varimp(
  forest,
  var.names = NULL,
  verbose = F,
  eval.fun = evaluateTree,
  method = "permutation",
  conditional = FALSE,
  ...
)
```

Arguments

- **forest**: A `semforest` object
- **var.names**: Covariates used in the forest creation process. NULL value will be automatically filled in by the function.
- **verbose**: Boolean to print messages while function is running.
- **eval.fun**: Default is `evaluateTree` function. The value of the -2LL of the leaf nodes is compared to baseline overall model.
- **method**: Experimental. Some alternative methods to compute importance. Default is "permutation".
- **conditional**: Conditional variable importance if TRUE, otherwise marginal variable importance.
- **...**: Optional arguments.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References

Index

* biodiversity
  biodiversity, 3
* datasets
  lgcm, 15
  semtree-package, 3
* models
  semforest, 27
  semtree, 28
* multivariate
  semforest, 27
  semtree, 28
* semforest
  biodiversity, 3
* tree
  semforest, 27
  semtree, 28
  .SCALE_METRIC (semtree-package), 3
  biodiversity, 3
  coef.semtree, 4
  computePval_maxLR, 4
  diversityMatrix, 5
  evaluate, 5
  evaluateDataLikelihood, 6, 6, 7
  evaluateTree, 7, 38
  findOtherSplits, 8
  fitSubmodels, 8
  getDepth, 9
  getHeight, 10
  getLeafs, 10
  getNodeById, 11
  getNumNodes, 11
  getParDiffForest, 12
  getParDiffTree, 13
  getTerminalNodes, 13
  hellinger (kl), 14
  isLeaf, 14
  kl, 14
  klsym (kl), 14
  lavaan, 29, 30
  lgcm, 15
  merge.semforest, 15
  modelEstimates, 16
  mxModel, 29
  nodeFunSemtree (semtree), 28
  OpenMx, 6, 29, 30
  outliers, 16
  parameters, 17, 17, 26, 30
  partialDependence, 18
  partialDependence_data, 19
  partialDependence_growth, 20
  plot.semforest (semforest), 27
  plot.semforest.varimp (varimp), 37
  plot.semtree (semtree), 28
  plotParDiffForest, 21
  plotParDiffTree, 22
  plotTreeStructure, 22
  predict.semforest, 23
  print.semforest (semforest), 27
  print.semforest.control (semtree.control), 28
  print.semforest.varimp (varimp), 37
  print.semtree (semtree), 28
  print.semtree.control (semtree.control), 31
  proximity, 6, 17, 24
  prune, 25
  prune.semtree, 30
se, 18, 26, 30, 36
semforest, 3, 5–7, 24, 27, 37, 38
semforest.control, 28
semforest_score_control
  (semforest.control), 28
semtree, 6–10, 14, 16–18, 24–27, 28, 29, 31,
  33, 34, 36
semtree-package, 3
semtree.constraints, 29, 30
semtree.control, 8, 18, 25, 26, 29, 30, 31, 36
strip, 34
subforest, 35
subtree, 30, 36
summary.semtree, 30
summary.semtree (semtree), 28
toLatex.semtree (semtree), 28
toTable, 37
varimp, 6, 35, 37
varimpConvergencePlot (varimp), 37