Package ‘semtree’

January 7, 2020

Type Package

Title Recursive Partitioning for Structural Equation Models

Author Andreas M. Brandmaier [aut, cre],
John J. Prindle [aut],
Manuel Arnold [aut]

Maintainer Andreas M. Brandmaier <andy@brandmaier.de>

Depends R (>= 2.10), OpenMx (>= 2.6.9)

Imports bitops, sets, digest, rpart, rpart.plot (>= 3.0.6), parallel,
plotrix, cluster, stringr, matrixcalc, expm, ggplot2, tidyr,
matrixStats, methods, MASS, mvtnorm

Suggests lavaan, knitr, rmarkdown

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 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.

License GPL-3

Encoding UTF-8

LazyLoad yes

Version 0.9.14

Date 2020-01-07

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-07 22:00:02 UTC
R topics documented:

- biodiversity ......................................................... 2
- Distances .............................................................. 3
- diversityMatrix ...................................................... 3
- evaluate ................................................................. 4
- evaluateDataLikelihood ............................................. 4
- evaluateTree .......................................................... 6
- findOtherSplits ....................................................... 7
- getDepth ................................................................. 7
- getHeight ............................................................... 8
- getLeafs ................................................................. 8
- getNodeById ........................................................... 9
- getNumNodes .......................................................... 10
- getTerminalNodes .................................................... 10
- isLeaf ................................................................. 11
- lgcm ................................................................. 11
- merge ................................................................. 12
- modelEstimates ....................................................... 12
- outliers ............................................................... 13
- parameters ........................................................... 14
- partialDependence .................................................. 15
- proximity .............................................................. 15
- prune ................................................................. 16
- se.semmtree .......................................................... 17
- semforest ............................................................ 18
- semforest.control .................................................... 19
- semtree ............................................................... 20
- semtree.constraints .................................................. 21
- semtree.control ....................................................... 22
- subforest ............................................................. 24
- subtree ............................................................... 25
- toTable ............................................................... 26
- varimp .............................................................. 26

Index ................................................................. 28

biodiversity .......................................................... Quantify biodiversity of a SEM Forest

Description

A function to calculate biodiversity of a semforest object.

Usage

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

Description

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

Usage

```r
klsym(mu1, cov1, mu2, cov2)
hellinger(mu1, cov1, mu2, cov2)
```

Arguments

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

```
diversityMatrix  Diversity Matrix
```

Description

Computes a diversity matrix using a distance function between trees

Usage

```r
diversityMatrix(forest, divergence, showProgressbar=TRUE)
```

Arguments

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

evaluateDataLikelihood

---

**Description**

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

**Usage**

```r
evaluate(x, data, ...)
```

**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`

---

**Description**

Evaluate Fitted Model Likelihood for Dataset. This helper function is used in the *semforest varimp* and *proximity* aggregate functions.

**Usage**

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

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

Examples

## Should be DIRECTLY executable !! ----
##-- ==> Define data, use random, 
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (model, data, data_type = "raw")
{
  model <- omxSetParameters(model, labels = names(omxGetParameters(model)),
  free = F)
  if (data_type == "raw") {
    data <- full_mxdata <- mxData(observed = data, type = "raw")
  }
  else if (data_type == "cov") {
    data <- full_mxdata <- mxData(observed = cov(data), type = "cov",
    numObs = dim(data)[1])
  }
  else if (data_type == "cor") {
    data <- full_mxdata <- mxData(observed = cor(data), type = "cor",
    numObs = dim(data)[1])
  }
  else {
    warning("data type is not supported!")
    return(NULL)
  }
  model <- setData(model, data)
  run <- OpenMx::mxRun(model, silent = T, useOptimizer = F,
  suppressWarnings = T)
evaluateTree <- function(tree, test_set, data_type = "raw", leaf_ids = NULL) {
  result <- OpenMx::mxEval(objective, run)
  return(result)
}

evaluateTree

Evaluate Tree -2LL

Description
A helper function to evaluate the -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
`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`
findOtherSplits

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


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

`getDepth(tree)`

Arguments

- `tree`: A `semtree` object
Author(s)
Andreas M. Brandmaier

References

getLeafs
Get a list of all leafs in a tree

Description
Get a list of all leafs 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 leafs that are predicted for each row of the given data.

Usage
getLeafs(tree, data=NULL)
**getNodeById**

**Arguments**

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

**Author(s)**

Andreas M. Brandmaier

**References**


---

**getNodeById**  

**Get Node By Id**

**Description**

Return a node matching a given node ID

**Usage**

`getNodeById(tree, id)`

**Arguments**

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

**Author(s)**

Andreas M. Brandmaier, John J. Prindle

**References**

getTerminalNodes

<table>
<thead>
<tr>
<th>getNumNodes</th>
<th>Tree Size</th>
</tr>
</thead>
</table>

**Description**
Counts the number of nodes in a tree.

**Usage**
```
getNumNodes(tree)
```

**Arguments**
- `tree` A SEM tree object.

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

**References**

---

getTerminalNodes

**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**

```r
isLeaf(tree)
```

**Arguments**

- `tree` A `semtree` object

**Author(s)**

Andreas M. Brandmaier

**References**


---

**lgcm**

*Simulated Linear Latent Growth Curve Data*

**Description**

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

**Usage**

```r
data(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

Merge two SEM forests

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

dots  
Extra arguments. Currently unused.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References

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

See Also

semtree

modelEstimates

Returns all estimates of a tree

Description
Return model estimates of the tree.

Usage

modelEstimates(tree, level = 0, ...)
outliers

Arguments

  tree A semtree object.
  level
  ... Optional arguments.

Author(s)

  Andreas M. Brandmaier, John J. Prindle

References


______________________________

outliers Find outliers based on case proximity

______________________________

Description

  Compute outlier score based on proximity matrix.

Usage

  outliers(prox)

Arguments

  prox A proximity matrix.

Author(s)

  Andreas M. Brandmaier, John J. Prindle

References


See Also

  proximity
**parameters**

<table>
<thead>
<tr>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEMtrees Parameter Estimates Table</td>
</tr>
</tbody>
</table>

**Description**

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

**Usage**

```r
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**

*Partial Dependence Plot*

**Description**

Partial dependence plot for the effect of an independent variable in the dataset on the dependent outcome parameter selected. Returns a partialDependence object that can be plotted via generic plot command.

**Usage**

```r
partialDependence(forest, reference.var, reference.param, support=NULL, cluster=NULL)
```

**Arguments**

- `forest` A SEM forest
- `reference.var` Label of the (independent) reference variable for which partial dependence is plotted
- `reference.param` Label of the (dependent) model parameter for which partial dependence is plotted
- `support` Number of grid points for interpolating the reference.var
- `cluster` A reference to a cluster from parallel package for parallel execution. Defaults to NULL for sequential computation.

**Author(s)**

Andreas M. Brandmaier

---

**proximity**

*SEM Forest Case Proximity*

**Description**

A `semforest` process to represent proportion of trees where each case are in the same leaf nodes. The values are bounded (0,1), where higher values are closer in proximity.

**Usage**

```r
proximity(forest, dataset = NULL, type = 0, aggregate = T, cluster = NULL, ...)
```
prune

Arguments

- **forest**: A `semforest` object.
- **dataset**: A dataset to compute proximity values for.
- **type**: Missingness accounted for. (0 = no, 1 = yes)
- **aggregate**: Boolean marker to compute aggregate proximity scores.
- **cluster**: An object of class "cluster" representing a parallel socket cluster. See package `makeCluster`.
- ... Optional arguments.

Value

A matrix with dimensions NxN is returned. The values of each cell are bounded (0,1) and represent proportion of trees where each case are in the same leaf nodes.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

`semforest`, `semtree`

---

prune 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

```r
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.semtree

**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**

```r
se(tree, leafs.only=TRUE)
```

**Arguments**

- `tree` A `SEMtrees` 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`.
semforest

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

semforest

Create a SEM Forest

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

Usage
semforest(model, data, control = NULL, predictors = NULL, constraints = NULL, cluster=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.
- cluster: An object of class "cluster" representing a parallel socket cluster. See package makeCluster.
- ...: 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)`

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.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References

**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.

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. "fair3" is a a variant of the previous method with an additional phase. The first two phases are done as described above. Then in the third phase, the complete sample is recombined and used to evaluate the best split point on
4. "crossvalidation" recursion method uses the "folds" option in `semtree.control` to partition the data into a specified number of subsamples (default = 5). to change this setting use `semtree.control(folds=n)` to change the number of folds to value n.

**Usage**

```r
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 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`.

`semtree` model specifications from `semtree.control` are input here. Any changes from the default setting can be specified here.

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.

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.

All other parameters controlling the tree growing process are available through a separate `semtree.control` object.

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

Andreas M. Brandmaier, John J. Prindle


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

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).
semtree.control

Usage

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. "crossvalidation" partitions the data for maximizing splits on each variable, then comparing maximum splits across each variable on the rest of the data.
Usage

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, test.type = "ml", score.tests =
  list(nominal = "LM", ordinal = "DM", metric = "DM"),
  min.bucket = 10)

Arguments

method Default: "naive". One out of c("fair","fair3","naive","cv") for either an unbiased two-step selection algorithm, three-step fair algorithm, a naive take-the-best, or a cross-validation 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.

verbose Default: FALSE. Option to turn on or off all model messages during tree growth.

bonferroni Default: FALSE. Correct for multiple tests with Bonferroni type correction.

seed Default: NA. Set a random number seed for repeating random fold generation in tree analysis.

custom.stopping.rule Default: NA. Otherwise, this can be a boolean function with a custom stopping rule for tree growing.

exclude.code Default: NA. NPSOL error code for exclusion from model fit evaluations when finding best split. Default: Models with errors during fitting are retained.

mtry Default: NA. Number of sample columns to use in SEMforest analysis.

report.level Default: 0. Values up to 99 can be used to increase the number of onscreen reports for semtree analysis.

use.all 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.

test.type Default: 'ml' for maximum-likelihood-based exhaustive split search.

score.tests A list of score test statistics to be used for different variable types.

min.bucket Minimum bucket size to continue splitting.
Value

A control object containing a list of the above parameters.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References


See Also

semtree

Examples

# 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)

----------------------------------------------------------------------------

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.
subtree

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

References

---

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

**Arguments**
- `tree`: A SEM Tree object.
- `added.param.cols`: Add extra columns with parameter estimates.
- `round.param`: Number of digits to round parameter estimates

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

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.  
method Experimental. Some alternative methods to compute importance. Default is "permutation".  
eval.fun Default is evaluateTree function. The value of the -2LL of the leaf nodes is compared to baseline overall model.  
cluster An object of class "cluster" representing a parallel socket cluster. See package makeCluster.  
conditional Conditional variable importance if TRUE, otherwise marginal variable importance.  
... Optional arguments.

Author(s)

Andreas M. Brandmaier, John J. Prindle

References

Index

*Topic biodiversity
  biodiversity, 2
*Topic datasets
  lgcm, 11
*Topic models
  semforest, 18
  semtree, 20
*Topic multivariate
  semforest, 18
  semtree, 20
*Topic semforest
  biodiversity, 2
*Topic tree
  semforest, 18
  semtree, 20

biodiversity, 2
Distances, 3
diversityMatrix, 3
evaluate, 4
evaluateDataLikelihood, 4, 4, 6
evaluateTree, 6, 27

findOtherSplits, 7
getDepth, 7
ggetHeight, 8
getLeafs, 8
getNodeById, 9
getNumNodes, 10
getTerminalNodes, 10

hellinger (Distances), 3
isLeaf, 11
klsym (Distances), 3
lavaan, 20, 21

lgcm, 11
makeCluster, 16, 18, 27
merge, 12
modelEstimates, 12
mxModel, 20, 21

nodeFunSemtree (semtree), 20
OpenMx, 5, 20, 21
outliers, 13

parameters, 14, 14, 17, 18, 21
partialDependence, 15
partialDependencePlot
  (partialDependence), 15
plot.partialDependence
  (partialDependence), 15
plot.proximity (proximity), 15
plot.semforest (semforest), 18
plot.semforest.proximity (proximity), 15
plot.semforest.varimp (varimp), 26
plot.semtree (semtree), 20
print.semforest (semforest), 18
print.semforest.control
  (semforest.control), 19
print.semforest.varimp (varimp), 26
print.semtree (semtree), 20
print.semtree.control
  (semtree.control), 22
proximity, 4, 13, 15
prune, 16
prune.semtree, 21

se, 14, 21, 25
se (se.semtree), 17
se.semtree, 17
semforest, 2–6, 15, 16, 18, 26, 27
semforest.control, 19
semtree, 4–7, 9, 11, 12, 14, 16–19, 20, 21, 22, 24, 25
INDEX

semtree.constraints, 21, 21
semtree.control, 7, 14, 17, 18, 20, 21, 22, 25
subforest, 24
subtree, 21, 25
summary.semtree, 21
summary.semtree (semtree), 20
toLatex.semtree (semtree), 20
toTable, 26

varimp, 4, 26
varimpConvergencePlot (varimp), 26