

# Package ‘ranktreeEnsemble’

May 24, 2024

**Type** Package

**Title** Ensemble Models of Rank-Based Trees with Extracted Decision Rules

**Version** 0.23

**Date** 2024-05-22

**Maintainer** Min Lu <luminwin@gmail.com>

**BugReports** <https://github.com/TransBioInfoLab/ranktreeEnsemble/issues/>

**License** GPL (>= 2)

**Depends** R (>= 3.5.0)

**Imports** Rcpp (>= 1.0.10),randomForestSRC,gbm,methods,data.tree

**LinkingTo** Rcpp

**Description** Fast computing an ensemble of rank-based trees via boosting or random forest on binary and multi-class problems. It converts continuous gene expression profiles into ranked gene pairs, for which the variable importance indices are computed and adopted for dimension reduction. Decision rules can be extracted from trees.

**URL** <https://github.com/TransBioInfoLab/ranktreeEnsemble/>

**LazyData** TRUE

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2024-05-24 03:50:02 UTC

**Author** Ruijie Yin [aut],  
Chen Ye [aut],  
Min Lu [aut, cre] (<<https://orcid.org/0000-0002-1386-1315>>)

## R topics documented:

extract.rules . . . . .	2
importance . . . . .	3

pair . . . . .	4
predict . . . . .	5
ranktreeEnsemble . . . . .	7
rboost . . . . .	8
rforest . . . . .	10
select.rules . . . . .	15
tnbc . . . . .	16

<b>Index</b>	<b>17</b>
--------------	-----------

---

extract.rules	<i>Extract Interpretable Decision Rules from a Random Forest Model</i>
---------------	--

---

## Description

Extract rules from a random forest (rfsrc) object

## Usage

```
extract.rules(object, subtrees = 5,
              treedepth = 2,
              digit = 2,
              pairs = TRUE)
```

## Arguments

object	A random forest (rfsrc) object
subtrees	Number of trees to extract rules
treedepth	Tree depth. The larger the number, the longer the extracted rules are.
digit	Digit to be displayed in the extracted rules.
pairs	Are variables in (object) generated from the pair function? Set pairs = FALSE to extract rules from regular random forest (rfsrc) object with continuous predictors.

## Value

rule	Interpretable extracted rules. Note that the performance score displayed is inaccurate based on few samples.
rule.raw	Rules directly extracted from trees for prediction purpose
data	Data used to grow trees from the argument (object).

## Author(s)

Ruijie Yin (Maintainer, <ruijieyin428@gmail.com>), Chen Ye and Min Lu

## References

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

## Examples

```
data(tnbc)
obj <- rforest(subtype~., data = tnbc[1:100,c(1:5,337)])
objr <- extract.rules(obj)
objr$rule[,1:3]

#### extract rules from a regular random forest
library(randomForestSRC)
obj2 <- rfsrc(subtype~., data = tnbc[1:100,c(1:5,337)])
objr2 <- extract.rules(obj2, pairs = FALSE)
objr2$rule[,1:3]
```

---

importance

*Variable Importance Index for Each Predictor*

---

## Description

The function computes variable importance for each predictor from a rank-based random forests model or boosting model. A higher value indicates a more important predictor. The random forest implementation was performed via the function `vimp` directly imported from the **randomForestSRC** package. Use the command `package?randomForestSRC` for more information. The boosting implementation was performed via the function `relative.influence` directly imported from the **gbm** package. For technical details, see the vignette: `utils::browseVignettes("gbm")`.

## Usage

```
importance(object, ...)
```

## Arguments

<code>object</code>	An object of class <code>rfsrc</code> generated from the function <code>rforest</code> or <code>gbm</code> generated from the function <code>rboost</code> .
<code>...</code>	Further arguments passed to or from other methods.

## Value

For the boosting model, a vector of variable importance values is given. For the random forest model, a matrix of variable importance values is given for the variable importance index for all the class labels, followed by the index for each class label.

**Author(s)**

Ruijie Yin (Maintainer,<ruijieyin428@gmail.com>), Chen Ye and Min Lu

**References**

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

**Examples**

```
data(tnbc)
#####
# Random Forest
#####
obj <- rforest(subtype~., data = tnbc[,c(1:10,337)])
importance(obj)
#####
# Boosting
#####
obj <- rboost(subtype~., data = tnbc[,c(1:10,337)])
importance(obj)
```

---

pair

*Transform Continuous Variables into Ranked Binary Pairs*

---

**Description**

The function transforms a dataset with  $p$  continuous predictors into  $\frac{p*(p-1)}{2}$  binary predictors of ranked pairs

**Usage**

```
pair(data, yvar.name = NULL)
```

**Arguments**

data	A dataset with $p$ continuous variables or with $p + 1$ variables including a dependent variable.
yvar.name	The column name of the independent variable in data. By default, there is no dependent variable.

**Value**

A data frame with the transformed data. The dependent variable is moved to the last column of the data.

**Note**

The function is efficiently coded in C++.

**Author(s)**

Ruijie Yin (Maintainer,<ruijieyin428@gmail.com>), Chen Ye and Min Lu

**References**

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

**Examples**

```
data(tnbc)
datp <- pair(tnbc[101:105,c(1:5,337)],"subtype")
datp
datp <- pair(tnbc[105:110,1:5])
datp
```

---

predict	<i>Prediction or Extract Predicted Values for Random Forest, Random Forest Rule or Boosting Models</i>
---------	--

---

**Description**

Obtain predicted values using a random forest (`rfsrc`), random forest extracted rule (`rules`) or boosting (`gbm`) object. If no new data is provided, it extracts the out-of-bag predicted values of the outcome for the training data.

**Usage**

```
predict(object,
        newdata = NULL,
        newdata.pair = FALSE, ...)
```

**Arguments**

object	An object of class <code>rfsrc</code> generated from the function <code>rforest</code> or <code>gbm</code> generated from the function <code>rboost</code> .
newdata	Test data. If missing, the original training data is used for extracting the out-of-bag predicted values without running the model again.
newdata.pair	Is <code>newdata</code> already converted into binary ranked pairs from the <code>pair</code> function?
...	Further arguments passed to or from other methods.

**Details**

For the boosting (gbm) object, the cross-validation predicted values are provided if `cv.folds>=2`.

**Value**

value	Predicted value of the outcome. For the random forest (rfsrc) object, it is the predicted probability. For the boosting (gbm) object, it is the fitted values on the scale of regression function (e.g. log-odds scale). For the random forest extracted rule (rules) object, it is empty.
label	Predicted label of the outcome.

**Author(s)**

Ruijie Yin (Maintainer,<ruijieyin428@gmail.com>), Chen Ye and Min Lu

**References**

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

**Examples**

```
data(tnbc)
#####
# Random Forest
#####
obj <- rforest(subtype~., data = tnbc[1:100,c(1:5,337)])
predict(obj)$label
predict(obj, tnbc[101:110,1:5])$label

datp <- pair(tnbc[101:110,1:5])
predict(obj, datp, newdata.pair = TRUE)$label
#####
# Random Forest Extracted Rule
#####
objr <- extract.rules(obj)
predict(objr)$label[1:5]
predict(obj, tnbc[101:110,1:5])$label
#####
# Boosting
#####
obj <- rboost(subtype~., data = tnbc[1:100,c(1:5,337)])
predict(obj)$label
predict(obj, tnbc[101:110,1:5])$label
```

---

ranktreeEnsemble	<i>Ensemble Models of Rank-Based Trees for Single Sample Classification with Interpretable Rules</i>
------------------	--

---

## Description

The package ranktreeEnsemble implements an ensemble of rank-based trees in boosting with the LogitBoost cost and random forests on both binary and multi-class problems. It converts continuous gene expression profiles into ranked gene pairs, for which the variable importance indices are computed and adopted for dimension reduction. Interpretable rules can be extracted from trees.

## Author(s)

Ruijie Yin (Maintainer, <ruijieyin428@gmail.com>), Chen Ye and Min Lu

## References

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

## Examples

```
library(ranktreeEnsemble)
data(tnbc)
##### performance of Random Rank Forest
obj <- rforest(subtype~., data = tnbc[,c(1:10,337)])
obj
# variable importance
importance(obj)
##### predict new data from Random Rank Forest
predict(obj, tnbc[101:110,1:10])$label
##### extract decision rules from rank-based trees
objr <- extract.rules(obj)
objr$rule[1:5,]
predict(objr, tnbc[101:110,1:10])$label
##### filter decision rules with higher performance
objrs <- select.rules(objr,tnbc[110:130,c(1:10,337)])
predict(objrs, tnbc[101:110,1:10])$label
```

rboost

*Generalized Boosted Modeling via Rank-Based Trees for Single Sample Classification with Gene Expression Profiles*

## Description

The function fits generalized boosted models via Rank-Based Trees on both binary and multi-class problems. It converts continuous gene expression profiles into ranked gene pairs, for which the variable importance indices are computed and adopted for dimension reduction. The boosting implementation was directly imported from the **gbm** package. For technical details, see the vignette: `utils::browseVignettes("gbm")`.

## Usage

```
rboost(
  formula,
  data,
  dimreduce = TRUE,
  datrank = TRUE,
  distribution = "multinomial",
  weights,
  ntree = 100,
  nodedepth = 3,
  nodesize = 5,
  shrinkage = 0.05,
  bag.fraction = 0.5,
  train.fraction = 1,
  cv.folds = 5,
  keep.data = TRUE,
  verbose = TRUE,
  class.stratify.cv = TRUE,
  n.cores = NULL
)
```

## Arguments

<code>formula</code>	Object of class 'formula' describing the model to fit.
<code>data</code>	Data frame containing the y-outcome and x-variables.
<code>dimreduce</code>	Dimension reduction via variable importance weighted forests. FALSE means no dimension reduction; TRUE means reducing 75% variables before binary rank conversion and then fitting a weighted forest; a numeric value x% between 0 and 1 means reducing x% variables before binary rank conversion and then fitting a weighted forest.
<code>datrank</code>	If using ranked raw data for fitting the dimension reduction model.



distribution	Either a character string specifying the name of the distribution to use: if the response has only 2 unique values, <code>bernoulli</code> is assumed; otherwise, if the response is a factor, <code>multinomial</code> is assumed.
weights	an optional vector of weights to be used in the fitting process. It must be positive but does not need to be normalized.
ntree	Integer specifying the total number of trees to fit. This is equivalent to the number of iterations and the number of basis functions in the additive expansion, which matches <code>n.tree</code> in the <code>gbm</code> package.
nodedepth	Integer specifying the maximum depth of each tree. A value of 1 implies an additive model. This matches <code>interaction.depth</code> in the <code>gbm</code> package.
nodesize	Integer specifying the minimum number of observations in the terminal nodes of the trees, which matches <code>n.minobsinnode</code> in the <code>gbm</code> package.. Note that this is the actual number of observations, not the total weight.
shrinkage	a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually work, but a smaller learning rate typically requires more trees. Default is 0.05.
bag.fraction	the fraction of the training set observations randomly selected to propose the next tree in the expansion. This introduces randomnesses into the model fit. If <code>bag.fraction &lt; 1</code> then running the same model twice will result in similar but different fits. <code>gbm</code> uses the R random number generator so <code>set.seed</code> can ensure that the model can be reconstructed. Preferably, the user can save the returned <code>gbm.object</code> using <code>save</code> . Default is 0.5.
train.fraction	The first <code>train.fraction * nrow(data)</code> observations are used to fit the <code>gbm</code> and the remaining observations are used for computing out-of-sample estimates of the loss function.
cv.folds	Number of cross-validation folds to perform. If <code>cv.folds &gt; 1</code> then <code>gbm</code> , in addition to the usual fit, will perform cross-validation and calculate an estimate of generalization error returned in <code>cv.error</code> .
keep.data	a logical variable indicating whether to keep the data and an index of the data stored with the object. Keeping the data and index makes subsequent calls to <code>gbm.more</code> faster at the cost of storing an extra copy of the dataset.
verbose	Logical indicating whether or not to print out progress and performance indicators (TRUE). If this option is left unspecified for <code>gbm.more</code> , then it uses <code>verbose</code> from <code>object</code> . Default is TRUE.
class.stratify.cv	Logical indicating whether or not the cross-validation should be stratified by class. The purpose of stratifying the cross-validation is to help avoid situations in which training sets do not contain all classes.
n.cores	The number of CPU cores to use. The cross-validation loop will attempt to send different CV folds off to different cores. If <code>n.cores</code> is not specified by the user, it is guessed using the <code>detectCores</code> function in the <code>parallel</code> package. Note that the documentation for <code>detectCores</code> makes clear that it is not failsafe and could return a spurious number of available cores.

**Value**

<code>fit</code>	A vector containing the fitted values on the scale of regression function (e.g. log-odds scale for bernoulli).
<code>train.error</code>	A vector of length equal to the number of fitted trees containing the value of the loss function for each boosting iteration evaluated on the training data.
<code>valid.error</code>	A vector of length equal to the number of fitted trees containing the value of the loss function for each boosting iteration evaluated on the validation data.
<code>cv.error</code>	If <code>cv.folds &lt; 2</code> this component is NULL. Otherwise, this component is a vector of length equal to the number of fitted trees containing a cross-validated estimate of the loss function for each boosting iteration.
<code>oobag.improve</code>	A vector of length equal to the number of fitted trees containing an out-of-bag estimate of the marginal reduction in the expected value of the loss function. The out-of-bag estimate uses only the training data and is useful for estimating the optimal number of boosting iterations. See <a href="#">gbm.perf</a> .
<code>cv.fitted</code>	If cross-validation was performed, the cross-validation predicted values on the scale of the linear predictor. That is, the fitted values from the i-th CV-fold, for the model having been trained on the data in all other folds.

**Author(s)**

Ruijie Yin (Maintainer, <ruijieyin428@gmail.com>), Chen Ye and Min Lu

**References**

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

**Examples**

```
data(tnbc)
obj <- rboost(subtype~., data = tnbc[,c(1:10,337)])
obj
```

---

rforest

*Random Forest via Rank-Based Trees for Single Sample Classification with Gene Expression Profiles*

---

**Description**

The function implements the ensembled rank-based trees in random forests on both binary and multi-class problems. It converts continuous gene expression profiles into ranked gene pairs, for which the variable importance indices are computed and adopted for dimension reduction. The random forest implementation was directly imported from the **randomForestSRC** package. Use the command `package?randomForestSRC` for more information.

**Usage**

```
rforest(formula, data,
        dimreduce = TRUE,
        datrank = TRUE,
        ntree = 500, mtry = NULL,
        nodesize = NULL, nodedepth = NULL,
        splitrule = NULL, nsplit = NULL,
        importance = c(FALSE, TRUE, "none", "anti", "permute", "random"),
        bootstrap = c("by.root", "none"),
        membership = FALSE,
        na.action = c("na.omit", "na.impute"), nimpute = 1,
        perf.type = NULL,
        xvar.wt = NULL, yvar.wt = NULL, split.wt = NULL, case.wt = NULL,
        forest = TRUE,
        var.used = c(FALSE, "all.trees", "by.tree"),
        split.depth = c(FALSE, "all.trees", "by.tree"),
        seed = NULL,
        statistics = FALSE,
        ...)

## convenient interface for growing a rank-based tree
rforest.tree(formula, data, dimreduce = FALSE,
             ntree = 1, mtry = ncol(data),
             bootstrap = "none", ...)
```

**Arguments**

formula	Object of class 'formula' describing the model to fit. Interaction terms are not supported.
data	Data frame containing the y-outcome and x-variables.
dimreduce	Dimension reduction via variable importance weighted forests. FALSE means no dimension reduction; TRUE means reducing 75% variables before binary rank conversion and then fitting a weighted forest; a numeric value x% between 0 and 1 means reducing x% variables before binary rank conversion and then fitting a weighted forest.
datrank	If using ranked raw data for fitting the dimension reduction model.
ntree	Number of trees.
mtry	Number of variables to possibly split at each node. Default is number of variables divided by 3 for regression. For all other families (including unsupervised settings), the square root of number of variables. Values are rounded up.
nodesize	Minimum size of terminal node. The defaults are: survival (15), competing risk (15), regression (5), classification (1), mixed outcomes (3), unsupervised (3). It is recommended to experiment with different nodesize values.
nodedepth	Maximum depth to which a tree should be grown. Parameter is ignored by default.
splitrule	Splitting rule (see below).

<code>nsplit</code>	Non-negative integer specifying number of random splits for splitting a variable. When zero, all split values are used (deterministic splitting), which can be slower. By default 10 is used.
<code>importance</code>	Method for computing variable importance (VIMP); see below. Default action is <code>importance="none"</code> but VIMP can be recovered later using <code>vimp</code> or <code>predict</code> .
<code>bootstrap</code>	Bootstrap protocol. Default is <code>by.root</code> which bootstraps the data by sampling without replacement. If <code>none</code> , the data is not bootstrapped (it is not possible to return OOB ensembles or prediction error in this case).
<code>membership</code>	Should terminal node membership and inbag information be returned?
<code>na.action</code>	Action taken if the data contains NA's. Possible values are <code>na.omit</code> or <code>na.impute</code> . The default <code>na.omit</code> removes the entire record if any entry is NA. Selecting <code>na.impute</code> imputes the data (see below for details). Also see the function <code>impute</code> for fast imputation.
<code>nimpute</code>	Number of iterations of the missing data algorithm. Performance measures such as out-of-bag (OOB) error rates are optimistic if <code>nimpute</code> is greater than 1.
<code>perf.type</code>	Optional character value specifying metric used for predicted value, variable importance (VIMP), and error rate. Reverts to the family default metric if not specified. Values allowed for univariate/multivariate classification are: <code>perf.type="misclass"</code> (default), <code>perf.type="brier"</code> and <code>perf.type="gmean"</code> .
<code>xvar.wt</code>	Vector of non-negative weights (does not have to sum to 1) representing the probability of selecting a variable for splitting. Default is uniform weights.
<code>yvar.wt</code>	Used for sending in features with custom splitting. For expert use only.
<code>split.wt</code>	Vector of non-negative weights used for multiplying the split statistic for a variable. A large value encourages the node to split on a specific variable. Default is uniform weights.
<code>case.wt</code>	Vector of non-negative weights (does not have to sum to 1) for sampling cases. Observations with larger weights will be selected with higher probability in the bootstrap (or subsampled) samples. It is generally better to use real weights rather than integers. See the breast data example below illustrating its use for class imbalanced data.
<code>forest</code>	Save key forest values? Used for prediction on new data and required by many of the package functions. Turn this off if you are only interested in training a forest.
<code>var.used</code>	Return statistics on number of times a variable split? Default is <code>FALSE</code> . Possible values are <code>all.trees</code> which returns total number of splits of each variable, and <code>by.tree</code> which returns a matrix of number a splits for each variable for each tree.
<code>split.depth</code>	Records the minimal depth for each variable. Default is <code>FALSE</code> . Possible values are <code>all.trees</code> which returns a matrix of the average minimal depth for a variable (columns) for a specific case (rows), and <code>by.tree</code> which returns a three-dimensional array recording minimal depth for a specific case (first dimension) for a variable (second dimension) for a specific tree (third dimension).
<code>seed</code>	Negative integer specifying seed for the random number generator.
<code>statistics</code>	Should split statistics be returned? Values can be parsed using <code>stat.split</code> .
<code>...</code>	Further arguments passed to or from other methods.

## Details

### *Splitting*

1. Splitting rules are specified by the option `splitrule`.
2. For all families, pure random splitting can be invoked by setting `splitrule="random"`.
3. For all families, computational speed can be increased using randomized splitting invoked by the option `nsplit`. See *Improving Computational Speed*.

### *Available splitting rules*

1. `splitrule="gini"` (default `splitrule`): Gini index splitting (Breiman et al. 1984, Chapter 4.3).
2. `splitrule="auc"`: AUC (area under the ROC curve) splitting for both two-class and multi-class settings. AUC splitting is appropriate for imbalanced data. See *imbalanced* for more information.
3. `splitrule="entropy"`: entropy splitting (Breiman et al. 1984, Chapter 2.5, 4.3).

## Value

An object of class `(rfsrc, grow)` with the following components:

<code>call</code>	The original call to <code>rfsrc</code> for growing the random forest object.
<code>family</code>	The family used in the analysis.
<code>n</code>	Sample size of the data (depends upon NA's, see <code>na.action</code> ).
<code>ntree</code>	Number of trees grown.
<code>mtry</code>	Number of variables randomly selected for splitting at each node.
<code>nodesize</code>	Minimum size of terminal nodes.
<code>nodedepth</code>	Maximum depth allowed for a tree.
<code>splitrule</code>	Splitting rule used.
<code>nsplit</code>	Number of randomly selected split points.
<code>yvar</code>	y-outcome values.
<code>yvar.names</code>	A character vector of the y-outcome names.
<code>xvar</code>	Data frame of x-variables.
<code>xvar.names</code>	A character vector of the x-variable names.
<code>xvar.wt</code>	Vector of non-negative weights for dimension reduction which specify the probability used to select a variable for splitting a node.
<code>split.wt</code>	Vector of non-negative weights specifying multiplier by which the split statistic for a covariate is adjusted.
<code>cause.wt</code>	Vector of weights used for the composite competing risk splitting rule.
<code>leaf.count</code>	Number of terminal nodes for each tree in the forest. Vector of length <code>ntree</code> . A value of zero indicates a rejected tree (can occur when imputing missing data). Values of one indicate tree stumps.

proximity	Proximity matrix recording the frequency of pairs of data points occur within the same terminal node.
forest	If forest=TRUE, the forest object is returned. This object is used for prediction with new test data sets and is required for other R-wrappers.
membership	Matrix recording terminal node membership where each column records node membership for a case for a tree (rows).
splitrule	Splitting rule used.
inbag	Matrix recording inbag membership where each column contains the number of times that a case appears in the bootstrap sample for a tree (rows).
var.used	Count of the number of times a variable is used in growing the forest.
imputed.indv	Vector of indices for cases with missing values.
imputed.data	Data frame of the imputed data. The first column(s) are reserved for the y-outcomes, after which the x-variables are listed.
split.depth	Matrix (i,j) or array (i,j,k) recording the minimal depth for variable j for case i, either averaged over the forest, or by tree k.
node.stats	Split statistics returned when statistics=TRUE which can be parsed using stat.split.
err.rate	Tree cumulative OOB error rate.
importance	Variable importance (VIMP) for each x-variable.
predicted	In-bag predicted value.
predicted.oob	OOB predicted value.
class	In-bag predicted class labels.
class.oob	OOB predicted class labels.

### Author(s)

Ruijie Yin (Maintainer,<ruijieyin428@gmail.com>), Chen Ye and Min Lu

### References

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

### Examples

```
data(tnbc)
##### performance of Random Rank Forest
obj <- rforest(subtype~., data = tnbc[,c(1:10,337)])
obj
```

---

select.rules	<i>Select Decision Rules to Achieve Higher Prediction Accuracy</i>
--------------	--

---

**Description**

Select rules from a `extrat.rules` (rules) object

**Usage**

```
select.rules(object, data, data.pair = FALSE)
```

**Arguments**

<code>object</code>	An extracted rule (rules) object generated from the <code>extract.rules</code> function.
<code>data</code>	A validation dataset for selecting rules.
<code>data.pair</code>	Is data already converted into binary ranked pairs from the <code>pair</code> function?

**Value**

<code>rule</code>	Interpretable selected rules. Note that the performance score displayed is inaccurate based on few samples from the original argument object.
<code>rule.raw</code>	Rules directly extracted from trees for prediction purpose
<code>data</code>	Data used to grow trees from the argument (object).

**Author(s)**

Ruijie Yin (Maintainer,<ruijieyin428@gmail.com>), Chen Ye and Min Lu

**References**

Lu M. Yin R. and Chen X.S. Ensemble Methods of Rank-Based Trees for Single Sample Classification with Gene Expression Profiles. *Journal of Translational Medicine*. 22, 140 (2024). doi: 10.1186/s12967-024-04940-2

**Examples**

```
data(tnbc)
obj <- rforest(subtype~., data = tnbc[1:100,c(1:5,337)])
objr <- extract.rules(obj)
predict(objr, tnbc[101:110,1:5])$label
objrs <- select.rules(objr,tnbc[110:130,c(1:5,337)])
predict(objrs, tnbc[101:110,1:5])$label
```

---

`tnbc`*Gene expression profiles in triple-negative breast cancer cell*

---

**Description**

Gene expression profiles in triple-negative breast cancer cells with 215 observations and 337 variables. Gene expression values were randomly chosen from the original dataset. The outcome variable is *subtype*.

**Usage**

```
data(tnbc)
```

**Source**

Chen, X., Li, J., Gray, W. H., Lehmann, B. D., Bauer, J. A., Shyr, Y., & Pietenpol, J. A. (2012). TNBCtype: a subtyping tool for triple-negative breast cancer. *Cancer informatics*, 11, CIN-S9983.

**Examples**

```
data(tnbc)
```



# Index

`extract.rules`, [2](#)

`gbm.more`, [9](#)

`gbm.object`, [9](#)

`gbm.perf`, [10](#)

importance, [3](#)

`pair`, [4](#)

`predict`, [5](#)

`ranktreeEnsemble`, [7](#)

`ranktreeEnsemble-package`  
(`ranktreeEnsemble`), [7](#)

`rboost`, [8](#)

`rforest`, [10](#)

`save`, [9](#)

`select.rules`, [15](#)

`tnbc`, [16](#)