# Package 'grandforest'

June 13, 2018

**Type** Package

**Title** A graph-guided random forest algorithm

**Version** 0.1

**Date** 2018-04-19

**Author** Simon J. Larsen [aut, cre], Marvin N. Wright [ctb], Stefan Wager [ctb], Philipp Probst [ctb]

**Maintainer** Simon J. Larsen <simonhffh@gmail.com>

**Description** Grand Forest is a graph-guided Random Forest algorithm, integrating secondary graph-structured data in order guide the feature selection towards interacting features. While it can be used for prediction, the main purpose of Grand Forest is descriptive, as it provides an efficient way of discovering highly informative subnetworks. Grand Forest is based on ranger.

**License** GPL-3

**Imports** Rcpp (>= 0.11.2), Matrix

**LinkingTo** Rcpp, RcppEigen

**Depends** R (>= 3.1)

**Suggests** survival, knitr, rmarkdown, tidyverse, data.table, simpIntLists, org.Hs.eg.db, geomnet, ComplexHeatmap, survminer

**RoxygenNote** 6.0.1

**URL** https://github.com/SimonLarsen/grandforest

**BugReports** https://github.com/SimonLarsen/grandforest/issues

**VignetteBuilder** knitr

## R topics documented:

---

`categorical` *Synthetic example data set for classification model.*

---

## Description

Synthetic example data set for classification model.

## Usage

```
categorical
```

## Format

A data frame with 100 rows and 21 columns.

---

`csrf` *Case-specific random forests.*

---

## Description

In case-specific random forests (CSRF), random forests are built specific to the cases of interest. Instead of using equal probabilities, the cases are weighted according to their difference to the case of interest.

## Usage

```
csrf(formula, training_data, test_data, params1 = list(), params2 = list())
```

## Arguments

| | |
|---|---|
| formula | Object of class formula or character describing the model to fit. |
| training_data | Training data of class data.frame. |
| test_data | Test data of class data.frame. |
| params1 | Parameters for the proximity random forest grown in the first step. |
| params2 | Parameters for the prediction random forests grown in the second step. |

## Details

The algorithm consists of 3 steps:

1. Grow a random forest on the training data
2. For each observation of interest (test data), the weights of all training observations are computed by counting the number of trees in which both observations are in the same terminal node.
3. For each test observation, grow a weighted random forest on the training data, using the weights obtained in step 2. Predict the outcome of the test observation as usual.

In total, n+1 random forests are grown, where n is the number observations in the test dataset. For details, see Xu et al. (2014).

## Value

Predictions for the test dataset.

## Author(s)

Marvin N. Wright

## References

Xu, R., Nettleton, D. & Nordman, D.J. (2014). Case-specific random forests. J Comp Graph Stat 25:49-65. https://doi.org/10.1080/10618600.2014.983641.

## Examples

```
## Split in training and test data
train.idx <- sample(nrow(iris), 2/3 * nrow(iris))
iris.train <- iris[train.idx, ]
iris.test <- iris[-train.idx, ]

## Run case-specific RF
csrf(Species ~ ., training_data = iris.train, test_data = iris.test,
     params1 = list(num.trees = 50, mtry = 4),
     params2 = list(num.trees = 5))
```

## getTerminalNodeIDs *Get terminal node IDs (deprecated)*

### Description

This function is deprecated. Please use predict() with `type = "terminalNodes"` instead. This function calls predict() now.

### Usage

```
getTerminalNodeIDs(rf, dat)
```

### Arguments

| | |
|---|---|
| `rf` | `grandforest` object. |
| `dat` | New dataset. Terminal node IDs for this dataset are obtained. |

### Value

Matrix with terminal nodeIDs for all observations in dataset and trees.

### Examples

```
library(grandforest)
rf <- grandforest(Species ~ ., data = iris, num.trees = 5, write.forest = TRUE)
getTerminalNodeIDs(rf, iris)
```

## grandforest *Grand Forest*

### Description

Grand Forest is a graph-guided Random Forest algorithm based on Ranger. Grand Forest integrates a feature interaction graph in order to guide the feature selection towards interacting features.

### Usage

```
grandforest(formula = NULL, data = NULL, graph_data = NULL,
  num.trees = 500, mtry = NULL, importance = "impurity",
  subgraph = "bfs", write.forest = TRUE, probability = FALSE,
  min.node.size = NULL, replace = TRUE, sample.fraction = ifelse(replace,
  1, 0.632), case.weights = NULL, splitrule = NULL, num.random.splits = 1,
  alpha = 0.5, minprop = 0.1, split.select.weights = NULL,
  always.split.variables = NULL, respect.unordered.factors = NULL,
  scale.permutation.importance = FALSE, keep.inbag = FALSE,
  holdout = FALSE, num.threads = NULL, save.memory = FALSE,
  verbose = TRUE, seed = NULL, dependent.variable.name = NULL,
  status.variable.name = NULL, classification = NULL)
```

## Arguments

| | |
|---|---|
| formula | Object of class formula or character describing the model to fit. Interaction terms supported only for numerical variables. |
| data | Training data of class data.frame, matrix or gwaa.data (GenABEL). |
| graph_data | Feature interaction graph. Must be two-column character matrix with feature names corresponding to column names in data. |
| num.trees | Number of trees. |
| mtry | Number of variables to possibly split at in each node. Default is the (rounded down) square root of the number variables. |
| importance | Variable importance mode, one of 'none', 'impurity', 'impurity_corrected', 'permutation'. The 'impurity' measure is the Gini index for classification, the variance of the responses for regression and the sum of test statistics (see splitrule) for survival. |
| subgraph | Feature subgraph selection mode. One of 'bfs', 'dfs', 'random'. |
| write.forest | Save grandforest.forest object, required for prediction. Set to FALSE to reduce memory usage if no prediction intended. |
| probability | Grow a probability forest as in Malley et al. (2012). |
| min.node.size | Minimal node size. Default 1 for classification, 5 for regression, 3 for survival, and 10 for probability. |
| replace | Sample with replacement. |
| sample.fraction | Fraction of observations to sample. Default is 1 for sampling with replacement and 0.632 for sampling without replacement. |
| case.weights | Weights for sampling of training observations. Observations with larger weights will be selected with higher probability in the bootstrap (or subsampled) samples for the trees. |
| splitrule | Splitting rule. For classification and probability estimation "gini" or "extratrees" with default "gini". For regression "variance", "extratrees" or "maxstat" with default "variance". For survival "logrank", "extratrees", "C" or "maxstat" with default "logrank". |
| num.random.splits | For "extratrees" splitrule.: Number of random splits to consider for each candidate splitting variable. |
| alpha | For "maxstat" splitrule: Significance threshold to allow splitting. |
| minprop | For "maxstat" splitrule: Lower quantile of covariate distribution to be considered for splitting. |
| split.select.weights | Numeric vector with weights between 0 and 1, representing the probability to select variables for splitting. Alternatively, a list of size num.trees, containing split select weight vectors for each tree can be used. |
| always.split.variables | Character vector with variable names to be always selected in addition to the mtry variables tried for splitting. |

`respect.unordered.factors`
: Handling of unordered factor covariates. One of 'ignore', 'order' and 'partition'. For the "extratrees" splitrule the default is "partition" for all other splitrules 'ignore'. Alternatively TRUE (='order') or FALSE (='ignore') can be used. See below for details.

`scale.permutation.importance`
: Scale permutation importance by standard error as in (Breiman 2001). Only applicable if permutation variable importance mode selected.

`keep.inbag`
: Save how often observations are in-bag in each tree.

`holdout`
: Hold-out mode. Hold-out all samples with case weight 0 and use these for variable importance and prediction error.

`num.threads`
: Number of threads. Default is number of CPUs available.

`save.memory`
: Use memory saving (but slower) splitting mode. No effect for survival and GWAS data. Warning: This option slows down the tree growing, use only if you encounter memory problems.

`verbose`
: Show computation status and estimated runtime.

`seed`
: Random seed. Default is `NULL`, which generates the seed from `R`.

`dependent.variable.name`
: Name of dependent variable, needed if no formula given. For survival forests this is the time variable.

`status.variable.name`
: Name of status variable, only applicable to survival data and needed if no formula given. Use 1 for event and 0 for censoring.

`classification`
: Only needed if data is a matrix. Set to `TRUE` to grow a classification forest.

## Details

Ranger is a fast implementation of Random Forest (Breiman 2001) or recursive partitioning, particularly suited for high dimensional data. Classification, regression, and survival forests are supported. Classification and regression forests are implemented as in the original Random Forest (Breiman 2001), survival forests as in Random Survival Forests (Ishwaran et al. 2008).

The tree type is determined by the type of the dependent variable. For factors classification trees are grown, for numeric values regression trees and for survival objects survival trees. The Gini index is used as default splitting rule for classification. For regression, the estimated response variances or maximally selected rank statistics (Wright et al. 2016) can be used. For Survival the log-rank test, a C-index based splitting rule (Schmid et al. 2015) and maximally selected rank statistics (Wright et al. 2016) are available. For all tree types, forests of extremely randomized trees (Geurts et al. 2006) can be grown.

With the `probability` option and factor dependent variable a probability forest is grown. Here, the node impurity is used for splitting, as in classification forests. Predictions are class probabilities for each sample. In contrast to other implementations, each tree returns a probability estimate and these estimates are averaged for the forest probability estimate. For details see Malley et al. (2012).

Note that for classification and regression nodes with size smaller than `min.node.size` can occur, as in original Random Forests. For survival all nodes contain at `min.node.size` samples.

Variables selected with always.split.variables are tried additionally to the mtry variables randomly selected. In split.select.weights variables weighted with 0 are never selected and variables with 1 are always selected. Weights do not need to sum up to 1, they will be normalized later. The weights are assigned to the variables in the order they appear in the formula or in the data if no formula is used. Names of the split.select.weights vector are ignored. The usage of split.select.weights can increase the computation times for large forests.

Unordered factor covariates can be handled in 3 different ways by using respect.unordered.factors: For 'ignore' all factors are regarded ordered, for 'partition' all possible 2-partitions are considered for splitting. For 'order' and 2-class classification the factor levels are ordered by their proportion falling in the second class, for regression by their mean response, as described in Hastie et al. (2009), chapter 9.2.4. For multiclass classification and survival outcomes, 'order' is experimental and should be used with care. The use of 'order' is recommended for 2-class classification and regression, as it computationally fast and can handle an unlimited number of factor levels. Note that the factors are only reordered once and not again in each split.

For a large number of variables and data frames as input data the formula interface can be slow or impossible to use. Alternatively dependent.variable.name (and status.variable.name for survival) can be used. Consider setting save.memory = TRUE if you encounter memory problems for very large datasets, but be aware that this option slows down the tree growing.

For GWAS data consider combining grandforest with the GenABEL package. See the Examples section below for a demonstration using Plink data. All SNPs in the GenABEL object will be used for splitting. To use only the SNPs without sex or other covariates from the phenotype file, use 0 on the right hand side of the formula. Note that missing values are treated as an extra category while splitting.

## Value

Object of class grandforest with elements

- `forest` Saved forest (If write.forest set to TRUE). Note that the variable IDs in the split.varIDs object do not necessarily represent the column number in R.
- `predictions` Predicted classes/values, based on out of bag samples (classification and regression only).
- `variable.importance` Variable importance for each independent variable.
- `prediction.error` Overall out of bag prediction error. For classification this is the fraction of missclassified samples, for regression the mean squared error and for survival one minus Harrell's c-index.
- `r.squared` R squared. Also called explained variance or coefficient of determination (regression only). Computed on out of bag data.
- `confusion.matrix` Contingency table for classes and predictions based on out of bag samples (classification only).
- `unique.death.times` Unique death times (survival only).
- `chf` Estimated cumulative hazard function for each sample (survival only).

| | |
|---|---|
| survival | Estimated survival function for each sample (survival only). |
| call | Function call. |
| num.trees | Number of trees. |
| num.independent.variables | Number of independent variables. |
| mtry | Value of mtry used. |
| min.node.size | Value of minimal node size used. |
| treetype | Type of forest/tree. classification, regression or survival. |
| importance.mode | Importance mode used. |
| num.samples | Number of samples. |
| inbag.counts | Number of times the observations are in-bag in the trees. |

## Author(s)

Simon J. Larsen

## References


- Wright, M. N. & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.
- Schmid, M., Wright, M. N. & Ziegler, A. (2016). On the use of Harrell's C for clinical risk prediction via random survival forests. Expert Syst Appl 63:450-459. https://doi.org/10.1016/j.eswa.2016.07.018.
- Wright, M. N., Dankowski, T. & Ziegler, A. (2017). Unbiased split variable selection for random survival forests using maximally selected rank statistics. Stat Med. https://doi.org/10.1002/sim.7212.
- Breiman, L. (2001). Random forests. Mach Learn, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324.
- Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. Ann Appl Stat 2:841-860. https://doi.org/10.1097/JTO.0b013e318233d835.
- Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., & Ziegler, A. (2012). Probability machines: consistent probability estimation using nonparametric learning machines. Methods Inf Med 51:74-81. https://doi.org/10.3414/ME00-01-0052.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning. Springer, New York. 2nd edition.
- Geurts, P., Ernst, D., Wehenkel, L. (2006). Extremely randomized trees. Mach Learn 63:3-42. https://doi.org/10.1007/s10994-006-6226-1.


## See Also

predict.grandforest

## Examples

```
require(grandforest)
data(network)
data(categorical)
data(survival)

## Training model with categorical response variable
grandforest(data=categorical, graph_data=network, dependent.variable.name="group")

## Model survival model
grandforest(
  data=survival, graph_data=network,
  dependent.variable.name="time", status.variable.name="event"
)

## Variable importance
model <- grandforest(data=categorical, graph_data=network, dependent.variable.name="group")
imp <- importance(model)
# Get 5 most important features
top5 <- tail(sort(imp), 5)
```

---

`grandforest_unsupervised`

*Train unsupervised Grand Forest model.*

---

## Description

Helper function for training Grand Forest model for unsupervised analysis. Generates a background data set by random sampling, then trains a Grand Forest model to distinguish the foreground from the background.

## Usage

```
grandforest_unsupervised(data, graph_data = NULL, replace = TRUE,
  importance = "impurity", ...)
```

## Arguments

| | |
|---|---|
| `data` | Training data of class `data.frame`, `matrix` or `gwaa.data` (GenABEL). |
| `graph_data` | Feature interaction graph. Must be two-column character `data.frame` or `matrix` with character strings corresponding to column names in `data`. |
| `replace` | Should background sampling be with replacement? |
| `importance` | Variable importance mode, one of 'none', 'impurity', 'impurity_corrected', 'permutation'. The 'impurity' measure is the Gini index. |
| `...` | Other arguments to be passed to the `grandforest` function during training. |

holdoutRF *Hold-out random forests*

## Description

Grow two random forests on two cross-validation folds. Instead of out-of-bag data, the other fold is used to compute permutation importance. Related to the novel permutation variable importance by Janitza et al. (2015).

## Usage

```
holdoutRF(...)
```

## Arguments

| | |
|---|---|
| `...` | All arguments are passed to `grandforest()` (except `importance`, `case.weights`, `replace` and `holdout`.). |

## Value

Hold-out random forests with variable importance.

## Author(s)

Marvin N. Wright

## References

Janitza, S., Celik, E. & Boulesteix, A.-L., (2015). A computationally fast variable importance test for random forests for high-dimensional data. Adv Data Anal Classif https://doi.org/10.1007/s11634-016-0276-4.

## See Also

`grandforest`

## importance.grandforest *Grand Forest variable importance*

### Description

Extract variable importance of grandforest object.

### Usage

```
## S3 method for class 'grandforest'
importance(x, ...)
```

### Arguments

| | |
|---|---|
| x | grandforest object. |
| ... | Further arguments passed to or from other methods. |

### Value

Variable importance measures.

### Author(s)

Marvin N. Wright

### See Also

grandforest

## importance_pvalues *Grand Forest variable importance p-values*

### Description

Compute variable importance with p-values.

### Usage

```
importance_pvalues(x, method = c("janitza", "altmann"),
  num.permutations = 100, formula = NULL, data = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | grandforest or holdoutRF object. |
| method | Method to compute p-values. Use "janitza" for the method by Janitza et al. (2015) or "altmann" for the non-parametric method by Altmann et al. (2010). |
| num.permutations | Number of permutations. Used in the "altmann" method only. |
| formula | Object of class formula or character describing the model to fit. Used in the "altmann" method only. |
| data | Training data of class data.frame or matrix. Used in the "altmann" method only. |
| ... | Further arguments passed to grandforest(). Used in the "altmann" method only. |

### Value

Variable importance and p-values.

### Author(s)

Marvin N. Wright

### References

Janitza, S., Celik, E. & Boulesteix, A.-L., (2015). A computationally fast variable importance test for random forests for high-dimensional data. Adv Data Anal Classif https://doi.org/10.1007/s11634-016-0276-4.
Altmann, A., Tolosi, L., Sander, O. & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure, Bioinformatics 26(10):1340-1347.

### See Also

grandforest

---

## network *Small 20-gene example network*

---

### Description

Each row corresponds to an edge (interaction) between the features (genes) in each column.

### Usage

```
network
```

### Format

A data frame with 100 rows and 2 columns.

## parse.formula *Parse formula*

### Description

Parse formula and return dataset containing selected columns. Interactions are supported for numerical columns only. An interaction column is the product of all interacting columns.

### Usage

```
parse.formula(formula, data)
```

### Arguments

| | |
|---|---|
| formula | Object of class formula or character describing the model to fit. |
| data | Training data of class data.frame. |

### Value

Dataset including selected columns and interactions.

## predict.grandforest *Grand Forest prediction*

### Description

Prediction with new data and a saved forest from Grand Forest

### Usage

```
## S3 method for class 'grandforest'
predict(object, data, predict.all = FALSE,
  num.trees = object$num.trees, type = "response", seed = NULL,
  num.threads = NULL, verbose = TRUE, ...)
```

### Arguments

| | |
|---|---|
| object | Grand Forest grandforest object. |
| data | New test data of class data.frame or gwaa.data (GenABEL). |
| predict.all | Return individual predictions for each tree instead of aggregated predictions for all trees. Return a matrix (sample x tree) for classification and regression, a 3d array for probability estimation (sample x class x tree) and survival (sample x time x tree). |
| num.trees | Number of trees used for prediction. The first num.trees in the forest are used. |

| | |
|---|---|
| type | Type of prediction. One of 'response', 'se', 'terminalNodes' with default 'response'. See below for details. |
| seed | Random seed used in Grand Forest |
| num.threads | Number of threads. Default is number of CPUs available. |
| verbose | Verbose output on or off. |
| ... | further arguments passed to or from other methods. |

## Details

For `type = 'response'` (the default), the predicted classes (classification), predicted numeric values (regression), predicted probabilities (probability estimation) or survival probabilities (survival) are returned. For `type = 'se'`, the standard error of the predictions are returned (regression only). The jackknife-after-bootstrap is used to estimate the standard errors based on out-of-bag predictions. See Wager et al. (2014) for details. For `type = 'terminalNodes'`, the IDs of the terminal node in each tree for each observation in the given dataset are returned.

For classification and `predict.all = TRUE`, a factor levels are returned as numerics. To retrieve the corresponding factor levels, use `rf$forest$levels`, if `rf` is the grandforest object.

## Value

Object of class `grandforest.prediction` with elements

| | |
|---|---|
| predictions | Predicted classes/values (only for classification and regression) |
| unique.death.times | Unique death times (only for survival). |
| chf | Estimated cumulative hazard function for each sample (only for survival). |
| survival | Estimated survival function for each sample (only for survival). |
| num.trees | Number of trees. |
| num.independent.variables | Number of independent variables. |
| treetype | Type of forest/tree. Classification, regression or survival. |
| num.samples | Number of samples. |

## Author(s)

Marvin N. Wright

## References


- Wright, M. N. & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.
- Wager, S., Hastie T., & Efron, B. (2014). Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife. J Mach Learn Res 15:1625-1651. http://jmlr.org/papers/v15/wager14a.html.


## See Also

grandforest

predict.grandforest.forest *Grand Forest prediction*

## Description

Prediction with new data and a saved forest from Grand Forest

## Usage

```
## S3 method for class 'grandforest.forest'
predict(object, data, predict.all = FALSE,
  num.trees = object$num.trees, type = "response", seed = NULL,
  num.threads = NULL, verbose = TRUE, inbag.counts = NULL, ...)
```

## Arguments

| | |
|---|---|
| object | Grand Forest grandforest.forest object. |
| data | New test data of class data.frame or gwaa.data (GenABEL). |
| predict.all | Return individual predictions for each tree instead of aggregated predictions for all trees. Return a matrix (sample x tree) for classification and regression, a 3d array for probability estimation (sample x class x tree) and survival (sample x time x tree). |
| num.trees | Number of trees used for prediction. The first num.trees in the forest are used. |
| type | Type of prediction. One of 'response', 'se', 'terminalNodes' with default 'response'. See below for details. |
| seed | Random seed used in grandforest |
| num.threads | Number of threads. Default is number of CPUs available. |
| verbose | Verbose output on or off. |
| inbag.counts | Number of times the observations are in-bag in the trees. |
| ... | further arguments passed to or from other methods. |

## Details

For type = 'response' (the default), the predicted classes (classification), predicted numeric values (regression), predicted probabilities (probability estimation) or survival probabilities (survival) are returned. For type = 'se', the standard error of the predictions are returned (regression only). The jackknife-after-bootstrap is used to estimate the standard errors based on out-of-bag predictions. See Wager et al. (2014) for details. For type = 'terminalNodes', the IDs of the terminal node in each tree for each observation in the given dataset are returned.

For classification and predict.all = TRUE, a factor levels are returned as numerics. To retrieve the corresponding factor levels, use rf$forest$levels, if rf is the grandforest object.

## Value

Object of class `grandforest.prediction` with elements

| | |
|---|---|
| predictions | Predicted classes/values (only for classification and regression) |
| unique.death.times | Unique death times (only for survival). |
| chf | Estimated cumulative hazard function for each sample (only for survival). |
| survival | Estimated survival function for each sample (only for survival). |
| num.trees | Number of trees. |
| num.independent.variables | Number of independent variables. |
| treetype | Type of forest/tree. Classification, regression or survival. |
| num.samples | Number of samples. |

## Author(s)

Marvin N. Wright

## References


- Wright, M. N. & Ziegler, A. (2017). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.
- Wager, S., Hastie T., & Efron, B. (2014). Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife. J Mach Learn Res 15:1625-1651. http://jmlr.org/papers/v15/wager14a.html.


## See Also

grandforest

---

# predictions.grandforest

*Grand Forest predictions*

---

## Description

Extract training data predictions of Grand Forest object.

## Usage

```
## S3 method for class 'grandforest'
predictions(x, ...)
```

## Arguments

| | |
|---|---|
| x | grandforest object. |
| ... | Further arguments passed to or from other methods. |

## Value

Predictions: Classes for Classification forests, Numerical values for Regressions forests and the estimated survival functions for all individuals for Survival forests.

### Author(s)

Marvin N. Wright

### See Also

grandforest

---

`predictions.grandforest.prediction`
*Grand Forest predictions*

---

### Description

Extract predictions of Grand Forest prediction object.

### Usage

```
## S3 method for class 'grandforest.prediction'
predictions(x, ...)
```

### Arguments

| | |
|---|---|
| x | Grand Forest prediction object. |
| ... | Further arguments passed to or from other methods. |

### Value

Predictions: Classes for Classification forests, Numerical values for Regressions forests and the estimated survival functions for all individuals for Survival forests.

### Author(s)

Marvin N. Wright

### See Also

grandforest

## print.grandforest *Print Grand Forest*

### Description

Print contents of Grand Forest object.

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class 'grandforest'. |
| ... | Further arguments passed to or from other methods. |

### Author(s)

Marvin N. Wright

### See Also

grandforest

## print.grandforest.forest *Print Grand Forest forest*

### Description

Print contents of Grand Forest forest object.

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class 'grandforest.forest'. |
| ... | further arguments passed to or from other methods. |

### Author(s)

Marvin N. Wright

## print.grandforest.prediction

*Print Grand Forest prediction*

### Description

Print contents of Greand Forest prediction object.

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class 'grandforest.prediction'. |
| ... | further arguments passed to or from other methods. |

### Author(s)

Marvin N. Wright

## proximity.grandforest.prediction

*Grand Forest sample proximity*

### Description

Extract sample proximities from grandforest.prediction object.

### Usage

```
## S3 method for class 'grandforest.prediction'
proximity(x, ...)
```

### Arguments

| | |
|---|---|
| x | grandforest.prediction object. |
| ... | Further arguments passed to or from other methods. |

### Details

The prediction object must have been called with type = "terminalNodes".

### Value

Numeric matrix with sample similarities.

## survival *Synthetic example data set for survival model.*

### Description

Synthetic example data set for survival model.

### Usage

```
survival
```

### Format

A data frame with 100 rows and 22 columns.

## timepoints.grandforest *Grand Forest timepoints*

### Description

Extract unique death times of Grand Forest Survival forest

### Usage

```
## S3 method for class 'grandforest'
timepoints(x, ...)
```

### Arguments

| | |
|---|---|
| x | Grand Forest Survival forest object. |
| ... | Further arguments passed to or from other methods. |

### Value

Unique death times

### Author(s)

Marvin N. Wright

### See Also

grandforest

`timepoints.grandforest.prediction` *Grand Forest timepoints*

## Description

Extract unique death times of Grand Forest Survival prediction object.

## Usage

```
## S3 method for class 'grandforest.prediction'
timepoints(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Grand Forest Survival prediction object. |
| `...` | Further arguments passed to or from other methods. |

## Value

Unique death times

## Author(s)

Marvin N. Wright

## See Also

`grandforest`

# Index