Package 'ranger'

April 7, 2016

Type Package

Title A Fast Implementation of Random Forests

Version 0.4.0

Date 2016-04-07

Author Marvin N. Wright

Maintainer Marvin N. Wright <wright@imbs.uni-luebeck.de>

Description A fast implementation of Random Forests, particularly suited for high dimensional data. Ensembles
of classification, regression, survival and probability prediction trees are supported. Data from genome-wide association studies can be analyzed efficiently. In addition to data frames, datasets of class 'gwaa.data' (R package 'GenABEL') can be directly analyzed.

License GPL-3

Imports Rcpp (>= 0.11.2)

LinkingTo Rcpp

Depends R (>= 3.1)

Suggests survival, testthat

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-04-07 18:27:32

R topics documented:

`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, in press. DOI: 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 of observations.*

Description

Get terminal node IDs of observations.

Usage

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

Arguments

rf	ranger 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(ranger)
rf <- ranger(Species ~ ., data = iris, write.forest = TRUE)
getTerminalNodeIDs(rf, iris)
```

importance.ranger *Ranger variable importance*

Description

Extract variable importance of Ranger object.

Usage

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

Arguments

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

Value

Variable importance measures.

Author(s)

Marvin N. Wright

See Also

ranger

predict.ranger *Ranger prediction*

Description

Prediction with new data and a saved forest from Ranger.

Usage

```
## S3 method for class 'ranger'
predict(object, data, predict.all = FALSE, seed = NULL,
  num.threads = NULL, verbose = TRUE, ...)
```

Arguments

object	Ranger ranger object.
data	New test data of class data.frame or gwaa.data (GenABEL).
predict.all	Return a matrix with individual predictions for each tree instead of aggregated predictions for all trees (classification and regression only).
seed	Random seed used in Ranger.
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 classification and predict.all = TRUE, a matrix of factor levels is returned. To retrieve the corresponding factor levels, use rf$forest$levels, if rf is the ranger object.

Value

Object of class ranger.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

See Also

ranger

---

predict.ranger.forest *Ranger prediction*

Description

Prediction with new data and a saved forest from Ranger.

Usage

```
## S3 method for class 'ranger.forest'
predict(object, data, predict.all = FALSE,
  seed = NULL, num.threads = NULL, verbose = TRUE, ...)
```

Arguments

object	Ranger ranger.forest object.
data	New test data of class data.frame or gwaa.data (GenABEL).
predict.all	Return a matrix with individual predictions for each tree instead of aggregated predictions for all trees (classification and regression only).
seed	Random seed used in Ranger.
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.

Value

Object of class ranger.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

See Also

ranger

---

predictions.ranger *Ranger predictions*

Description

Extract training data predictions of Ranger object.

Usage

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

Arguments

x	Ranger 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

ranger

---

predictions.ranger.prediction

Ranger predictions

Description

Extract predictions of Ranger prediction object.

Usage

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

Arguments

x	Ranger 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

ranger

`print.ranger` *Print Ranger*

Description

Print contents of Ranger object.

Usage

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

Arguments

x	Object of class 'ranger'.
...	Further arguments passed to or from other methods.

Author(s)

Marvin N. Wright

See Also

ranger

`print.ranger.forest` *Print Ranger forest*

Description

Print contents of Ranger forest object.

Usage

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

Arguments

x	Object of class 'ranger.forest'.
...	further arguments passed to or from other methods.

Author(s)

Marvin N. Wright

print.ranger.prediction *Print Ranger prediction*

Description

Print contents of Ranger prediction object.

Usage

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

Arguments

x	Object of class 'ranger.prediction'.
...	further arguments passed to or from other methods.

Author(s)

Marvin N. Wright

ranger *Ranger*

Description

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).

Usage

```
ranger(formula = NULL, data = NULL, num.trees = 500, mtry = NULL,
  importance = "none", write.forest = FALSE, probability = FALSE,
  min.node.size = NULL, replace = TRUE, sample.fraction = ifelse(replace,
  1, 0.632), splitrule = NULL, case.weights = NULL,
  split.select.weights = NULL, always.split.variables = NULL,
  respect.unordered.factors = FALSE, scale.permutation.importance = FALSE,
  keep.inbag = 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.
data	Training data of class data.frame, matrix or gwaa.data (GenABEL).
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', 'permutation'. The 'impurity' measure is the Gini index for classification and the variance of the responses for regression. For survival, only 'permutation' is available.
write.forest	Save ranger.forest object, needed for prediction.
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.
splitrule	Splitting rule, survival only. The splitting rule can be chosen of "logrank" and "C" with default "logrank".
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.
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 tried for splitting.
respect.unordered.factors	Regard unordered factor covariates as unordered categorical variables. If FALSE, all factors are regarded ordered.
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.
num.threads	Number of threads. Default is number of CPUs available.
save.memory	Use memory saving (but slower) splitting mode. No effect for GWAS data.
verbose	Verbose output on or off.
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

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 splitting rule for classification, the estimated response variances for regression and the log-rank test for survival. For Survival the log-rank test or an AUC-based splitting rule are available.

With the `probability` option and factor dependent variable a probability forest is grown. Here, the estimated response variances are used for splitting, as in regression forests. Predictions are class probabilities for each sample. For details see Malley et al. (2012).

Note that for classification and regression nodes with size smaller than min.node.size can occur, like in original Random Forest. For survival all nodes contain at least min.node.size samples. Variables selected with `always.split.variables` are tried additionaly 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 usage of `split.select.weights` can increase the computation times for large forests.

For a large number of variables and data frame 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.

For GWAS data consider combining `ranger` 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.

See https://github.com/mnwright/ranger for the development version.

Notes:

- Multithreading is currently not supported for Microsoft Windows platforms.

Value

Object of class `ranger` 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).

`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).

`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)

Marvin N. Wright

References

Wright, M. N. & Ziegler, A. (2016). ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. Journal of Statistical Software, in press. http://arxiv.org/abs/1508.04409.

Breiman, L. (2001). Random forests. Mach Learn, 45(1), 5-32.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. Ann Appl Stat, 841-860.

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(1), 74.

See Also

`predict.ranger`

Examples

```
require(ranger)

## Classification forest with default settings
ranger(Species ~ ., data = iris)

## Prediction
train.idx <- sample(nrow(iris), 2/3 * nrow(iris))
iris.train <- iris[train.idx, ]
iris.test <- iris[-train.idx, ]
rg.iris <- ranger(Species ~ ., data = iris.train, write.forest = TRUE)
pred.iris <- predict(rg.iris, dat = iris.test)
table(iris.test$Species, pred.iris$predictions)

## Variable importance
rg.iris <- ranger(Species ~ ., data = iris, importance = "impurity")
rg.iris$variable.importance

## Survival forest
require(survival)
rg.veteran <- ranger(Surv(time, status) ~ ., data = veteran)
plot(rg.veteran$unique.death.times, rg.veteran$survival[1,])

## Alternative interface
ranger(dependent.variable.name = "Species", data = iris)

## Not run:
## Use GenABEL interface to read Plink data into R and grow a classification forest
## The ped and map files are not included
library(GenABEL)
convert.snp.ped("data.ped", "data.map", "data.raw")
dat.gwaa <- load.gwaa.data("data.pheno", "data.raw")
phdata(dat.gwaa)$trait <- factor(phdata(dat.gwaa)$trait)
ranger(trait ~ ., data = dat.gwaa)

## End(Not run)
```

---

timepoints.ranger *Ranger timepoints*

Description

Extract unique death times of Ranger Survival forest

Usage

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

Arguments

x	Ranger Survival forest object.
...	Further arguments passed to or from other methods.

Value

Unique death times

Author(s)

Marvin N. Wright

See Also

ranger

---

timepoints.ranger.prediction
Ranger timepoints

Description

Extract unique death times of Ranger Survival prediction object.

Usage

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

Arguments

x	Ranger Survival prediction object.
...	Further arguments passed to or from other methods.

Value

Unique death times

Author(s)

Marvin N. Wright

See Also

ranger

Index