

# Package ‘tuneRanger’

January 22, 2019

**Type** Package

**Title** Tune Random Forest of the 'ranger' Package

**Description** Tuning random forest with one line. The package is mainly based on the packages 'ranger' and 'mlrMBO'.

**Version** 0.4

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 3.0.2), ranger (>= 0.8.0), mlrMBO (>= 1.1.1), parallel (>= 3.3.3), lubridate (>= 1.6.0), lhs (>= 0.14)

**Imports** mlr (>= 2.11), smooof (>= 1.5.1), ParamHelpers (>= 1.10), methods (>= 3.3.3), BBmisc (>= 1.11), DiceKriging (>= 1.5.5)

**LazyData** yes

**ByteCompile** yes

**Date** 2019-01-22

**RoxygenNote** 6.1.1

**Suggests** testthat

**NeedsCompilation** no

**Author** Philipp Probst [aut, cre],  
Simon Klau [ctb]

**Maintainer** Philipp Probst <philipp\_probst@gmx.de>

**Repository** CRAN

**Date/Publication** 2019-01-22 14:30:03 UTC

## R topics documented:

estimateTimeTuneRanger . . . . .	2
restartTuneRanger . . . . .	2
tuneRanger . . . . .	3

<b>Index</b>	<b>6</b>
--------------	----------

---

```
estimateTimeTuneRanger
```

```
estimateTimeTuneRanger
```

---

### Description

estimateTimeTuneRanger

### Usage

```
estimateTimeTuneRanger(task, iters = 100, num.threads = 1,
  num.trees = 1000, respect.unordered.factors = "order")
```

### Arguments

task	The mlr task created by makeClassifTask or makeRegrTask.
iters	Number of iterations.
num.threads	Number of threads. Default is 1.
num.trees	Number of trees.
respect.unordered.factors	Handling of unordered factor covariates. One of 'ignore', 'order' and 'partition'. 'order' is the default.

### Value

estimated time for the tuning procedure

### Examples

```
estimateTimeTuneRanger(iris.task)
```

---

```
restartTuneRanger
```

```
restartTuneRanger
```

---

### Description

Restarts the tuning process if an error occurred.

### Usage

```
restartTuneRanger(save.file.path = "optpath.RData", task,
  measure = NULL)
```

**Arguments**

save.file.path	File name in the current working directory to which interim results were saved by <code>tuneRanger</code> .
task	The mlr task created by <code>makeClassifTask</code> or <code>makeRegrTask</code> .
measure	Performance measure that was already used in the original <code>tuneRanger</code> process.

**Value**

	A list with elements
recommended.pars	Recommended hyperparameters.
results	A data.frame with all evaluated hyperparameters and performance and time results for each run.
	No model is build.

**Examples**

```
## Not run:
library(tuneRanger)
library(mlr)

# iris is a bit nonsense here
# A mlr task has to be created in order to use the package
# the already existing iris task is used here
estimateTimeTuneRanger(iris.task)
# temporarily file name to save results
path = tempfile()
res = tuneRanger(iris.task, measure = list(multiclass.brier), num.trees = 1000,
  num.threads = 8, iters = 70, save.file.path = path)

# Mean of best 5 % of the results
res

# Restart after failing in one of the iterations:
res = restartTuneRanger(save.file.path = path, iris.task,
  measure = list(multiclass.brier))
## End(Not run)
```

---

tuneRanger

*tuneRanger*


---

**Description**

Automatic tuning of random forests of the `ranger` package with one line of code.

## Usage

```
tuneRanger(task, measure = NULL, iters = 70, iters.warmup = 30,
  time.budget = NULL, num.threads = NULL, num.trees = 1000,
  parameters = list(replace = FALSE, respect.unordered.factors =
    "order"), tune.parameters = c("mtry", "min.node.size",
    "sample.fraction"), save.file.path = NULL, build.final.model = TRUE,
  show.info = getOption("mlrMBO.show.info", TRUE))
```

## Arguments

task	The mlr task created by <a href="#">makeClassifTask</a> or <a href="#">makeRegrTask</a> .
measure	Performance measure to evaluate/optimize. Default is brier score for classification and mse for regression. Can be changed to accuracy, AUC or logarithmic loss by setting it to <code>list(acc)</code> , <code>list(auc)</code> or <code>list(logloss)</code> . Other possible performance measures from mlr can be looked up in the <a href="#">mlr tutorial</a> .
iters	Number of iterations. Default is 70.
iters.warmup	Number of iterations for the warmup. Default is 30.
time.budget	Running time budget in seconds. Note that the actual mbo run can take more time since the condition is checked after each iteration. The default NULL means: There is no time budget.
num.threads	Number of threads. Default is number of CPUs available.
num.trees	Number of trees.
parameters	Optional list of fixed named parameters that should be passed to <a href="#">ranger</a> .
tune.parameters	Optional character vector of parameters that should be tuned. Default is <code>mtry</code> , <code>min.node.size</code> and <code>sample.fraction</code> . Additionally <code>replace</code> and <code>respect.unordered.factors</code> can be included in the tuning process.
save.file.path	File to which interim results are saved (e.g. "optpath.RData") in the current working directory. Default is NULL, which does not save the results. If a file was specified and one iteration fails the algorithm can be started again with <a href="#">restartTuneRanger</a> .
build.final.model	[logical(1)] Should the best found model be fitted on the complete dataset? Default is TRUE.
show.info	Verbose mlrMBO output on console? Default is TRUE.

## Details

Model based optimization is used as tuning strategy and the three parameters `min.node.size`, `sample.fraction` and `mtry` are tuned at once. Out-of-bag predictions are used for evaluation, which makes it much faster than other packages and tuning strategies that use for example 5-fold cross-validation. Classification as well as regression is supported. The measure that should be optimized can be chosen from the list of measures in mlr: [mlr tutorial](#)

**Value**

recommended.pars	A list with elements Recommended hyperparameters.
results	A data.frame with all evaluated hyperparameters and performance and time results for each run.
model	The final model if build.final.model set to TRUE.

**See Also**

[estimateTimeTuneRanger](#) for time estimation and [restartTuneRanger](#) for continuing the algorithm if there was an error.

**Examples**

```
## Not run:
library(tuneRanger)
library(mlr)

# A mlr task has to be created in order to use the package
data(iris)
iris.task = makeClassifTask(data = iris, target = "Species")

# Estimate runtime
estimateTimeTuneRanger(iris.task)
# Tuning
res = tuneRanger(iris.task, measure = list(multiclass.brier), num.trees = 1000,
  num.threads = 2, iters = 70, save.file.path = NULL)

# Mean of best 5 % of the results
res
# Model with the new tuned hyperparameters
res$model
# Prediction
predict(res$model, newdata = iris[1:10,])
## End(Not run)
```

# Index

`estimateTimeTuneRanger`, [2](#), [5](#)

`makeClassifTask`, [3](#), [4](#)

`makeRegrTask`, [3](#), [4](#)

`ranger`, [3](#), [4](#)

`restartTuneRanger`, [2](#), [4](#), [5](#)

`tuneRanger`, [3](#), [3](#)