

# Package ‘OOBCurve’

May 7, 2026

**Type** Package

**Title** Out of Bag Learning Curve

**Description** Provides functions to calculate the out-of-bag learning curve for random forests for any measure that is available in the 'mlr' package. Supported random forest packages are 'randomForest' and 'ranger' and trained models of these packages with the train function of 'mlr'. The main function is OOBCurve() that calculates the out-of-bag curve depending on the number of trees. With the OOBCurvePars() function out-of-bag curves can also be calculated for 'mtry', 'sample.fraction' and 'min.node.size' for the 'ranger' package.

**URL** <https://github.com/PhilippPro/OOBCurve>

**BugReports** <https://github.com/PhilippPro/OOBCurve/issues>

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 3.3.3), mlr (>= 2.11)

**Imports** randomForest, ranger

**LazyData** yes

**ByteCompile** yes

**Version** 0.3

**Date** 2018-08-30

**RoxygenNote** 6.0.1

**Suggests** testthat

**NeedsCompilation** no

**Author** Philipp Probst [aut, cre]

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

**Repository** CRAN

**Date/Publication** 2018-08-30 15:00:03 UTC

## Contents

OOBCurve	2
OOBCurvePars	3

---

OOBCurve	<i>Out of Bag Learning curve</i>
----------	----------------------------------

---

### Description

With the help of this function the out of bag learning curve for random forests can be created for any measure that is available in the mlr package.

### Usage

```
OOBCurve(mod, measures = list(auc), task, data)
```

### Arguments

mod	An object of class <code>randomForest</code> or <code>ranger</code> , as that created by the function <a href="#">randomForest/ranger</a> with option <code>keep.inbag = TRUE</code> . Alternatively you can also use a <code>randomForest</code> or <code>ranger</code> model trained with <a href="#">train</a> of <code>mlr</code> .
measures	List of performance measure(s) of mlr to evaluate. Default is <code>auc</code> only. See the <a href="#">mlr tutorial</a> for a list of available measures for the corresponding task.
task	Learning task created by the function <a href="#">makeClassifTask</a> or <a href="#">makeRegrTask</a> of <code>mlr</code> .
data	Original data that was used for training the random forest.

### Value

Returns a dataframe with a column for each desired measure.

### See Also

[OOBCurvePars](#) for out-of-bag curves of other parameters.

### Examples

```
library(mlr)
library(ranger)

# Classification
data = getTaskData(sonar.task)
sonar.task = makeClassifTask(data = data, target = "Class")
lrn = makeLearner("classif.ranger", keep.inbag = TRUE, par.vals = list(num.trees = 100))
mod = train(lrn, sonar.task)

# Alternatively use ranger directly
# mod = ranger(Class ~., data = data, num.trees = 100, keep.inbag = TRUE)
# Alternatively use randomForest
# mod = randomForest(Class ~., data = data, ntree = 100, keep.inbag = TRUE)
```

```

# Application of the main function
results = OOBCurve(mod, measures = list(mmce, auc, brier), task = sonar.task, data = data)
# Plot the generated results
plot(results$mmce, type = "l", ylab = "oob-mmce", xlab = "ntrees")
plot(results$auc, type = "l", ylab = "oob-auc", xlab = "ntrees")
plot(results$brier, type = "l", ylab = "oob-brier-score", xlab = "ntrees")

# Regression
data = getTaskData(bh.task)
bh.task = makeRegrTask(data = data, target = "medv")
lrn = makeLearner("regr.ranger", keep.inbag = TRUE, par.vals = list(num.trees = 100))
mod = train(lrn, bh.task)

# Application of the main function
results = OOBCurve(mod, measures = list(mse, mae, rsq), task = bh.task, data = data)
# Plot the generated results
plot(results$mse, type = "l", ylab = "oob-mse", xlab = "ntrees")
plot(results$mae, type = "l", ylab = "oob-mae", xlab = "ntrees")
plot(results$rsq, type = "l", ylab = "oob-mae", xlab = "ntrees")

```

---

OOBCurvePars

*OOBCurvePars*


---

## Description

With the help of this function the out of bag curves for parameters like `mtry`, `sample.fraction` and `min.node.size` of random forests can be created for any measure that is available in the `mlr` package.

## Usage

```
OOBCurvePars(lrn, task, pars = c("mtry"), nr.grid = 10, par.vals = NULL,
  measures = list(auc))
```

## Arguments

<code>lrn</code>	The learner created with <code>makeLearner</code> . Currently only <code>ranger</code> is supported. <code>num.trees</code> has to be set sufficiently high to produce smooth curves.
<code>task</code>	Learning task created by the function <code>makeClassifTask</code> or <code>makeRegrTask</code> of <code>mlr</code> .
<code>pars</code>	One of the hyperparameter "mtry", "sample.fraction" or "min.node.size".
<code>nr.grid</code>	Number of points on hyperparameter space that should be evaluated (distributed equally)
<code>par.vals</code>	Optional vector of hyperparameter points that should be evaluated. If set, <code>nr.grid</code> is not used anymore. Default is <code>NULL</code> .
<code>measures</code>	List of performance measure(s) of <code>mlr</code> to evaluate. Default is <code>mmce</code> for classification and <code>mse</code> for regression. See the <code>mlr tutorial</code> for a list of available measures for the corresponding task.

**Value**

Returns a list with parameter values and a list of performances.

**See Also**

[OOBCurve](#) for out-of-bag curves dependent on the number of trees.

**Examples**

```
## Not run:
library(mlr)
task = sonar.task

lrn = makeLearner("classif.ranger", predict.type = "prob", num.trees = 1000)
results = OOBCurvePars(lrn, task, measures = list(auc))
plot(results$par.vals, results$performances$auc, type = "l", xlab = "mtry", ylab = "auc")

lrn = makeLearner("classif.ranger", predict.type = "prob", num.trees = 1000, replace = FALSE)
results = OOBCurvePars(lrn, task, pars = "sample.fraction", measures = list(mmce))
plot(results$par.vals, results$performances$mmce, type = "l", xlab = "sample.fract.", ylab = "mmce")

results = OOBCurvePars(lrn, task, pars = "min.node.size", measures = list(mmce))
plot(results$par.vals, results$performances$mmce, type = "l", xlab = "min.node.size", ylab = "mmce")
## End(Not run)
```

# Index

`makeClassifTask`, [2](#), [3](#)

`makeLearner`, [3](#)

`makeRegrTask`, [2](#), [3](#)

`OOBCurve`, [2](#), [4](#)

`OOBCurvePars`, [2](#), [3](#)

`randomForest`, [2](#)

`ranger`, [2](#), [3](#)

`train`, [2](#)