

# Package ‘RLT’

October 12, 2022

**Version** 3.2.4

**Date** 2022-01-24

**Title** Reinforcement Learning Trees

**Author** Ruoqing Zhu <teazrq@gmail.com>

**Maintainer** Ruoqing Zhu <teazrq@gmail.com>

**Suggests** randomForest, survival

**Description** Random forest with a variety of additional features for regression, classification and survival analysis. The features include: parallel computing with OpenMP, embedded model for selecting the splitting variable (based on Zhu, Zeng & Kosorok, 2015), subject weight, variable weight, tracking subjects used in each tree, etc.

**License** GPL (>= 2)

**URL** <https://cran.r-project.org/package=RLT>

**NeedsCompilation** yes

**Repository** CRAN

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Date/Publication** 2022-01-25 10:12:42 UTC

## R topics documented:

MuteRate	2
predict.RLT	2
print.RLT	3
RLT	3

<b>Index</b>	<b>8</b>
--------------	----------

---

MuteRate	<i>Muting rate</i>
----------	--------------------

---

**Description**

Get the muting rate based on sample size N and dimension P. This is an experimental feature. When P is too small, this is not recommended.

**Usage**

```
MuteRate(N, P, speed = NULL, info = FALSE)
```

**Arguments**

N	sample size
P	dimension
speed	Muting speed: moderate or aggressive
info	Whether to output detailed information

**Value**

A suggested muting rate

**Examples**

```
MuteRate(500, 100, speed = "aggressive")
```

---

predict.RLT	<i>Prediction function for reinforcement learning trees</i>
-------------	---

---

**Description**

Predict future subjects with a fitted RLT model

**Usage**

```
## S3 method for class 'RLT'
predict(object, testx, ...)
```

**Arguments**

object	A fitted RLT object
testx	Testing data
...	...

**Value**

The predicted values. For survival model, it returns the fitted survival functions

**Examples**

```
x = matrix(rnorm(100), ncol = 10)
y = rowMeans(x)
fit = RLT(x, y, ntrees = 5)
predict(fit, x)
```

---

print.RLT	<i>Print a RLT object</i>
-----------	---------------------------

---

**Description**

Print a RLT object

**Usage**

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

**Arguments**

x	A fitted RLT object
...	...

**Examples**

```
x = matrix(rnorm(100), ncol = 10)
y = rowMeans(x)
fit = RLT(x, y, ntrees = 5)
fit
```

---

RLT	<i>Main function of reinforcement learning trees</i>
-----	--

---

**Description**

Fit models for regression, classification and survival analysis using reinforced splitting rules

**Usage**

```

RLT(
  x,
  y,
  censor = NULL,
  model = "regression",
  print.summary = 0,
  use.cores = 1,
  ntrees = if (reinforcement) 100 else 500,
  mtry = max(1, as.integer(ncol(x)/3)),
  nmin = max(1, as.integer(log(nrow(x)))),
  alpha = 0.4,
  split.gen = "random",
  nsplit = 1,
  resample.prob = 0.9,
  replacement = TRUE,
  npermute = 1,
  select.method = "var",
  subject.weight = NULL,
  variable.weight = NULL,
  track.obs = FALSE,
  importance = TRUE,
  reinforcement = FALSE,
  muting = -1,
  muting.percent = if (reinforcement) MuteRate(nrow(x), ncol(x), speed = "aggressive",
    info = FALSE) else 0,
  protect = as.integer(log(ncol(x))),
  combsplit = 1,
  combsplit.th = 0.25,
  random.select = 0,
  embed.n.th = 4 * nmin,
  embed.ntrees = max(1, -atan(0.01 * (ncol(x) - 500))/pi * 100 + 50),
  embed.resample.prob = 0.8,
  embed.mtry = 1/2,
  embed.nmin = as.integer(nrow(x)^(1/3)),
  embed.split.gen = "random",
  embed.nsplit = 1
)

```

**Arguments**

<code>x</code>	A matrix or data.frame for features
<code>y</code>	Response variable, a numeric/factor vector or a Surv object
<code>censor</code>	The censoring indicator if survival model is used
<code>model</code>	The model type: regression, classification or survival
<code>print.summary</code>	Whether summary should be printed
<code>use.cores</code>	Number of cores

<code>ntrees</code>	Number of trees, <code>ntrees = 100</code> if use reinforcement, <code>ntrees = 1000</code> otherwise
<code>mtry</code>	Number of variables used at each internal node, only for reinforcement = FALSE
<code>nmin</code>	Minimum number of observations required in an internal node to perform a split. Set this to twice of the desired terminal node size.
<code>alpha</code>	Minimum number of observations required for each child node as a portion of the parent node. Must be within $(0, 0.5]$ .
<code>split.gen</code>	How the cutting points are generated
<code>nsplit</code>	Number of random cutting points to compare for each variable at an internal node
<code>resample.prob</code>	Proportion of in-bag samples
<code>replacement</code>	Whether the in-bag samples are sampled with replacement
<code>npermute</code>	Number of imputations (currently not implemented, saved for future use)
<code>select.method</code>	Method to compare different splits
<code>subject.weight</code>	Subject weights
<code>variable.weight</code>	Variable weights when randomly sample <code>mtry</code> to select the splitting rule
<code>track.obs</code>	Track which terminal node the observation belongs to
<code>importance</code>	Should importance measures be calculated
<code>reinforcement</code>	If reinforcement splitting rules should be used. There are default values for all tuning parameters under this feature.
<code>muting</code>	Muting method, <code>-1</code> for muting by proportion, positive for muting by count
<code>muting.percent</code>	Only for <code>muting = -1</code> the proportion of muting
<code>protect</code>	Number of protected variables that will not be muted. These variables are adaptively selected for each tree.
<code>combsplit</code>	Number of variables used in a combination split. <code>combsplit = 1</code> gives regular binary split; <code>combsplit &gt; 1</code> produces linear combination splits.
<code>combsplit.th</code>	The minimum threshold (as a relative measurement compared to the best variable) for a variable to be used in the combination split.
<code>random.select</code>	Randomly select a variable from the top variable in the linear combination as the splitting rule.
<code>embed.n.th</code>	Number of observations to stop the embedded model and choose randomly from the current protected variables.
<code>embed.ntrees</code>	Number of embedded trees
<code>embed.resample.prob</code>	Proportion of in-bag samples for embedded trees
<code>embed.mtry</code>	Number of variables used for embedded trees, as proportion
<code>embed.nmin</code>	Terminal node size for embedded trees
<code>embed.split.gen</code>	How the cutting points are generated in the embedded trees
<code>embed.nsplit</code>	Number of random cutting points for embedded trees

**Value**

A RLT object; a list consisting of

FittedTrees	Fitted tree structure
FittedSurv, timepoints	Terminal node survival estimation and all time points, if survival model is used
AllError	All out-of-bag errors, if importance = TRUE
VarImp	Variable importance measures, if importance = TRUE
ObsTrack	Registration of each observation in each fitted tree
...	All the tuning parameters are saved in the fitted RLT object

**References**

Zhu, R., Zeng, D., & Kosorok, M. R. (2015) "Reinforcement Learning Trees." *Journal of the American Statistical Association*. 110(512), 1770-1784.

Zhu, R., & Kosorok, M. R. (2012). Recursively imputed survival trees. *Journal of the American Statistical Association*, 107(497), 331-340.

**Examples**

```

N = 600
P = 100

X = matrix(runif(N*P), N, P)
Y = rowSums(X[,1:5]) + rnorm(N)

trainx = X[1:200,]
trainy = Y[1:200]
testx = X[-c(1:200),]
testy = Y[-c(1:200)]

# Regular ensemble trees (Extremely Randomized Trees, Geurts, et. al., 2006)

RLT.fit = RLT(trainx, trainy, model = "regression", use.cores = 6)

barplot(RLT.fit$VarImp)
RLT.pred = predict(RLT.fit, testx)
mean((RLT.pred$Prediction - testy)^2)

# Reinforcement Learning Trees, using an embedded model to find the splitting rule
## Not run:
Mark0 = proc.time()
RLT.fit = RLT(trainx, trainy, model = "regression", use.cores = 6, ntrees = 100,
              importance = TRUE, reinforcement = TRUE, combsplit = 3, embed.ntrees = 25)
proc.time() - Mark0

barplot(RLT.fit$VarImp)
RLT.pred = predict(RLT.fit, testx)
mean((RLT.pred$Prediction - testy)^2)

```

## End(Not run)

# Index

MuteRate, [2](#)

predict.RLT, [2](#)

print.RLT, [3](#)

RLT, [3](#)