

# Package ‘nproc’

March 4, 2017

**Type** Package

**Title** Neyman-Pearson Receiver Operating Characteristics

**Version** 2.0.6

**Date** 2017-03-02

**Author** Yang Feng, Jessica Li and Xin Tong

**Maintainer** Yang Feng <yang.feng@columbia.edu>

**Imports** glmnet, e1071, randomForest, MASS, parallel, ada, stats,  
graphics, ROCR, tree

**Description** Given a sample of class 0 and class 1 and a classification method, the package generates the corresponding Neyman-Pearson classifier with a pre-specified type-I error control and Neyman-Pearson Receiver Operating Characteristics.

**License** GPL-2

**LazyData** TRUE

**RoxygenNote** 5.0.1

**URL** <http://arxiv.org/abs/1608.03109>

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2017-03-04 17:12:32

## R topics documented:

compare . . . . .	2
lines.nproc . . . . .	3
npc . . . . .	3
nproc . . . . .	6
plot.nproc . . . . .	8
predict.npc . . . . .	9
rocCV . . . . .	10
<b>Index</b>	<b>12</b>

---

compare

---

*Compare two NP classification methods at different thresholds.*


---

### Description

compare compares NP classification methods and provide the regions where one method is better than the other. The two NP-ROC curves are both required to have band = TRUE.

### Usage

```
compare(roc1, roc2, plot = TRUE, col1 = "black", col2 = "red")
```

### Arguments

roc1	the first nproc object.
roc2	the second nproc object.
plot	whether to generate the two NP-ROC plots and mark the area of significant difference. Default = 'TRUE'.
col1	the color of the region where roc1 is significantly better than roc2. Default = 'black'.
col2	the color of the region where roc2 is significantly better than roc1. Default = 'red'.

### Value

A list with the following items.

alpha1	the alpha values where roc1 is significantly better than roc2.
alpha2	the alpha values where roc2 is significantly better than roc1.
alpha3	the alpha values where roc1 and roc2 are not significantly different.
confidence	represents the confidence level.

### References

Xin Tong, Yang Feng, and Jingyi Jessica Li (2016), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC) curves, manuscript, <http://arxiv.org/abs/1608.03109>.

### See Also

[npc](#), [nproc](#), [predict.npc](#) and [plot.nproc](#)

**Examples**

```

n = 1000
set.seed(1)
x1 = c(rnorm(n), rnorm(n) + 1)
x2 = c(rnorm(n), rnorm(n)*sqrt(6) + 1)
y = c(rep(0,n), rep(1,n))
fit1 = nproc(x1, y, band = TRUE, method = 'lda')
fit2 = nproc(x2, y, band = TRUE, method = 'lda')
v = compare(fit1, fit2)
legend('topleft', legend=c('x1', 'x2'), col=1:2, lty=c(1,1))

```

---

lines.nproc	<i>Add NP-ROC curves to the current plot object.</i>
-------------	--

---

**Description**

Add NP-ROC curves to the current plot object.

**Usage**

```

## S3 method for class 'nproc'
lines(x, ...)

```

**Arguments**

x	fitted NP-ROC object using nproc.
...	additional arguments.

**See Also**

[npc](#), [nproc](#) and [plot.nproc](#).

---

npc	<i>Calculate the Neyman-Pearson Classifier from a sample of class 0 and class 1.</i>
-----	--

---

**Description**

Give a type I error upper bound  $\alpha$  and the violation upper bound  $\delta$ , npc calculate the Neyman-Pearson Classifier which control the type I error with in  $\alpha$  with probability at least  $1-\delta$

**Usage**

```
npc(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "nb", "ada", "tree", "custom"), score = NULL, alpha = 0.05,
  delta = 0.05, split = 1, split.ratio = 0.5, n.cores = 1,
  nproc.param = NULL, randSeed = 0, ...)
```

**Arguments**

x	n * p observation matrix. n observations, p covariates.
y	n 0/1 observations.
method	classification method. <ul style="list-style-type: none"> <li>• logistic: Logistic regression. <a href="#">glm</a> function with family = 'binomial'</li> <li>• penlog: Penalized logistic regression with LASSO penalty. <a href="#">glmnet</a> in <a href="#">glmnet</a> package</li> <li>• svm: Support Vector Machines. <a href="#">svm</a> in <a href="#">e1071</a> package</li> <li>• randomforest: Random Forest. <a href="#">randomForest</a> in <a href="#">randomForest</a> package</li> <li>• Linear Discriminant Analysis. <a href="#">lda</a> in <a href="#">MASS</a> package</li> <li>• nb: Naive Bayes. <a href="#">naiveBayes</a> in <a href="#">e1071</a> package</li> <li>• ada: Ada-Boost. <a href="#">ada</a> in <a href="#">ada</a> package</li> <li>• custom: a custom classifier. score vector needed.</li> </ul>
score	score vector corresponding to y. Required when method = 'custom'.
alpha	the desirable control on type I error. Default = 0.05.
delta	the violation rate of the type I error. Default = 0.05.
split	the number of splits for the class 0 sample. Default = 1. For ensemble version, choose split > 1. When method = 'custom', split = 0 always.
split.ratio	the ratio of splits used for the class 0 sample to train the classifier. Default = 0.5.
n.cores	number of cores used for parallel computing. Default = 1. WARNING: windows machine is not supported.
nproc.param	parameters passed from a nproc call. Default = NULL.
randSeed	the random seed used in the algorithm.
...	additional arguments.

**Value**

An object with S3 class npc.

fits	a list of length max(1,split), represents the fit during each split.
method	the classification method.
split	the number of splits used.

**References**

Xin Tong, Yang Feng, and Jingyi Jessica Li (2016), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), manuscript, <http://arxiv.org/abs/1608.03109>.

**See Also**

[nproc](#) and [predict.npc](#)

**Examples**

```

set.seed(1)
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
xtest = matrix(rnorm(n*2),n,2)
ctest = 1+3*xtest[,1]
ytest = rbinom(n,1,1/(1+exp(-ctest)))

##Use svm classifier and the default type I error control with alpha=0.05
fit = npc(x, y, method = 'svm')
pred = predict(fit,xtest)
fit.score = predict(fit,x)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ', accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')

##Ensembled svm classifier with split = 11, alpha=0.05
#fit = npc(x, y, method = 'svm', split = 11)
#pred = predict(fit,xtest)
#accuracy = mean(pred$pred.label==ytest)
#cat('Overall Accuracy: ', accuracy,'\n')
#ind0 = which(ytest==0)
#typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
#cat('Type I error: ', typeI, '\n')

##Now, change the method to logistic regression and change alpha to 0.1
fit = npc(x, y, method = 'logistic', alpha = 0.1)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ', accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')

##Now, change the method to adaboost
#fit = npc(x, y, method = 'ada', alpha = 0.1)
#pred = predict(fit,xtest)
#accuracy = mean(pred$pred.label==ytest)
#cat('Overall Accuracy: ', accuracy,'\n')
#ind0 = which(ytest==0)
#typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
#cat('Type I error: ', typeI, '\n')

##A 'custom' npc classifier with y and score.
```

```
#fit2 = npc(y = y, score = fit.score$pred.score, method = 'custom')
```

---

nproc

---

*Calculate the Neyman-Pearson Receiver Operating Characteristics*


---

## Description

nproc calculate the Neyman-Pearson Receiver Operating Characteristics curve for a given sequence of type I error values.

## Usage

```
nproc(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "nb", "ada", "tree", "custom"), score = NULL, band = FALSE,
  typeI.lower = FALSE, delta = 0.05, split = 1, split.ratio = 0.5,
  n.cores = 1, randSeed = 0, ...)
```

## Arguments

x	n * p observation matrix. n observations, p covariates.
y	n 0/1 observatons.
method	classification method(s). <ul style="list-style-type: none"> <li>logistic: Logistic regression. <a href="#">glm</a> function with family = 'binomial'</li> <li>penlog: Penalized logistic regression with LASSO penalty. <a href="#">glmnet</a> in <a href="#">glmnet</a> package</li> <li>svm: Support Vector Machines. <a href="#">svm</a> in <a href="#">e1071</a> package</li> <li>randomforest: Random Forest. <a href="#">randomForest</a> in <a href="#">randomForest</a> package</li> <li>Linear Discriminant Analysis. <a href="#">lda</a> in <a href="#">MASS</a> package</li> <li>nb: Naive Bayes. <a href="#">naiveBayes</a> in <a href="#">e1071</a> package</li> <li>ada: Ada-Boost. <a href="#">ada</a> in <a href="#">ada</a> package</li> <li>custom: a custom classifier. score vector needed.</li> </ul>
score	score vector corresponding to y. Required when method = 'custom'.
band	whether to generate two NP-ROC curves representing a confidence band. Default = FALSE.
typeI.lower	whether to generate the data-driven type-I error lower bound. NOTE: experimental feature. Default = FALSE.
delta	the violation rate of the type I error. Default = 0.05.
split	the number of splits for the class 0 sample. Default = 1. For ensemble version, choose split > 1. When method = 'custom', split = 0 always.
split.ratio	the ratio of splits used for the class 0 sample to train the classifier. Default = 0.5.
n.cores	number of cores used for parallel computing. Default = 1.
randSeed	the random seed used in the algorithm.
...	additional arguments.

**Value**

An object with S3 class nproc.

typeI.u	sequence of upper bound of type I error.
typeII.l	sequence of lower bound of type I error.
typeII.u	sequence of upper bound of type II error.
auc.l	the auc value of the lower NP-ROC curve.
auc.u	the auc value of the upper NP-ROC curve.
band	whether the upper NP-ROC curve is generated.
method	the classification method implemented.
delta	the violation rate.

**References**

Xin Tong, Yang Feng, and Jingyi Jessica Li (2016), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), manuscript, <http://arxiv.org/abs/1608.03109>

**See Also**

[npc](#)

**Examples**

```
n = 200
x = matrix(rnorm(n*2),n,2)
c = 1 - 3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
#fit = nproc(x, y, method = 'svm')
fit2 = nproc(x, y, method = 'penlog')
##Plot the nproc curve
plot(fit2)

##custom method
fit.npc = npc(x, y, method = 'svm')
fit.score = predict(fit.npc,x)$pred.score
fit.custom = nproc(y = y, score = fit.score, method = 'custom')

#fit3 = nproc(x, y, method = 'penlog')

##Plot the nproc curve
#plot(fit3)

#fit3 = nproc(x, y, method = 'penlog', n.cores = 2)
#In practice, replace 2 by the number of cores available 'detectCores()'
#fit4 = nproc(x, y, method = 'penlog', n.cores = detectCores())

#Testing the custom method for nproc.
#fit = npc(x, y, method = 'lda', split = 0, n.cores = 2) #use npc to get score list.
#obj = nproc(x = NULL, y = fit$y, method = 'custom', split = 0,
```

```
#score = fit$score, n.cores = 2)

#Confidence nproc curves
#fit6 = nproc(x, y, method = 'lda', band = TRUE)

#nproc ensembled version
#fit7 = nproc(x, y, method = 'lda', split = 11)
```

---

plot.nproc	<i>Plot the nproc curve(s).</i>
------------	---------------------------------

---

## Description

Plot the nproc curve(s).

## Usage

```
## S3 method for class 'nproc'
plot(x, ...)
```

## Arguments

x	fitted nproc object using nproc.
...	additional arguments.

## See Also

[npc](#), [nproc](#)

## Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = nproc(x, y, method = 'svm')
plot(fit)
```



---

predict.npc	<i>Predicting the outcome of a set of new observations using the fitted npc object.</i>
-------------	---

---

## Description

Predicting the outcome of a set of new observations using the fitted npc object.

## Usage

```
## S3 method for class 'npc'
predict(object, newx = NULL, ...)
```

## Arguments

object	fitted npc object using npc.
newx	a set of new observations.
...	additional arguments.

## Value

A list containing the predicted label and score.

pred.label	Predicted label vector.
pred.score	Predicted score vector.

## See Also

[npc](#) and [nproc](#)

## Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
xtest = matrix(rnorm(n*2),n,2)
ctest = 1+3*xtest[,1]
ytest = rbinom(n,1,1/(1+exp(-ctest)))

##Use logistic classifier and the default type I error control with alpha=0.05
#fit = npc(x, y, method = 'logistic')
#pred = predict(fit,xtest)
#fit.score = predict(fit,x)
#accuracy = mean(pred$pred.label==ytest)
#cat('Overall Accuracy: ', accuracy,'\n')
#ind0 = which(ytest==0)
#ind1 = which(ytest==1)
```

```
#typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
#cat('Type I error: ', typeI, '\n')
#typeII = mean(pred$pred.label[ind1]!=ytest[ind1]) #type II error on test set
#cat('Type II error: ', typeII, '\n')
```

---

rocCV	<i>Calculate the Receiver Operating Characteristics with Cross-validation or Subsampling</i>
-------	--

---

## Description

rocCV calculate the receiver operating characterisitic with cross-validation

## Usage

```
rocCV(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "nb", "ada", "tree"), metric = "CV", n.folds = 5,
  train.frac = 0.5, n.cores = 1, randSeed = 0, ...)
```

## Arguments

x	n * p observation matrix. n observations, p covariates.
y	n 0/1 observatons.
method	classification method(s). <ul style="list-style-type: none"> <li>• logistic: Logistic regression. <a href="#">glm</a> function with family = 'binomial'</li> <li>• penlog: Penalized logistic regression with LASSO penalty. <a href="#">glmnet</a> in <a href="#">glmnet</a> package</li> <li>• svm: Support Vector Machines. <a href="#">svm</a> in <a href="#">e1071</a> package</li> <li>• randomforest: Random Forest. <a href="#">randomForest</a> in <a href="#">randomForest</a> package</li> <li>• Linear Discriminant Analysis. lda: <a href="#">lda</a> in <a href="#">MASS</a> package</li> <li>• nb: Naive Bayes. <a href="#">naiveBayes</a> in <a href="#">e1071</a> package</li> <li>• ada: Ada-Boost. <a href="#">ada</a> in <a href="#">ada</a> package</li> </ul>
metric	metric used for averging performance. Includes 'CV' and 'SS' as options. Default = 'CV'.
n.folds	number of folds used for cross-validation or the number of splits in the subsampling. Default = 5.
train.frac	fraction of training data in the subsampling process. Default = 0.5.
n.cores	number of cores used for parallel computing. Default = 1.
randSeed	the random seed used in the algorithm. Default = 0.
...	additional arguments.

**Value**

A list.

fpr                    sequence of false positive rate.

tpr                    sequence of true positive rate.

**References**

Xin Tong, Yang Feng, and Jingyi Jessica Li (2016), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), manuscript, <http://arxiv.org/abs/1608.03109>

**See Also**

[nproc](#)

**Examples**

```
n = 200
x = matrix(rnorm(n*2),n,2)
c = 1 - 3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = rocCV(x, y, method = 'svm')
fit2 = rocCV(x, y, method = 'penlog')
fit3 = rocCV(x, y, method = 'penlog', metric = 'SS')
```

# Index

ada, [4](#), [6](#), [10](#)

compare, [2](#)

glm, [4](#), [6](#), [10](#)

glmnet, [4](#), [6](#), [10](#)

lda, [4](#), [6](#), [10](#)

lines.nproc, [3](#)

naiveBayes, [4](#), [6](#), [10](#)

npc, [2](#), [3](#), [3](#), [7–9](#)

nproc, [2](#), [3](#), [5](#), [6](#), [8](#), [9](#), [11](#)

plot.nproc, [2](#), [3](#), [8](#)

predict.npc, [2](#), [5](#), [9](#)

randomForest, [4](#), [6](#), [10](#)

rocCV, [10](#)

svm, [4](#), [6](#), [10](#)