

# Package ‘ROC’

April 12, 2022

**Version** 1.70.0

**Title** utilities for ROC, with microarray focus

**Author** Vince Carey <stvjc@channing.harvard.edu>, Henning Redestig for  
C++ language enhancements

**Maintainer** Vince Carey <stvjc@channing.harvard.edu>

**Description** Provide utilities for ROC, with microarray focus.

**Depends** R (>= 1.9.0), utils, methods

**Imports** knitr

**Suggests** rmarkdown, Biobase

**License** Artistic-2.0

**URL** <http://www.bioconductor.org>

**LazyLoad** Yes

**biocViews** DifferentialExpression

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/ROC>

**git\_branch** RELEASE\_3\_14

**git\_last\_commit** 44fd639

**git\_last\_commit\_date** 2021-10-26

**Date/Publication** 2022-04-12

## R topics documented:

AUC . . . . .	2
plot-methods . . . . .	3
rocc-class . . . . .	4
rocdemo.sca . . . . .	5
trapezint . . . . .	6
<b>Index</b>	<b>8</b>

---

AUC

*functionals of ROC curve*

---

### Description

various functionals of ROC (Receiver Operating Characteristic) curves

### Usage

```
AUC(rocobj)
AUCi(rocobj)
pAUC(rocobj, t0)
pAUCi(rocobj, t0)
```

### Arguments

rocobj	element of class rocc
t0	FPR point at which TPR is evaluated or limit in (0,1) to integrate to

### Details

AUC, pAUC, AUCi and pAUCi compute the Area Under the Curve.

AUC and pAUC employ the trapezoidal rule. AUCi and pAUCi use integrate().

AUC and AUCi compute the area under the curve from 0 to 1 on the x-axis (i.e., the 1 - specificity axis).

pAUC and pAUCi compute the area under the curve from 0 to argument t0 on the x-axis (i.e., the 1 - specificity axis).

Elements of class rocc can be created by rocdemo.sca() or other constructors you might make using the code of rocdemo.sca() as a template.

### Author(s)

Vince Carey (stvjc@channing.harvard.edu)

### References

Rosner, B., 2000, *Fundamentals of Biostatistics, 5th Ed.*, pp. 63–65

Duda, R. O., Hart, P. E., Stork, D. G., 2001 *Pattern Classification, 2nd Ed.*, p. 49

### See Also

rocdemo.sca

**Examples**

```

set.seed(123)
R1 <- rocdemo.sca( rbinom(40,1,.3), rnorm(40), dxrule.sca,
  caseLabel="new case", markerLabel="demo Marker" )
print(AUC(R1))
print(pAUC(R1,.3))
print(pAUCi(R1,.3))
print(ROC(R1,.3))

```

---

plot-methods

*plot method for ROC curves*


---

**Description**

plot method for ROC curves

**Methods**

**x = rocc** plots an ROC curve object, with additional parameters available:

**show.thresh (logical):** should marker threshold values be plotted?

**jit (logical):** should plotted points be jittered?

**add (logical):** increment to current plot?

**line (logical):** plot points or lines?

**threshCex (numeric):** if showing threshold values, set character expansion in text call to this value

**threshYsh (numeric):** if showing threshold values, add this quantity to y coordinate of curve to plot the threshold value (should be negative for printing below point)

**threshDig (numeric):** if showing threshold values, use this as the digits parameter to round to display the threshold

... extra parameters passed to base plot, lines or points as needed

**Examples**

```

set.seed(123)
R1 <- rocdemo.sca( rbinom(40,1,.3), rnorm(40), dxrule.sca,
  caseLabel="new case", markerLabel="demo Marker" )
plot(R1, line=TRUE, show.thresh=TRUE, lwd=2, threshDig=2)
R2 <- rocdemo.sca( rbinom(40,1,.3), rnorm(40), dxrule.sca,
  caseLabel="new case", markerLabel="demo Marker" )
plot(R2, line=TRUE, add=TRUE, col="green", lwd=2 )
R3 <- rocdemo.sca( rbinom(40,1,.4), rnorm(40), dxrule.sca,
  caseLabel="new case", markerLabel="demo Marker" )
points(R3, col="red", pch=19)

```

---

rocc-class	<i>Class rocc, ROC curve representation</i>
------------	---

---

### Description

object representing ROC curve, typically created using rocdemo.sca

### Creating Objects

```
new('rocc',  
    sens = ..., # Object of class numeric  
    spec = ..., # Object of class numeric  
    rule = ..., # Object of class function  
    cuts = ..., # Object of class numeric  
    markerLabel = ..., # Object of class character  
    caseLabel = ..., # Object of class character  
)
```

### Slots

sens: Object of class "numeric" sensitivity values  
spec: Object of class "numeric" specificity values  
rule: Object of class "function" rule to classify objects  
cuts: Object of class "numeric" thresholds defining curve  
markerLabel: Object of class "character" name of measured marker  
caseLabel: Object of class "character" name of condition

### Methods

**plot** (rocc, missing): a plotting function with some additional parameters

### Examples

```
set.seed(123)  
R1 <- rocdemo.sca( rbinom(40,1,.3), rnorm(40), dxrule.sca,  
    caseLabel="new case", markerLabel="demo Marker" )  
plot( R1, show.thresh=TRUE )
```

---

rocdemo.sca                      *function to build objects of class 'rocc'*

---

### Description

rocdemo.sca – demonstrate 'rocc' class construction using a scalar marker and simple functional rule

### Usage

```
rocdemo.sca(truth, data, rule=NULL,
            cutpts=NA,
            markerLabel="unnamed marker", caseLabel="unnamed diagnosis",
            quiet=TRUE)
```

### Arguments

truth	true classification of objects. Must take values 0 or 1.
data	quantitative markers used to classify
rule	rule: a function with arguments (x, thresh) returning 0 or 1. If no rule is provided or the standard rule <code>dxrule.sca</code> is passed, a faster C-based implementation is used to compute sensitivity and specificity.
cutpts	values of thresholds; no NA allowed, or they will be recomputed using smallest gap between data points with distinct values
markerLabel	textual label describing marker
caseLabel	textual label describing classification
quiet	defaults to TRUE, suppressing message about discovery of NA in cutpts

### Details

`dxrule.sca` is function (x, thresh) `ifelse(x > thresh, 1, 0)`

The default value of argument `cutpts` is a point less than `min(data)`, points separating the unique values of data and a point greater than `max(data)`.

### Value

an object of S4 class `rocc`

### Author(s)

Vince Carey ([stvjc@channing.harvard.edu](mailto:stvjc@channing.harvard.edu))

### See Also

AUC

**Examples**

```

set.seed(123)
R1 <- rocdemo.sca( rbinom(40,1,.3), rnorm(40), caseLabel="new case", markerLabel="demo Marker" )
plot(R1, line=TRUE, show.thresh=TRUE)

truth <- c(0, 1, 0, 1, 1, 0, 1, 1)
data <- c(2, 3, 4, 4, 5, 6, 7, 8)
R2 <- rocdemo.sca(truth, data, dxrule.sca)
plot(R2, line=TRUE, show.thresh=TRUE)
R3 <- rocdemo.sca(truth, data, function(x, thresh) 1 - dxrule.sca(x, thresh))
if (AUC(R2) + AUC(R3) != 1) stop('Sum of AUCs should be 1.')
#
# more involved
#
set.seed(1234)
x = runif(1000)
w = runif(1000)
z = rbinom(1000, 1, plogis(-2.7+6.2*x + .3*w))
m1 = glm(z~x, fam=binomial)
demorule.glm.clo = function(model) function(w,thresh)
  ifelse(predict(model, newdata=list(x=w), type="response")>thresh, 1, 0)
demorule.glm = demorule.glm.clo(m1)
R4 = rocdemo.sca(z, x, demorule.glm )
plot(R4)

```

---

trapezint

*trapezoidal rule for AUC*


---

**Description**

trapezoidal rule for AUC

**Usage**

```
trapezint(x, y, a, b)
```

**Arguments**

x	x - abscissae
y	y - ordinates
a	a - lower limit of integration
b	b - upper limit of integration

**Details**

uses approx

*trapezint*

7

**Value**

estimated AUC

**Examples**

```
x <- sort(runif(30))
y <- sin(x)
print(trapezint(x,y,0,1))
```

# Index

## \* **methods**

plot-methods, 3

rocc-class, 4

## \* **models**

AUC, 2

rocdemo.sca, 5

trapezint, 6

.initClasses (rocc-class), 4

AUC, 2

AUCi (AUC), 2

dxrule.sca (rocdemo.sca), 5

lines, rocc-method (plot-methods), 3

pAUC (AUC), 2

pAUCi (AUC), 2

plot, rocc-method (plot-methods), 3

plot-methods, 3

points, rocc-method (plot-methods), 3

ROC (AUC), 2

rocc-class, 4

rocdemo.sca, 5

trapezint, 6