Package 'Comp2ROC'

July 21, 2025

Title Compare Two ROC Curves that Intersect

Version 1.1.4

Date 2016-05-18

Author Ana C. Braga with contributions from Hugo Frade, Sara Carvalho and Andre M. Santiago

Maintainer Ana C. Braga `<acb@dps.uminho.pt>`

Description Comparison of two ROC curves through the methodology proposed by Ana C. Braga.

License GPL-2

Depends R (>= 2.15.1), ROCR, boot

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-01 01:17:58

Contents

`Comp2ROC-package` *Comparation of Two ROC Curves that Intersect*

Description

Comaparation of ROC Curves using the methodology devoloped by Braga.

Details

Package:	Comp2ROC
Type:	Package
Version:	1.1.2
Date:	2016-05-18
License:	GPL-2

Author(s)

Ana C. Braga, with contributions from Hugo Frade, Sara Carvalho and Andre M Santiago.

Maintainer: Ana C. Braga <acb@dps.uminho.pt>; Andre M. Santiago <andreportugalsantiago@gmail.com>;

References

BRAGA, A. C. AND COSTA, L. AND OLIVEIRA, P. 2011. An alternative method for global and partial comparasion of two diagnostic system based on ROC curves In Journal of Statistical Computation and Simulation.

Examples

```
# This is a simple example on how to use the package with the given dataset ZHANG (paired samples):
nameE = "Zhang"
modality1DataColumn = "modality1"
modality2DataColumn = "modality2"
data(zhang)
results = roc.curves.boot(zhang, 10, 0.05, name=nameE,
                          mod1=modality1DataColumn, mod2=modality2DataColumn)
rocboot.summary(results, "modality1", "modality2")

# This is another simple example on how to use the package with the given
# dataset CAS2015 (unpaired samples):
nameE = "CAS2015"
modality1DataColumn = "CRIBM"
modality2DataColumn = "CRIBF"
paired = FALSE
```

```
data(cas2015)
results = roc.curves.boot(cas2015, 1000, 0.05, name=nameE,
                          mod1=modality1DataColumn, mod2=modality2DataColumn, paired)
rocboot.summary(results, modality1DataColumn, modality2DataColumn)
```

`areatriangles` *Triangle Areas*

Description

This function allows to calculate the triangles area formed with two points that was next to each other and the reference point. It also allows to calculate the total area based on the previous triangles.

Usage

```
areatriangles(line.slope, line.dist1)
```

Arguments

`line.slope`	Vector with all sampling lines slope
`line.dist1`	Vector with the ROC Curves and sampling lines intersection points, the distance between this points and the reference point

Value

This function return a list with:

`auctri`	Total area
`areatri`	Vector with all triangles areas

See Also

`lineslope` `linedistance` `curvesegslope` `curvesegsloperef`

`cas2015` *CAS2015 Dataset*

Description

This dataset was created by Braga, A. C. and allows the comparison of two independent samples.

Usage

```
data(cas2015)
```

Format

A data frame with a total of 800 observations on the following 2 variables and respectives status.

`mod1` CRIBM

`status1` Result1

`mod2` CRIBF

`status2` Result2

Details

The dataset contains the values of the indicator (CRIB) for 2 different groups (sex: M/F) and respective results, from 0 (alive) to 1 (deceased). These samples are unpaired, therefore presenting different statuses for each one.

Source

COELHO, S. AND BRAGA, A. C.: Performance Evaluation of Two Software for Analysis Through ROC Curves: Comp2ROC vs SPSS. Computational Science and Its Applications – ICCSA 2015; p. 144-156; Springer International Publishing., ISBN: 978-3-319-21406-1.

`comp.roc.curves` *Calculate distribution*

Description

This funtion calculates by bootstrapping the real distribution for the entire length set.

Usage

```
comp.roc.curves(result, ci.flag = FALSE, graph.flag = FALSE, nome)
```

Arguments

`result`	List of statistical measures obtaind throught `rocsampling`
`ci.flag`	Flag that indicates if the user wants to calculate the confidance intervals
`graph.flag`	Flag that indicates if the user wants to draw the graph
`nome`	Name to put on the graph

Details

In this function `ci.flag` and `graph.flag` are set `FALSE` by defaut

Value

`boot`	statistics test
`p-value`	p-value for one-sided
`p-value2`	p-value for two-sided
`ci`	confidance interval

See Also

`rocsampling`

`comp.roc.delong` *Calculate areas and stats*

Description

This function allows to calculate the areas under the curve for each curve and some statistical measures.

Usage

```
comp.roc.delong(sim1.ind, sim1.sta, sim2.ind, sim2.sta, related = TRUE)
```

Arguments

`sim1.ind`	Vector with the data for Curve 1
`sim1.sta`	Vector with the status for Curve 1
`sim2.ind`	Vector with the data for Curve 2
`sim2.sta`	Vector with the status for Curve 2
`related`	Boolean parameter that represents if the two modalities are related or not

Details

This function calculates the Wilcoxon Mann Whitney matrix for each modality, areas, standard deviations, variances and global correlations.

Value

This function returns a list with:

`Z`	Hanley Z calculation
`pvalue`	p-value for this Z
`AUC`	Area under curve for each modality
`SE`	Standard error
`S`	Variance for each modality
`R`	Correlation Coeficient

Examples

```
data(zhang)
modality1DataColumn = "modality1"
modality2DataColumn = "modality2"
data = read.manually.introduced(zhang, modality1DataColumn, TRUE,
                                modality2DataColumn, TRUE, "status", TRUE)
sim1.ind = unlist(data[1])
sim2.ind = unlist(data[2])
sim1.sta = unlist(data[3])
sim2.sta = unlist(data[4])
comp.roc.delong(sim1.ind, sim1.sta, sim2.ind, sim2.sta)
```

`curvesegslope` *Segment Slopes*

Description

This function allows to calculate the ROC curve segments slope through the points that are given by parameter.

Usage

```
curvesegslope(curve.fpr, curve.tpr)
```

Arguments

`curve.fpr`	False positive rate vector with all points of the given Curve
`curve.tpr`	True positive rate vector with all points of the given Curve

Value

This function returns a vector with all segments slopes

`curvesegsloperef` *Segment Slopes to Reference Point*

Description

This function allows to calculate the segments slope that connect the ROC curve segments with the reference point (1,0).

Usage

```
curvesegsloperef(curve.fpr, curve.tpr, ref.point)
```

Arguments

`curve.fpr`	False positive rate vector with all points of the given Curve
`curve.tpr`	True positive rate vector with all points of the given Curve
`ref.point`	Reference point where we start drawing the sample lines

Value

This function returns a vector with all segments slopes that connect the ROC curve points to the reference point.

`diffareatriangles` *Difference Between Area Triangles*

Description

This function allows to calculate the difference between triangles areas formed by the same sampling lines in two different ROC curves. It also allows to calculate the difference between total areas.

Usage

```
diffareatriangles(area.triangle1, area.triangle2)
```

Arguments

`area.triangle1`	Vector with all triangles areas of the Curve 1
`area.triangle2`	Vector with all triangles areas of the Curve 2

Value

This function returns a list with:

`diffareas`	Difference between each triangle area
`diffauc`	Difference between total areas

See Also

`areatriangles`

`linedistance` *Intersection Points*

Description

This function allows to calculate the intersection points between the ROC curve and the sampling lines. Also calculates the distance between this points and the reference point.

Usage

```
linedistance(curve.fpr, curve.tpr, curve.segslope, curve.slope, line.slope, ref.point)
```

Arguments

`curve.fpr`	False positive rate vector with all points of the given Curve
`curve.tpr`	True positive rate vector with all points of the given Curve
`curve.segslope`	Vector with all segments slope of the ROC curves
`curve.slope`	Vector with all the slope of all segments that connect the ROC curve with the reference point
`line.slope`	Vector with the slope of all sampling lines
`ref.point`	Reference point where we start drawing the sampling lines

Value

This function returns a list with:

`dist`	Vector with distances between the intersection points and the reference points
`x`	Vector with all x coordinates of intersection points
`y`	Vector with all y coordinates of intersection points

See Also

`lineslope` `curvesegslope` `curvesegsloperef`

`lineslope` *Sampling Lines Slope*

Description

This function allows to calculate the sample lines slope that were drawn beginning at the reference point.

Usage

```
lineslope(K)
```

Arguments

`K`	Number of sampling lines that we want to create

Value

This function returns a vector with all slopes of the sampling lines that we create

Examples

```
K = 100
lineslope(K)
```

`read.file` *Read data from file*

Description

This function allows to read data from a file.

Usage

```
read.file(name.file.csv, header.status = TRUE, separator = ";", decimal = ",", modality1,
testdirection1, modality2, testdirection2, status1, related = TRUE, status2 = NULL)
```

Arguments

`name.file.csv`	Name of the file with data. The file must be in `csv` or `txt` format
`header.status`	Indicates if the file has a header row
`separator`	Indicates what is the column separator
`decimal`	Indicates what is the decimal separator
`modality1`	Name of the column of dataframe that represents the first modality

`testdirection1`	Indicates the direction of the test for modality 1. If `TRUE` means that larger test results represent more positive test
`modality2`	Name of the column of dataframe that represents the second modality
`testdirection2`	Indicates the direction of the test for modality 2. If `TRUE` means that larger test results represent more positive test
`status1`	Name of the column of dataframe that represents the Status 1
`related`	Boolean parameter that represents if the two modalities are related or not
`status2`	Name of the column of dataframe that represents the Status 2

Details

The default column separator is `";"`. And the default decimal separator is `"."`. `header.status` has also a default value that is `TRUE`. By default, the related parameter is set to `TRUE`. In this case the `status2` is not necessary (by default set to `(NULL)`, because in related modalities the status is the same. Otherwise, if related is set to `FALSE`, its necessary to indicate the name of `status2` column. In the data must be listed first all values of the distribution of negative cases (0), followed by the positive ones (1).

Value

This functions returns a list with the following data:

`sim1.ind`	Vector with the data for Curve 1
`sim2.ind`	Vector with the data for Curve 2
`sim1.sta`	Vector with the status for Curve 1
`sim2.sta`	Vector with the status for Curve 2

See Also

`read.manually.introduced`

Examples

```
# This is a simple example how to read a file:

data.filename = "zhang.csv"
modality1DataColumn = "modality1"
modality2DataColumn = "modality2"
modality2StatusHeader = "status"  # if different from modality1's header
                                  # (a.k.a they are independent)
zhang = read.file(data.filename, TRUE, ";", ".", modality1, TRUE, modality2, TRUE, "status")
```

`read.manually.introduced` *Read data manually introduced*

Description

This function allows to read the testing data.

Usage

```
read.manually.introduced(dat, modality1, testdirection1, modality2,
testdirection2, status1, related = TRUE, status2 = NULL)
```

Arguments

`dat`	Dataframe of data to anlyse
`modality1`	Name of the column of dataframe that represents the first modality
`testdirection1`	Indicates the direction of the test for modality 1. If `TRUE` means that larger test results represent more positive test
`modality2`	Name of the column of dataframe that represents the second modality
`testdirection2`	Indicates the direction of the test for modality 2. If `TRUE` means that larger test results represent more positive test
`status1`	Name of the column of dataframe that represents the Status 1
`related`	Boolean parameter that represents if the two modalities are related or not
`status2`	Name of the column of dataframe that represents the Status 2

Details

By default, the related parameter is set to `TRUE`. In this case the `status2` is not necessary (by default set to `(NULL)`, because in related modalities the status is the same. Otherwise, if related is set to `FALSE`, its necessary to indicate the name of `status2` column. In the data must be listed first all values of the distribution of negative cases (0), followed by the positive ones (1).

Value

This functions returns a list with the following data:

`sim1.ind`	Vector with the data for Curve 1
`sim2.ind`	Vector with the data for Curve 2
`sim1.sta`	Vector with the status for Curve 1
`sim2.sta`	Vector with the status for Curve 2

Examples

```
data(zhang)
moda1 = "modality1"
moda2 = "modality2"
data = read.manually.introduced(zhang, moda1, TRUE, moda2, TRUE, "status", TRUE)
```

`roc.curves.boot` *Compare curves*

Description

This is the function which control the whole package.This uses all functions except the reading ones and `rocboot.summary` and `save.file.summary`.

Usage

```
roc.curves.boot(data, nb = 1000, alfa = 0.05, name, mod1, mod2, paired)
```

Arguments

`data`	Data obtained throught `read.file` or `read.manually.introduced`
`nb`	Number of permutations
`alfa`	Confidance level for parametric methods
`name`	Name too show in graphs
`mod1`	Name of Modality 1
`mod2`	Name of Modality 2
`paired`	Boolean parameter that represents if the two modalities are related or not

Value

This function returns a list with:

`Area1`	Area of Curve 1
`SE1`	Standard error of Curve 1
`Area2`	Area of Curve 2
`SE2`	Standard error of Curve 2
`CorrCoef`	Correlation Coeficient
`diff`	Difference Between Areas (TS)
`zstats`	Z Statistic
`pvalue1`	p-value of Z Statistics
`TrapArea1`	Area of curve 1 using the Trapezoidal rule
`TrapArea2`	Area of curve 2 using the Trapezoidal rule

`bootpvalue`	p-value of bootstrapping
`nCross`	Number of Crossings
`ICLB1`	Confidance Interval: Lower Bound for Curve 1
`ICUB1`	Confidance Interval: Upper Bound for Curve 1
`ICLB2`	Confidance Interval: Lower Bound for Curve 2
`ICUB2`	Confidance Interval: Upper Bound for Curve 2
`ICLBDiff`	Confidance Interval: Lower Bound for Difference between areas
`ICUBDiff`	Confidance Interval: Upper Bound for Difference between areas

Examples

```
data(zhang)
nameE = "new_Zhang"
modality1DataColumn = "modality1"
modality2DataColumn = "modality2"
data = read.manually.introduced(zhang, moda1, TRUE, moda2, TRUE, "status", TRUE)
results = roc.curves.boot(zhang, 1000, 0.05, name=nameE,
                          mod1=modality1DataColumn, mod2=modality2DataColumn)
```

`roc.curves.plot` *Plot ROC curves*

Description

This function allows to plot the two roc curves in comparasion.

Usage

```
roc.curves.plot(sim1.curve, sim2.curve, mod1, mod2)
```

Arguments

`sim1.curve`	Curve 1 created using the function `performance`.
`sim2.curve`	Curve 2 created using the function `performance`.
`mod1`	Name of Modality 1
`mod2`	Name of Modality 2

See Also

`read.file` `read.manually.introduced`

Examples

```
data(zhang)
moda1 = "modality1"
moda2 = "modality2"
data = read.manually.introduced(zhang, moda1, TRUE, moda2, TRUE, "status", TRUE)

sim1.ind = unlist(data[1])
sim2.ind = unlist(data[2])
sim1.sta = unlist(data[3])
sim2.sta = unlist(data[4])

sim1.pred = prediction(sim1.ind, sim1.sta)
sim2.pred = prediction(sim2.ind, sim2.sta)

sim1.curve = performance(sim1.pred, "tpr", "fpr")
sim2.curve = performance(sim2.pred, "tpr", "fpr")

roc.curves.plot(sim1.curve, sim2.curve, mod1=moda1, mod2=moda2)
```

`rocboot.summary` *Summary of Comparation*

Description

This function allows to see the information obtained throught function `roc.curve.boot`.

Usage

```
rocboot.summary(result, mod1, mod2)
```

Arguments

`result`	List of statistical measures obtaind throught `roc.curves.boot`
`mod1`	Name of the column of dataframe that represents the first modality
`mod2`	Name of the column of dataframe that represents the second modality

See Also

`save.file.summary`

Examples

```
data(zhang)
moda1 = "modality1"
moda2 = "modality2"
nameE = "new_Zhang"
data = read.manually.introduced(zhang, moda1, TRUE, moda2, TRUE, "status", TRUE)
```

```
results = roc.curves.boot(data, name=nameE, mod1=moda1, mod2=moda2)
rocboot.summary(results, moda1, moda2)
```

`rocsampling` *ROC Sampling*

Description

This function allows to calculate some statistical measures like extension and location.

Usage

```
rocsampling(curve1.fpr, curve1.tpr, curve2.fpr, curve2.tpr, K = 100)
```

Arguments

`curve1.fpr`	False positive rate vector with all points of the Curve 1
`curve1.tpr`	True positive rate vector with all points of the Curve 1
`curve2.fpr`	False positive rate vector with all points of the Curve 2
`curve2.tpr`	True positive rate vector with all points of the Curve 2
`K`	Number of sampling lines

Details

This function uses functions like `areatriangles`, `curvesegslope`, `curvesegsloperef`, `diffareatriangles`, `linedistance` and `lineslope to calculate` that measures. By default the number of sampling lines is 100, beacause it was proved by Braga that it was the optimal number.

Value

This funcion returns a list with the following components:

`AUC1`	Total Area of Curve 1 (using triangles)
`AUC2`	Total Area of Curve 2 (using triangles)
`propc1`	Proportion of Curve1
`propc2`	Proportion of Curve2
`propties`	Proportion of ties
`locc1`	Location of Curve 1
`locc2`	Location of Curve 2
`locties`	Location of Ties
`K`	Number of sampling lines
`lineslope`	Slopes of sampling lines
`diffareas`	Difference of area of triangles
`dist1`	Distance of the intersection points of Curve 1 to reference point
`dist2`	Distance of the intersection points of Curve 2 to reference point

See Also

`areatriangles` `curvesegslope` `curvesegsloperef` `diffareatriangles` `linedistance` `lineslope`

`rocsampling.summary` *Summary of ROC Sampling*

Description

This function allows to see with a simple interface the results obtained in `rocsampling`.

Usage

```
rocsampling.summary(result, mod1, mod2)
```

Arguments

`result`	List with results obtained throught the use of `rocsampling`
`mod1`	Name of the column of dataframe that represents the first modality
`mod2`	Name of the column of dataframe that represents the second modality

See Also

`rocsampling`

`save.file.summary` *Save File*

Description

This functions allow to save the information on a file.

Usage

```
save.file.summary(result, name, app = TRUE, mod1, mod2)
```

Arguments

`result`	List of statistical measures obtaind throught roc.curves.boot
`name`	File name
`app`	Indicates if the user wants to append information on the same file
`mod1`	Name of the column of dataframe that represents the first modality
`mod2`	Name of the column of dataframe that represents the second modality

Details

The user don't need to fill the `app` parameter, because by default it was set to `TRUE`. This parameter allow the user to choose if he wants the results of differents performances in the same file, or each time that he starts a new performance the file will be new.

Value

This functions saves on the file with name `name` the performance parameters of the test.

Examples

```
# If the user wants to append the results
save.file.summary(results, nameE, mod1=moda1, mod2=moda2)

# If the user does not want to append the results
save.file.summary(results, nameE, app=FALSE, moda1, moda2)
```

`zhang` *Zhang Dataset*

Description

This dataset was created by Zhang and we use it as example on our package

Usage

`data(zhang)`

Format

A data frame with 2410 observations on the following 3 variables.

`mod1` modality 1

`status` status

`mod2` modality 2

Details

This modalities are related to each other, so they have the same status

Source

ZHANG, D. AND ZHOU, X.AND FREEMAN, D. AND FREEMAN, J. 2002. A nonparametric method for the comparison of partial areas under ROC curves and its application to large health care data sets In Stat. Med., Vol. 21 N. 5 701-715.

Index