Package 'bimixt'

August 25, 2015

Type Package

Title Estimates Mixture Models for Case-Control Data

Version 1.0

Date 2015-08-24

Author Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

Maintainer Michelle Winerip <mwinerip@asu.edu>

Description Estimates non-Gaussian mixture models of case-control data. The four types of models supported are binormal, two component constrained, two component unconstrained, and four component. The most general model is the four component model, under which both cases and controls are distributed according to a mixture of two unimodal distributions. In the four component model, the two component distributions of the control mixture may be distinct from the two components of the case mixture distribution. In the two component unconstrained model, the components of the control and case mixtures are the same; however the mixture probabilities may differ for cases and controls. In the two component constrained model, all controls are distributed according to one of the two components while cases follow a mixture distribution of the two components. In the binormal model, cases and controls are distributed according to distinct unimodal distributions. These models assume that Box-Cox transformed case and control data with a common lambda parameter are distributed according to Gaussian mixture distributions. Model parameters are estimated using the expectation-maximization (EM) algorithm. Likelihood ratio test comparison of nested models can be performed using the lr.test function. AUC and PAUC values can be computed for the model-based and empirical ROC curves using the auc and pauc functions, respectively. The model-based and empirical ROC curves can be graphed using the roc.plot function. Finally, the model-based density estimates can be visualized by plotting a model object created with the bimixt.model function.

Depends pROC

License GPL (>= 3)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-08-25 00:54:26

R topics documented:

`bimixt-package` *bimixt*

Description

The bimixt package contains tools for estimating non-Gaussian mixture models of case-control data. The four types of models supported are binormal, two component constrained, two component unconstrained, and four component. The most general model is the four component model, under which both cases and controls are distributed according to a mixture of two unimodal distributions. In the four component model, the two component distributions of the control mixture may be distinct from the two components of the case mixture distribution. In the two component unconstrained model, the components of the control and case mixtures are the same; however the mixture probabilities may differ for cases and controls. In the two component constrained model, all controls are distributed according to one of the two components while cases follow a mixture distribution of the two components. In the binormal model, cases and controls are distributed according to distinct unimodal distributions. These models assume that Box-Cox transformed case

and control data with a common lambda parameter are distributed according to Gaussian mixture distributions. Model parameters are estimated using the expectation-maximization (EM) algorithm. Likelihood ratio test comparison of nested models can be performed using the lr.test function. AUC and PAUC values can be computed for the model-based and empirical ROC curves using the auc and pauc functions, respectively. The model-based and empirical ROC curves can be graphed using the roc.plot function. Finally, the model-based density estimates can be visualized by plotting a model object created with the bimixt.model function.

Details

Package:	bimixt
Type:	Package
Version:	1.0
Date:	2015-06-21
License:	GPL (>=3)
LazyLoad:	yes

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer Maintainer: Michelle Winerip <mwinerip@asu.edu>

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 7, 77. DOI: 10.1186/1471-2105-12-77.

See Also

pROC

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model1=bimixt.model(case=case,control=control, type="4c")
model2=bimixt.model(case=case, control=control, type="2cu")
model3=bimixt.model(case=case, control=control, type="2cc")
model4=bimixt.model(case=case, control=control, type="binorm")
par(mfrow=c(2,2))
plot(model1)
```

```
plot(model2)
plot(model3)
plot(model4)
```

`bc.binorm` *bc.binorm*

Description

Implementation of binormal model. The binormal model estimates a single unimodal component for the cases and a single unimodal component for the controls.

Usage

```
bc.binorm(case, control, lambda.bounds = c(-5, 5))
```

Arguments

`case`	a numeric vector of case values
`control`	a numeric vector of control values
`lambda.bounds`	numeric vector of bounds: `c(upper bound, lower bound)`. Specifies the range for `optim` to search for the optimization of lambda. Default: `c(-5,5)`.

Value

`lambda`	Box-Cox transformation parameter
`type`	model type ("binorm")
`mu.cases`	mean of the Box-Cox transformed case component
`sig.cases`	standard deviation of the Box-Cox transformed case component
`pi.cases`	proportion of cases in each case component (always equal to 1 for binorm since all cases are forced into one component)
`mu.controls`	mean value of the Box-Cox transformed control component
`sig.controls`	standard deviation of the Box-Cox transformed control component
`pi.controls`	proportion of controls in each control component (always equal to 1 for binorm since all controls are forced into one component)
`max.loglike`	the maximum log likelihood value for the model
`case`	original case values
`control`	original control values
`mu.cases.unt`	an estimate of the untransformed mean of the case component. Based on Monte Carlo simulations. Values will differ by computer seed.
`sig.cases.unt`	an estimate of the untransformed standard deviation of the case component. Based on Monte Carlo simulations. Values will differ by computer seed.

`mu.controls.unt`
: an estimate of the untransformed mean of the control component. Based on Monte Carlo simulations. Values will differ by computer seed.

`sig.controls.unt`
: an estimate of the untransformed standard deviation of the control component. Based on Monte Carlo simulations. Values will differ by computer seed.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

`bc.twocomp` `bc.fourcomp` `em.twocomp.m1` `em.twocomp.m2` `em.twocomp.m3`

`bc.fourcomp` *bc.fourcomp*

Description

Implementation of four component model. The four component model estimates an upper and lower component for the cases and an upper and lower component for the controls.

Usage

```
bc.fourcomp(x.cases, x.controls, lambda.bounds = c(-5, 5),

start.vals.cases=NULL, start.vals.controls=NULL)
```

Arguments

`x.cases`
: a numeric vector of case values

`x.controls`
: a numeric vector of control values

`lambda.bounds`
: numeric vector of bounds: `c(upper bound, lower bound)`. Specifies the range for `optim` to search for the optimization of lambda. Default: `c(-5,5)`.

`start.vals.cases`
: starting values for the EM algorithm for the cases. If `NA`, the starting values are estimated from the data.

`start.vals.controls`
: starting values for the EM algorithm for the controls. If `NA`, the starting values are estimated from the data.

Value

`lambda`	Box-Cox transformation parameter
`type`	model type ("4c")
`mu.cases`	means of the Box-Cox transformed case components
`sig.cases`	standard deviations of the Box-Cox transformed case components
`pi.cases`	proportion of cases in each case component
`max.loglike.cases`	the maximum log likelihood value for the fit of the cases
`mu.controls`	means of the Box-Cox transformed control components
`sig.controls`	standard deviations of the Box-Cox transformed control components
`pi.controls`	proportion of controls in each control component
`max.loglike.controls`	the maximum log likelihood value for the fit of the controls
`max.loglike`	the maximum log likelihood value for the model
`mu.cases.unt`	an estimate of the untransformed means of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
`sig.cases.unt`	an estimate of the untransformed standard deviations of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
`mu.controls.unt`	an estimate of the untransformed means of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
`sig.controls.unt`	an estimate of the untransformed standard deviations of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
`case`	original case values
`control`	original control values
`time`	running time for the model fit

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

`bc.binorm` `bc.twocomp` `em.twocomp.m1` `em.twocomp.m2` `em.twocomp.m3`

bc.twocomp *bc.twocomp*

Description

Implementation of two component models. In the two component unconstrained model, the components of the control and case mixtures are the same; however the mixture probabilities may differ for cases and controls. In the two component constrained model, all controls are distributed according to one of the two components while cases follow a mixture distribution of the two components.

Usage

```
bc.twocomp(x.cases, x.controls, constrained = T, lambda.bounds = c(-5, 5),

control.comp = 1, start.vals=NULL)
```

Arguments

x.cases	a numeric vector of case values
x.controls	a numeric vector of control values
constrained	Boolean indicating whether the two component constrained model should be used (default T) or the two component unconstrained model should be used (F)
lambda.bounds	numeric vector of bounds: c(upper bound, lower bound). Specifies the range for optim to search for the optimization of lambda. Default: c(-5,5).
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

Value

lambda	Box-Cox transformation parameter
type	model type ("2cc" or "2cu")
mu.cases	means of the Box-Cox transformed case components
sig.cases	standard deviations of the Box-Cox transformed case components
pi.cases	proportion of cases in each case component
mu.controls	means of the Box-Cox transformed control components
sig.controls	standard deviations of the Box-Cox transformed control components
pi.controls	proportion of controls in each control component (always equal to 1 for 2cc since all controls are forced into one component)
max.loglike	the maximum log likelihood value for the model
mu.cases.unt	an estimate of the untransformed means of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.

sig.cases.unt	an estimate of the untransformed standard deviations of the case components. Based on Monte Carlo simulations. Values will differ by computer seed.
mu.controls.unt	an estimate of the untransformed means of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
sig.controls.unt	an estimate of the untransformed standard deviations of the control components. Based on Monte Carlo simulations. Values will differ by computer seed.
case	original case values
control	original control values
time	running time for the model fit

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

bc.binorm bc.fourcomp em.twocomp.m1 em.twocomp.m2 em.twocomp.m3

---

bimixt.model *bimixt.model*

Description

Estimates mixture model components based on model type.

Usage

```
bimixt.model(case, control, type = "binorm", start.vals=NULL)
```

Arguments

case	a numeric vector of case values. NA's will be omitted.
control	a numeric vector of control values. NA's will be omitted.
type	a string specifying the mixture model to be used to fit the data. Valid types are "binorm", "2cc", "2cu", or "4c". These correspond to binormal, two component constrained, two component unconstrained, and four component models respectively. Defaults to "binorm".
start.vals	an (optional) list of starting values for the EM algorithm used in the "2cc", "2cu", and "4c" models. If not specified by the user, starting values are estimated from the data using kmeans clustering. The format of the lists are described in the details section.

Details

Starting values for the EM algorithm can be provided by the user. The starting values must be given as lists. Each element in the list is a named numeric vector of length 2 containing starting estimates for the model parameters. Names must match the names given below exactly (See examples section for "4c" model example).

For "2cc" `start.vals` is a list of 3 named vectors:

mu	Starting estimates for component means
sig	Starting estimates for component standard deviations
pi	Starting estimates for component proportions. Must sum to 1.

For "2cu", `start.vals` is a list of length 4:

mu	Starting estimates for component means.
sig	Starting estimates for component standard deviations.
pi.cs	Starting estimates for case component proportions. Must sum to 1.
pi.ctrl	Starting estimates for control component proportions. Must sum to 1.

For "4c", `start.vals` is a list of length 6:

mu.cs	Starting estimates for case component means.
mu.ctrl	Starting estimates for control component means.
sig.cs	Starting estimates for case component standard deviations.
sig.ctrl	Starting estimates for control component standard deviations.
pi.cs	Starting estimates for component proportions for cases. Must sum to 1.
pi.ctrl	Starting estimates for component proportions for controls. Must sum to 1.

Value

Returns an object of type model with parameters specified by bc.binorm, bc.twocomp, or bc.fourcomp.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

`plot.model` `print.model` `summary.model`

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model1=bimixt.model(case=case, control=control, type="4c", start.vals=list(mu.cs=c(10,15),
```

```
mu.ctrl=c(10,15),sig.cs=c(1.2,1),sig.ctrl=c(1.2,1),pi.cs=c(.7,.3),pi.ctrl=c(.95,.05)))
model2=bimixt.model(case=case, control=control, type="2cu")
model3=bimixt.model(case=case, control=control, type="2cc")
model4=bimixt.model(case=case, control=control, type="binorm")
```

---

boxcox *boxcox*

Description

Implementation of the Box-Cox normalization transformation method. Called internally in `bc.twocomp` and `bc.fourcomp`.

Usage

```
boxcox(x, lambda)
```

Arguments

`x`	a numeric vector or scalar
`lambda`	Box-Cox transformation variable

Value

A vector or scalar of the transformed values of x.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

See Also

`boxcox.inv` `boxcox.deriv` `boxcox.inv.density`

boxcox.deriv *boxcox.deriv*

Description

Derivative of the Box-Cox transformation function.

Usage

```
boxcox.deriv(x, lambda)
```

Arguments

x	a numeric vector or scalar
lambda	Box-Cox transformation variable.

Value

A vector or scalar of the derivative of the Box-Cox function evaluated at x.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

See Also

boxcox boxcox.inv boxcox.inv.density

boxcox.inv *boxcox.inv*

Description

Inverse of the Box-Cox transformation. Called internally in bc.twocomp and bc.fourcomp.

Usage

```
boxcox.inv(y, lambda)
```

Arguments

y	a numeric vector or scalar
lambda	Box-Cox transformation variable.

Value

A vector or scalar of the untransformed values of x.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

See Also

boxcox boxcox.deriv boxcox.inv.density

---

boxcox.inv.density *boxcox.inv.density*

Description

A variable transformation that gives the probability density function (PDF) of the inverse Box-Cox transformation of a normal random variable. Called internally in plot.model.

Usage

```
boxcox.inv.density(y, lambda, mu, sig)
```

Arguments

y	a numeric vector or scalar
lambda	the transformation parameter
mu	the mean of the transformed component
sig	the standard deviation of the transformed component

Value

A vector or scalar of the untransformed x values.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Box, George EP, and David R. Cox. "An analysis of transformations." Journal of the Royal Statistical Society. Series B (Methodological) (1964): 211-252.

See Also

boxcox boxcox.inv boxcox.deriv

---

em.twocomp.m1 *em.twocomp.m1*

Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixtures in which all controls are constrained to one component and the cases follow a mixture of the two components (two component constrained model). This is used as an internal method and is called from bc.twocomp.

Usage

```
em.twocomp.m1(x.all, case.indicator, max.iters = 1000, errtol = 1e-09,

control.comp = 1, start.vals=NULL)
```

Arguments

x.all	vector of cases and controls
case.indicator	a vector of equal length to x.all with 1's in the case positions and 0's in the control positions
max.iters	the maximum number of iterations to run
errtol	Error tolerance level. Approximates convergence of the maximum log likelihood value.
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

Value

max.loglike	the maximum log likelihood value for the algorithm
mu	estimated means for each component
sig	estimated standard deviations for each component
pi	estimated proportion of cases in each component
n.iters	the number of iterations the algorithm took to converge
control.comp	indicator of which component contains the controls (1 or 2)

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

See Also

bc.binorm bc.twocomp bc.fourcomp em.twocomp.m2 em.twocomp.m3

---

em.twocomp.m2 *em.twocomp.m2*

Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixture models. This is used as an internal method and is called twice from bc.fourcomp: once for the cases and once for the controls (four component model).

Usage

```
em.twocomp.m2(x.all, max.iters = 1000, errtol = 1e-09, start.vals=NULL)
```

Arguments

x.all	vector of data
max.iters	the maximum number of iterations to run
errtol	Error tolerance level. Approximates convergence of the maximum log likelihood value.
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

Value

max.loglike	the maximum log likelihood value for the algorithm
mu	estimated means for each component
sig	estimated standard deviation for each component
pi	estimated proportion of data in each component
n.iters	the number of iterations the algorithm took to converge

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

See Also

bc.binorm bc.twocomp bc.fourcomp em.twocomp.m1 em.twocomp.m3

---

em.twocomp.m3 *em.twocomp.m3*

Description

Expectation maximization (EM) algorithm for estimating two-component Gaussian mixtures with different mixture proportions for cases and controls (two component unconstrained model). This is used as an internal method and is called from bc.twocomp.

Usage

```
em.twocomp.m3(x.all, case.indicator, max.iters = 1000, errtol = 1e-09,

control.comp = 1, start.vals=NULL)
```

Arguments

x.all	vector of cases and controls
case.indicator	a vector of equal length to x.all with 1's in the case positions and 0's in the control positions
max.iters	the maximum number of iterations to run
errtol	Error tolerance level. Approximates convergence of the maximum log likelihood value.
control.comp	indicator of which component contains the controls (1 or 2)
start.vals	starting values for the EM algorithm. If NA, the starting values are estimated from the data.

Value

max.loglike	the maximum log likelihood value for the algorithm
mu	estimated means for each component
sig	estimated standard deviations for each component
pi.cs	estimated proportion of cases in each component
pi.ctrl	estimated proportion of controls in each component
n.iters	the number of iterations the algorithm took to converge
control.comp	indicator of which component contains the controls (1 or 2)

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

See Also

bc.binorm bc.twocomp bc.fourcomp em.twocomp.m1 em.twocomp.m2

---

lambda *lambda*

Description

An accessor function. Retrieves the transformation parameter, lambda, of a model object.

Usage

```
lambda(model)
```

Arguments

model	an object of type model from bimixt.model

Value

The numeric value for the Box-Cox transformation parameter, lambda.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

mn stdev prop maxll type

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
lambda(model)
```

lr.test *lr.test*

Description

Computes the likelihood ratio test to compare two bimixt models.

Usage

```
lr.test(model1, model2)
```

Arguments

model1	an object of type model from bimixt.model.
model2	an object of type model from bimixt.model

Details

The model fits for model1 and model2 will be compared using the likelihood ratio test. Models must have been fit on the same data sets.

Value

Returns a p-value indicating the significance of the likelihood ratio test.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

ROCauc ROCpauc ROCplot

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model1=bimixt.model(case=case,control=control, type="4c")
model2=bimixt.model(case=case,control=control, type="binorm")
lr.test(model1, model2)
```

maxll *maxll*

Description

An accessor function. Retrieves the maximum log likelihood value of a model object.

Usage

```
maxll(model)
```

Arguments

model	an object of type model from bimixt.model

Value

The numeric value for the maximum log likelihood value for the model.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

lambda mn stdev prop type

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
maxll(model)
```

mn *mn*

Description

An accessor function. Retrieves the component means of a model object.

Usage

```
mn(model, transformed = F)
```

Arguments

model	an object of type model from bimixt.model
transformed	A Boolean indicating whether to return the mean values on the transformed scale (TRUE) or the original scale (FALSE default). The transformed means are estimates of the Gaussian component means. The original scale means are Monte Carlo estimates of the mean of the distribution of the inverse Box-Cox function applied to the estimated Gaussian component distribution.

Value

cases	A vector (or scalar) of numeric values for the mean of each case component in the model.
controls	A vector (or scalar) of numeric values for the mean of each control component in the model.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

lambda stdev prop maxll type

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
mn(model)
```

---

plot.model *plot.model*

Description

Plot method for a mixture model object.

Usage

```
## S3 method for class 'model'
plot(x, histogram = T, breaks = "Sturges", main = model$type,

cols = c("#008ED6", "#990033"), ylab = "Density", xlab = "", ...)
```

Arguments

x	an object of type model from bimixt.model
histogram	a Boolean indicating whether to plot a histogram of the original data (default = true). Histogram is plotted using the hist function.
breaks	the types of breaks to be used in hist
main	a character string to be used as the title of the plot
cols	a vector of length 2 specifying the colors of the components c(color of control component, color of
ylab	y label of the plot
xlab	x label of the plot
...	Not used.

Value

Plots a model object.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

summary.model print.model

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
plot(model)
```

---

print.model *print.model*

Description

Print method for a mixture model object.

Usage

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

Arguments

x	an object of type model from bimixt.model
...	Not used.

Value

Values used in fitting a mixture model object.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

summary.model plot.model

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
print(model)
```

---

prop *prop*

Description

An accessor function. Retrieves the case component proportions and control component proportions of a model object.

Usage

```
prop(model)
```

Arguments

model	an object of type model from bimixt.model

Value

cases	A vector (or scalar) of numeric values for the proportion of cases in each case component of the model.
controls	A vector (or scalar) of numeric values for the proportion of controls in each control component of the model.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

lambda mn stdev maxll type

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
prop(model)
```

---

rmix *rmix*

Description

Creates a random sample from a normal mixture distribution with two components.

Usage

```
rmix(n, mu1, s1, mu2, s2, p1)
```

Arguments

n	size of random sample
mu1	mean of first component
s1	standard deviation of first component
mu2	mean of second component
s2	standard deviation of second component
p1	proportion of values in the first component

Value

A vector of n numeric values from a sample mixture distribution.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

Examples

```
rmix(30,5,1,10,1.2,.95)
```

ROCauc *ROCauc*

Description

Finds the area under the ROC curve.

Usage

```
ROCauc(model, direction = "auto")
```

Arguments

model	an object of type model from bimixt.model
direction	same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).

Value

Returns the area under the curve (AUC) for the fitted and empirical receiver operator characteristic (ROC) curves. The empirical AUC value is calculated using the pROC package.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Fr?d?rique Lisacek, Jean-Charles Sanchez and Markus M?ller (2011). "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77

See Also

auc ROCpauc ROCcoords ROCplot

Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCauc(model)
```

ROCcoords *ROCcoords*

Description

Takes in a threshold, specificity, or sensitivity value and calculates the other two values.

Usage

```
ROCcoords(model, direction = "auto", x, input)
```

Arguments

model	an object of type model from bimixt.model
direction	same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).
x	The numeric value for the input. If input is "sensitivity" or "specificity" x must be between 0 and 1.
input	A string that defines what the input type is. Valid inputs are "sensitivity", "specificity", or "threshold". These can be shortened to "sens", "spec", "thr" or "se", "sp", "t".

Value

Returns a numeric vector with the values of threshold, specificity, and sensitivity.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

coords ROCauc ROCpauc ROCplot

Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCcoords(model,x=.95,input="sens")
ROCcoords(model,x=.95,input="spec")
ROCcoords(model,x=9,input="thr")
```

ROCpauc *ROCpauc*

Description

Finds the partial area under the ROC curve.

Usage

```
ROCpauc(model, spec.lower = 0.95, spec.upper = 1, direction = "auto")
```

Arguments

model	an object of type model from bimixt.model
spec.lower	a value between 0 and 1 that serves as the lower bound of the specificity to be used in the PAUC calculation
spec.upper	a value between 0 and 1 that serves as the upper bound of the specificity to be used in the PAUC calculation
direction	same as roc: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the predictor values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < t <= cases).

Value

Returns the partial area under the curve (pAUC) for the fitted and empirical receiver operator characteristic (ROC) curves between spec.lower and spec.upper. The empirical pAUC value is calculated using the pROC package.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 7, 77. DOI: 10.1186/1471-2105-12-77.

See Also

auc ROCauc ROCcoords ROCplot

Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model= bimixt.model(cases,controls,"4c")
ROCpauc(model, spec.lower = .85, spec.upper = 1)
```

ROCplot *ROCplot*

Description

Creates a ROC plot.

Usage

```
ROCplot(model, direction = "auto")
```

Arguments

model	an object of type model from bimixt.model
direction	same as roc: same as pROC: the direction in which to make the comparison. "auto" (default): automatically define in which group the median is higher and take the direction accordingly. ">": if the values for the control group are higher than the values of the case group (controls > t >= cases). "<": if the values for the control group are lower than the values of the case group (controls < t <= cases).

Value

Plots empirical and model-based estimates of the receiver operator characteristic (ROC) curve. The empirical plot comes from the pROC package.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

References

Xavier Robin, Natacha Turck, Alexandre Hainard, et al. (2011) "pROC: an open-source package for R and S+ to analyze and compare ROC curves". BMC Bioinformatics, 7, 77. DOI: 10.1186/1471-2105-12-77.

See Also

auc ROCcoords ROCauc ROCpauc

Examples

```
cases=rmix(50,10,1.2,20,1.3,.7)
controls=rmix(50,9,1.1,17,1.3,.95)
model=bimixt.model(cases,controls,"4c")
ROCplot(model)
```

---

stdev *stdev*

Description

An accessor function. Retrieves the component standard deviations of a model object.

Usage

```
stdev(model, transformed = F)
```

Arguments

model	an object of type model from bimixt.model
transformed	A Boolean indicating whether to return the standard deviation values on the transformed scale (TRUE) or the original scale (FALSE default). The transformed standard deviations are estimates of the Gaussian component standard deviations. The original scale standard deviations are Monte Carlo estimates of the standard deviation of the inverse Box-Cox of the estimated Gaussian component distribution.

Value

cases	A vector (or scalar) of numeric values for the standard deviation of each case component in the model.
controls	A vector (or scalar) of numeric values for the standard deviation of each control component in the model.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

lambda mn prop maxll type

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
stdev(model)
```

summary.model *summary.model*

Description

Summary method for a mixture model object.

Usage

```
## S3 method for class 'model'
summary(object, ...)
```

Arguments

object	an object of type model from bimixt.model
...	Not used.

Value

Gives a table with the estimated means and standard deviations of the Gaussian components (following the Box-Cox transformation), the estimated means and standard deviations of the untransformed components (before transforming for normality), and the estimated case and control proportions for each component in the mixture model.

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

plot.model print.model

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="4c")
summary(model)
```

type *type*

Description

An accessor function. Retrieves the model type of a model object.

Usage

```
type(model)
```

Arguments

model	an object of type model from bimixt.model

Value

Returns the type of the model, either "4c", "2cu", "2cc", or "binorm".

Author(s)

Michelle Winerip, Garrick Wallstrom, Joshua LaBaer

See Also

lambda mn stdev prop maxll

Examples

```
case=rmix(50,10,1.2,15,1,.7)
control=rmix(50,10,1.2,15,1,.95)
model=bimixt.model(case=case,control=control, type="2cu")
type(model)
```

Index