Package 'bivrp'

December 16, 2016

Type Package

Title Bivariate Residual Plots with Simulation Polygons

Version 1.0

Date 2016-12-13

Author Rafael de Andrade Moral [aut, cre], John Hinde [aut], Clarice Garcia Borges Demetrio [aut]

Maintainer Rafael de Andrade Moral <rafael_moral@yahoo.com.br>

Depends R (>= 3.0.0), MASS (>= 7.3-35), methods, graphics, stats

Suggests mvtnorm (>= 1.0-3)

Description Generates bivariate residual plots with simulation polygons for any diagnostics and bivariate model from which functions to extract the desired diagnostics, simulate new data and refit the models are available.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-16 23:36:29

R topics documented:

bivrp-package *Bivariate Residual Plots with Simulation Polygons*

Description

Generates bivariate residual plots with simulation polygons for any diagnostics and bivariate model from which functions to extract the desired diagnostics, simulate new data and refit the models are available.

Details

Package:	bivrp
Type:	Package
Title:	Bivariate Residual Plots with Simulation Polygons
Version:	1.0
Date:	2016-12-13
Authors@R:	c(person("Rafael", "de Andrade Moral", role = c("aut", "cre"), email = "rafael_moral@yahoo.com.br"), person
Author:	Rafael de Andrade Moral [aut, cre], John Hinde [aut], Clarice Garcia Borges Demetrio [aut]
Maintainer:	Rafael de Andrade Moral <rafael_moral@yahoo.com.br>
Depends:	R (>= 3.0.0), MASS (>= 7.3-35), methods, graphics, stats
Suggests:	mvtnorm (>= 1.0-3)
Description:	Generates bivariate residual plots with simulation polygons for any diagnostics and bivariate model from whic
License:	GPL (>=2)

Index of help topics:

Author(s)

Rafael de Andrade Moral [aut, cre], John Hinde [aut], Clarice Garcia Borges Demetrio [aut]

Maintainer: Rafael de Andrade Moral <rafael_moral@yahoo.com.br>

bivrp *Bivariate Residual Plots with Simulation Polygons*

Description

description

Usage

```
bivrp(obj, sim = 99, conf = 0.95, diagfun, simfun, fitfun,
      verb = F, add.dplots = T, theta.sort = T, add.polygon = F,
      reduce.polygon = T, kernel = F, chp = F, superpose.points = F,
      one.dim = F, xlab, ylab, main, clear.device = F, ...)
```

Arguments

obj	fitted model object
sim	number of simulations used to compute envelope. Default is 99
conf	confidence level of the simulated polygons. Default is 0.95
diagfun	user-defined function used to obtain the diagnostic measures from the fitted model object
simfun	user-defined function used to simulate a random sample from the model estimated parameters
fitfun	user-defined function used to re-fit the model to simulated data
verb	logical. If TRUE, prints each step of the simulation procedure
add.dplots	logical. If TRUE, adds the marginal density plots
theta.sort	logical. If TRUE, produces a simulated polygon for each point
add.polygon	logical. If TRUE, plots the simulated polygons as well
reduce.polygon	logical. If TRUE, reduces the polygon area using the algorithm in get.k and get.newpolygon. If FALSE, performs convex hull peeling to reduce the area
kernel	logical. If TRUE, instead of using polygons for each point, computes 2d kernels and plots the contours
chp	logical. If TRUE, instead of using polygons for each point, performs convex hull peeling over all simulated points
superpose.points	only used if kernel or chp is TRUE. Logical argument, if TRUE, plots all simulated bivariate diagnostics
one.dim	logical. If TRUE, plots only the marginal density plots
xlab	argument passed to par
ylab	argument passed to par
main	argument passed to par
clear.device	logical. If TRUE, clears the plotting device after producing the bivariate residual plot with simulation polygons
...	further arguments passed to par

Details

This approach relies on the same strategy used for producing half-normal plots with simulation envelopes. Given a vector of bivariate model diagnostics, the angle each point makes with the origin is calculated to order them. Then, by default 99 bivariate response variables are simulated from the fitted model, using the same model matrices, error distribution and fitted parameters. The model is refitted to each simulated sample, obtaining the same type of model diagnostics, again ordered by the angle they form with the origin. We have, for each bivariate diagnostic, 99 simulated bivariate diagnostics forming the whole cloud of simulated diagnostics. By default, we then obtain the convex hulls of each set of the s sets of points and obtain a reduced polygon whose area is 95% of the original convex hull's area, forming the simulated polygon. The points are then connected to the centroids of their respective simulated polygons and, if they lie outside the polygons, they are drawn in red. For the final display, the polygons are erased so as to ease visualization.

There is no automatic implementation of a bivariate model in this function, and hence users must provide three functions for `bivrp`. The first function, `diagfun`, must extract the desired model diagnostics from a model fit object. The second function, `simfun`, must return the response variable, simulated using the same error distributions and estimated parameters from the fitted model. The third and final function, `fitfun`, must return a fitted model object. See the Examples section.

Value

The function returns an object of class "bivrp", which is a list containing the following components:

`reslist.ord`	list of ordered diagnostics from model refitting to each simulated dataset
`res.original.ord`	original model diagnostics
`res1`	diagnostics from variable 1
`res2`	diagnostics from variable 2
`add.polygon`	logical. Equals TRUE if `add.polygon=TRUE` in the bivrp call
`res.original1`	original model diagnostics for variable 1
`res.original2`	original model diagnostics for variable 2
`theta.sort`	logical. Equals TRUE if `theta.sort=TRUE` in the bivrp call
`conf`	confidence level of the simulated polygons
`superpose.points`	logical. Equals TRUE if `superpose.points=TRUE` in the bivrp call
`kernel`	logical. Equals TRUE if `kernel=TRUE` in the bivrp call
`one.dim`	logical. Equals TRUE if `one.dim=TRUE` in the bivrp call
`chp`	logical. Equals TRUE if `chp=TRUE` in the bivrp call
`add.dplots`	logical. Equals TRUE if `add.dplots=TRUE` in the bivrp call
`reduce.polygon`	logical. Equals TRUE if `reduce.polygon=TRUE` in the bivrp call

Author(s)

Rafael A. Moral <rafael_moral@yahoo.com.br>, John Hinde and Clarice G. B. Demétrio

Examples

```
## simulating a bivariate normal response variable

require(mvtnorm)
n <- 80
beta1 <- c(2, .4)
beta2 <- c(.2, .2)
x <- seq(1, 10, length=n)
X <- model.matrix(~ x)
mu1 <- X%*%beta1
mu2 <- X%*%beta2
sig1 <- 2
sig2 <- 3
sig12 <- -1.7
Sig1 <- diag(rep(sig1), n)
Sig2 <- diag(rep(sig2), n)
Sig12 <- diag(rep(sig12), n)
V <- rbind(cbind(Sig1, Sig12),
           cbind(Sig12, Sig2))

set.seed(2016)
Y <- as.numeric(rmvnorm(1, c(mu1, mu2), V))

## code for fitting the model estimating covariance or not
loglik.bn <- function(theta, Y, X, covariance=TRUE) {
  n <- length(Y)/2
  beta1 <- theta[1:2]
  beta2 <- theta[3:4]
  sig1 <- exp(theta[5])
  sig2 <- exp(theta[6])
  if(covariance) sig12 <- theta[7] else sig12 <- 0
  mu1 <- X%*%beta1
  mu2 <- X%*%beta2
  Sig1 <- diag(rep(sig1), n)
  Sig2 <- diag(rep(sig2), n)
  Sig12 <- diag(rep(sig12), n)
  V <- rbind(cbind(Sig1, Sig12),
             cbind(Sig12, Sig2))
  llik <- dmvnorm(Y, c(mu1, mu2), V, log=TRUE)
  return(-llik)
}

bivnormfit <- function(Y, X, covariance, init) {
  if(covariance) ni <- 7 else ni <- 6
  if(missing(init)) init <- rep(0, ni)
  fit <- optim(init, loglik.bn, Y=Y, X=X, covariance=covariance, method="BFGS")
  coefs <- fit$par
  coefs[5:6] <- exp(coefs[5:6])
  fitted <- c(X%*%coefs[1:2], X%*%coefs[3:4])
  resid <- Y - fitted
  ret <- list("coefs"=coefs, "covariance"=covariance, "n"=length(Y)/2,
              "X"=X, "fitted"=fitted, "resid"=resid, "loglik"=fit$value, "Y"=Y)
```

```
  class(ret) <- "bivnormfit"
  return(ret)
}

## fitting bivariate models with and without estimating covariance
fit0 <- bivnormfit(Y, X, covariance=FALSE)
fit1 <- bivnormfit(Y, X, covariance=TRUE)
## likelihood-ratio test
2*(fit0$loglik - fit1$loglik)
pchisq(43.33, 1, lower=FALSE)

## function for extracting diagnostics (raw residuals)
dfun <- function(obj) {
  r <- obj$resid
  n <- obj$n
  return(list(r[1:n], r[(n+1):(2*n)]))
}

## function for simulating new response variables
sfun <- function(obj) {
  n <- obj$n
  fitted <- obj$fitted
  sig1 <- obj$coefs[5]
  sig2 <- obj$coefs[6]
  if(obj$covariance) sig12 <- obj$coefs[7] else sig12 <- 0
  Sig1 <- diag(rep(sig1), n)
  Sig2 <- diag(rep(sig2), n)
  Sig12 <- diag(rep(sig12), n)
  V <- rbind(cbind(Sig1, Sig12),
             cbind(Sig12, Sig2))
  beta1 <- obj$coefs[1:2]
  beta2 <- obj$coefs[3:4]
  mu1 <- obj$X%*%beta1
  mu2 <- obj$X%*%beta2
  Y <- as.numeric(rmvnorm(1, c(mu1, mu2), V))
  return(list(Y[1:n], Y[(n+1):(2*n)], "X"=obj$X, "covariance"=obj$covariance))
}

## function for refitting the model to simulated data
ffun <- function(new.obj) {
  Ynew <- c(new.obj[[1]], new.obj[[2]])
  bivnormfit(Ynew, new.obj$X, new.obj$covariance)
}

## Not run:
## Bivariate residual plot for model 1 (without estimating covariance)
plot1 <- bivrp(fit0, diagfun=dfun, simfun=sfun, fitfun=ffun, verb=T)
## without polygon area reduction
plot(plot1, conf=1)
## drawing polygons
plot(plot1, add.polygon=T)
## without ordering
plot(plot1, theta.sort=F, kernel=T, add.dplots=T, superpose=T)
```

```
## Bivariate residual plot for model 2 (estimating covariance)
plot2 <- bivrp(fit1, diagfun=dfun, simfun=sfun, fitfun=ffun, verb=T)
## without polygon area reduction
plot(plot2, conf=1)
## drawing polygons
plot(plot2, add.polygon=T, conf=1)
## without ordering
plot(plot2, theta.sort=F, kernel=T, add.dplots=T, superpose=T)

## End(Not run)
```

`bivrp-internal` *Internal functions to prepare* `bivrp` *objects*

Description

Internal functions used to prepare `bivrp` objects for plotting

Author(s)

Rafael A. Moral <rafael_moral@yahoo.com.br>, John Hinde and Clarice G. B. Demétrio

---

`is.point.inside` *Determine if point is inside or outside a simple polygon area*

Description

Returns whether a point is inside or outside the convex polygon formed with the coordinates in a data frame or matrix

Usage

```
is.point.inside(point, polyg)
```

Arguments

`point`	vector of two values for a point in the Cartesian plane
`polyg`	data frame or matrix with the coordinates forming the convex polygon

Details

The algorithm used here draws a ray from the point and counts the number of intersections made with the polygon. If the number of intersections is only one, then this means the point is inside the convex polygon.

Value

This function returns TRUE, if the point is inside and FALSE, otherwise.

Author(s)

Rafael A. Moral <rafael_moral@yahoo.com.br>, John Hinde and Clarice G. B. Demétrio

Examples

```
my.polygon <- data.frame(c(1, 2, 3, 4, 3),
                         c(1, 0, .5, 3, 4))
points.to.test <- list(c(0, 0), c(2.5, 1), c(3.5, 4))

unlist(lapply(points.to.test, is.point.inside, my.polygon))
```

---

plot.bivrp *Plot Method for bivrp Objects*

Description

Plots the bivariate residual plot with simulation polygons from a bivrp object

Usage

```
## S3 method for class 'bivrp'
plot(x, kernel, superpose.points, chp, add.dplots,
          theta.sort, add.polygon, reduce.polygon, one.dim, pch = 16, cex = 0.8,
           conf, xlab, ylab, main, ...)
```

Arguments

x	object of class bivrp
kernel	logical. If TRUE, instead of using polygons for each point, computes 2d kernels and plots the contours
superpose.points	only used if kernel or chp is TRUE. Logical argument, if TRUE, plots all simulated bivariate diagnostics
chp	logical. If TRUE, instead of using polygons for each point, performs convex hull peeling over all simulated points
add.dplots	logical. If TRUE, adds the marginal density plots
theta.sort	logical. If TRUE, produces a simulated polygon for each point
add.polygon	logical. If TRUE, plots the simulated polygons as well
reduce.polygon	logical. If TRUE, reduces the polygon area using the algorithm in get.k and get.newpolygon. If FALSE, performs convex hull peeling to reduce the area

one.dim	logical. If TRUE, plots only the marginal density plots
pch	argument passed to par
cex	argument passed to par
conf	confidence level of the simulated polygons. Default is 0.95
xlab	argument passed to par
ylab	argument passed to par
main	argument passed to par
...	further arguments passed to par

Author(s)

Rafael A. Moral <rafael_moral@yahoo.com.br>, John Hinde and Clarice G. B. Demétrio

---

polygon-operations *Polygon operations*

Description

Convex polygon operations - determination of area, centre of mass, and area reduction

Usage

```
polygon.area(P)
get.k(P, conf)
get.newpolygon(k, P)
```

Arguments

P	2-column matrix or data.frame with the coordinates of the vertices of the convex polygon
conf	proportion of the area of polygon P
k	distance from the centre of mass to the vertices to be reduced so that new polygon has proportion conf of the area of polygon P

Author(s)

Rafael A. Moral <rafael_moral@yahoo.com.br>, John Hinde and Clarice G. B. Demétrio

See Also

is.point.inside polygon

Examples

```
oldPolygon <- data.frame(x=c(2,1,3,4.5,5), y=c(1,3,5,4.5,2))

# area
polygon.area(oldPolygon)$area
# centre of mass
polygon.area(oldPolygon)$centre

# get a new polygon with 50% of the area of the old one
k <- get.k(P=oldPolygon, conf=0.5)
newPolygon <- get.newpolygon(k=k, P=oldPolygon)
polygon.area(newPolygon)$area/polygon.area(oldPolygon)$area

# illustration with conf=0.4
plot(oldPolygon, xlim=c(0,6), ylim=c(0,6), main="(a)", pch=16)
polygon(oldPolygon, lwd=2, col="#00000033")
text(oldPolygon, c(expression(P[1]), expression(P[2]),
                   expression(P[3]), expression(P[4]),
                   expression(P[5])), pos=c(1,2,3,4,4), cex=2)
polygon(newPolygon, border=4, lwd=2, col="#52A3E199")
points(newPolygon, pch=16, col=4)
text(newPolygon, c(expression(paste(P[1],minute)), expression(paste(P[2],minute)),
           expression(paste(P[3],minute)), expression(paste(P[4],minute)),
           expression(paste(P[5],minute))), pos=c(1,3,2,4,4), col=4, cex=2)

C <- polygon.area(oldPolygon)$centre
text(C[1], C[2], "C", pos=4, cex=2)
for(i in 1:5) lines(c(C[1], oldPolygon[i,1]),
                    c(C[2], oldPolygon[i,2]), lty=2, lwd=2, type="b")
```

Index