Package 'gamm4.test'

February 12, 2018

Type Package

Title Comparing Nonlinear Curves and Surface Estimations by Semiparametric Methods

Version 0.1.0

Author Shi Zhao <shizhao@iu.edu>

Maintainer Shi Zhao <shizhao@iu.edu>

Description To compare nonlinear curves and surface estimations between groups using semiparametric methods for cross-sectional and longitudinal dataset.

Date 2018-02-09

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Depends R (>= 2.14.0)

Imports gamm4, mgcv, doParallel, foreach, parallel, plotly, Matrix, RColorBrewer

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-12 10:10:48 UTC

R topics documented:

gam.grptest — *Test the equality of nonlinear curves and surface estimations by semiparametric method*

Description

This function tests the equality of nonlinear curves and surface estimations based on L2 distance. The semiparametric estimation uses 'mgcv' package. The specific model considered here is

Usage

```
gam.grptest(formula, test, data, N.boot = 200, m = 225, parallel = FALSE)
```

Arguments

`formula` A GAM formula. This is like the formula for a glm except that smooth terms (s and t2 but not te) can be added to the right hand side of the formula.

`test` An indicator of variable for testing nonlinear curves or surface estimations

`data` A data frame or list containing the model response variable and covariates required by the formula.

`N.boot` the number of bootstrap replicates. This should be a single positive integer.

`m` the number of the sampling points for the Monte-Carlo integration.

`parallel` Parallel computation of semiparametric estimations with bootstrap samples for getting test statistics under null hypothesis.

Details

y_ij= m_i(x_ij) + e_ij,

where m_i(.), are semiparametric smooth functions; e_ij are subject-specific errors. The errors e_ij do not have to be independent N(0, sigma^2) errors. The errors can be heteroscedastic, i.e., e_ij = sigma_i(x_ij) * u_ij, where u_ij are independent identically distributed errors with mean 0 and variance 1.

We are interested in the problem of testing the equality of the regression curves (when x is one-dimensional) or surfaces (when x is two-dimensional),

H_0: m_1(.) = m_2(.) = ... v.s. H_1: otherwise

The problem can also be viewed as the test of the equality in the one-sample problem for functional data.

A bootstrap algorithm is applied to test the equality of semiparametric curves or surfaces based on L2 distance.

See Also

`gam` `gamm4.grptest` `plot.gamtest` `T.L2c`

Examples

```
n1 <- 200
x1 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- sin(2*x1) + cos(2*x1) + e1

n2 <- 120
x2 <- runif(n2, min=0, max=3)
sd2 <- 0.25
e2 <- rnorm(n2, sd=sd2)
y2 <- sin(2*x2) + cos(2*x2) + x2 + e2

data.bind <- rbind(cbind(x1,y1,1), cbind(x2,y2,2))
data.bind <- data.frame(data.bind)
colnames(data.bind)=c('x','y','group')

t1 <- gam.grptest(y~s(x,bs="cr"), test=~group, data=data.bind, parallel=FALSE)
t1
plot(t1)

########
## Semiparametric test the equality for regression surfaces
## Simulate data sets

n1 <- 500
x11 <- runif(n1,min=0, max=3)
x12 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- 2*x11^2 + 3*x12^2  + e1

n2 <- 420
x21 <- runif(n2, min=0, max=3)
x22 <- runif(n2, min=0, max=3)
sd2 <- 0.25
e2 <- rnorm(n2, sd=sd2)
y2 <- 2*x21^2 + 3*x22^2 + 6*sin(2*pi*x21) + e2

n3 <- 550
x31 <- runif(n3,min=0, max=3)
x32 <- runif(n3,min=0, max=3)
sd3 <- 0.2
e3 <- rnorm(n3,sd=sd1)
y3 <- 2*x31^2 + 3*x32^2  + e3

data.bind <- rbind(cbind(x11, x12 ,y1,1), cbind(x21, x22, y2,2), cbind(x31, x32, y3,3))
data.bind <- data.frame(data.bind)
colnames(data.bind)=c('x1','x2', 'y','group')
```

```
tspl <- gam.grptest(y~s(x1,x2), test=~group, data=data.bind, N.boot=200, m=225, parallel=FALSE)
tspl$p.value #p-value
plot(tspl, test.statistic = TRUE)
plot(tspl, type="contour")
plot(tspl, type="persp")
plot(tspl, type="plotly.persp")
plot(tspl, type="plotly.persp",data.pts=TRUE)

########
## Data analyses with internal "outchild" dataset

data("outchild")
child<- outchild[order(outchild$SID,outchild$age),]
bs <- aggregate(.~SID, child, FUN=head, 1)

childcur <- bs[,c("SEX","WEIGHT","age")]
test.grpsex1 <- gam.grptest(WEIGHT~s(age), test=~SEX, data=childcur)
test.grpsex1
plot(test.grpsex1)
plot(test.grpsex1,test.statistic=TRUE)

childsurf <- bs[,c("SEX","HEIGHT","WEIGHT","age")]
test.grpsex2 <- gam.grptest(WEIGHT~s(HEIGHT,age), test=~SEX, data=childsurf)
test.grpsex2
plot(test.grpsex2)
plot(test.grpsex2, type="plotly.persp")
plot(test.grpsex2, type="plotly.persp",data.pts=TRUE)
```

gamm4.grptest *Test the equality of nonlinear curves and surface estimations by semiparametric methods with correlated data*

Description

This function tests the equality of nonlinear curves and surface estimations with correlated data based on L2 distance. The semiparametric estimation uses ’gamm4’ package with a compond symmetry correlation structure to adjust correlated observations. The specific model considered here is

Usage

```
gamm4.grptest(formula, random, test, data, N.boot = 200, m = 225,
  parallel = TRUE)
```

Arguments

`formula` A GAM formula. This is like the formula for a glm except that smooth terms (s and t2 but not te) can be added to the right hand side of the formula. Note that ids for smooths and fixed smoothing parameters are not supported.

`random` An optional formula specifying the random effects structure in lmer style.

`test` An indicator of variable for testing nonlinear curves or surface estimations

`data` A data frame or list containing the model response variable and covariates required by the formula.

`N.boot` the number of bootstrap replicates. This should be a single positive integer.

`m` the number of the sampling points for the Monte-Carlo integration.

`parallel` Parallel computation of semiparametric estimations with bootstrap samples for getting test statistics under null hypothesis

Details

y_ij= m_i(x_ij) + b_i + e_ij,

where m_i(.), are semiparametric smooth functions; b_i are subject-specific random intercept; e_ij are subject-specific errors. The errors e_ij do not have to be independent N(0, sigma^2) errors. The errors can be heteroscedastic, i.e., e_ij = sigma_i(x_ij) * u_ij, where u_ij are independent identically distributed errors with mean 0 and variance 1.

We are interested in the problem of testing the equality of the regression curves (when x is one-dimensional) or surfaces (when x is two-dimensional),

H_0: m_1(.) = m_2(.) = ...v.s.H_1: otherwise

The problem can also be viewed as the test of the equality in the one-sample problem for functional data.

A bootstrap algorithm is applied to test the equality of semiparametric curves or surfaces based on L2 distance.

See Also

`gamm4` `gam.grptest` `plot.gamtest`

Examples

```
#Test the equality of three nonlinear curves
m1 <- 120 #number of subjects in group 1
m2 <- 100 #number of subjects in group 2
m3 <- 110 #number of subjects in group 3
n1 <- 3 #number of repeated measurements for each subject in group 1
n2 <- 4 #number of repeated measurements for each subject in group 2
n3 <- 2 #number of repeated measurements for each subject in group 3
sigma1 <- 0.3
sigma2 <- 0.2
sigma3 <- 0.2
sigma.noise1 <- sigma.noise2 <- sigma.noise3 <- 0.1
f1 <- function(u) sin(2*pi*u)
f2 <- f3 <- function(u) sin(2*pi*u)+u/3
N1 <- m1*n1
N2 <- m2*n2
N3 <- m3*n3
```

```
x11 <- runif(N1,0,1)
b1i <- rnorm(m1,0,sigma1)
b1 <- rep(b1i, each=n1)
id1 <- rep(1:m1, each=n1)
y1 <- f1(x11) + b1 + rnorm(N1, 0, sigma.noise1)

x21 <- runif(N2,0,1)
b2i <- rnorm(m2,0,sigma2)
b2 <- rep(b2i,each=n2)
id2 <- rep((m1+1):(m1+m2),each=n2)
y2 <- f2(x21) + b2 + rnorm(N2,0,sigma.noise2)

x31 <- runif(N3,0,1)
b3i <- rnorm(m3,0,sigma3)
b3 <- rep(b3i,each=n3)
id3 <- rep((m1+m2+1):(m1+m2+m3),each=n3)
y3 <- f3(x31) + b3 + rnorm(N3,0,sigma.noise2)

dat <- data.frame(rbind(cbind(id1, x11,y1,1), cbind(id2, x21, y2,2), cbind(id3, x31, y3,3)))
colnames(dat)=c('id','x', 'y','grp')
testout <- gamm4.grptest(formula=y~s(x,k=6,bs=”cr”), test=~grp,
                        random=~(1|id), data=dat, N.boot=200, m=225, parallel = TRUE)
testout
plot(testout)

dat0 <- data.frame(rbind(cbind(id3, x31, y3, 3), cbind(id2, x21, y2, 2)))
colnames(dat0)=c('id', 'x', 'y', 'grp')
testout0 <- gamm4.grptest(formula=y~s(x,k=6,bs=”cr”), test=~grp,
       random=~(1|id), data=dat0, N.boot=200, m=225, parallel= TRUE)
testout0$p.value
plot(testout0, test.statistic = TRUE)

########
## Semiparametric test the equality for regression surfaces with longitudinal data
## Simulate data sets
f1 <- function(u,v) 2*u^2+3*v^2
f2 <- function(u,v) 2*u^2+3*v^2+sin(2*pi*u)

m1 <- 100 #number of subjects in group 1
n1 <- 4 #number of repeated measurements for each subject in group 1
m2 <- 120 #number of subjects in group 2
n2 <- 3 #number of repeated measurements for each subject in group 2
N1 <- m1*n1
N2 <- m2*n2
sigma1 <- 0.2
sigma2 <- 0.15
sigma.noise1 <- 0.04
sigma.noise2 <- 0.05
x11 <- runif(N1,0,1)
x12 <- runif(N1,0,1)
b1i <- rnorm(m1,0,sigma1)
b1 <- rep(b1i,each=n1)
```

```
id1 <- rep(1:m1,each=n1)
y1 <- f1(x11,x12) + b1 + rnorm(N1,0, sigma.noise1)

x21 <- runif(N2,0,1)
x22 <- runif(N2,0,1)
b2i <- rnorm(m2,0,sigma2)
b2 <- rep(b2i,each=n2)
id2 <- rep((m1+1):(m1+m2),each=n2)
y2 <- f2(x21,x22) + b2 + rnorm(N2,0,sigma.noise2)

y3 <- f1(x21,x22) + b2 + rnorm(N2,0,sigma.noise2)

dat <- data.frame(rbind(cbind(id1, x11, x12,y1,1), cbind(id2, x21, x22, y2,2)))
colnames(dat)=c('id','x1','x2', 'y','grp')

test.spline1 <- gamm4.grptest(formula=y~t2(x1,x2), test=~grp,
               random=~(1|id), data=dat, N.boot=200, m=225, parallel=TRUE)
plot(test.spline1)
plot(test.spline1, type=”plotly.persp”)
plot(test.spline1, type=”plotly.persp”, data.pts=TRUE)

dat0 <- data.frame(rbind(cbind(id1, x11, x12 , y1, 1), cbind(id2, x21, x22, y3, 2)))
colnames(dat0)=c('id','x1','x2', 'y','grp')

test.spline0 <- gamm4.grptest(y~t2(x1,x2), test=~grp,
               random=~(1|id), data=dat0, N.boot=200, m=225, parallel=TRUE)
test.spline0
plot(test.spline0, test.statistic = FALSE)
plot(test.spline0)
plot(test.spline0, type=”plotly.persp”)

########
## Data analyses with internal ”outchild” dataset

data(”outchild”)
outchild1016 <- outchild[(outchild$age<=16 & outchild$age>10),]
child.repw <- outchild1016[(outchild1016$RACE==1),]
child.reptest1 <- gamm4.grptest(HEIGHT~s(age), random=~(1|SID),
                               test=~SEX, data=child.repw, parallel = TRUE)
child.reptest1
plot(child.reptest1)
plot(child.reptest1,test.statistic = FALSE)

child.reptest2 <- gamm4.grptest(WEIGHT~t2(age,HEIGHT), random=~(1|SID),
                              test=~SEX, data = child.repw, parallel = TRUE)
plot(child.reptest2,type=”plotly.persp”)
plot(child.reptest2,type=”contour”)
```

outchild *Longitudinal health data of 1065 children from age 6 to 15.*

Description

A dataset containing the demographic and health related data of 1065 children.

Usage

```
data(outchild)
```

Format

A data frame with 6681 rows and 21 variables:

SID Child ID

SEX 1=male,2=female

RACE 1=white,2=black

WEIGHT Weight of the child

HEIGHT Height of the child

age Age of a child

dias Diastolic blood pressure

sys Systolic blood pressure

heartrate Heart rate

pulse Blood pulse

BMI Body Mass Index (BMI)

HeightZ Normalized height

HeightPct Percentile of height over standardized population

WeightZ Normalized weight

WeightPct Percentile of weight over standardized population

BMIZ Normalized BMI

BMIPCT Percentile of BMI over standardized population

SBPpct Systolic blood pressure percentile

DBPpct Diastolic blood pressure percentile

HTN_ANY 1=Hypertension 0=No hypertension

HTNpre_ANY 1=Pre-hypertension 0=No pre-hypertension

plot.gamtest *Plot a gamtest Object*

Description

This function plots the semiparametric estimation of nonlinear curves and surface.

Usage

```
## S3 method for class 'gamtest'
plot(x, test.statistic = FALSE,
  test.stat.type = "density", main = "", n = 256,
  legend.position = "topright", se.est = FALSE, data.pts = FALSE,
  type = "contour", ...)
```

Arguments

`x`	A gamtest object.
`test.statistic`	If TRUE, plot the density of the test statistic under null hypothesis; if FALSE, plot the estimated curves/surfaces.
`test.stat.type`	must have "test.statistic=TRUE". Default is "test.stat.type=density". If "test.stat.type=hist", plot the histogram of the test statistic under null.
`main`	The title of the plot.
`n`	The number of points that are used to draw the curves or surfaces in the plot.
`legend.position`	the position of legend in the plot: "topright", "topleft", "bottomright", "bottomleft", etc.
`se.est`	If TRUE, plot the pointwise 95% confidence intervals of curves; if FALSE, don't plot the pointwise confidence intervals.
`data.pts`	If TRUE, plot raw data points. If FALSE, only plot estimated curves/surfaces.
`type`	Only used for ploting surfaces. If "type=contour",then contour plot from vis.gam function in mgcv package; if "type=persp", then plot persp from vis.gam function; if "type=plotly.persp", then plot persp from plotly package.
`...`	Other options from package “mgcv” “vis.gam()” function.

Details

This function is to plot a gamtest object. If "test.statistic=TRUE", a density plot of the test statistic under null hypothesis will be generated; if "test.statistic=FALSE", the estimated curves/surfaces for all groups are drawn.

See Also

`gam` `gamm4` `gamm4.grptest` `gam.grptest`

Examples

```
n1 <- 200
x1 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- sin(2*x1) + cos(2*x1) + e1

n2 <- 120
x2 <- runif(n2, min=0, max=3)
sd2 <- 0.25
e2 <- rnorm(n2, sd=sd2)
y2 <- sin(2*x2) + cos(2*x2) + x2 + e2

data.bind <- rbind(cbind(x1,y1,1), cbind(x2,y2,2))
data.bind <- data.frame(data.bind)
colnames(data.bind)=c('x','y','group')

t1 <- gam.grptest(y~s(x,bs=”cr”),test=~group,data=data.bind)
t1
plot(t1)
plot(t1,test.statistic=TRUE)

########
## Semiparametric test the equality for regression surfaces
## Simulate data sets

n1 <- 200
x11 <- runif(n1,min=0, max=3)
x12 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- 2*x11^2 + 3*x12^2  + e1

n2 <- 120
x21 <- runif(n2, min=0, max=3)
x22 <- runif(n2, min=0, max=3)
sd2 <- 0.25
e2 <- rnorm(n2, sd=sd2)
y2 <- 2*x21^2 + 3*x22^2 + 4*sin(2*pi*x21) + e2

n3 <- 150
x31 <- runif(n3,min=0, max=3)
x32 <- runif(n3,min=0, max=3)
sd3 <- 0.2
e3 <- rnorm(n3,sd=sd1)
y3 <- 2*x31^2 + 3*x32^2  + e3

data.bind <- rbind(cbind(x11, x12 ,y1,1), cbind(x21, x22, y2,2), cbind(x31, x32, y3,3))
data.bind <- data.frame(data.bind)
colnames(data.bind)=c('x1','x2', 'y','group')
```

```
tspl <- gam.grptest(y~te(x1,x2),test=~group,data=data.bind,N.boot=200,m=225,parallel=FALSE)
tspl$p.value #p-value
plot(tspl)
plot(tspl,test.statistic = TRUE)
plot(tspl,type="plotly.persp")
plot(tspl,type="plotly.persp", data.pts=TRUE)
```

print.gamtest *Print a gamtest Object*

Description

This function print the semiparametric estimation of nonlinear curves and surface.

Usage

```
## S3 method for class 'gamtest'
print(x, digits = 4, ...)
```

Arguments

`x`	A gamtest object.
`digits`	Number of digits for test statistic and p-value.
`...`	Other generic options.

T.L2c *Test the equality of nonparametric curves or surfaces based on L2 distance*

Description

This function tests the equality of nonparametric curves and surface estimations based on L2 distance. The specific model considered here is

Usage

```
T.L2c(formula, test, data, N.boot = 200, degree = 1, criterion = c("aicc",
  "gcv"), family = c("gaussian", "symmetric"), m = 225, user.span = NULL,
  ...)
```

Arguments

`formula` A regression formula. This is like the formula for a lm.

`test` An indicator of variable for testing nonparametric curves or surface estimations

`data` A data frame or list containing the model response variable and covariates required by the formula.

`N.boot` the number of bootstrap replicates. This should be a single positive integer.

`degree` the degree of the local polynomials to be used. It can ben 0, 1 or 2.

`criterion` the criterion for automatic smoothing parameter selection: “aicc” denotes bias-corrected AIC criterion, “gcv” denotes generalized cross-validation.

`family` if “gaussian” fitting is by least-squares, and if “symmetric” a re-descending M estimator is used with Tukey’s biweight function.

`m` the number of the sampling points for the Monte-Carlo integration.

`user.span` the user-defined parameter which controls the degree of smoothing.

`...` other options from “loess” package.

Details

y_ij= m_i(x_ij) + e_ij,

where m_i(.), are semiparametric smooth functions; e_ij are subject-specific errors. The errors e_ij do not have to be independent N(0, sigma^2) errors. The errors can be heteroscedastic, i.e., e_ij = sigma_i(x_ij) * u_ij, where u_ij are independent identically distributed errors with mean 0 and variance 1.

We are interested in the problem of testing the equality of the regression curves (when x is one-dimensional) or surfaces (when x is two-dimensional),

H_0: m_1(.) = m_2(.) = ... v.s. H_1: otherwise

The problem can also be viewed as the test of the equality in the one-sample problem for functional data.

A bootstrap algorithm is applied to test the equality of semiparametric curves or surfaces based on L2 distance.

See Also

`gam.grptest`

Examples

```
n1 <- 200
x1 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- sin(2*x1) + cos(2*x1) + e1

n2 <- 120
x2 <- runif(n2, min=0, max=3)
sd2 <- 0.25
```

```
e2 <- rnorm(n2, sd=sd2)
y2 <- sin(2*x2) + cos(2*x2) + x2 + e2

dat <- data.frame(rbind(cbind(x1,y1,1), cbind(x2,y2,2)))
colnames(dat)=c('x','y','group')

t1 <- T.L2c(formula=y~x,test=~group,data=dat)
t1$p.value
########
## Semiparametric test the equality for regression surfaces
## Simulate data sets

n1 <- 200
x11 <- runif(n1,min=0, max=3)
x12 <- runif(n1,min=0, max=3)
sd1 <- 0.2
e1 <- rnorm(n1,sd=sd1)
y1 <- 2*x11^2 + 3*x12^2  + e1

n2 <- 120
x21 <- runif(n2, min=0, max=3)
x22 <- runif(n2, min=0, max=3)
sd2 <- 0.25
e2 <- rnorm(n2, sd=sd2)
y2 <- 2*x21^2 + 3*x22^2 + sin(2*pi*x21) + e2

n3 <- 150
x31 <- runif(n3,min=0, max=3)
x32 <- runif(n3,min=0, max=3)
sd3 <- 0.2
e3 <- rnorm(n3,sd=sd1)
y3 <- 2*x31^2 + 3*x32^2  + e3

data.bind <- data.frame(rbind(cbind(x11, x12 ,y1,1), cbind(x21, x22, y2,2), cbind(x31, x32, y3,3)))
colnames(data.bind)=c('x1','x2', 'y','group')

T.L2c(formula=y~x1+x2,test=~group,data=data.bind)
```

Index