Package 'gamlss.foreach'

October 13, 2022

Type Package

Title Parallel Computations for Distributional Regression

Version 1.1-6

Date 2022-08-28

Author Mikis Stasinopoulos [aut, cre, cph], Bob Rigby [aut], Fernanda De Bastiani [aut]

Description Computational intensive calculations for Generalized Additive Models for Location Scale and Shape, <doi:10.1111/j.1467-9876.2005.00510.x>.

Maintainer Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>

LazyLoad yes

Depends R (>= 2.2.1), gamlss, foreach, doParallel, methods

Imports gamlss.data, gamlss.dist, glmnet

License GPL-2 | GPL-3

URL https://www.gamlss.com/

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-28 10:00:02 UTC

R topics documented:

gamlss.foreach-package

Computational Intensive Functions within GAMLSS

Description

This package is intended for functions needed parallel computations provided by the package **foreach**.

At the moment the following functions exist:

`centiles.boot()`, which is designed get bootstrap confidence intervals for centile curves

`fitRolling()`, rolling regression which is common in time series analysis when one step ahead forecasts is required.

`fitPCR()`, for univariate principal component regression. I

Details

The DESCRIPTION file:

Package:	gamlss.foreach
Type:	Package
Title:	Parallel Computations for Distributional Regression
Version:	1.1-6
Date:	2022-08-28
Author:	Mikis Stasinopoulos [aut, cre, cph], Bob Rigby [aut], Fernanda De Bastiani [aut]
Description:	Computational intensive calculations for Generalized Additive Models for Location Scale and Shape, <d
Maintainer:	Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>
LazyLoad:	yes
Depends:	R (>= 2.2.1), gamlss, foreach, doParallel, methods
Imports:	gamlss.data, gamlss.dist, glmnet
License:	GPL-2 | GPL-3
URL:	https://www.gamlss.com/
NeedsCompilation:	no
Packaged:	2021-03-04 15:27:15 UTC; dimitriosstasinopoulos

Index of help topics:

gamlss.foreach-package	
	Computational Intensive Functions within GAMLSS
pc	Functions to Fit Principal Component Regression in GAMLSS
which.Data.Corr	Detecting Hight Pair-Wise Correlations in Data

Author(s)

Mikis Stasinopoulos, <d.stasinopoulos@londonmet.ac.uk>,and Bob Rigby <r.rigby@londonmet.ac.uk>

Maintainer: Mikis Stasinopoulos, <d.stasinopoulos@londonmet.ac.uk>

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

(see also https://www.gamlss.com/).

See Also

gamlss,centiles,centiles.pred

Examples

```
library(gamlss.foreach)
# fixed degrees of freedom
cl <- makePSOCKcluster(2)
registerDoParallel(2)
data(db)
nage <- with(db, age^0.33)
ndb <- data.frame(db, nage)
m1 <- gamlss(head~cs(nage, 12), sigma.fo=~cs(nage,4), nu.fo=~nage,
             tau.fo=~nage, family=BCT, data=ndb)
test1 <- centiles.boot(m1, xname="nage", xvalues=seq(0.01,20,0.2),B=10, power=0.33)
test1
plot(test1)
# degrees of freedom varying
m2 <- gamlss(head~pb(nage), sigma.fo=~pb(nage), nu.fo=~pb(nage),
             tau.fo=~pb(nage), family=BCT, data=ndb)
test2 <- centiles.boot(m2, xname="nage", xvalues=seq(0.01,20,0.2),B=10, power=0.33)
stopImplicitCluster()
test2
```

```
plot(test2)
```

`BayesianBoot` *Non parametric and Bayesian Bootstrapping for GAMLSS models*

Description

The function takes a GAMLSS fitted model and bootstrap it to create `B` bootstrap samples.

Usage

```
NonParametricBoot(obj, data = NULL, B = 100, newdata = NULL)

BayesianBoot(obj, data = NULL, B = 100, newdata = NULL)
```

Arguments

`obj`	a `gamlss` fitted model
`data`	a data frame
`B`	the number of boostrap samples
`newdata`	new data for `predictAll()`

Details

The function `NonParametric()` perform non-parametric bootstraping, Efron and Tibshirani (1993) while the function `BayesianBoot()` perform Bayesian bootstrap Rubin (1981)

Value

An `Bayesian.boot` object with elements

`boot`	the bootstrap samples
`B`	the required number of boostraps
`trueB`	the actual number of boostraps
`par`	the distribution parameters
`orig.coef`	the fitted coeficients from the GAMLSS model
`orig.call`	the call from the GAMLSS model

Author(s)

Mikis Stasinopoulos, <d.stasinopoulos@londonmet.ac.uk>

References

Efron, B. and Tibshirani, R, (1993), *An introduction to the bootstrap*, Chapman and Hall New York, Monographs on statistics and applied probability, vulume 57.

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Rubin, D. B. (1981) the bayesian bootstrap. *The annals of statistics*, pp. 130-134.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

(see also https://www.gamlss.com/).

See Also

gamlss

Examples

```
m1 <-gamlss(y~x+qrt, data=aids, family=NBI())
registerDoParallel(cores = 2)
B1 <- BayesianBoot(m1)
summary(B1)
plot(B1)
B2 <- NonParametricBoot(m1)
stopImplicitCluster()
summary(B2)
plot(B2)
```

---

centiles.boot *Bootstrapping centiles curves estimated using GAMLSS*

Description

This is a function designed for non-parametric bootstrapping centile curves (growth curves) when the fitted model is fitted using GAMLSS with a single explanatory variable (usually age). Non parametric bootstrapping resample the data with replacement. The model is refitted for each bootstraps sample. Notes that if smoothing is used in the model, it is advisable (but not necessary) that the smoothing degree of freedom are fixed throughout.

Usage

```
centiles.boot(obj, data = NULL, xname = NULL, xvalues = NULL,
  power = NULL, cent = c(2.5, 50, 97.5), B = 100, calibration = FALSE,
  ...)

## S3 method for class 'centiles.boot'
print(x, ...)
## S3 method for class 'centiles.boot'
summary(object, fun = "mean", ...)
## S3 method for class 'centiles.boot'
plot(x, quantiles = c(0.025, 0.975),
       ylab = NULL, xlab = NULL, location = "median", original = FALSE,
       scheme = c("shaded", "lines"), col.cent = "darkred",
       col.se = "orange", col.shaded = "gray", lwd.center = 1.5, ...)
```

Arguments

obj	a fitted gamlss object for the function centiles.boot()
data	a data frame containing the variables occurring in the formula. If it is missing, then it will try to get the data frame from the GAMLSS object
xname	the name (as character) of the unique explanatory variable (it has to be the same as in the original fitted model)
xvalues	a vector containing the new x-variable values for which bootstrap simulation predictions will be made
power	if power transformation is needed (but see example below)
cent	a vector of centile values for which the predicted centiles have to be evaluated, by default is: 2.5, 50 and 97.5
B	the number of bootstraps
calibration	whether to calibrate the centiles, default is FALSE
...	for extra arguments, for the centiles.pred() function
x	an a centiles.boot object
object	an a centiles.boot object
fun	for the summary() function this is a summary statistics function. The "mean" is the default
quantiles	specify which quantiles (in the plot() function) of the bootstrap distribution to plot
location	which location parameter to plot, with default the mean
original	logical if TRUE the original predicted centile values at the given xvalues are plotted (the default is FALSE)
ylab	y label for the plot
xlab	x label for the plot
scheme	which scheme of plotting to use "shaded" or "lines"

col.cent	the colour of the centile in the "shaded" scheme, with "darkred" as default
col.se	the colour of the standard errors for the "lines" scheme with default "orange"
col.shaded	the colour of the standard errors for the "shaded" scheme with default "gray"
lwd.center	the width of the central line

Details

This function is designed for bootstrapping centiles curves. It can be used to provide bootstrap means, standard deviations and quantiles, so the variability of the centile curves can be accessed (eg. by deriving confidence bands for centile curves).

Value

The function returns an object centile.boot which has its own print(), summary(), and plot() functions. The object centile.boot is a list with elements:

boot0	Containing centile predictions from the fitted model to the original data using the centile.pred() function on the new xvalues. This can be compared with the mean of the object to assess the bias
boot	A list of length trueB, each containing a matrix of dimension length(xvalues) by (length(cent)+1)
B	The number of bootstrap samples requested
trueB	The number of actual bootstrapping simulations performed. It is equal to B-number of failed simulations
xvalues	The new x-variable values for which the bootstrap simulation has taken place
cent	The centile values requested
original.call	The call of the original gamlss fitted model
yname	The name of the response variable, used in the plot() function
xname	The name of the x-variable, used in the plot() function
failed	A vector containing values identifying which of the bootstrap simulations had failed to converge and therefore have not included in the list boot

Note

See example below of how to use the function when a power transformation is used for the x-variable

Do not forget to use registerDoParallel(cores = NUMBER) or cl <- makeCluster(NUMBER) and registerDoParallel(cl) before calling the function centiles.boot(). Use closeAllConnections() after the fits to close the connections. Where NUMBER depends on the machine used.

Author(s)

Mikis Stasinopoulos, <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk> and Calliope Akantziliotou

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

(see also https://www.gamlss.com/).

See Also

gamlss,centiles,centiles.pred

Examples

```
# bring the data and fit the model
data(abdom)
m1<-gamlss(y~poly(x,2), sigma.fo=~x, data=abdom, family=BCT)
# perform the bootstrap simulation
# (only 10 bootstrap samples here)
registerDoParallel(cores = 2)
boC<-centiles.boot(m1,xname="x", xvalues=c(15,20,25,30,35,40,45), B=10)
stopImplicitCluster()
boC
# get summaries
summary(boC, fun="mean")
#summary(boC, "median")
#summary(boC, "quantile", 0.025)
plot(boC)

# with transformation in x within the formula
# unsuitable for large data set since it is slow
m2<-gamlss(y~cs(x^0.5),sigma.fo=~cs(x^0.5), data=abdom, family=BCT)
boC<-centiles.boot(m2,xname="x", xvalues=c(15,20,25,30,35,40,45), B=10)
summary(boC)
plot(boC)
#
# now with x-variable previously transformed
# better for large data set as it is faster
nx<-abdom$x^0.5
newd<-data.frame(abdom, nx=abdom$x^0.5)
m3<-gamlss(y~cs(nx),sigma.fo=~cs(nx), data=newd, family=BCT)
boC <- centiles.boot(m3, xname="nx", xvalues=c(15,20,25,30,35,40,45), data=newd, power=0.5, B=10)
```

```
summary(boC)
#plot(boC)
# the original variables can be added in
#points(y~x,data=abdom)
```

`fitPCR` *Function to fit simple Principal Component Regression.*

Description

This function is a univariate (i.e. one response) version of a principal component regression. It is based on the function `svdpc.fit()` of package **pls** but it has been generalised to take prior weights. It gets a (single) response variable `y` (n x 1) and a matrix of explanatory variables of dimensions n x p and fits different principal component regressions up to principal component M. Note that M can be less or equal to p (if $n > p$) or less or equal to n if $n < p$, that is, when there they are less observations than variables.

The function is used by the GAMLSS additive term function `pcr()` to fit a principal component regression model within `gamlss()`.

Usage

```
fitPCR(x = NULL, y = NULL, weights = rep(1, n),
       M = NULL, df = NULL, supervised = FALSE,
       k = 2, r = 0.2, plot = TRUE)
```

Arguments

`x`	a matrix of explanatory variables
`y`	the response variable
`weights`	prior weights
`M`	if set specifies the maximum number of components to be considered
`df`	if set specifies the number of components
`supervised`	whether supervised PCR should be used or not, `default=FALSE`
`k`	the penalty of GAIC
`r`	a correlation value (between zero and one) used smoothing parameter when `supervised=TRUE`
`plot`	Whether to plot the coefficient path

Details

More details here

Value

It returns a object PCR which can be used with methods print(), summary(), plot(), fitted() and coef(). The object has elements:

coefficients	The beta coefficients for 1 to M principal components
scores	the n x M dimensional matrix T o=f scores
loadings	the p x M dimensional matrix P of loadings
gamma	the first M principal component coefficients
se.gamma	the standard errors of the M principal component coefficients
center	the location parameters used to scale the x's
scale	the scale parameters used to scale the x's
fitted.values	matrix of n x M dimensions
Xvar	sum of squares of the scores i.e. diag(T'T)
gaic	The GAIC values
pc	number of PC i.e. which value of GAIC has the minimum
k	which penalty for GAIC
M	the maximum of PC tried
sigma	The estimated sigma from the M fitted components
residuals	The n x M matrix of the residuals
call	the function call

Author(s)

Mikis Stasinopoulos, Robert Rigby and Fernanda De Bastiani.

References

Bjorn-Helge Mevik, Ron Wehrens and Kristian Hovde Liland (2019). pls: Partial Least Squares and Principal Component Regression. R package version 2.7-2. https://CRAN.R-project.org/package=pls

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

Stasinopoulos, M. D., Rigby, R. A., Georgikopoulos N., and De Bastiani F., (2021) Principal component regression in GAMLSS applied to Greek-German government bond yield spreads, *Statistical Modelling* doi:10.1177/1471082X211022980.

(see also https://www.gamlss.com/).

See Also

pc

Examples

```
library(glmnet)
data(QuickStartExample)
attach(QuickStartExample)
hist(y, main="(a)")
if (is.null(rownames(x))) colnames(x) <- paste0("X",
    seq(1:dim(x)[2]))
############################################################
# fitPCR
############################################################
# fitting
registerDoParallel(cores = 2)
MM<- fitPCR(x,y, k=log(100))
stopImplicitCluster()
points(MM$coef[,16]~rep(16,20))
names(MM)
MM
#-----------------------------------------------------------
# plotting
plot(MM)
plot(MM, "gaic")
#-----------------------------------------------------------
print(MM)
#-----------------------------------------------------------
coef(MM)                        # the gammas
coef(MM, param="beta")          # the betas
coef(MM, param="beta", pc=1)  # at position 1
#-----------------------------------------------------------
# plotting y and and fitted balues at different points
plot(y)
points(fitted(MM, pc=3), col=2)
points(fitted(MM, pc=20), col=3)
#-----------------------------------------------------------
# variance covariance
vcov(MM, type="se", pc=1)
vcov(MM, type="se", pc=2)
vcov(MM, type="se", pc=20)
# library(corrplot)
# corrplot(vcov(MM, type="cor", pc=10))
# corrplot(vcov(MM, type="cor", pc=20))
#-----------------------------------------------------------
summary(MM)
summary(MM, param="beta", pc=15)
summary(MM, param ="beta", pc=3)
summary(MM, param ="beta") # at default
#-----------------------------------------------------------
predict(MM, newdata= x[1:5,])
fitted(MM)[1:5]
```

`fitRolling` *Function to Fit Rolling Regression in gamlss*

Description

Rolling regression is common in time series analysis when one step ahead forecasts is required. The function `fitRolling()` works as follows: A model is fitted first to the whole data set using `gamlss()`. Then the function `fitRolling()` can be used. The function uses a fixed size rolling widow i.e. 365 days. The model is refitted repeatedly for the different windows over time (like a local regression in smoothing). Each time one step ahead forecast of distribution parameters are saved together with the prediction global deviance. The result is presented as a matrix with time as rows and parameters and the prediction deviance as columns.

Usage

```
fitRolling(obj, data, window = 365, as.time = NULL)
```

Arguments

`obj`	a gamlss fitted model
`data`	the original data of the fitted model
`window`	the number of observation to include in the window (typically this will be a year)
`as.time`	if a column indicating time exist in the data set this can be specified here

Details

If the total observations are `N` and the window size `n` then we will need `N-n` different fits. The parallelization of the fits is achieved using the function `foreach()` from the package **foreach**.

Value

Returns a matrix containing as columns the one ahead prediction parameters of the distribution as well as the prediction global deviance.

Note

Do not forget to use `registerDoParallel(cores = NUMBER)` or `cl <- makeCluster(NUMBER)` and `registerDoParallel(cl)` before calling the function `fitRolling()` and `closeAllConnections()` after the fits. Where `NUMBER` depends on the machine used.

Author(s)

Mikis Stasinopoulos, <d.stasinopoulos@londonmet.ac.uk>

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

(see also https://www.gamlss.com/).

See Also

gamlss

Examples

```
# fitting the aids data 45 observations
m1 <- gamlss(formula = y ~ pb(x) + qrt, family = NBI, data = aids)
# get rolling regression with a window of 30
# there are 45-40=15 fits to do
# declaring cores (not needed for small data like this)
registerDoParallel(cores = 2)
FF <- fitRolling(m1, data=aids, window=30)
FF
stopImplicitCluster()
# check the first prediction
m30_1 <-update(m1, data=aids[1:30,])
predictAll(m30_1, newdata=aids[31,],output="matrix")
FF[1,]
# plot all the data
plot(y~x, data=aids, xlim=c(0,45), ylim=c(0, 700), col=gray(.8))
# the first 30 observations
points(y~x, data=aids[1:30,], xlim=c(0,45))
# One step ahead forecasts
lines(FF[,"mu"]~as.numeric(rownames(FF)), col="red")
lines(fitted(m1)~aids$x, col="blue")
```

`fitted.PCR` *Methods for PCR objects*

Description

The functions below are methods for PCR objects

Usage

```
## S3 method for class 'PCR'
fitted(object, pc = object$pc, ...)
## S3 method for class 'PCR'
plot(x, type = c("path", "gaic"),
                   labels = TRUE, cex = 0.8, ...)
## S3 method for class 'PCR'
coef(object, param = c("gamma", "beta"),
              pc = object$pc, ...)
## S3 method for class 'PCR'
predict(object, newdata = NULL,
                pc = object$pc, ...)
## S3 method for class 'PCR'
vcov(object, pc = object$pc,
        type = c("vcov", "cor", "se", "all"),
                       ...)
```

Arguments

`object, x`	an PCR object
`pc`	the number of PC (by default the one minimising the local GAIC)
`type`	for `plot()` whether to plot the path of coefficients or the path of GAIC and for `vcov()` whether variance covariance, correlation or se's
`param`	getting the gamma or the beta coefficients
`newdata`	new data for prediction
`labels`	whether to plot the labels of the variables
`cex`	the size of the text when plotting the labels of the variables
`...`	for extra arguments

Author(s)

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby and Fernada De Bastiani

References

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, https://www.jstatsoft.org/v23/i07/.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC.

(see also https://www.gamlss.com/).

See Also

fitPCR

---

pc *Functions to Fit Principal Component Regression in GAMLSS*

Description

The functions `pcr()` and `pc()` can be use to fit principal component regression (PCR) within a GAMLSS model. They can be used as an extra additive term (together with other additive terms for example `pb()`) but the idea is mainly to be used on their own as a way of reducing the dimensionality of the the (scaled) x-variables. The functions can be used even when the number of the explanatory variables say `p` is greater than the number of observations `n`.

The two functions differ on the way PCR is implemented within the GAMLSS algorithm see for example Stasinopoulos et.al (2021). In the function `pc()` the singular value decomposition of the scaled x's is done in the beginning and different re-weighted linear models are fitted on the PC scores see algorithm 1 in Stasinopoulos et al. (2021). In the function `pcr()` at each iteration a new weighted PCR is performed using the function `fitPCR()` see algorithm 2 in Stasinopoulos et al. (2021).

The functions `gamlss.pcr()` and `gamlss.pc()` are supporting functions. The are not intended to be called directly by users. The function `gamlss.pc()` is using the linear model function `lm()` to fit the first principal components while the function codegamlss.pcr() uses `fitPCR()`.

The function `getSVD()` creates a singular value decomposition of a design matrix `X` using the R function `La.svd()`.

Usage

```
pc(x.vars = NULL, x = NULL, x.svd = NULL, df = NULL,
   center = TRUE, scale = TRUE, tol = NULL,
   max.number = min(p, n), k = log(n),
```

```
   method = c( "t-values","GAIC","k-fold"))

pcr(x.vars = NULL, x = NULL, df = NULL,
    M = min(p, n), k = log(n),
    r = 0.2, method = c("GAIC", "t-values", "SPCR"))

gamlss.pc(x, y, w, xeval = NULL, ...)

gamlss.pcr(x, y, w, xeval = NULL, ...)

getSVD(x = NULL, nu = min(n, p), nv = min(n, p))
```

Arguments

x.vars	A character vector showing the names of the x-variables. The variables should exist in the original data argument declared in the gamlss() function
x	For the function pc() and getSVD() x is a design matrix of dimensions n x p contain all the explanatory variables terms. For the function gamlss.pc(), x is a vector of zeros which curries all in information needed for the principal components fits in its attributes
x.svd	A list created by the function getSVD(). This will speed up the time of fitting, (especial for large data sets), since all the principal components calculation are done in advance. Also if all the parameters of the distribution are modelled by principal components the calculation needed to be done only once.
df	(if is not NULL) the number of principal components to be fitted. If it is NULL the number of principal components is automatically calculated using a GAIC criterion.
center	whether to center the explanatory variables with default TRUE
scale	whether to scale the explanatory variables with default TRUE
r	the cut point for correlation coefficient to be use SPCR
tol	CHECK THIS?????
max.number, M	The maximum number of principal component to be used in the fit.
method	method used for choosing the number of components
k	the penalty for GAIC
y	the iterative response variable
w	the iterative weights
xeval	used in prediction
...	for extra arguments
nu	the number of left singular vectors to be computed. This must between 0 and n = nrow(x).
nv	the number of right singular vectors to be computed. This must be between 0 and p = ncol(x).

Details

There are three different ways of declaring the list of x-variables (two for the function `pcr()`):

x.vars: this should be a character vector having the names of the explanatory variables. The names should be contained in the names of variables of the `data` argument of the function `gamlss()`, see example below.

x: This should be a design matrix (preferable unscaled since this could create problems when try to predict), see examples.

x.svd: This should be a list created by the function `getSVD()` which is used as an argument a design matrix, see examples.

Value

For the function `pc()` returns an object `pc` with elements `"coef"`, `"beta"`, `"pc"`, `"edf"`, `"AIC"`. The object `pc` has methods `plot()`, `coef()` and `print()`.

For the function `pcr()` returns an object `PCR` see for the help for function `fitPCR`.

Note

Do not forget to use `registerDoParallel(cores = NUMBER)` or `cl <- makeCluster(NUMBER)` and `registerDoParallel(cl)` before calling the function `pc()` without specifying the degrees of freedom. Use `closeAllConnections()` after the fits to close the connections. The `NUMBER` depends on the machine used.

Author(s)

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby

References

Bjorn-Helge Mevik, Ron Wehrens and Kristian Hovde Liland (2019). pls: Partial Least Squares and Principal Component Regression. R package version 2.7-2. https://CRAN.R-project.org/package=pls

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

Stasinopoulos, M. D., Rigby, R. A., Georgikopoulos N., and De Bastiani F., (2021) Principal component regression in GAMLSS applied to Greek-German government bond yield spreads, *Statistical Modelling* doi:10.1177/1471082X211022980.

(see also https://www.gamlss.com/).

See Also

centiles.boot, fitRolling

Examples

```
# the pc() function
# fitting the same model using different arguments
# using x.vars
p1 <- gamlss(y~pc(x.vars=c("x1","x2","x3","x4","x5","x6")), data=usair)
registerDoParallel(cores = 2)
t1 <- gamlss(y~pcr(x.vars=c("x1","x2","x3","x4","x5","x6")), data=usair)
# using x
X <- model.matrix(~x1+x2+x3+x4+x5+x6, data=usair)[,-1]
p2 <- gamlss(y~pc(x=X), data=usair)
t2 <- gamlss(y~pcr(x=X), data=usair)
# using x.svd
svdX <- getSVD(X)
p3 <- gamlss(y~pc(x.svd=svdX), data=usair)
# selecting the componets
p3 <- gamlss(y~pc(x.svd=svdX, df=3), data=usair)
stopImplicitCluster()
plot(getSmo(t2))
plot(getSmo(t2), "gaic")
```

---

which.Data.Corr *Detecting Hight Pair-Wise Correlations in Data*

Description

There are two function here.

The function `which.Data.Corr()` is taking as an argument a `data.frame` or a data matrix and it reports the pairs of variables which have higher correlation than `r`.

The function `which.yX.Corr()` it takes as arguments a continuous response variable, `y`, and a set of continuous explanatory variables, `x`, (which may include first order interactions), and it creates a `data.frame` containing all variables with a pair-wise correlation above `r`. If the set of the continuous explanatory variables contains first order interactions, then by default, (`hierarchical = TRUE`), the main effects of the first order interactions will be also included so hierarchy will be preserved.

Usage

```
which.Data.Corr(data, r = 0.9, digits=3)
which.yX.Corr(y, x, r = 0.5, plot = TRUE,
              hierarchical = TRUE, print = TRUE, digits=3)
```

Arguments

data	A data.frame or a matrix
r	a correlation values (acting as a lower limit)
y	the response variable (continuous)
x	the (continuous) explanatory variables
plot	whether to plot the results or not
print	whether to print the dim of the new matrix or not
hierarchical	This is designed for make sure that if first order interactions are included in the list the main effects will be also included
digits	the number of digits to print.

Value

The function which.Data.Corr() creates a matrix with three columns. The first two columns contain the names of the variables having pair-wise correlation higher than r and the third column show their correlation.

The function which.yX.Corr() creats a design matrix which containts variables which have

Author(s)

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby and Fernada De Bastiani.

References

Bjorn-Helge Mevik, Ron Wehrens and Kristian Hovde Liland (2019). pls: Partial Least Squares and Principal Component Regression. R package version 2.7-2. https://CRAN.R-project.org/package=pls

Rigby, R. A. and Stasinopoulos D. M. (2005). Generalized additive models for location, scale and shape, (with discussion), *Appl. Statist.*, **54**, part 3, pp 507-554.

Rigby, R. A., Stasinopoulos, D. M., Heller, G. Z., and De Bastiani, F. (2019) *Distributions for modeling location, scale, and shape: Using GAMLSS in R*, Chapman and Hall/CRC, doi:10.1201/9780429298547. An older version can be found in https://www.gamlss.com/.

Stasinopoulos D. M. Rigby R.A. (2007) Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, Vol. **23**, Issue 7, Dec 2007, doi:10.18637/jss.v023.i07.

Stasinopoulos D. M., Rigby R.A., Heller G., Voudouris V., and De Bastiani F., (2017) *Flexible Regression and Smoothing: Using GAMLSS in R*, Chapman and Hall/CRC. doi:10.1201/b21973

Stasinopoulos, M. D., Rigby, R. A., and De Bastiani F., (2018) GAMLSS: a distributional regression approach, *Statistical Modelling*, Vol. **18**, pp, 248-273, SAGE Publications Sage India: New Delhi, India. doi:10.1177/1471082X18759144

Stasinopoulos, M. D., Rigby, R. A., Georgikopoulos N., and De Bastiani F., (2021) Principal component regression in GAMLSS applied to Greek-German government bond yield spreads, *Statistical Modelling* doi:10.1177/1471082X211022980.

(see also https://www.gamlss.com/). .

See Also

pc

Examples

```
data(oil, package="gamlss.data")
dim(oil)
# which variables are highly correlated?
CC<- which.Data.Corr(oil, r=0.999)
head(CC)
# 6 of them
# get the explanatory variables
form1 <- as.formula(paste("OILPRICE ~ ",
          paste(names(oil)[-1],collapse='+')))
# no interactions
X <- model.matrix(form1, data=oil)[,-1]
dim(X)
sX <- which.yX.Corr(oil$OILPRICE,x=X, r=0.4)
dim(sX)
# first order interactions
form2 <- as.formula(paste("OILPRICE ~ ",
        paste0(paste0("(",paste(names(oil)[-1],
        collapse='+')), ")^2")))
form2
XX <- model.matrix(form2, data=oil)[,-1]
dim(XX)
which.yX.Corr(oil$OILPRICE,x=XX, r=0.4)
```

Index