Package ‘gamlss.add’

October 13, 2022

Description
Interface for extra smooth functions including tensor products, neural networks and decision trees.

Title Extra Additive Terms for Generalized Additive Models for Location Scale and Shape

LazyLoad yes
Version 5.1-6
Date 2020-02-03
Depends R (>= 2.15.0), gamlss.dist, gamlss (>= 2.4.0), mgcv, nnet, rpart, graphics, stats, utils, grDevices, methods
Suggests lattice
Author Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby, Vlassios Voudouris, Daniil Kiose
Maintainer Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>
License GPL-2 | GPL-3
URL http://www.gamlss.org/
NeedsCompilation no
Repository CRAN
Date/Publication 2020-02-03 17:10:02 UTC

R topics documented:

- gamlss.add-package ........................................... 2
- centilesTwo ....................................................... 3
- fitFixedKnots .................................................... 5
- fk ................................................................. 8
- ga ............................................................... 10
- gamlss_fk ....................................................... 14
- gamlss_ga ....................................................... 16
- gamlss_mn ....................................................... 17
- nn ............................................................... 18
- plot.nnet ......................................................... 20
- tr ............................................................... 23
gamlss.add-package

Index

gamlss.add-package  Extra Additive Terms for Generalized Additive Models for Location Scale and Shape

Description

Interface for extra smooth functions including tensor products, neural networks and decision trees.

Details

The DESCRIPTION file:

Package:  gamlss.add
Description:  Interface for extra smooth functions including tensor products, neural networks and decision trees.
Title:  Extra Additive Terms for Generalized Additive Models for Location Scale and Shape
LazyLoad:  yes
Version:  5.1-6
Date:  2020-02-03
Depends:  R (>= 2.15.0), gamlss.dist, gamlss (>= 2.4.0), mgcv, nnet, rpart, graphics, stats, utils, grDevices, methods
Suggests:  lattice
Author:  Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby, Vlasios Voudouris, Daniil Kiose
Maintainer:  Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>
License:  GPL-2 | GPL-3
URL:  http://www.gamlss.org/

Index of help topics:

centilesTwo  Centiles contour plots in GAMLSS
fitFixedKnots  Functions to Fit Univariate Break Point Regression Models
fk  A function to fit break points within GAMLSS
ga  A interface functions to use Simon Wood's gam() and bam() functions within GAMLSS
gamlss.add-package  Extra Additive Terms for Generalized Additive Models for Location Scale and Shape
gamlss.fk  Support for Function fk()
gamlss.ga  Support for Function ga() and ba()
gamlss.nn  Support for Function nn()
nn  A interface function to use nnet() function within GAMLSS
plot.nnet  Plotting fitted neural networks
tr  A interface function to use rpart() function within GAMLSS
centilesTwo

Author(s)
Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby, Vlasios Voudouris, Daniil Kiose
Maintainer: Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>

References
(see also [http://www.gamlss.com/](http://www.gamlss.com/)).

See Also
- [gamlss](#), [gamlss.family](#)

Examples
```r
library(gamlss)
gn <- gamlss(R~ga(~te(Fl,A)), data=rent, family=GA)
```

---

centilesTwo

Centiles contour plots in GAMLSS

Description
This function centilesTwo() plots two dimensional centiles contour plots for GAMLSS models.

Usage
```r
centilesTwo(x, grid.x1, grid.x2, x1.name, x2.name,
    cent = 0.05, dist = 0.01, points = TRUE,
    other = list(), point.col = 1, point.pch = ",.",
    image = FALSE, image.col = heat.colors(12), ...)
```
Arguments

- **object**: an gamlss object
- **grid.x1**: grid values for x-variable one
- **grid.x2**: grid values for x-variable two
- **x1.name**: the name of x-variable one
- **x2.name**: the name of x-variable two
- **cent**: the required centiles
- **dist**: the distance
- **points**: whether to plot the data points
- **other**: a list having other explanatory variables at fixed values
- **point.col**: the colour of the data points
- **point.pch**: the type of the data point
- **image**: whether to plot using the image9 function
- **image.col**: the colour scheme
- **...**: for extra arguments for the contour() function

Details

The function uses the function exclude.too.far() of the package mgcv.

Value

Produce a contour plot.

Author(s)

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

References


(see also [http://www.gamlss.com/](http://www.gamlss.com/)).

fitFixedKnots

Functions to Fit Univariate Break Point Regression Models

Description

There are two main functions here. The functions fitFixedKnots allows the fit a univariate regression using piecewise polynomials with "known" break points while the function fitFreeKnots estimates the break points.

Usage

fitFixedKnots(y, x, weights = NULL, knots = NULL, data = NULL, degree = 3, fixed = NULL, base=c("trun","Bbase"), ...)  
fitFreeKnots(y, x, weights = NULL, knots = NULL, degree = 3, fixed = NULL, trace = 0, data = NULL, base=c("trun","Bbase"), ...)  

Arguments

x       the x variable (explanatory)  
y       the response variable  
weights       the prior weights  
knots       the position of the interior knots for fitFixedKnots or starting values for fitFreeKnots  
data       the data frame  
degree       the degree if the piecewise polynomials  
fixed       this is to be able to fit fixed break points  
base       The basis for the piecewise polynomials, turn for truncated (default) and Bbase for B-base piecewise polynomials  
trace       controlling the trace of of optim()  
...       for extra arguments

See Also

centiles

Examples

### Not run:  
data(plasma)  
m1 <- gamlss(betadiet ~ ga(~te(age, fiber)), sigma.formula = ~1,  
    nu.formula = ~ga(~te(age, fiber)), tau.formula = ~1,  
    family = BCTo, data = plasma)  
centilesTwo(m1, 18:90, seq(2.5,38, 0.5), age, fiber, cent=0.05, dist=.1,  
    xlab="age", ylab="fiber")  
centilesTwo(m1, 18:90, seq(2.5,38, 0.5), age, fiber, cent=0.95, dist=.1)

### End(Not run)
Details

The functions `fitFreeKnots()` is loosely based on the `curfit.free.knot()` function of package `DierckxSpline` of Sundar Dorai-Raj and Spencer Graves.

Value

The functions `fitFixedKnots` and `fitFreeKnots` return an object `FixBreakPointsReg` and `FreeBreakPointsReg` respectively with the following items:

- `fitted.values`: the fitted values of the model
- `residuals`: the residuals of the model
- `df`: the degrees of freedom fitted in the model
- `rss`: the residuals sum of squares
- `knots`: the knots used in creating the beta-function base
- `fixed`: the fixed break points if any
- `breakPoints`: the interior (estimated) break points (or knots)
- `coef`: the coefficients of the linear part of the model
- `degree`: the degree of the piecewise polynomial
- `y`: the y variable
- `x`: the x variable
- `w`: the prior weights

Note

The prediction function in piecewise polynomials using the B-spline basis is tricky because by adding the newdata for x to the current one the B-basis function for the piecewise polynomials changes. This does not seems to be the case with the truncated basis, that is, when the option `base="turn"` is used (see the example below).

If the newdata are outside the range of the old x then there could a considerable discrepancies between the all fitted values and the predicted ones if the option `base="Bbase"` is used. The prediction function for the objects `FixBreakPointsReg` or `FreeBreakPointsReg` has the option `old.x.range=TRUE` which allow the user two choices:

The first is to use the old end-points for the creation of the new B-basis which were determinate from the original range of x. This choice is implemented as a default in the predict method for `FixBreakPointsReg` and `FreeBreakPointsReg` objects with the argument `old.x.range=TRUE`.

The second is to create new end-points from the new and old data x values. In this case the range of x will be bigger that the original one if the newdata has values outside the original x range. In this case (`old.x.range=FALSE`) the prediction could be possible better outside the x range but would not coincide with the original predictions i.e. `fitted(model)` since basis have changed.

Author(s)

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


(see also [http://www.gamlss.com/](http://www.gamlss.com/)).

Examples

```r
# creating a linear + linear function
x <- seq(0,10, length.out=201)
knot <- 5
set.seed(12543)
mu <- ifelse(x<=knot,5+0.5*x,5+0.5*x+(x-knot))
y <- rNO(201, mu=mu, sigma=.5)
# plot the data
plot(y~x, xlim=c(-1,13), ylim=c(3,18))
# fit model using fixed break points
m1 <- fitFixedKnots(y, x, knots=5, degree=1)
knots(m1)
lines(fitted(m1)~x, col="red")
# now estimating the knot
m2 <- fitFreeKnots(y, x, knots=5, degree=1)
knots(m2)
summary(m2)
# now predicting
plot(y~x, xlim=c(-5,13), ylim=c(3,18))
lines(fitted(m2)~x, col="green", lwd=3)
points(-2:13,predict(m2, newdata=-2:13), col="red",pch = 21, bg="blue")
points(-2:13,predict(m2, newdata=-2:13, old.x.range=FALSE), col="red",pch = 21, bg="grey")
# fit different basis
m21 <- fitFreeKnots(y, x, knots=5, degree=1, base="Bbase")
deviance(m2)
deviance(m21) # should be identical
# predicting with m21
plot(y~x, xlim=c(-5,13), ylim=c(3,18))
lines(fitted(m21)~x, col="green", lwd=3)
points(-2:13,predict(m21, newdata=-2:13), col="red",pch = 21, bg="blue")
points(-2:13,predict(m21, newdata=-2:13, old.x.range=FALSE), col="red",pch = 21, bg="grey")
```
fk

Description

The fk() function is an additive function to be used for GAMLSS models. It is an interface for the fitFreeKnots() function of package gamlss.util. The functions fitFreeKnots() was first based on the curfit.free.knot() function of package DierckxSpline of Sundar Dorai-Raj and Spencer Graves. The function fk() allows the user to use the free knots function fitFreeKnots() within gamlss. The great advantage of course comes from the fact GAMLSS models provide a variety of distributions and diagnostics.

Usage

fk(x, start=NULL, control=fk.control(...), ...)
fk.control(degree = 1, all.fixed = FALSE, fixed = NULL, base = c("trun", "Bbase"))

Arguments

x the x-variable
start starting values for the breakpoints. If are set the number of break points is also determined by the length of start
control the degree of the spline function fitted
... for extra arguments
degree the degree of the based function
all.fixed whether to fix all parameter
fixed the fixed break points
base Which base should be used

Details

Note that fk itself does no smoothing; it simply sets things up for the function gamlss() which in turn uses the function additive.fit() for backfitting which in turn uses gamlss.fk(). Note that, finding the break points is not a trivial problem and therefore multiple maximum points can occur. More details about the free knot splines can be found in package Dierckx, (1991).

The gamlss algorithm used a modified backfitting in this case, that is, it fits the linear part fist. Note that trying to predict outside the x-range can be dangerous as the example below shows.

Value

The gamlss object saved contains the last fitted object which can be accessed using obj$par.coefSmo where obj is the fitted gamlss object par is the relevant distribution parameter.

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org>, Bob Rigby
## References


### See Also

`gamlss.fk`

### Examples

```r
## creating a linear + linear function
x <- seq(0,10, length.out=201)
knot <- 5
set.seed(12543)
u <- ifelse(x<=knot,5+0.5*x,5+0.5*x+1.5*(x-knot))
y <- rNO(201, mu=mu, sigma=.5)
## plot the data
plot(y~x, xlim=c(-1,13), ylim=c(3,23))
## fit model using curfit
m1 <- fitFreeKnots(y, x, knots=3, degree=1)
knots(m1)
## fitted values
lines(fitted(m1)~x, col="red", lwd="3")
## predict
pm1<-predict(m1, newdata=-1:12)
points(-1:12,pm1, col="red",pch = 21, bg="blue")
```

```r
#------------------------------------------------
## now gamlss
#------------------------------------------------
## now negative binomial data
knot=4
eta1 <- ifelse(x<=knot,0.8+0.08*x,.8+0.08*x+.3*(x-knot))
plot(eta1~x)
set.seed(143)
y <- rNBI(201, mu=exp(eta1), sigma=.1)
da <- data.frame(y=y,x=x)
plot(y~x, data=da)
## getting the break point using profile deviance
m1 <- quote(gamlss(y ~ x+I((x>this)*(x-this)), family=NBI, data=da))
prof.term(m1, min=1, max=9, criterion="GD", start.prev=FALSE)
## now fit the model using fk
g1 <- gamlss(y~fk(x, degree=1, start=c(4)), data=da, family=NBI)
## get the breakpoint
knots(getSmo(g1))
## summary of the gamlss object FreeBreakPointsReg object
getSmo(g1)
```
# plot fitted model
plot(y~x, data=da)
lines(fitted(g1)~x, data=da, col="red")

## the aids data as example where things can go wrong
## using fk()

data(aids)
a1<-gamlss(y~x+f(x, degree=1, start=25)+qrt, data=aids, family=NBI)
knots(getSmo(a1))

# using profile deviance
aids.1 <- quote(gamlss(y ~ x+I((x>this)*(x-this))+qrt,family=NBI,data=aids))
prof.term(aids.1, min=16, max=21, step=.1, start.prev=FALSE)

## The Maximum Likelihood estimator is 18.33231 not 17.37064
## plotting the fit
with(aids, plot(x,y,pch=21,bg=c("red","green3","blue","yellow")[unclass(qrt)]))
lines(fitted(a1)~aids$x)

---

**ga**

A interface functions to use Simon Wood’s gam() and bam() functions within GAMLSS

### Description

The ga() and ba() functions are additive functions to be used within GAMLSS models. They are interfaces for the gam() and the bam() functions of package mgcv of Simon Wood. The functions gam() and the bam() allows the user to use all the available smoothers of the package mgcv() within gamlss. The great advantage of course come from fitting models outside the exponential family.

### Usage

```r
ga(formula, control = ga.control(...), ...)

ba(formula, control = ba.control(...), ...)
```

**ga.control**

- `offset = NULL`
- `method = "REML"`
- `optimizer = c("outer", "newton")`
- `control = list()`
- `scale = 0`
- `select = FALSE`
- `knots = NULL`
- `sp = NULL`
- `min.sp = NULL`
- `H = NULL`
- `gamma = 1`
- `paraPen = NULL`
- `in.out = NULL`
- `drop.unused.levels = TRUE`
- `drop.intercept = NULL`
- `discrete = FALSE`

**ba.control**

- `offset = NULL`
- `method = "fREML"`
- `control = list()`
- `select = FALSE`
- `scale = 0`
- `gamma = 1`
- `knots = NULL`
- `sp = NULL`
- `min.sp = NULL`
- `paraPen = NULL`
- `chunk.size = 10000`
- `rho = 0`
- `AR.start = NULL`
- `discrete = TRUE`
- `cluster = NULL`
- `nthreads = 2`
Arguments

formula  A formula containing s() and te functions i.e. ~s(x1)+ te(x2,x3).
offset  the offset in the formula
method  the method argument in gam() and bam()
optimizer  the method optimizer in gam()
control  values for the gam.control()
scale  for the scale parameter
select  the select argument in gam() and bam()
knots  the knots argument in gam() and bam()
sp  the sp argument in gam() and bam()
min.sp  the min.sp argument in gam() and bam()
H  a user supplied fixed quadratic penalty on the parameters in gam()
gamma  the gamma argument in gam() and bam()
paraPen  the paraPen argument in gam() and bam()
in.out  the in.out argument in gam()
drop.unused.levels  by default unused levels are dropped from factors before fitting for gam() and bam()
drop.intercept  set to TRUE to force the model to really not have the a constant in the parametric model part for gam() and bam()
discrete  see bam and gam for details
chunk.size  see the help for bam().
rho  for an AR1 error model, see the help for bam()
AR.start  for an AR1 error model, see the help for bam()
cluster  see the help for bam()
threads  Number of threads to use for non-cluster computation see the help for bam()
gc.level  keeping the memory footprint down, see the help for bam()
use.chol  see the help for bam()
samfrac  see the help for bam()
coef  initial values for model coefficients
...  extra options to pass to gam.control()
Details

Note that ga itself does no smoothing; it simply sets things up for the function gamlss() which in turn uses the function additive.fit() for back-fitting which in turn uses gamlss.ga()

Note that, in our (limited) experience, for normal errors or exponential family, the fitted models using gam() and ga() within gamlss() are identical or at least very similar. This is particularly true if the default values for gam() are used.

Value

the fitted values of the smoother is returned, endowed with a number of attributes. The smoother fitted values are used in the construction of the overall fitted values of the particular distribution parameter. The attributes can be use to obtain information about the individual fit. In particular the coefSmo within the parameters of the fitted model contains the final additive fit.

Warning

The function is experimental so please report any peculiar behaviour to the authors

Author(s)

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

References


(see also [http://www.gamlss.com/](http://www.gamlss.com/)).


Examples

```r
library(mgcv)
data(rent)
#---------------------------------------------------------
## normal errors one x-variable
# normal errors one x-variable
ga1 <- gam(R~s(Fl, bs="ps", k=20), data=rent, method="REML")
gn1 <- gamlss(R~ga(~s(Fl, bs="ps", k=20), method="REML"), data=rent) # additive
gb1 <- gamlss(R~pb(Fl), data=rent) # additive
AIC(ga1,gn1, gb1, k=0)
AIC(ga1,gn1, gb1)
```
```r
# normal error additive in Fl and A
ga2 <- gam(R~s(Fl)+s(A), method="REML", data=rent)
gn2 <- gamlss(R~ga(~s(Fl)+s(A), method="REML"), data=rent) # additive
gb2 <- gamlss(R~pb(Fl)+pb(A), data=rent) # additive
AIC(ga2,gn2, gb2, k=0)
AIC(ga2,gn2, gb2)

# Not run:
# gamma error additive in Fl and A
ga3 <- gam(R~s(Fl)+s(A), method="REML", data=rent, family=Gamma(log))
gn3 <- gamlss(R~ga(~s(Fl)+s(A), method="REML"), data=rent, family=GA) # additive
gb3 <- gamlss(R~pb(Fl)+pb(A), data=rent, family=GA) # additive
AIC(ga3,gn3, gb3, k=0)
AIC(ga3,gn3, gb3)

# gamma error surface fitting
ga4 <- gam(R~s(Fl,A), method="REML", data=rent, family=Gamma(log))
gn4 <- gamlss(R~ga(~s(Fl,A), method="REML"), data=rent, family=GA)
AIC(ga4,gn4, k=0)
AIC(ga4,gn4)

# plot the fitted surfaces
op<-par(mfrow=c(1,2))
vis.gam(ga4)
vis.gam(getSmo(gn4))
par(op)

# contour plot using mgcv's plot() function
plot(getSmo(gn4))

# predict
newrent <- data.frame(expand.grid(Fl=seq(30,120,5), A=seq(1890,1990,5 )))
newrent1 <-newrent2 <- newrent
newrent1$pred <- predict(ga4, newdata=newrent, type="response")
newrent2$pred <- predict(gn4, newdata=newrent, type="response")
library(lattice)
wf1<wireframe(pred~Fl*A, newrent1, aspect=c(1,0.5), drape=TRUE,
               colorkey=list(space="right", height=0.6), main="gam()")
wf2<wireframe(pred~Fl*A, newrent2, aspect=c(1,0.5), drape=TRUE,
               colorkey=list(space="right", height=0.6), main="gamlss()")
print(wf1, split=c(1,1,2,1), more=TRUE)
print(wf2, split=c(2,1,2,1))

# gamma error two variables te() function
ga5 <- gam(R~te(Fl,A), data=rent, family=Gamma(log))
gn5 <- gamlss(R~ga(~te(Fl,A)), data=rent, family=GA)
AIC(ga5,gn5)
AIC(ga5,gn5, k=0)
op<-par(mfrow=c(1,2))
vis.gam(ga5)
vis.gam(getSmo(gn5))
par(op)

# use of Markov random fields
```
## example from package mgcv of Simon Wood

## Load Columbus Ohio crime data (see ?columb for details and credits)
data(columb)  ## data frame
data(columb.polys) ## district shapes list
xt <- list(polys=columb.polys) ## neighbourhood structure info for MRF

## First a full rank MRF...
b <- gam(crime ~ s(district,bs="mrf",xt=xt),data=columb,method="REML")
bb <- gamlss(crime~ ga(~s(district,bs="mrf",xt=xt), method="REML"), data=columb)
AIC(b,bb, k=0)
op<-par(mfrow=c(2,2))
plot(b,scheme=1)
plot(bb$mu.coefSmo[[1]], scheme=1)

## Compare to reduced rank version...
b <- gam(crime ~ s(district,bs="mrf",k=20,xt=xt),data=columb,method="REML")
bb <- gamlss(crime~ ga(~s(district,bs="mrf",k=20,xt=xt), method="REML")
                        , data=columb)
AIC(b,bb, k=0)
plot(b,scheme=1)
plot(bb$mu.coefSmo[[1]], scheme=1)
par(op)

## An important covariate added...
b <- gam(crime ~ s(district,bs="mrf",k=20,xt=xt)+s(income),
                        data=columb,method="REML")
bb <- gamlss(crime~ ga(~s(district,bs="mrf",k=20,xt=xt)+s(income),
                        method="REML"), data=columb)
bbb <- gamlss(crime~ ga(~s(district,bs="mrf"), method="REML")+pb(income), data=columb)
AIC(b,bb,bbb)

## plotting the fitted models
op<-par(mfrow=c(2,2))
plot(b,scheme=c(0,1))
plot(getSmo(bb), scheme=c(0,1))
par(op)
plot(getSmo(bbb, which=2))

## plot fitted values by district
op<- par(mfrow=c(1,2))
fv <- fitted(b)
names(fv) <- as.character(columb$district)
fv1 <- fitted(bbb)
names(fv1) <- as.character(columb$district)
polys.plot(columb.polys,fv)
polys.plot(columb.polys,fv1)
par(op)

## End(Not run)

## bam
Description

This is support for the functions fk(). It is not intended to be called directly by users. The function gamlss.fk is calling on the R function curfit.free.knot() of Sundar Dorai-Raj

Usage

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

Arguments

x the design matrix
y the response variable
w prior weights
xeval used in prediction
... for extra arguments

Author(s)

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

References


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

See Also

fk
Support for Function ga() and ba()

Description

This is support for the smoother functions ga() and ba() interfaces for Simon Wood’s gam() and bam() functions from package mgcv. It is not intended to be called directly by users.

Usage

gamlss.ga(x, y, w, xeval = NULL, ...)
gamlss.ba(x, y, w, xeval = NULL, ...)

Arguments

x the explanatory variables
y iterative y variable
w iterative weights
xeval if xeval=TRUE then prediction is used
... for extra arguments

Author(s)

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

References


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

Support for Function nn()

Description
This is support for the smoother function nn() an interface for Brian Reply’s nnet() function. It is not intended to be called directly by users.

Usage
```
gamlss.nn(x, y, w, xeval = NULL, ...)```

Arguments
- `x`: the explanatory variables
- `y`: iterative y variable
- `w`: iterative weights
- `xeval`: if xeval=TRUE then prediction is used
- `...`: for extra arguments

Author(s)
Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby

References
  (see also [http://www.gamlss.com/](http://www.gamlss.com/)).

See Also
- fk
**Description**

The `nn()` function is a additive function to be used for GAMLSS models. It is an interface for the `nnet()` function of package `nnet` of Brian Ripley. The function `nn()` allows the user to use neural networks within `gamlss`. The great advantage of course comes from the fact GAMLSS models provide a variety of distributions and diagnostics.

**Usage**

```r
nn(formula, control = nn.control(...), ...)  
nn.control(size = 3, linout = TRUE, entropy = FALSE, softmax = FALSE,  
            censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,  
            maxit = 100, Hess = FALSE, trace = FALSE,  
            MaxNWts = 1000, abstol = 1e-04, reltol = 1e-08)
```

**Arguments**

- `formula`: A formula containing the explanatory variables i.e. ~x1+x2+x3.
- `control`: control to pass the arguments for the `nnet()` function
- `...`: for extra arguments
- `size`: number of units in the hidden layer. Can be zero if there are skip-layer units
- `linout`: switch for linear output units. Default is TRUE, identify link
- `entropy`: switch for entropy (= maximum conditional likelihood) fitting. Default by least-squares.
- `softmax`: switch for softmax (log-linear model) and maximum conditional likelihood fitting. linout, entropy, softmax and censored are mutually exclusive.
- `censored`: A variant on softmax, in which non-zero targets mean possible classes. Thus for softmax a row of (0, 1, 1) means one example each of classes 2 and 3, but for censored it means one example whose class is only known to be 2 or 3.
- `skip`: switch to add skip-layer connections from input to output
- `rang`: Initial random weights on [−rang, rang]. Value about 0.5 unless the inputs are large, in which case it should be chosen so that rang * max(|x|) is about 1
- `decay`: parameter for weight decay. Default 0.
- `maxit`: parameter for weight decay. Default 0.
- `Hess`: If true, the Hessian of the measure of fit at the best set of weights found is returned as component Hessian.
- `trace`: switch for tracing optimization. Default FALSE
- `MaxNWts`: The maximum allowable number of weights. There is no intrinsic limit in the code, but increasing MaxNWts will probably allow fits that are very slow and time-consuming.
abstol  Stop if the fit criterion falls below abstol, indicating an essentially perfect fit.
reltol  Stop if the optimizer is unable to reduce the fit criterion by a factor of at least 1
        - reltol.

Details
Note that, neural networks are over parameterized models and therefor notorious for multiple max-
imum. There is no guarantee that two identical fits will produce identical results.

Value
Note that nn itself does no smoothing; it simply sets things up for the function gamlss() which in
        turn uses the function additive.fit() for backfitting which in turn uses gamlss.nn()

Warning
You may have to fit the model several time to unsure that you obtain a reasonable minimum

Author(s)
Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby based on work of Ven-
ables & Ripley wich also based on work by Kurt Hornik and Albrecht Gebhardt.

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.
Location, Scale and Shape: Using GAMLSS in R, Chapman and Hall/CRC.
org/v23/i07.
Regression and Smoothing: Using GAMLSS in R, Chapman and Hall/CRC. (see also http://www.
gamlss.com/).

Examples
library(nnet)
data(rock)
area1<- with(rock,area/10000)
peri1<- with (rock,peri/10000)
rock1<- with(rock, data.frame(perm, area=area1, peri=peri1, shape))
# fit nnet
r1 <- nnet(log(perm)~area+peri+shape, rock1, size=3, decay=1e-3, linout=TRUE,
        skip=TRUE, max=1000, Hess=TRUE)
summary(r1)
# get gamlss
library(gamlss)
c <- nn.control(size=3, decay=1e-3, linout=TRUE, skip=TRUE, max=1000,
Hess=TRUE)
g1 <- gamlss(log(perm)~nn(~area+peri+shape, size=3, control=c), data=rock1)
summary(g1$mu.coefSmo[[1]])

# predict
Xp <- expand.grid(area=seq(0.1,1.2,0.05), peri=seq(0,0.5, 0.02), shape=0.2)
rocknew <- cbind(Xp, fit=predict(r1, newdata=Xp))
library(lattice)
w1 <- wireframe(fit~area+peri, rocknew, screen=list(z=60, x=-60),
aspect=c(1, 0.5), drape=TRUE, main="nnet()")
rocknew1 <- cbind(Xp, fit=predict(g1, newdata=Xp))
w2 <- wireframe(fit~area+peri, rocknew1, screen=list(z=60, x=-60),
aspect=c(1, 0.5), drape=TRUE, main="nn()")
print(w1, split=c(1,1,2,1), more=TRUE)
print(w2, split=c(2,1,2,1))

# ---------------------------------------------------------------
data(rent)
mr1 <- gamlss(R~nn(~Fl+A, size=5, decay=0.001), data=rent, family=GA)
library(gamlss.add)
m1 <- gamlss(R~ga(~s(Fl,A)), data=rent, family=GA)
AIC(mr1,mg1)
newrent <- data.frame(expand.grid(Fl=seq(30,120,5),
A=seq(1890,1990,5 )))
newrent1$fit <- predict(mr1, newdata=newrent, type="response")##nn
newrent2$fit <- predict(m1, newdata=newrent, type="response")# gam
library(lattice)
w1 <- wireframe(fit~Fl+A, newrent1, aspect=c(1,0.5), drape=TRUE,
colorkey=list(space="right", height=0.6)), main="nn()")
w2 <- wireframe(fit~Fl+A, newrent2, aspect=c(1,0.5), drape=TRUE,
colorkey=list(space="right", height=0.6)), main="ga()")
print(w1, split=c(1,1,2,1), more=TRUE)
print(w2, split=c(2,1,2,1))

# Not run:
data(db)
mb1 <- gamlss(h~nn(~age, size=20, decay=0.001), data=db)
plot(h~age, data=db)
points(fitted(mbd1)-db$age, col="red")
mb2 <- gamlss(h~nn(~age, size=20, decay=0.001), data=db, family=BCT)
plot(h~age, data=db)
points(fitted(mbd2)-db$age, col="red")

# End(Not run)
Description

A function to plot the results of a neural network fit based on the `plotnet()` function of the package `NeuralNetTools`.

Usage

```r
## S3 method for class 'nnet'
## S3 method for class 'nnet'
plot(x, nid = TRUE, all.out = TRUE, all.in = TRUE, bias = TRUE,
wts.only = FALSE, rel.rsc = 5, circle.cex = 5, node.labs = TRUE,
var.labs = TRUE, x.lab = NULL, y.lab = NULL, line.stag = NULL,
struct = NULL, cex.val = 1, alpha.val = 1, circle.col = "lightblue",
pos.col = "black", neg.col = "grey", max.sp = FALSE, ...)
```

Arguments

- `x`: A neural network fitted model.
- `nid`: logical value indicating if neural interpretation diagram is plotted, default is `TRUE`.
- `all.out`: character string indicating names of response variables for which connections are plotted, default all.
- `all.in`: character string indicating names of input variables for which connections are plotted, default all.
- `bias`: logical value indicating if bias nodes and connections are plotted, not applicable for networks from `mlp` function, default `TRUE`.
- `wts.only`: logical value indicating if connections weights are returned rather than a plot, default `FALSE`.
- `rel.rsc`: numeric value indicating maximum width of connection lines, default 5.
- `circle.cex`: numeric value indicating size of nodes, passed to `cex` argument, default 5.
- `node.labs`: logical value indicating if text labels are plotted, default `TRUE`.
- `var.labs`: logical value indicating if variable names are plotted next to nodes, default `TRUE`.
- `x.lab`: character string indicating names for input variables, default from model object.
- `y.lab`: character string indicating names for output variables, default from model object.
- `line.stag`: numeric value that specifies distance of connection weights from nodes.
- `struct`: numeric value of length three indicating network architecture (no nodes for input, hidden, output), required only if `mod.in` is a numeric vector.
- `cex.val`: numeric value indicating size of text labels, default 1.
- `alpha.val`: numeric value (0-1) indicating transparency of connections, default 1.
- `circle.col`: text value indicating colour of nodes default "lightblue".
- `pos.col`: text value indicating colour of the positive connections, default "black".
- `neg.col`: text value indicating colour of the negative connections, default "gray".
- `max.sp`: logical value indication whether the space between nodes in each layers is maximised.
- `...`: for further arguments.
Details

The function `plot.nnet()` is (almost) identical to the function `plot.nnet()` created by Marcus W. Beck it was first published in the web but now is part of the `NeuralNetTools` package in R under the name `plotnet()` Here we modify the function it so it works within the `gamlss.add` package. This involves of borrowing the functions `rescale()`, `zero_range()` and `alpha()` from package `scales`.

Value

The function is producing a plot

Author(s)

Marcus W. Beck <mbafs2012@gmail.com> modified by Mikis Stasinopoulos

References


Hadley Wickham (2014). scales: Scale functions for graphics. R package version 0.4.0. https://cran.r-project.org/package=scales

See Also

`nn`

Examples

```r
r1 <- gamlss(R~nn(~Fl+A+H+loc, size=10, decay=0.2), data=rent, family=GA, gd.tol=1000, n.cyc=5)
getSmo(r1)
plot(getSmo(r1), y.lab=expression(eta[1]))
plot(getSmo(r1), y.lab=expression(g[1](mu)))
## Not run:
r2 <- gamlss(R~nn(~Fl+A+H+loc, size=10, decay=0.2), family=GA, gd.tol=1000, n.cyc=5)
plot(getSmo(r2), y.lab=expression(g[1](mu)))
plot(getSmo(r2, what="sigma"), y.lab=expression(g[2](sigma)));
## End(Not run)
```
Description

The tr() function is a additive function to be used for GAMLSS models. It is an interface for the rpart() function of package rpart. The function tr() allows the user to use regression trees within gamlss. The great advantage of course comes from the fact GAMLSS models provide a variety of distributions and diagnostics. Note that the function gamlss.tr is not used by the user but it needed for the backfitting.

Usage

tr(formula, method = c("rpart"), control = rpart.control(...), ...)  
gamlss.tr(x, y, w, xeval = NULL, ...)

Arguments

  formula  A formula containing the expolanatory variables i.e. ~x1+x2+x3.  
  method    only method "rpart" is supported at the moment  
  control   control here is equivalent to rpart.control() function od package rpart  
  x         object passing informatio to the function  
  y         the iterative y variable  
  w         the iterative weights  
  xeval     whether prediction or not is used  
  ...       additional arguments

Details

Note that, the gamlss fit maybe would not coveredg. Also occasionly the gd.tol argument in gamlss has to be increased. The

Value

Note that tr itself does no smoothing; it simply sets things up for the function gamlss() which in turn uses the function additive.fit() for backfitting which in turn uses gamlss.tr() The result is a rpart object.

Author(s)

Mikis Stasinopoulos <mikis.stasinopoulos@gamlss.org>, Bob Rigby based on work of Therneau and Atkison (2015)
References


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


See Also

See Also as nn

Examples

data(rent)
#--- fitting gamlss+tree Normal
library(rpart)
data(rent)
rg1 <- gamlss(R ~ tr(~A+Fl), data=rent, family=NO)
plot(rg1)
plot(getSmo(rg1))
text(getSmo(rg1))
## Not run:
# fitting Gamma errors
rg2 <- gamlss(R ~ tr(~A+Fl), data=rent, family=GA)
plot(rg2)
plot(getSmo(rg2))
text(getSmo(rg2))
#--- fitting also model in the variance
rg3 <- gamlss(R ~ tr(~A+Fl), sigma.fo=tr(~Fl+A), data=rent, family=GA, gd.tol=100, c.crit=0.1)
plot(rg3)
plot(getSmo(rg3))
text(getSmo(rg3))
plot(getSmo(rg3, what="sigma"))
text(getSmo(rg3, what="sigma"))
## End(Not run)
Index

* package
  gamlss.add-package, 2

* regression
  centilesTwo, 3
  fitFixedKnots, 5
  fk, 8
  ga, 10
  gamlss_fk, 14
  gamlss ga, 16
  gamlss nn, 17
  nn, 18
  plot.nnet, 20
  tr, 23

ba (ga), 10

centiles, 5
centilesTwo, 3

fitFixedKnots, 5
fitFreeKnots (fitFixedKnots), 5
fk, 8, 15, 17

ga, 10

gamlss, 3

gamlss.add (gamlss.add-package), 2

gamlss.add-package, 2

gamlss.ba (gamlss.ga), 16

gamlss.family, 3

gamlss_fk, 9, 14

gamlss ga, 16

gamlss.nn, 17

gamlss.tr (tr), 23

nn, 18, 22, 24

plot.nnet, 20

tr, 23

25