

# Package ‘gamlss.add’

July 2, 2014

**Description** Extra additive terms for GAMLSS models.

**Title** GAMLSS Additive.

**LazyLoad** yes

**Version** 4.3-0

**Date** 2014-06-03

**Depends** R (>= 2.15.0), gamlss (>= 2.4.0), mgcv, nnet, rpart

**Suggests** lattice

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

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

**License** GPL-2 | GPL-3

**URL** <http://www.gamlss.org/>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-06-03 19:43:39

## R topics documented:

blag . . . . .	2
fitFixedKnots . . . . .	4
fk . . . . .	6
ga . . . . .	9
gamlss.fk . . . . .	12
gamlss.ga . . . . .	13
gamlss.nn . . . . .	14
la . . . . .	15
nn . . . . .	17
penLags . . . . .	20
tr . . . . .	23

---

blag	<i>Functions to create lag values</i>
------	---------------------------------------

---

**Description**

The function `blag()` creates a basis for lag values of `x`, (a matrix of lag values of `x`). The function `llag()` creates a list with two components i) a basis matrix and ii) weights to be used as prior weights in any regression analysis. The function `wlag()` can take a "mlags" object (created by `blag()`) or a vector and returns a vector with ones and zeros. This can be used as prior weights in any analysis which uses `blag()`.

**Usage**

```
blag(x, lags = 1, from.lag=0, omit.na = FALSE,
     value = NA, ...)
llag(x, ...)
wlag(x, lags = NULL)
```

**Arguments**

<code>x</code>	For <code>blag()</code> and <code>llag()</code> <code>x</code> is the vector for creating lags. For <code>wlag()</code> <code>x</code> is an <code>mlags</code> object created by <code>blag()</code> .
<code>lags</code>	how many lags are required
<code>from.lag</code>	where the lags are starting from. The default values is zero which indicates that the <code>x</code> is also included as a first column. If you want <code>x</code> not to included in the matrix use <code>from.lag=1</code>
<code>omit.na</code>	if true the first "lag" rows of the resulting matrix are omitted
<code>value</code>	<code>value</code> : what values should be set in the beginning of the lags columns, by default is set to NA
<code>...</code>	additional arguments

**Details**

Those three functions are design for helping a user to fit regression model using lags by generating the appropriate structures. The function `blag()` creates a basis for lag values of `x`. It assumed that time runs from the oldest to the newest observations. That is, the latest observations are the most recent ones. The function `wlag()` take a basis matrix of lags and creates a vector of weights which can be used as a prior weights for any regression type analysis which has the matrix as explanatory variable. The function `llag()` creates a list with the matrix base for lags and the appropriate weights.

**Value**

The function `blag()` returns a "mlags" object (matrix of lag values). The function `llag()` returns a list with components:

<code>matrix</code>	The basis of the lag matrix
<code>weights</code>	The weights vector

The function `wlag()` returns a vector of prior weights having, The vector starts with zeros (as many as the number of lags) and continues with ones.

**Author(s)**

Mikis Stasinopoulos <<d.stasinopoulos@londonmet.ac.uk>>, Bob Rigby <<r.rigby@londonmet.ac.uk>>  
Vlasios Voudouris <<v.voudouris@londonmet.ac.uk>>, Majid Djennad, Paul Eilers.

**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.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).

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, <http://www.jstatsoft.org/v23/i07>.

**See Also**

[penLags](#)

**Examples**

```
library(stats)
y <- arima.sim(500, model=list(ar=c(.4,.3,.1)))
X <- blag(y, lags=5, from.lag=1, value=0)
head(X)
w<-wlag(X)
library(gamlss)
m1<-gamlss(y~X, weights=w )
summary(m1)
plot(y)
lines(fitted(m1)~as.numeric(time(y)), col="blue")
```

---

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

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

### 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:

<code>fitted.values</code>	the fitted values of the model
<code>residuals</code>	the residuals of the model
<code>df</code>	the degrees of freedom fitted in the model
<code>rss</code>	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 seem 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 be 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 allows the user two choices:

The first is to use the old end-points for the creation of the new B-basis which were determined 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 than the original one if the newdata has values outside the original x range. In this case (`old.x.range=FALSE`) the prediction could be possibly better outside the x range but would not coincide with the original predictions i.e. `fitted(model)` since the basis has changed.

### Author(s)

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

### References

- Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications
- 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, <http://www.jstatsoft.org/v23/i07>.
- 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.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>)

## Examples

```

# 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

*A function to fit break points within GAMLSS*


---

## 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 first. 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 <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>

**References**

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).

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, <http://www.jstatsoft.org/v23/i07>.

**See Also**

[gamlss.fk](#)

## Examples

```

## 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+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")
#-----
## 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
n1 <- quote(gamlss(y ~ x+I((x>this)*(x-this)), family=NBI, data=da))
prof.term(n1, 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+fk(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 function to use Simon Wood's gam() function within GAMLSS*

---

## Description

The `ga()` function is a additive function to be used within GAMLSS models. It is an interface for the `gam()` function of package `mgcv` of Simon Wood. The function `ga()` allows the user to use all the available smoothers of `gam()` within `gamLSS`. The great advantage of course come from fitting models outside the exponential family.

## Usage

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

```
ga.control(offset = NULL, method = "GCV.Cp", optimizer = c("outer", "newton"),
  control = list(), select = FALSE, knots = NULL,
  sp = NULL, min.sp = NULL, H = NULL, gamma = 1,
  fit = TRUE, paraPen = NULL, G = NULL, in.out = NULL, ...)
```

## Arguments

formula	A formula containing <code>s()</code> and <code>te</code> functions i.e. <code>~s(x1)+ te(x2,x3)</code> .
control	this allow to specify argument within the function <code>gam()</code> of <code>mgcv</code>
...	arguments used by the <code>gam()</code> function.
offset	the <code>offset</code> argument in <code>gam()</code>
method	the <code>method</code> argument in <code>gam()</code>
optimizer	the <code>method optimizer</code> in <code>gam()</code>
select	the <code>select</code> argument in <code>gam()</code>
knots	the <code>knots</code> argument in <code>gam()</code>
sp	the <code>sp</code> argument in <code>gam()</code>
min.sp	the <code>min.sp</code> argument in <code>gam()</code>
H	the <code>H</code> argument in <code>gam()</code>
gamma	the <code>gamma</code> argument in <code>gam()</code>
fit	the <code>fit</code> argument in <code>gam()</code>
paraPen	the <code>paraPen</code> argument in <code>gam()</code>
G	the <code>G</code> argument in <code>gam()</code>
in.out	the <code>in.out</code> argument in <code>gam()</code>

## 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 backfitting 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

## 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.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.
- Wood S.N. (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC Press.

## Examples

```
library(mgcv)
data(rent)
#-----
## normal errors one x-variable
ga1 <- gam(R~s(F1, bs="ps", k=20), data=rent, method="REML")
gn1 <- gamlss(R~ga(~s(F1, bs="ps", k=20), method="REML"), data=rent) # additive
gb1 <- gamlss(R~pb(F1), data=rent) # additive
AIC(ga1,gn1, gb1, k=0)
AIC(ga1,gn1, gb1)
#-----
## normal error additive in F1 and A
ga2 <- gam(R~s(F1)+s(A), method="REML", data=rent)
```

```

gn2 <- gamlss(R~ga(~s(F1)+s(A), method="REML"), data=rent) # additive
gb2 <- gamlss(R~pb(F1)+pb(A), data=rent) # additive
AIC(ga2,gn2, gb2, k=0)
AIC(ga2,gn2, gb2)
#-----
## Not run:
## gamma error additive in F1 and A
ga3 <- gam(R~s(F1)+s(A), method="REML", data=rent, family=Gamma(log))
gn3 <- gamlss(R~ga(~s(F1)+s(A), method="REML"), data=rent, family=GA)# additive
gb3 <- gamlss(R~pb(F1)+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(F1,A), method="REML", data=rent, family=Gamma(log))
gn4 <- gamlss(R~ga(~s(F1,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(F1=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~F1*A, newrent1, aspect=c(1,0.5), drape=TRUE,
               colorkey=(list(space="right", height=0.6)), main="gam()")
wf2<-wireframe(pred~F1*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(F1,A), data=rent, family=Gamma(log))
gn5 <- gamlss(R~ga(~te(F1,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 ?columbus 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")
## x in gam()
bb <- gamlss(crime~ ga(~s(district,bs="mrf",k=20,xt=xt)+s(income),
        method="REML"), data=columb)
## x in gamlss()
bbb <- gamlss(crime~ ga(~s(district,bs="mrf",k=20,xt=xt),
        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)

```

---

gamlss.fk

*Support for Function fk()*


---

### 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 <r.rigby@londonmet.ac.uk>

**References**

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).

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, <http://www.jstatsoft.org/v23/i07>.

**See Also**

[fk](#)

---

gamlss.ga	<i>Support for Function ga()</i>
-----------	----------------------------------

---

**Description**

This is support for the smoother function `ga()` an interface for Simon Wood's `gam()` function. It is not intended to be called directly by users.

**Usage**

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

**Arguments**

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

**Author(s)**

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@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.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.
- Wood S.N. (2006) *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC Press.

---

gamlss.nn

*Support for Function nn()*

---

**Description**

This is support for the smoother function nn() an interface for Brian Ripley'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 predicion is used
...	for extra arguments

**Author(s)**

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@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.
- Ripley, B. D. (1996) Pattern Recognition and Neural Networks. Cambridge.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.
- Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

**See Also**

[ga](#)

---

la *An Additive term for fitting Penalised Lags within a GAMLSS model*

---

**Description**

The function `la()` can be used for fitting penalised lags for explanatory variables.

**Usage**

```
la(x, control = la.control(...), ...)
la.control(lags = 10, from.lag = 0, df = NULL, lambda = NULL,
           start.lambda = 10, order = 1, plot = FALSE,
           method = c("ML", "GAIC"), k = 2, ...)
gamlss.la(x, y, w, xeval = NULL, ...)
```

**Arguments**

x	the name of the explanatory variable
y	internally evaluated (iterative working variable)
w	internally evaluated (iterative weights)
xeval	internally evaluated no need for specification here
control	a list of a number of control parameters for the fitting function <code>penLags()</code>
lags	the number of lags

<code>from.lag</code>	from which lag value to start, the default is zero which means include the original $x$ in the basis
<code>df</code>	use this if you want to fix the degrees of freedom
<code>lambda</code>	use this if you would like to fix the smoothing parameter
<code>start.lambda</code>	initial starting value for lambda
<code>order</code>	the order of the penalty
<code>plot</code>	whether you would like a plot of the data
<code>method</code>	method of fitting if lambda or df are not specified
<code>k</code>	the penalty used if method "GAIC" is used
<code>...</code>	for further arguments

### Details

The idea of penalised lags is that we use a large amount of lags but we penalised their fitted coefficients and therefore we use few degrees of freedom. The penalty and method of fitting are the same as in the `pb()` function of `gamlss`. This function does not do the fitting this is achieved by the function `gamlss.la()` which uses the function `penLags` for the fitting

### Value

a vector of zeros is returned, endowed with a number of attributes. The vector itself is used in the construction of the model matrix (contributing nothing), while the attributes are needed for the back-fitting algorithms of the additive fit.

### Note

Note that an appropriate prior weight is needed in the `gamlss` fit

### Author(s)

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>, Vlasios Voudouris <<v.voudouris@londonmet.ac.uk>>, Majid Djennad, and Paul Eilers.

### References

- Benjamin M. A., Rigby R. A. and Stasinopoulos D.M. (2003) Generalised Autoregressive Moving Average Models. *J. Am. Statist. Ass.*, 98, 214-223.
- 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.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.



**See Also**[penLags](#)**Examples**

```
## the data
dax <- EuStockMarkets[,"DAX"]
plot(dax)
## using a penalised autoregressive model
w <- wlag(dax, lag=20)
m1 <- gamlss(dax~ la(dax, lags=20, order=1, from.lag=1), weights=w)
lines(fitted(m1)~as.numeric(time(dax)), col=2)
wp(m1, ylim.all=1) # not very good model
## Not run:
## Try modelling the variance
m2 <- gamlss(dax~ la(dax, lags=20, order=1, from.lag=1),
             sigma.fo=~la(dax^2, lags=10, order=1, from.lag=1), weights=w)
wp(m2, ylim.all=1)# maybe the tails
m3 <- gamlss(dax~ la(dax, lags=20, order=1, from.lag=1),
             sigma.fo=~la(dax^2, lags=10, order=1, from.lag=1),
             weights=w, family=TF)
wp(m3, ylim.all=1) # better model
plot(m3, ts=TRUE) # autocorrelation OK
## using FTSE to predict DAX
ftse <- EuStockMarkets[,"FTSE"]
# fitting using penLags
l1 <- penLags(dax, ftse, lags=30, plot=TRUE)
# similar model within gamlss
w <- wlag(ftse, lag=30)
g1 <- gamlss(dax~ la(ftse, lags=30, order=1), weights=w)
lines(fitted(m1)~as.numeric(time(dax)))
op <- par(mfrow=c(2,1))
# plotting the fitted coefficients of the AR terms
plot(coef(l1, "AR"), type="h")
plot(coef(g1$mu.coefSmo[[1]])[-1], type="h")
par(op)
g2 <- gamlss(dax~ la(ftse, lags=30, order=1)+la(dax, lags=20, order=1, from.lag=1) , weights=w)
g3 <- gamlss(dax~ la(ftse, lags=30, order=1)+la(dax, lags=20, order=1, from.lag=1) ,
             sigma.fo=~la(dax^2, lags=10, order=1, from.lag=1), weights=w)

## End(Not run)
```

nn

*A interface function to use nnet() function within GAMLSS***Description**

The `nn()` function is an 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

```
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

<code>formula</code>	A formula containing the explanatory variables i.e. $\sim x_1 + x_2 + x_3$ .
<code>control</code>	control to pass the arguments for the <code>nnet()</code> function
<code>...</code>	for extra arguments
<code>size</code>	number of units in the hidden layer. Can be zero if there are skip-layer units
<code>linout</code>	switch for linear output units. Default is TRUE, identity link
<code>entropy</code>	switch for entropy (= maximum conditional likelihood) fitting. Default by least-squares.
<code>softmax</code>	switch for softmax (log-linear model) and maximum conditional likelihood fitting. <code>linout</code> , <code>entropy</code> , <code>softmax</code> and <code>censored</code> are mutually exclusive.
<code>censored</code>	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.
<code>skip</code>	switch to add skip-layer connections from input to output
<code>rang</code>	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
<code>decay</code>	parameter for weight decay. Default 0.
<code>maxit</code>	parameter for weight decay. Default 0.
<code>Hess</code>	If true, the Hessian of the measure of fit at the best set of weights found is returned as component Hessian.
<code>trace</code>	switch for tracing optimization. Default FALSE
<code>MaxNWts</code>	The maximum allowable number of weights. There is no intrinsic limit in the code, but increasing <code>MaxNWts</code> will probably allow fits that are very slow and time-consuming.
<code>abstol</code>	Stop if the fit criterion falls below <code>abstol</code> , indicating an essentially perfect fit.
<code>reltol</code>	Stop if the optimizer is unable to reduce the fit criterion by a factor of at least 1 - <code>reltol</code> .

### Details

Note that, neural networks are over parameterized models and therefore notorious for multiple maximum. 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 <r.rigby@londonmet.ac.uk> based on work of Venables & 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.
- Ripley, B. D. (1996) Pattern Recognition and Neural Networks. Cambridge.
- Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.
- Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

**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)
cc <- 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=cc), 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)
wf1<-wireframe(fit~area+peri, rocknew, screen=list(z=160, x=-60),
```

```

      aspect=c(1, 0.5), drape=TRUE, main="nnet()")
rocknew1 <- cbind(Xp, fit=predict(g1, newdata=Xp))
wf2<-wireframe(fit~area+peri, rocknew1, screen=list(z=160, x=-60),
      aspect=c(1, 0.5), drape=TRUE, main="nn()")
print(wf1, split=c(1,1,2,1), more=TRUE)
print(wf2, split=c(2,1,2,1))
#-----
data(rent)
mr1 <- gamlss(R~nn(~F1+A, size=5, decay=0.001), data=rent, family=GA)
library(gamlss.add)
mg1<-gamlss(R~ga(~s(F1,A)), data=rent, family=GA)
AIC(mr1,mg1)
newrent <- newrent1 <- newrent2 <- data.frame(expand.grid(F1=seq(30,120,5),
      A=seq(1890,1990,5 )))
newrent1$fit <- predict(mr1, newdata=newrent, type="response") ##nn
newrent2$fit <- predict(mg1, newdata=newrent, type="response")# gam
library(lattice)
wf1<-wireframe(fit~F1+A, newrent1, aspect=c(1,0.5), drape=TRUE,
      colorkey=(list(space="right", height=0.6)), main="nn()")
wf2<-wireframe(fit~F1+A, newrent2, aspect=c(1,0.5), drape=TRUE,
      colorkey=(list(space="right", height=0.6)), main="ga()")
print(wf1, split=c(1,1,2,1), more=TRUE)
print(wf2, split=c(2,1,2,1))
#-----
## Not run:
data(db)
mdb1 <- gamlss(head~nn(~age,size=20, decay=0.001), data=db)
plot(head~age, data=db)
points(fitted(mdb1)~db$age, col="red")

# do not run
#mdb2 <- gamlss(head~nn(~age,size=20, decay=0.001), data=db, family=BCT)
#plot(head~age, data=db)
#points(fitted(mdb2)~db$age, col="red")

## End(Not run)

```

---

penLags

*Penalised Lag Regression Function*


---

## Description

The function `penLags()` fits a regression model to lags of an explanatory variable `x` or to lags of `y` itself. The estimated coefficients of the lags are penalised using a quadratic penalty similar to P-splines.

## Usage

```

penLags(y, x, lags = 10, from.lag=0, weights = NULL, data = NULL, df = NULL,
      lambda = NULL, start.lambda = 10, order = 1,
      plot = FALSE, method = c("ML", "GAIC"), k = 2, ...)

```

**Arguments**

y	The response variable
x	The explanatory variable which can be the response itself if autoregressive model is required
lags	The number of lags required
from.lag	from which lag value to start, the default is zero which means include the original x in the basis
weights	The prior weights
data	The data frame if needed
df	If not NULL this argument sets the required effective degrees of freedom for the penalty
lambda	If not NULL this argument sets the required smoothing parameter of the penalty
start.lambda	Starting values for lambda for the local ML estimation
order	The order of the penalties in the beta coefficients
plot	Whether to plot the data and the fitted values
method	The method of estimating the smoothing parameter with two alternatives, i) ML: the local maximum likelihood estimation method (or PQL method) ii) GAIC: the generalised Akaike criterion method of estimating the smoothing parameter
k	The penalty required if the method GAIC is used i.e. k=2 for AIC or k=log(n) if BIC (or SBC).
...	for further arguments

**Details**

This function is designed for fitting a simple penalised lag regression model to a response variable. The meaning of simple in this case is that only one explanatory variable can be used (whether it is a true explanatory or the response variable itself) and only a normal assumption for the response is made. For multiple explanatory variables and for different distributions within `gamLSS` use the additive function `la`.

**Value**

Returns `penLags` objects which has several methods.

**Author(s)**

Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>, Vlasios Voudouris <<v.voudouris@londonmet.ac.uk>>, Majid Djennad, and Paul Eilers.

**References**

- Benjamin M. A., Rigby R. A. and Stasinopoulos D.M. (2003) Generalised Autoregressive Moving Average Models. *J. Am. Statist. Ass.*, 98, 214-223.
- 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.

Stasinopoulos D. M., Rigby R.A. and Akantziliotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).

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, <http://www.jstatsoft.org/v23/i07>.

## Examples

```
# generating data
y <- arima.sim(500, model=list(ar=c(.9,-.8)))
#-----
#fitting model with different order
m0 <- penLags(y,y, lag=20, order=0)
m1 <- penLags(y,y, lag=20, order=1)
m2 <- penLags(y,y, lag=20, order=2)
m3 <- penLags(y,y, lag=20, order=3)
# choosing the order
AIC(m0, m1, m2, m3)
#-----
# look at the AR coefficients of the models
op <- par(mfrow=c(2,2))
plot(coef(m0,"AR"), type="h")
plot(coef(m1, "AR"), type="h")
plot(coef(m2, "AR"), type="h")
plot(coef(m3,"AR"), type="h")
par(op)
#-----
# refit and plotting model
m1 <- penLags(y,y, lag=20, order=1, plot=TRUE)

# looking at the residuals
plot(resid(m1))
acf(resid(m1))
pacf(resid(m1))
# or better use plot, wp or dtop
plot(m1, ts=TRUE)
wp(m1)
dtop(m1)
# the coefficients
coef(m1)
coef(m1, "AR")
coef(m1, 'varComp')
#
print(m1)
#summary(m1)
# use prediction
plot(ts(c(y, predict(m1,100))))
```

---

tr *A interface function to use rpart() function within GAMLSS*

---

### Description

The `tr()` function is an 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 is needed for the backfitting.

### Usage

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

### Arguments

<code>formula</code>	A formula containing the explanatory variables i.e. $\sim x_1 + x_2 + x_3$ .
<code>method</code>	only method "rpart" is supported at the moment
<code>control</code>	control here is equivalent to <code>rpart.control()</code> function of package <code>rpart</code>
<code>x</code>	object passing information to the function
<code>y</code>	the iterative y variable
<code>w</code>	the iterative weights
<code>xeval</code>	whether prediction or not is used
<code>...</code>	additional arguments

### Details

Note that, the `gamlss` fit maybe would not be covered. Also occasionally 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 <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>  
based on work of Therneau and Atkinson (19)

## References

- Ripley, B. D. (1996) Pattern Recognition and Neural Networks. Cambridge.
- Stasinopoulos D. M., Rigby R.A. and Akantziotou C. (2006) Instructions on how to use the GAMLSS package in R. Accompanying documentation in the current GAMLSS help files, (see also <http://www.gamlss.org/>).
- 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, <http://www.jstatsoft.org/v23/i07>.
- Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S. Fourth edition. Springer.

## See Also

See Also as [nn](#)

## Examples

```
data(rent)
#--- fitting gamlss+tree Normal
library(rpart)
data(rent)
rg1 <- gamlss(R ~ tr(~A+F1), data=rent, family=NO)
plot(rg1)
plot(getSmo(rg1))
text(getSmo(rg1))
## Not run:
# fitting Gamma errors
rg2 <- gamlss(R ~ tr(~A+F1), data=rent, family=GA)
plot(rg2)
plot(getSmo(rg2))
text(getSmo(rg2))
#--- fitting also model in the variance
rg3 <- gamlss(R ~ tr(~A+F1), sigma.fo=~tr(~F1+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

## \*Topic **regression**

- blag, [2](#)
- fitFixedKnots, [4](#)
- fk, [6](#)
- ga, [9](#)
- gamlss.fk, [12](#)
- gamlss.ga, [13](#)
- gamlss.nn, [14](#)
- la, [15](#)
- nn, [17](#)
- penLags, [20](#)
- tr, [23](#)

## \*Topic **ts**

- blag, [2](#)
- la, [15](#)

blag, [2](#)

fitFixedKnots, [4](#)  
fitFreeKnots (fitFixedKnots), [4](#)  
fk, [6](#), [13](#)

ga, [9](#), [15](#)  
gamlss.fk, [7](#), [12](#)  
gamlss.ga, [13](#)  
gamlss.la (la), [15](#)  
gamlss.nn, [14](#)  
gamlss.tr (tr), [23](#)

la, [15](#)  
llag (blag), [2](#)

nn, [17](#), [24](#)

penLags, [3](#), [17](#), [20](#)

tr, [23](#)

wlag (blag), [2](#)