

# Package ‘gamlss.util’

February 15, 2012

**Description** Extra utilities for GAMLSS models.

**Title** GAMLSS Utilities.

**LazyLoad** yes

**Version** 4.1-1

**Date** 2012-02-15

**Depends** R (>= 2.4.0), gamlss, colorspace, Matrix, MASS

**Author** Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob  
Rigby <r.rigby@londonmet.ac.uk>, Paul Eilers <p.eilers@erasmusmc.nl>

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

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

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

**Repository** CRAN

**Date/Publication** 2012-02-15 15:50:58

## R topics documented:

fitFixBP . . . . .	2
Locmean . . . . .	4
penLS . . . . .	7
penReg . . . . .	9
scattersmooth . . . . .	11

<b>Index</b>	<b>14</b>
--------------	-----------

fitFixBP

*Functions to Fit Univariate Break Point Regression Models***Description**

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

**Usage**

```
fitFixBP(x, y, w = NULL, knots = NULL, data = NULL, degree = 3,
         fixed = NULL, base=c("trun","Bbase"), ...)
fitFreeKnots(x, y, w = 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>w</code>	the prior weights
<code>knots</code>	the position of the interior knots for <code>fitFixBP</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 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 `fitFixBP` 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.com/>)

**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 <- fitFixBP(x, y, knots=5, degree=1)
knots(m1)
lines(fitted(m1)~x, col="red")
# estimated the knot
m2 <- fitFreeKnots(x, y, 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(x, y, 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")

```

---

Locmean

*Functions to fit local regression*


---

**Description**

There are four function here to illustrate the fitting of local regressions. i) Locmean, which uses local means within a symmetric local window, ii) Locpoly, which uses a local polynomial fit within a symmetric local window. iii) WLocmean, which uses a Gaussian kernel and iv) WLocpoly, which uses local polynomials weighted by a Gaussian kernel

**Usage**

```

Locmean(y, x = seq(1, length(y)), w = rep(1, length(y)), span = 0.5)
Locpoly(y, x = seq(1, length(y)), w = rep(1, length(y)), span = 0.5, order = 1)
WLocmean(y, x = seq(1, length(y)), w = rep(1, length(y)), lambda = 0.5)
WLocpoly(y, x = seq(1, length(y)), w = rep(1, length(y)), lambda = 0.5, order = 1)

```

**Arguments**

y	the response variable
x	the x-variable
w	prior weights
span	the side of the local window compare as a proportion to the total number of observations
lambda	the smoothing parameter for the Gaussian kernel
order	the order of the polynomial

**Details**

Those functions can be used for illustration of the basic concepts of smoothing using small data sets. Do not use them with large data because are computationally inefficient.

**Value**

The functions return a locW object with values

fitted.values	the fitted values
residuals	the residuals
edf	the effective degrees of freedom
rss	the residual sum of squares
lambda	the smoothing parameter
y	the y variable
x	the x variable
w	the prior weights

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

**See Also**

[loess](#), [ksmooth](#)

**Examples**

```

library(MASS)
data(mcycle)
# local means
m0<-Locmean(mcycle$accel, mcycle$times, span=.1)
m1<-Locmean(mcycle$accel, mcycle$times, span=.2)
m2<-Locmean(mcycle$accel, mcycle$times, span=.3)
span <- c("span=0.1", "span=0.2", "span=0.3")
plot(accel~times, data=mcycle,main="local mean")
lines(fitted(m0)~mcycle$times, col=1, lty=1)
lines(fitted(m1)~mcycle$times, col=2, lty=2)
lines(fitted(m2)~mcycle$times, col=3, lty=3)
legend(1.5,50, legend = span, col = 1:3,
      lty = 1:3, cex = .8, y.intersp = 1)
# kernel estimation
k0<-WLocmean(mcycle$accel, mcycle$times, lambda=1)
k1<-WLocmean(mcycle$accel, mcycle$times, lambda=2)
k2<-WLocmean(mcycle$accel, mcycle$times, lambda=3)
lambda <- c("lambda=1", "lambda=2", "lambda=3")
plot(accel~times, data=mcycle,main="Gaussian kernel fit")
lines(fitted(k0)~mcycle$times, col=1, lty=1)
lines(fitted(k1)~mcycle$times, col=2, lty=2)
lines(fitted(k2)~mcycle$times, col=3, lty=3)
legend(1.5,50, legend = lambda, col = 1:3,
      lty = 1:3, cex = .8, y.intersp = 1)
# local polynomials
l1<-Locpoly(mcycle$accel, mcycle$times, span=.1)
l2<-Locpoly(mcycle$accel, mcycle$times, span=.2)
l3<-Locpoly(mcycle$accel, mcycle$times, span=.3)

span <- c("span=0.1", "span=0.2", "span=0.3")
plot(accel~times, data=mcycle,main="local linear fit")
lines(fitted(l1)~mcycle$times, col=1, lty=1)
lines(fitted(l2)~mcycle$times, col=2, lty=2)
lines(fitted(l3)~mcycle$times, col=3, lty=3)
legend(1.5,50, legend = span, col = 1:3,
      lty = 1:3, cex = .8, y.intersp = 1)
# weighted local polynomials
lw1<-WLocpoly(mcycle$accel, mcycle$times, lambda=1.5, order=1)
lw2<-WLocpoly(mcycle$accel, mcycle$times, lambda=1.5, order=2)
lw3<-WLocpoly(mcycle$accel, mcycle$times, lambda=1.5, order=3)

span <- c("linear", "quadratic", "cubic")
plot(accel~times, data=mcycle,main="Weighted local linear, quadratic and cubic fits")
lines(fitted(lw1)~mcycle$times, col=1, lty=1)
lines(fitted(lw2)~mcycle$times, col=2, lty=2)
lines(fitted(lw3)~mcycle$times, col=3, lty=3)
legend(1.5,50, legend = span, col = 1:3,
      lty = 1:3, cex = .8, y.intersp = 1)

```

penLS

*Function to fit penalised least squares***Description**

The function `penLS()` can be used to fit a penalised least square to a response variable `y`. There is no explanatory variable here. The underline model is a random walk.

**Usage**

```
penLS(y, w = rep(1, length(y)), df = NULL, lambda = NULL,
      start = 10, order = 1, plot = FALSE,
      type = c("level", "trend"),
      method = c("ML", "GAIC", "GCV"), k = 2, ...)
```

**Arguments**

<code>y</code>	the response variable usually a time series
<code>w</code>	prior weights if needed otherwise 1
<code>df</code>	effective degrees of freedom
<code>lambda</code>	the smoothing parameter
<code>start</code>	the lambda starting value if the local methods are use
<code>order</code>	the required difference in the vector of coefficients, see below
<code>plot</code>	whether to plot the data and the fitted function
<code>type</code>	the type of X matrix, if "level" X is a diagonal matrix of 1's if "trend" X is a diagonal matrix 1:n
<code>method</code>	The method used in the estimation of the smoothing parameter
<code>k</code>	the penalty used in "GAIC" and "GCV"
<code>...</code>	for extra arguments

**Details**

The order refers to differences in the penalty matrix, (i) order = 0 : white noise random effects (ii) order = 1 : random walk (iii) order = 2 : random walk of order 2 (iv) order = 3 : random walk of order 3

**Value**

Returns a fitted object of class `penLS`. The object contains 1) the fitted coefficients 2) the `fitted.values` 3) the response variable `y`, 4) the smoothing parameter `lambda`, 8) the effective degrees of freedom `df`, 5) the estimate for sigma `sigma`, 6) the residual sum of squares `rss`, 7) the Akaike information criterion `aic`, 8) the Bayesian information criterion `sbc` and 9) the deviance

**Warning**

The function can be slow for large response variable

**Author(s)**

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

**References**

Eilers, P. H. C. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties (with comments and rejoinder). *Statist. Sci*, **11**, 89-121.

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. (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**

[penReg](#)

**Examples**

```
set.seed(1234)
x<-seq(0,10,length=200); y<-(yt<-1+2*x+.6*x^2-.1*x^3)+rnorm(200,0, 16)
yts <- ts(y)
plot(yts)
#-----
#lambda fix
m1<-penLS(yts,lambda=1) ; deviance(m1)
#-----
# fixing df
m2<-penLS(yts, df=10) ; deviance(m2)
#-----
# estimating lambda - ML
m3<-penLS(yts) ; deviance(m3)
#-----
# estimating lambda - GAIC
m4<-penLS(yts, method="GAIC", k=3) ; deviance(m4)
#-----
# different order
PPP <- par(mfrow=c(2,2))
penLS(yts, plot=TRUE, order=0, main="order=0")
penLS(yts, plot=TRUE, order=1, main="order=1")
penLS(yts, plot=TRUE, order=2, main="order=2")
penLS(yts, plot=TRUE, order=3, main="order=3")
par(PPP)
```

---

penReg *Function to fit penalised regression*

---

### Description

The function `penReg()` can be used to fit a P-spline. It can be used as demonstration of how the penalised B-splines can be fitted to one explanatory variable. For more that explanatory variable use the function `pb()` in **gamlss**.

### Usage

```
penReg(y, x, w = rep(1, length(y)), df = NULL, lambda = NULL, start = 10,
       inter = 20, order = 2, degree = 3, plot = FALSE,
       method = c("ML", "ML-1", "GAIC", "GCV", "EM"), k = 2, ...)
```

### Arguments

<code>y</code>	the response variable
<code>x</code>	the unique explanatory variable
<code>w</code>	prior weights
<code>df</code>	effective degrees of freedom
<code>lambda</code>	the smoothing parameter
<code>start</code>	the lambda starting value if the local methods are used
<code>inter</code>	the no of break points (knots) in the x-axis
<code>order</code>	the required difference in the vector of coefficients
<code>degree</code>	the degree of the piecewise polynomial
<code>plot</code>	whether to plot the data and the fitted function
<code>method</code>	The method used in the (local) performance iterations. Available methods are "ML", "ML-1", "EM", "GAIC" and "GCV"
<code>k</code>	the penalty used in "GAIC" and "GCV"
<code>...</code>	for extra arguments

### Value

Returns a fitted object of class `penReg`. The object contains 1) the fitted coefficients 2) the fitted values 3) the response variable `y`, 4) the label of the response variable `ylabel` 5) the explanatory variable `x`, 6) the label of the explanatory variable 7) the smoothing parameter `lambda`, 8) the effective degrees of freedom `df`, 9) the estimate for sigma `sigma`, 10) the residual sum of squares `rss`, 11) the Akaike information criterion `aic`, 12) the Bayesian information criterion `sbc` and 13) the deviance

### Author(s)

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

## References

- Eilers, P. H. C. and Marx, B. D. (1996). Flexible smoothing with B-splines and penalties (with comments and rejoinder). *Statist. Sci.*, **11**, 89-121.
- 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. (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

[penLS](#)

## Examples

```
set.seed(1234)
x <- seq(0,10,length=200); y<-(yt<-1+2*x+.6*x^2-.1*x^3)+rnorm(200, 4)
library(gamlss)
#-----
# df fixed
g1<-gamlss(y~pb(x, df=4))
m1<-penReg(y,x, df=4)
cbind(g1$mu.coefSmo[[1]]$lambda, m1$lambda)
cbind(g1$mu.df, m1$df)
cbind(g1$aic, m1$aic)
cbind(fitted(g1), fitted(m1))[1:10,]
# identical
#-----
# estimate lambda using ML
g2<-gamlss(y~pb(x))
m2<-penReg(y,x)
cbind(g2$mu.df, m2$df)
cbind(g2$mu.lambda, m2$lambda)
cbind(g2$aic, m2$aic) # different lambda
cbind(fitted(g2), fitted(m2))[1:10,]
# identical
#-----
# estimate lambda using GCV
g3 <- gamlss(y~pb(x, method="GCV"))
m3 <- penReg(y,x, method="GCV")
cbind(g3$mu.df, m3$df)
cbind(g3$mu.lambda, m3$lambda)
cbind(g3$aic, m3$aic)
cbind(fitted(g3), fitted(m3))[1:10,]
# almost identical
#-----
# estimate lambda using EM
g4<-gamlss(y~pb(x, method="EM"))
m4<-penReg(y,x, method="EM")
cbind(g4$mu.df, m4$df )
cbind(g4$mu.lambda, m4$lambda)
```

```

cbind(g4$aic, m4$aic)
cbind(fitted(g4), fitted(m4))[1:10,]
# almost identical
#-----
# estimate lambda using GAIC(#=3)
g5<-gamlss(y~pb(x, method="GAIC", k=3))
m5<-penReg(y,x, method="GAIC", k=3)
cbind(g5$mu.df, m5$df )
cbind(g5$mu.lambda, m5$lambda)
cbind(g5$aic, m5$aic)
cbind(g5$mu.df, m5$df)
cbind(g5$mu.lambda, m5$lambda)
cbind(fitted(g5), fitted(m5))[1:10,]
#-----
plot(y~x)
lines(fitted(m1)~x, col="green")
lines(fitted(m2)~x, col="red")
lines(fitted(m3)~x, col="blue")
lines(fitted(m4)~x, col="yellow")
lines(fitted(m4)~x, col="grey")

```

---

scattersmooth

*Two dimensional Smooth scatter plots*


---

### Description

The function produced two dimensional smooth scatter plots. The method used is described in Eilers and Goeman (2004).

### Usage

```

scattersmooth(x, y, nbin = 100, lambda = 1, ndot = 500,
              csize = 0.3, ticks = TRUE, xlim = c(min(x),
              max(x)), ylim = c(min(y), max(y)), show = TRUE,
              save = FALSE, data = NULL, xlab = NULL,
              ylab = NULL, ...)

```

### Arguments

x	the x-variable
y	the y-variable
nbin	the number of bins required
lambda	the smoothing parameter
ndot	how many data points to show
csize	the size of the data points
ticks	whether ticks in the x and y axis appear in the plot

xlim	the x limit
ylim	the y limit
show	whether to show the graph or not
save	whether to save the output as a list or not
data	the data file data
xlab	the x label as character string
ylab	the y label as character string
...	for extra arguments

### Details

The function is similar to the function `smoothScatter()` in **graphics** but it used penelized bin smoother as described in Eilers and Goeman (2004) rather than kernel smoother.

### Value

the function produces a two dimensional smooth plot and saves if `save=TRUE` a list with the following components:

Hraw	A nbib by nbib matrix containing the bin row data
Hsmooth	A nbib by nbib matrix containing the smooth two dimensional histogram
xgrid	the x-grid
ygrid	the y-grid
xbin	the bin for x values
ybin	the bin for y values
nmiss	number of missing values
seldots	the values of the plotted dots

### Author(s)

Paul Eilers <p.eilers@erasmusmc.nl>

### References

- Eilers, P. H. C. and Goeman, J. J. (2004). Enhancing scatterplots with smoothed density. *Bioinformatics*, Vol **20** no 5, pp 623-628.
- 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. (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

[smoothScatter](#),[gamlss](#)

**Examples**

```
m <- 1000
set.seed(pi)
phi <- 2 * pi * runif(m)
rho <- rchisq(m, df = 6)
x <- cos(phi) * rho
y <- sin(phi) * rho
H <- scattersmooth(x, y)
```

# Index

## \*Topic **regression**

fitFixBP, [2](#)

Locmean, [4](#)

penLS, [7](#)

penReg, [9](#)

scattersmooth, [11](#)

fitFixBP, [2](#)

fitFreeKnots (fitFixBP), [2](#)

gamlss, [12](#)

ksmooth, [5](#)

Locmean, [4](#)

Locpoly (Locmean), [4](#)

loess, [5](#)

penLS, [7](#), [10](#)

penReg, [8](#), [9](#)

scattersmooth, [11](#)

smoothScatter, [12](#)

WLocmean (Locmean), [4](#)

WLocpoly (Locmean), [4](#)