Package ‘gamlss.mx’

May 18, 2016

Type Package
Title Fitting Mixture Distributions with GAMLSS
Version 4.3-5
Date 2016-05-18
Author Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>, Bob Rigby <r.rigby@londonmet.ac.uk>
Maintainer Mikis Stasinopoulos <d.stasinopoulos@londonmet.ac.uk>
LazyLoad yes
Depends R (>= 2.2.1), gamlss.dist, gamlss, nnet, stats, graphics
Suggests MASS
Description The main purpose of this package is to allow fitting of mixture distributions with GAMLSS models.
License GPL-2 | GPL-3
URL http://www.gamlss.org/
NeedsCompilation no
Repository CRAN
Date/Publication 2016-05-18 14:22:30

R topics documented:

  gamlss.mx-package ........................................ 2
dMX ......................................................... 3
enzyme ..................................................... 4
gamlssMX .................................................. 5
gamlssNP .................................................. 7
MX.control ............................................... 11
NP.control ............................................... 12
plotMP ................................................... 13

Index 14
The GAMLSS add on package for mixture distributions

Description

The main purpose of this package is to allow the user of the GAMLSS models to fit mixture distributions.

Details

Package: gamlss.mx
Type: Package
Version: 0.0
Date: 2005-08-3
License: GPL (version 2 or later)

This package has two main function the `gamlssmx()` which is loosely based on the package `flexmix` of R and the function `gamlssNP()` which is based on the `npmlreg` package of Jochen Einbeck, Ross Darnell and John Hinde (2006) which in turns is based on several GLIM4 macros originally written by Murray Aitkin and Brian Francis. It also contains the function `gqz()` which is written by Nick Sofroniou and the function `gauss.quad()` written by Gordon Smyth.

Author(s)

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

References

Jochen Einbeck, Ross Darnell and John Hinde (2006) npmlreg: Nonparametric maximum likelihood estimation for random effect models, R package version 0.34

See Also

gamlss.gamlss.family

Examples

data(enzyme)
mmNO <- gamlssmx(enzyme$act~1, family=NO, K=2)
### Evaluate the d (pdf) and p (cdf) functions from GAMLSS mixtures

**Description**

The functions `dMX()` and `pMX()` can be used to evaluate the pdf (pdf function) and the cdf (cdf function) receptively from a `gamlss.family` mixture. The functions `getpdfMX()` and `getpdfNP()` can be used to evaluate the fitted d function at a specified observation and therefore for plotting the fitted distribution of a fitted model at this observation.

**Usage**

```r
\begin{verbatim}
dMX(y, mu = list(mu1 = 1, mu2 = 5), sigma = list(sigma1 = 1, sigma2 = 1),
    nu = list(nu1 = 1, nu2 = 1), tau = list(tau1 = 1, tau2 = 1),
    pi = list(pi1 = 0.2, pi2 = 0.8), family = list(fam1 = "NO", fam2 = "NO"),
    log = FALSE, ...)
pMX(q, mu = list(mu1 = 1, mu2 = 5), sigma = list(sigma1 = 1, sigma2 = 1),
    nu = list(nu1 = 1, nu2 = 1), tau = list(tau1 = 1, tau2 = 1),
    pi = list(pi1 = 0.2, pi2 = 0.8), family = list(fam1 = "NO", fam2 = "NO"),
    log = FALSE, ...)
getpdfMX(object = NULL, observation = 1)
getpdfNP(object = NULL, observation = 1)
\end{verbatim}
```

**Arguments**

- `y,q`: vector of quantiles
- `mu`: a vector of mu's
- `sigma`: a vector of sigma's
- `nu`: a vector of nu's
- `tau`: a vector of tau's
- `pi`: a vector of pi's
- `family`: a vector of GAMLSS family's
- `log`: whether the log of the function or not
Data used in gamlss.mx

enzyme

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enzyme : The data comprise independent measurement of enzyme activity in the blood of 245 individuals. The data were analysed by Bechker et al. (1993).</td>
</tr>
<tr>
<td>brains : the brain size, brain, and body weight, body, for 28 different animals.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data(enzyme)</td>
</tr>
<tr>
<td>data(brains)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>enzyme : data frame with 245 observations on the following variable act.</td>
</tr>
<tr>
<td>brains : data frame with 28 observations on the following variables. body, brain</td>
</tr>
<tr>
<td>act a numeric vector showing enzyme activity in the blood of 245 individuals.</td>
</tr>
<tr>
<td>body a numeric vector showing the body weight of 28 different animals</td>
</tr>
<tr>
<td>brain a numeric vector showing the brain size of 28 different animals</td>
</tr>
</tbody>
</table>
**gamlssMX**

**Examples**

```r
data(enzyme)
hist(enzyme$act)
data(brains)
brains$lbrain<-log(brains$brain)
brains$lbody<-log(brains$body)
with(brains, plot(lbrain~lbody))
```

---

**gamlssMX**  
*Function to fit finite mixture of gamlss family distributions*

**Description**

The function `gamlssMX` is designed for fitting a K fold non-parametric mixture of gamlss family distributions.

**Usage**

```r
gamlssMX(formula = formula(data), pi.formula = ~1,  
family = "NO", weights, K = 2, prob = NULL,  
data = sys.parent(), control = MX.control(...),  
g.control = gamlss.control(trace = FALSE, ...),  
zero.component = FALSE, ...)
gamlssMXfits(n = 5, formula = formula(data), pi.formula = ~1,  
family = "NO", weights, K = 2, prob = NULL,  
data = sys.parent(), control = MX.control(),  
g.control = gamlss.control(trace = FALSE),  
zero.component = FALSE, ...)
```

**Arguments**

- **formula**  
  This argument should be a formula (or a list of formulae of length K) for modelling the $\mu$ parameter of the model. Note that modelling the rest of the distributional parameters can be done by using the usual ... which passes the arguments to `gamlss()`

- **pi.formula**  
  This should be a formula for modelling the prior probabilities as a function of explanatory variables. Note that no smoothing of other additive terms are allowed here only the usual linear terms. The modelling here is done using the `multinom()` function from package `nnet`

- **family**  
  This should be a `gamlss.family` distribution (or a list of distributions). Note that if different distributions are used here their parameters should be comparable for ease of interpretation.

- **weights**  
  Prior weights if needed

- **K**  
  The number of finite mixtures with default K=2

- **prob**  
  Prior probabilities if required for starting values
data  the data frame needed for the fit. Note that this is compulsory if pi.formula is used.

control  This argument sets the control parameters for the EM iterations algorithm. The default setting are given in the MX.control function.

g.control  This argument can be used to pass to gamlss() control parameters, as in gamlss.control.

n  the number of fits required in gamlssMXfits()

zero.component  whether zero component models exist, default is FALSE.

...  for extra arguments

Author(s)

Mikis Stasinopoulos and Bob Rigby

References


See Also

gamlss, gamlss.family

Examples

library(MASS)
data(geyser)

# fitting 2 finite normal mixtures
m1<-gamlssMX(waiting~1, data=geyser, family=NO, K=2)

## Not run:
# fitting 2 finite gamma mixtures
m2<-gamlssMX(waiting~1, data=geyser, family=GA, K=2)

# fitting a model for pi
# first create a data frame
geyer1<-matrix(0, ncol=2, nrow=298)
geyer1[,1]<-geyser$waiting[-1]
geyer1[,2]<-geyser$duration[-299]
colnames(geyer1)<-c("waiting", "duration")
geyer1<-data.frame(geyer1)

# get the best of 5 fits
m3<-gamlssMXfits(n=5, waiting~1, pi.formula=~duration, data=geyer1, family=NO, K=2)
m3

## End(Not run)
gamlssNP  A function to fit finite mixtures using the gamlss family of distributions

Description

This function will fit a finite (or normal) mixture distribution where the kernel distribution can belong to any gamlss family of distributions using the EM algorithm. The function is based on functions alldist() and allvc of the npmlreg package of Jochen Einbeck, John Hinde and Ross Darnell.

Usage

gamlssNP(formula, random = ~1, family = NO(), data = NULL, K = 4,
mixture = c("np", "gq"),
tol = 0.5, weights, pluginz, control = NP.control(...),
g.control = gamlss.control(trace = FALSE, ...), ...)

Arguments

formula    a formula defining the response and the fixed effects for the mu parameters
random     a formula defining the random part of the model
family     a gamlss family object
data       the data frame which for this function is mandatory even if it the data are attached
K          the number of mass points/integration points (supported values are 1:10,20)
mixture    the mixing distribution, "np" for non-parametric or "gq" for Gaussian Quadrature
tol        the toletance scalar ussualy between zero and one
weights    prior weights
pluginz    optional
control    this sets the control parameters for the EM iterations algorithm. The default setting is the NP.control function
g.control  the gamlss control function, gamlss.control, passed to the gamlss fit
...        for extra arguments

Details

The function gamlssNP() is a modification of the R functions alldist() and allvc created by Jochen Einbeck and John Hinde. Both functions were originally created by Ross Darnell (2002). Here the two functions are merged to one gamlssNP and allows finite mixture from gamlss family of distributions.

The following are comments from the original Einbeck and Hinde documentation.
The nonparametric maximum likelihood (NPML) approach was introduced in Aitkin (1996) as a tool to fit overdispersed generalized linear models. Aitkin (1999) extended this method to generalized linear models with shared random effects arising through variance component or repeated measures structure. Applications are two-stage sample designs, when firstly the primary sampling units (the upper-level units, e.g. classes) and then the secondary sampling units (lower-level units, e.g. students) are selected, or longitudinal data. Models of this type have also been referred to as multi-level models (Goldstein, 2003). This R function is restricted to 2-level models. The idea of NPML is to approximate the unknown and unspecified distribution of the random effect by a discrete mixture of $k$ exponential family densities, leading to a simple expression of the marginal likelihood, which can then be maximized using a standard EM algorithm. When option ’gq’ is set, then Gauss-Hermite masses and mass points are used and considered as fixed, otherwise they serve as starting points for the EM algorithm. The position of the starting points can be concentrated or extended by setting tol smaller or larger than one, respectively. Variance component models with random coefficients (Aitkin, Hinde & Francis, 2005, p. 491) are also possible, in this case the option random.distribution is restricted to the setting ’np’. The weights have to be understood as frequency weights, i.e. setting all weights equal to 2 will duplicate each data point and hence double the disparity and deviance. Warning: There might be some options and circumstances which had not been tested and where the weights do not work.” Note that in keeping with the gamlss notation disparity is called global deviance.

Value

The function `gamlssNP` produces an object of class "gamlssNP". This object contain several components.

- **family**: the name of the gamlss family
- **type**: the type of distribution which in this case is "Mixture"
- **parameters**: the parameters for the kernel gamlss family distribution
- **call**: the call of the gamlssNP function
- **y**: the response variable
- **bd**: the binomial denominator, only for BI and BB models
- **control**: the NP.control settings
- **weights**: the vector of weights of te expanded fit
- **G.deviance**: the global deviance
- **N**: the number of observations in the fit
- **rqres**: a function to calculate the normalized (randomized) quantile residuals of the object (here is the gamlss object rather than gamlssNP and it should change??)
- **iter**: the number of external iterations in the last gamlss fitting (?? do we need this?)
- **type**: the type of the distribution or the response variable here set to "Mixture"
- **method**: which algorithm is used for the gamlss fit, RS(), CG() or mixed()
- **contrasts**: the type of contrasts use in the fit
- **converged**: whether the gamlss fit has converged
- **residuals**: the normalized (randomized) quantile residuals of the model
mu.fv the fitted values of the extended mu model, also sigma.fv, nu.fv, tau.fv for the other parameters if present
mu.lp the linear predictor of the extended mu model, also sigma.lp, nu.lp, tau.lp for the other parameters if present
mu.wv the working variable of the extended mu model, also sigma.wv, nu.wv, tau.wv for the other parameters if present
mu.wt the working weights of the mu model, also sigma.wt, nu.wt, tau.wt for the other parameters if present
mu.link the link function for the mu model, also sigma.link, nu.link, tau.link for the other parameters if present
mu.terms the terms for the mu model, also sigma.terms, nu.terms, tau.terms for the other parameters if present
mu.x the design matrix for the mu, also sigma.x, nu.x, tau.x for the other parameters if present
mu.qr the QR decomposition of the mu model, also sigma.qr, nu.qr, tau.qr for the other parameters if present
mu.coefficients the linear coefficients of the mu model, also sigma.coefficients, nu.coefficients, tau.coefficients for the other parameters if present
mu.formula the formula for the mu model, also sigma.formula, nu.formula, tau.formula for the other parameters if present
mu.df the mu degrees of freedom also sigma.df, nu.df, tau.df for the other parameters if present
mu.nl.df the non linear degrees of freedom, also sigma.nl.df, nu.nl.df, tau.nl.df for the other parameters if present
df.fit the total degrees of freedom used by the model
df.residual the residual degrees of freedom left after the model is fitted
data the original data set
EMiter the number of EM iterations
EMconverged whether the EM has converged
allresiduals the residuals for the long fit
mass.points the estimates mass point (if "np" mixture is used)
K the number of mass points used
post.prob contains a matrix of posteriori probabilities,
prob the estimated mixture probabilities
aic the Akaike information criterion
sbc the Bayesian information criterion
formula the formula used in the expanded fit
random the random effect formula
pweights prior weights
the Empirical Bayes Predictions (Aitkin, 1996b) on the scale of the linear predictor.

Note that in case of Gaussian quadrature, the coefficient given at 'z' in coefficients corresponds to the standard deviation of the mixing distribution.

As a by-product, gamlssNP produces a plot showing the global deviance against the iteration number. Further, a plot with the EM trajectories is given. The x-axis corresponds to the iteration number, and the y-axis to the value of the mass points at a particular iteration. This plot is not produced when mixture is set to "gq".

Author(s)
Mikis Stasinopoulos based on function created by Jochen Einbeck John Hinde and Ross Darnell

References

See Also
gamlss, gamlss.family

Examples

data(enzyme)
# equivalent model using gamlssNP
mmNP1 <- gamlssNP(act~1, data=enzyme, random=-1, family=NO, K=2)
mmNP2 <- gamlssNP(act~1, data=enzyme, random=-1, sigma.fo=-MASS, family=NO, K=2)
AIC(mmmNP1, mmNP2)
**MX.control**

*The control function for gamlssMX*

**Description**

The function sets controls for the gamlssMX function.

**Usage**

```r
MX.control(cc = 1e-04, n.cyc = 200, trace = FALSE, 
            seed = NULL, plot = TRUE, sample = NULL, ...)
```

**Arguments**

- `cc`: convergent criterion for the EM
- `n.cyc`: number of cycles for EM
- `trace`: whether to print the EM iterations
- `seed`: a number for setting the seeds for starting values
- `plot`: whether to plot the sequence of global deviance up to convergence
- `sample`: how large the sample to be in the starting values
- `...`: for extra arguments

**Value**

Returns a list

**Author(s)**

Mikis Stasinopoulos and Bob Rigby

**References**


**See Also**

`gamlss`, `gamlssMX`, `gamlssMXfits`
Description

This is a control function for `gamlssNP` function.

Usage

```r
NP.control(EMcc = 0.001, EMn.cyc = 200, damp = TRUE,
           trace = TRUE, plot.opt = 3, ...)
```

Arguments

- `EMcc` convergence criterion for the EM
- `EMn.cyc` number of cycles for the EM
- `damp` Not in used
- `trace` whether to print the EM iterations
- `plot.opt` plotting the
- `...` for extra arguments

Value

Returns a list.

Author(s)

Mikis Stasinopoulos

References


See Also

`gamlss`, `gamlssNP`
Description

A utility function for plotting two dimension non-parametric distribution. The function uses the `persp()` function.

Usage

```r
plotMP(x, y, prob, theta = 20, phi = 20, expand = 0.5, col = "lightblue",
       xlab = "intercept", ylab = "slope", ...)
```

Arguments

- **x**: a vector containing points in the x axis
- **y**: a vector containing points in the y axis
- **prob**: vector containing probabilities which should add up to one
- **theta, phi, expand, col**: arguments to pass to the `persp()` function
- **xlab**: the x label
- **ylab**: the y label
- **...**: additional argument to be passed to `persp()`

Details

The function call

Value

A graph is produced.

Author(s)

Mikis Stasinopoulos

See Also

`gamlssNP`, `persp`

Examples

```r
gamma_0 <- c(-4.4, -3, -2.2, -.5, 0.1, 1, 1.5, 2.2, 3.5, 4.1)
gamma_1 <- c(2.2, 1.2, 0.1, -1, -2.3, -4.6, 5.1, -3.2, 0.1, -1.2)
prob <- c(0.1, .05, .12, 0.25, 0.08, 0.12, 0.10, 0.05, 0.10, 0.03)
plotMP(gamma_0, gamma_1, prob)
```
Index

*Topic datasets
  enzyme, 4

*Topic distribution
dMX, 3
  gamlss.mx-package, 2

*Topic package
gamlss.mx-package, 2

*Topic regression
dMX, 3
  gamlss.mx-package, 2
  gamlssMX, 5
  gamlssNP, 7
  MX.control, 11
  NP.control, 12
  plotMP, 13

brains (enzyme), 4

dMX, 3

enzyme, 4

gamlss, 2, 6, 10–12
  gamlss.family, 2, 6, 10
  gamlss.mx (gamlss.mx-package), 2
  gamlss.mx-package, 2
  gamlssMX, 5, 11
  gamlssMXfits, 11
  gamlssMXfits (gamlssMX), 5
  gamlssNP, 7, 12, 13
  getpdfMX (dMX), 3
  getpdfNP (dMX), 3

  MX.control, 11

  NP.control, 12

  persp, 13
  plotMP, 13
  pMX (dMX), 3