Flexible Regression and Smoothing”
Packages, Diagnostics and Algorithms

Bob Rigby  Mikis Stasinopoulos

Graz University of Technology, Austria, November 2016
1 The R packages

2 The \texttt{gamlss} package and the \texttt{gamlss()} function

3 Diagnostics
   - Normalised quantile residuals
   - Worm plots

4 Algorithms

5 End
The R packages

- **gamlss** the original package
- **gamlss.dist** all `gamlss.family` distributions
- **gamlss.data** different sets of data
- **gamlss.add** for extra additive terms
- **gamlss.cens** for censored (left, right or interval) response variables
- **gamlss.demo** demos for distributions and smoothing
  - **gamlss.nl** non-linear term fitting
  - **gamlss.tr** generating truncated distributions
  - **gamlss.mx** finite mixtures distributions and random effects
- **gamlss.spatial** for spatial models
- **gamlss.util** for extra utilities
The `gamlss()` function

```r
gamlss(formula = ~1, sigma.formula = ~1, 
u.nu.formula = ~1, tau.formula = ~1, 
family = NO(), 
data = sys.parent(), weights = NULL, 
contrasts = NULL, method = RS(), start.from = NULL, 
mu.start = NULL, sigma.start = NULL, 
u.nu.start = NULL, tau.start = NULL, 
mu.fix = FALSE, sigma.fix = FALSE, nu.fix = FALSE, 
tau.fix = FALSE, control = gamlss.control(...), 
i.control = glim.control(...), ...)```
Arguments of the `gamlss()` function

- `formula` = $y \sim x1 + x3$
- `sigma.fo` = $\sim x1$
- `nu.fo` = $\sim x2$
- `tau.fo` = $\sim 1$
- `data` = `abdom`
- `family` = `LO`
- `weights` = `freq`
- `method` = `mixed(10,50)`
- `control` = `gamlss.control(trace=FALSE)`
Starting values

Generally are not needed:

\[
\text{start.mu} = 2 \quad \text{or} \quad \text{start.mu} = \text{fitted(m1, "mu")}
\]

The same applies for other parameters

\[
\text{start.sigma, start.nu, start.tau}
\]

Starting from a previous model

\[
\text{start.from} = \text{model1}
\]
The `gamlss` package

Available functions

- Fitting or Updating a Model
- Extracting Information from the Fitted Model
- Selecting a Model
- Plotting and Diagnostics
- Centile Estimation
The `gamlss` package and the `gamlss()` function

Fitting or Updating a Model

- `gamlss()` for fitting and creating a `gamlss` object
- `refit()` to refit a `gamlss` object (i.e. continue iterations)
- `update()` to update a given `gamlss` model object
- `gamlssML()` fitting a parametric distribution to a single (response) variable
- `histDist()` to fit and plot a parametric distribution
- `fitDist()` select a parametric distribution from an appropriate list
Extracting Information from the Fitted Model

- **GAIC()** generalised Akaike information criterion (or AIC)
- **coef()** the linear coefficients
- **deviance()** the global deviance $-2 \log L$
- **fitted()** the fitted values for a distribution parameter
- **predict()** to predict from new data individual distribution parameter values
- **predictAll()** to predict from new data all the distribution parameter values
- **print()** : to print a gamlss object
- **residuals()** to extract the normalised (randomised) quantile residuals
- **summary()** to summarise the fit in a gamlss object
- **vcov()** to extract the variance-covariance matrix of the beta estimates.
Selecting a Model

add1() drop1() to add or drop a single term
stepGAIC() to select explanatory terms in one parameter
stepGAICA11.A() to select explanatory terms in all the parameters (strategy A)
stepGAICA11.B() to select explanatory terms in all the parameters (strategy B)
gamlssCV() to evaluate a model using cross validation
add1TGD() drop1TGD() to add or drop a single term using a validation or test data set
stepTGD() selecting variables using a test data set the global deviance for new (test) data set given a fitted gamlss model.
Diagnostics

`plot()` a plot of four graphs for the normalized (randomized) quantile residuals

`pdf.plot()` for plotting the pdf functions for a given fitted `gamlss` object or a given `gamlss.family` distribution

`Q.stats()` for printing and plotting the Q statistics of Royston and Wright (2000).

`rqres.plot()` for plotting QQ-plots of different realisations of randomised residuals (for discrete distributions)

`wp()` worm plot of the residuals from a fitted `gamlss` object

`dtop()` detrended Own’s plot of the residuals
Centile estimation

\texttt{lms()} a function trying to automate the process of fitting growth curves

\texttt{centiles()} to plot centile curves against an x-variable.

\texttt{centiles.com()} to compare centiles curves for more than one object.

\texttt{centiles.split()} as for \texttt{centiles()}, but splits the plot at specified values of x.

\texttt{centiles.pred()} to predict and plot centile curves for new x-values.

\texttt{centiles.fan()} fan plot od centile curves

\texttt{fitted.plot()} to plot fitted values for all the parameters against an x-variable
Other useful functions

**prof.dev()** the profile global deviance of one of the distribution parameters

**prof.term()** for plotting the profile global deviance of one of the model (beta) parameters

**show.link()** for showing available link functions

**term.plot()** for plotting additive (smoothing) terms

**gen.likelihood()** generates the likelihood from a GAMLSS fitted model [used in vcov()]
If $y_i$ is an observation from a continuous response variable then

$$\hat{u}_i = F(y_i | \hat{\theta}_i)$$

where $F(y_i | \theta_i)$ is the assumed cumulative distribution function for case $i$.

$$r_i = \Phi^{-1}(u_i)$$

So $r_i$ will have an approximately standard normal distribution

$$r_i = "z\text{-scores}"$$
Normalised quantile residuals—continuous response
If $y_i$ is an observation from a discrete integer response then

$$\hat{u}_i$$

is a random value from the uniform distribution on the interval

$$[u_1, u_2] = \left[ F(y_i - 1|\hat{\theta}_i), F(y_i|\hat{\theta}_i) \right]$$

$$\hat{r}_i = \Phi^{-1}(\hat{u}_i)$$
Normalised quantile residuals-continuous response
using plot()
Worm plots: using wp() van Buuren and Fredriks [2001]
Worm plots: different types

(a) resid mean too small

(b) resid mean too large

(c) resid variance too high

(d) resid variance too low

(e) resid negative skewness

(f) resid positive skewness

(g) resid lepto–kyrtotic

(h) resid platy–kyrtotic
Multiple worm plots against different range of x-variable
Multiple worm plots against different range of two x-variables
Worm plot conclusion

Worm plot are useful in:

- checking the assumed distribution of the response variable,
- checking whether the distribution is fitted correctly in all part of explanatory variable(s)
The model

\[ y \overset{\text{ind}}{\sim} D(\mu, \sigma, \nu, \tau) \]

\[ g_1(\mu) = X_1\beta_1 + s_{11}(x_{11}) + \ldots + s_{1J_1}(x_{1J_1}) \]

\[ g_2(\sigma) = X_2\beta_2 + s_{21}(x_{21}) + \ldots + s_{2J_2}(x_{2J_2}) \]

\[ g_3(\nu) = X_3\beta_3 + s_{31}(x_{31}) + \ldots + s_{3J_3}(x_{3J_3}) \]

\[ g_4(\tau) = X_4\beta_4 + s_{41}(x_{41}) + \ldots + s_{4J_4}(x_{4J_4}) \]
Algorithms: GAMLSS model random effects

\[ y \overset{\text{ind}}{\sim} D(\mu, \sigma, \nu, \tau) \]
\[ g_1(\mu) = X_1 \beta_1 + Z_{11} \gamma_{11} + \ldots + Z_{1k_1} \gamma_{1J_1} \]
\[ g_2(\sigma) = X_2 \beta_2 + Z_{21} \gamma_{21} + \ldots + Z_{2k_2} \gamma_{2J_2} \]
\[ g_3(\nu) = X_3 \beta_3 + Z_{31} \gamma_{31} + \ldots + Z_{3k_3} \gamma_{3J_3} \]
\[ g_4(\tau) = X_4 \beta_4 + Z_{41} \gamma_{41} + \ldots + Z_{4k_4} \gamma_{4J_4} \]
\[ \gamma_{kJ} \sim N(0, \lambda_{kJ}^{-1} G^{-1}) \] (1)
we will need estimates for the ‘betas’,

$$\beta = (\beta_1, \beta_2, \beta_3, \beta_4)$$

the ‘gammas’

$$\gamma = (\gamma_{11}, \ldots, \gamma_{1J_1}, \gamma_{21}, \ldots, \gamma_{4J_4})$$

and the ‘lambdas’

$$\lambda = (\lambda_{11}, \ldots, \lambda_{1J_1}, \lambda_{21}, \ldots, \lambda_{4J_4})$$

For parametric models: **ML estimators** for $\beta$

For smoothing (random effect models): **MAP estimators** for $\beta$ and $\gamma$ for fixed $\lambda$
Algorithms: posterior probability

\[ f(\beta, \gamma, |y, \lambda) \propto f(y|\beta, \gamma)f(\gamma|\lambda) \]
\[ \propto L(\beta, \gamma) \prod_{k} \prod_{j} f(\gamma_{kj}|\lambda_{kj}), \]

prior for \( \gamma \)
Algorithms: penalised likelihood

\[
\log f(\beta, \gamma, | y, \lambda) = \ell(\beta, \gamma) + \sum_k \sum_j \log f(\gamma_{kj} | \lambda_{kj}) + c(y, \lambda)
\]

\[
= \ell_h(\beta, \gamma, | \lambda) + c(y, \lambda)
\]

\[
= \ell(\beta, \gamma) - \frac{1}{2} \sum_k \sum_j \lambda_{kj} \gamma_{kj}^T G_{kj} \gamma_{kj} + c_1(y, \lambda)
\]

\[
= \ell_p(\beta, \gamma, | \lambda) + c_1(y, \lambda)
\]

\[
\text{log-likelihood} \quad \text{constant}
\]

\[
\text{hierarchical log-likelihood} \quad \text{constant}
\]

\[
\text{log-normal exponent}
\]

\[
\text{penalised log-likelihood} \quad \text{constant}
\]
Algorithms: Estimating $\beta$ and $\gamma$ for fixed $\lambda$

- **RS** a generalisation of the MADAM algorithm, Rigby and Stasinopoulos (1996a)
- **CG** a generalisation of Cole and Green (1992) algorithm
- **mixed** a mixture of RS+CG (i.e. $j$ iterations of RS, followed by $k$ iterations of CG)
Algorithms

(a) RS

(b) CG

Bob Rigby, Mikis Stasinopoulos
Flexible Regression and Smoothing
2015
Algorithms: The RS algorithm

- the outer iteration
- the inner iteration (or local scoring or GLIM algorithm)
- the modified backfitting algorithm
Algorithms: outer iteration

1. Initialise
2. Fit \( \mu \) given \( \hat{\sigma}, \hat{\nu}, \hat{\tau} \) using inner iter.
3. Fit \( \sigma \) given \( \hat{\mu}, \hat{\nu}, \hat{\tau} \) using inner iter.
4. Fit \( \nu \) given \( \hat{\mu}, \hat{\sigma}, \hat{\tau} \) using inner iter.
5. Fit \( \tau \) given \( \hat{\mu}, \hat{\sigma}, \hat{\nu} \) using inner iter.
6. Check global deviance converged:
   - If No, go back to step 2.
   - If Yes, finish.
Algorithms: inner iteration

given the current $\mu, \sigma, \nu, \hat{\nu}$

calculate $z_k$ and $w_k$

fit the linear explanatory variables and smoothers to $z_k$ with weights $w_k$ using modified backfitting and recalculate $\hat{\nu}_k$ and $\hat{\theta}_k$

No

global deviance converged

Yes

finish
Algorithms: modified backfitting

given current $z_k$, $w_k$, $\gamma_{k1}$ and $\gamma_{k2}$ for parameter $\theta_k$

1. calculate $\tau = z_k - Z_{k1}\hat{\gamma}_{k1} - Z_{k2}\hat{\gamma}_{k2}$
2. fit WLS to $\tau$ against $X_k$ using weights $w_k$, to get $\hat{\beta}_k$
3. calculate $\tau = z_k - X_k\hat{\beta}_k - Z_{k2}\hat{\gamma}_{k2}$
4. fit PWLS to $\tau$ against $Z_{k1}$ using weights $w_k$, to get new $\hat{\gamma}_{k1}$
5. calculate $\tau = z_k - X_k\hat{\beta}_k - Z_{k1}\hat{\gamma}_{k1}$
6. fit PWLS to $\tau$ against $Z_{k2}$ using weights $w_k$, to get new $\hat{\gamma}_{k2}$

Yes

do the parameters $\hat{\beta}_k$, $\hat{\gamma}_{k1}$, $\hat{\gamma}_{k2}$ change?

No

finish
Advantages of algorithms

1. flexible modular fitting procedure
2. easy implementation of new distributions
3. easy implementation of new additive terms
4. simple starting values for $(\mu, \sigma, \nu, \tau)$ easily found
5. stable and reliable algorithms
6. fast fitting (for fixed hyperparameters)
**Algorithms: Estimating $\lambda$**

**locally:** when the method of estimation of each $\lambda_{kj}$ is applied each time within the backfitting algorithm.

**globally:** when the method is applied outside the RS or CG GAMLSS algorithm.

Different methodologies for estimating the smoothing hyper-parameters:

- Generalised cross validation (GCV),
- Generalised Akaike information criterion (GAIC), and
- Maximum likelihood based methods (ML/REML).
END

for more information see

www.gamlss.org