

Flexible Regression and Smoothing"

Packages, Diagnostics and Algorithms

Bob Rigby Mikis Stasinopoulos

Dipartimento di Scienze Cliniche e di Comunità, Università degli studi di Milano, Italy, January 2018

- 1 The R packages
- 2 The (gamlss) package and the `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.inf` models for inflated 0 to 1 or for adjusted 0 to ∞
- `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

```
gamlss( formula = ~1, sigma.formula= ~1,  
        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,  
        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 ~ x1 + x3  
sigma.fo = ~ x1  
nu.fo = ~ x2  
tau.fo = ~ 1  
data = abdom  
family = LO  
weights = freq  
method = mixed(10,50)  
control = gamlss.control(trace=FALSE)
```

Starting values

Generally are not needed:

```
start.mu= 2 or start.mu=fitted(m1, "mu")
```

The same applies for other parameters

```
start.sigma , start.nu, start.tau
```

Starting from a previous model

```
start.from= 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

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

`stepGAICAll.A()` to select explanatory terms in all the parameters
(strategy A)

`stepGAICAll.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

`lms()` a function trying to automate the process of fitting growth curves

`centiles()` to plot centile curves against an x-variable.

`centiles.com()` to compare centiles curves for more than one object.

`centiles.split()` as for `centiles()`, but splits the plot at specified values of x.

`centiles.pred()` to predict and plot centile curves for new x-values.

`centiles.fan()` fan plot of centile curves

`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()`]

Normalised quantile residuals-continuous response

If y_i is an observation from a continuous response variable then

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

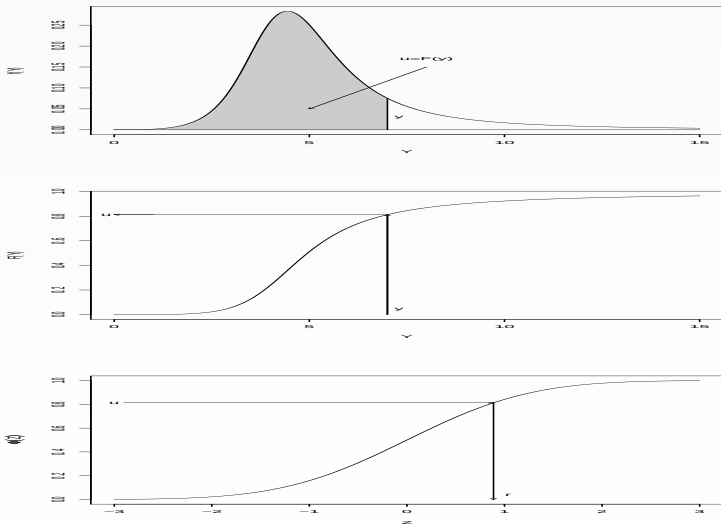
where $F(y_i | \theta_i)$ is the assumed cumulative distribution function for case i .

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

So r_i will have an approximately standard normal distribution

$$r_i = \text{"z-scores"}$$

Normalised quantile residuals-continuous response



Normalised quantile residuals-discrete response

If y_i is an observation from a discrete integer response then

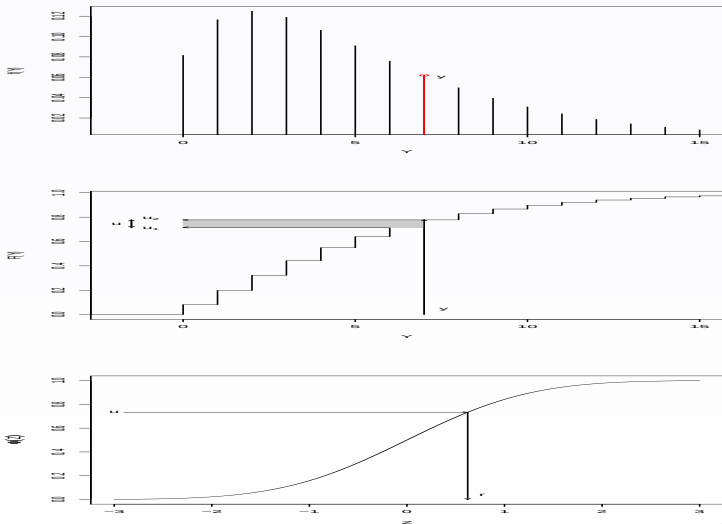
$$\hat{u}_i$$

is a random value from the uniform distribution on the interval

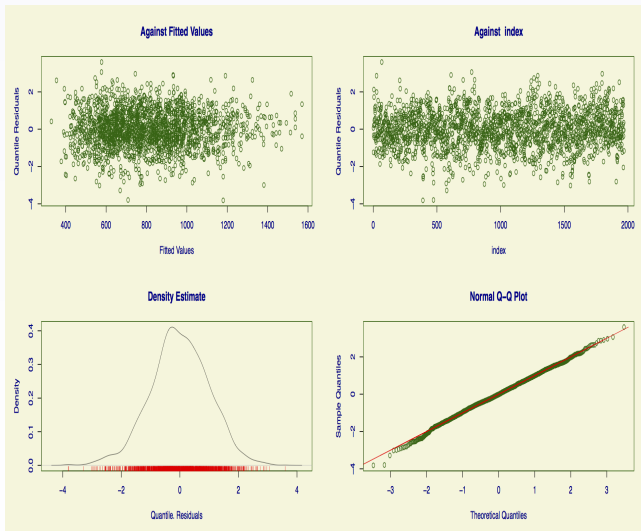
$$[u_1, u_2] = \left[F(y_i - 1 | \hat{\theta}_i), F(y_i | \hat{\theta}_i) \right]$$

$$\hat{r}_i = \Phi^{-1}(\hat{u}_i)$$

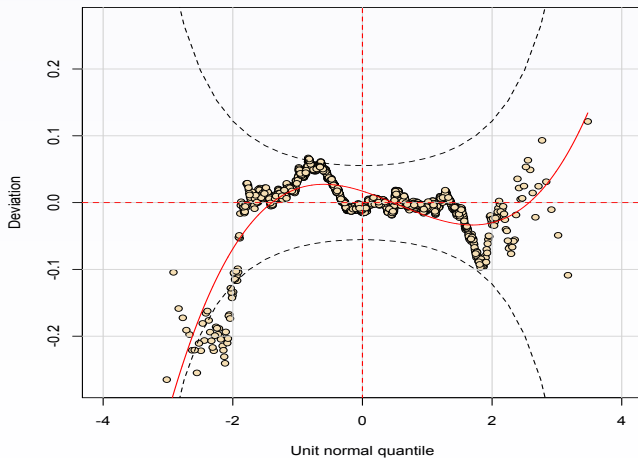
Normalised quantile residuals-continuous response



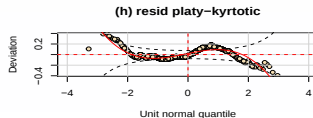
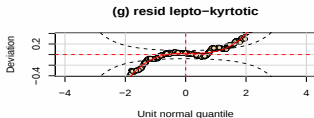
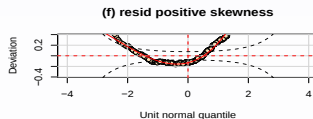
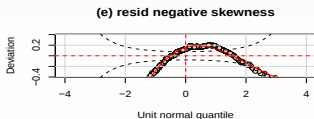
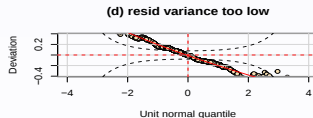
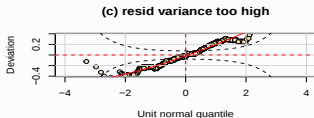
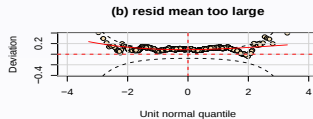
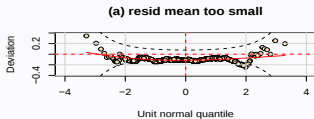
using plot()



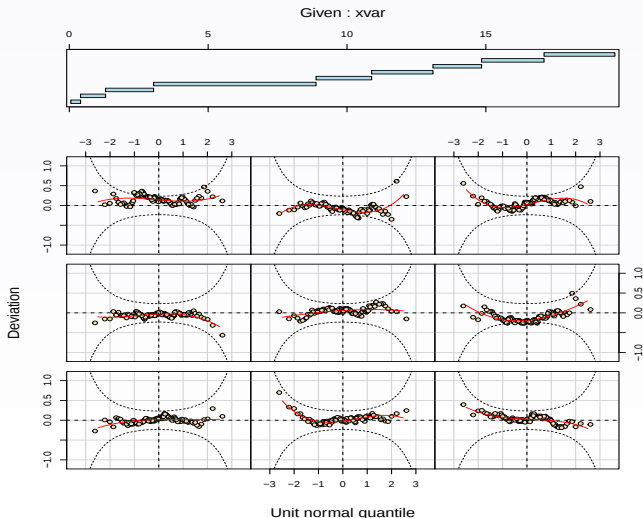
Worm plots: using `wp()` van Buuren and Fredriks [2001]



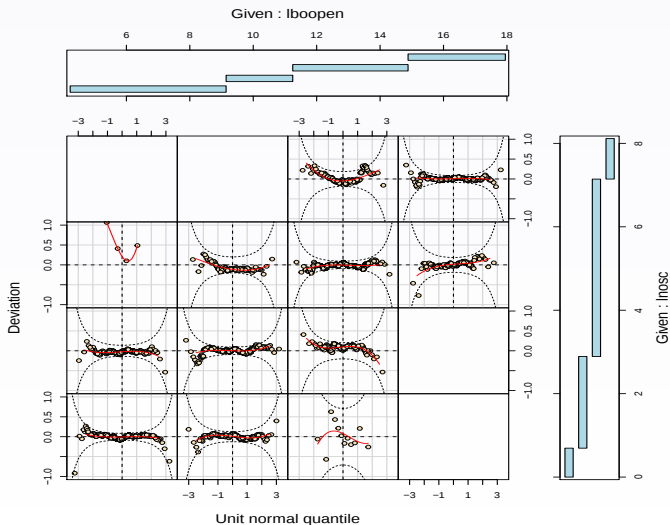
Worm plots: different types



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)

Algorithms: GAMLSS model

The model

$$\begin{aligned}
 \mathbf{y} &\overset{\text{ind}}{\sim} D(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}) \\
 g_1(\boldsymbol{\mu}) &= \mathbf{X}_1 \boldsymbol{\beta}_1 + s_{11}(\mathbf{x}_{11}) + \dots + s_{1J_1}(\mathbf{x}_{1J_1}) \\
 g_2(\boldsymbol{\sigma}) &= \mathbf{X}_2 \boldsymbol{\beta}_2 + s_{21}(\mathbf{x}_{21}) + \dots + s_{2J_2}(\mathbf{x}_{2J_2}) \\
 g_3(\boldsymbol{\nu}) &= \mathbf{X}_3 \boldsymbol{\beta}_3 + s_{31}(\mathbf{x}_{31}) + \dots + s_{3J_3}(\mathbf{x}_{3J_3}) \\
 g_4(\boldsymbol{\tau}) &= \mathbf{X}_4 \boldsymbol{\beta}_4 + s_{41}(\mathbf{x}_{41}) + \dots + s_{4J_4}(\mathbf{x}_{4J_4})
 \end{aligned}$$

Algorithms: GAMLSS model random effects

$$\begin{aligned}
 \mathbf{y} &\stackrel{\text{ind}}{\sim} D(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}) \\
 g_1(\boldsymbol{\mu}) &= \mathbf{X}_1 \boldsymbol{\beta}_1 + \mathbf{Z}_{11} \boldsymbol{\gamma}_{11} + \dots + \mathbf{Z}_{1k_1} \boldsymbol{\gamma}_{1J_1} \\
 g_2(\boldsymbol{\sigma}) &= \mathbf{X}_2 \boldsymbol{\beta}_2 + \mathbf{Z}_{21} \boldsymbol{\gamma}_{21} + \dots + \mathbf{Z}_{2k_2} \boldsymbol{\gamma}_{2J_2} \\
 g_3(\boldsymbol{\nu}) &= \mathbf{X}_3 \boldsymbol{\beta}_3 + \mathbf{Z}_{31} \boldsymbol{\gamma}_{31} + \dots + \mathbf{Z}_{3k_3} \boldsymbol{\gamma}_{3J_3} \\
 g_4(\boldsymbol{\tau}) &= \mathbf{X}_4 \boldsymbol{\beta}_4 + \mathbf{Z}_{41} \boldsymbol{\gamma}_{41} + \dots + \mathbf{Z}_{4k_4} \boldsymbol{\gamma}_{4J_4}, \\
 \boldsymbol{\gamma}_{kJ} &\sim N(\mathbf{0}, \lambda_{kJ}^{-1} \mathbf{G}^{-1})
 \end{aligned} \tag{1}$$

Algorithms: Estimation

we will need estimates for the ‘betas’,

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

the ‘gammas’

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

and the ‘lambdas’

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

For parametric models: **ML** estimators for β

For smoothing (random effect models) : **MAP** estimators for β and γ for fixed λ

Algorithms: posterior probability

$$\begin{aligned}
 f(\beta, \gamma, | \mathbf{y}, \boldsymbol{\lambda}) &\propto f(\mathbf{y} | \beta, \gamma) f(\gamma | \boldsymbol{\lambda}) \\
 &\propto \underbrace{L(\beta, \gamma)}_{\text{Likelihood}} \underbrace{\prod_k \prod_j f(\gamma_{kj} | \lambda_{kj})}_{\text{prior for } \gamma},
 \end{aligned}$$

Algorithms: penalised likelihood

$$\begin{aligned}
 \log f(\beta, \gamma, \mathbf{y}, \lambda) &= \underbrace{\ell(\beta, \gamma)}_{\text{log-likelihood}} + \sum_k \sum_j \log f(\gamma_{kj} | \lambda_{kj}) + \underbrace{c(\mathbf{y}, \lambda)}_{\text{constant}} \\
 &= \underbrace{\ell_h(\beta, \gamma, \lambda)}_{\text{hierarchical log-likelihood}} + \underbrace{c(\mathbf{y}, \lambda)}_{\text{constant}} \\
 &= \underbrace{\ell(\beta, \gamma)}_{\text{log-likelihood}} - \underbrace{\frac{1}{2} \sum_k \sum_j \lambda_{kj} \gamma_{kj}^\top \mathbf{G}_{kj} \gamma_{kj}}_{\text{log-normal exponent}} + \underbrace{c_1(\mathbf{y}, \lambda)}_{\text{constant}} \\
 &= \underbrace{\ell_p(\beta, \gamma, \lambda)}_{\text{penalised log-likelihood}} + \underbrace{c_1(\mathbf{y}, \lambda)}_{\text{constant}}
 \end{aligned}$$

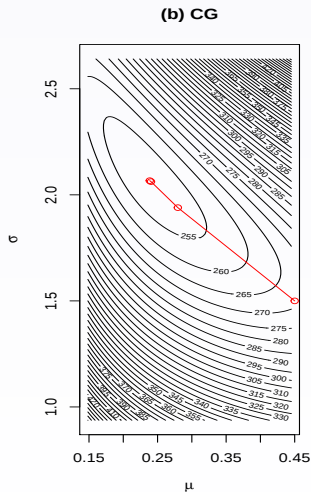
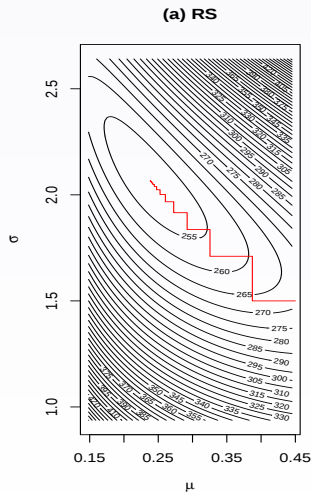
Algorithms: Estimating β and γ for fixed λ

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)

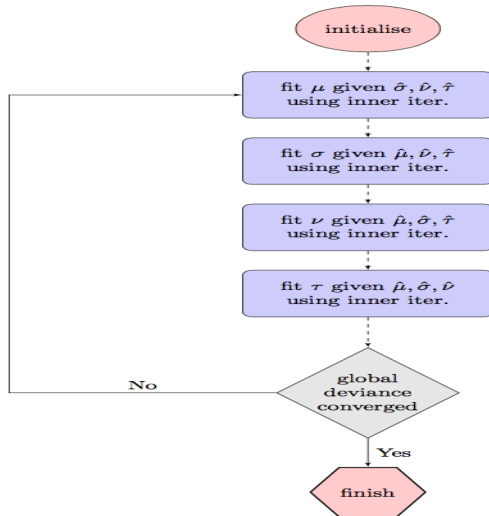
Algorithms



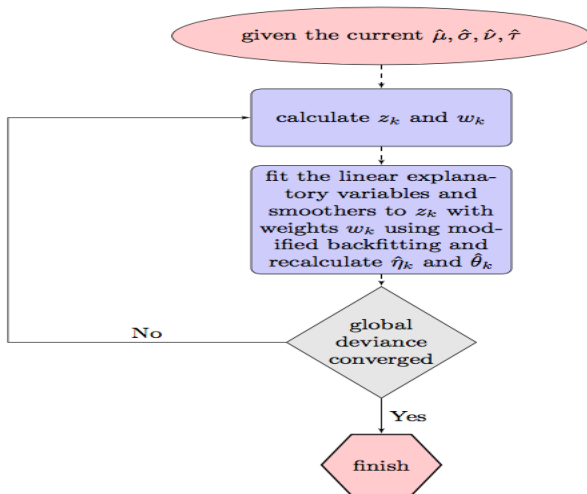
Algorithms: The RS algorithm

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

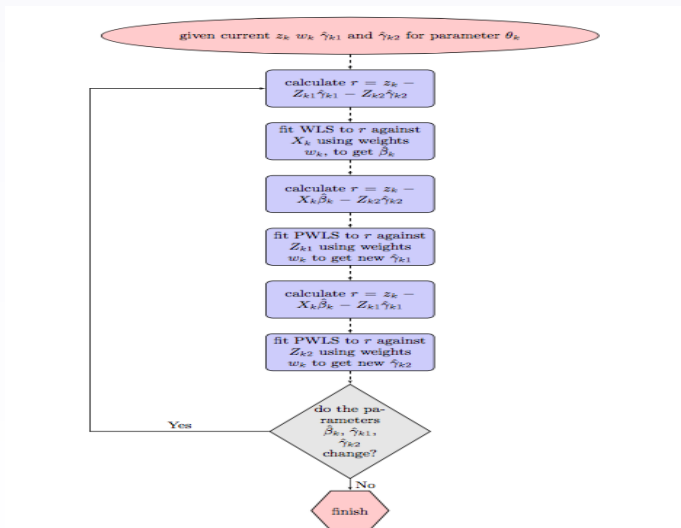
Algorithms: outer iteration



Algorithms: inner iteration



Algorithms: modified backfitting



Algorithms: properties

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 (μ, σ, ν, τ) easily found
- 5 stable and reliable algorithms
- 6 fast fitting (for fixed hyperparameters)

Algorithms: Estimating λ

locally: when the method of estimation of each λ_{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