Journal of Statistical Software

December 2007, Volume 23, Issue 7. *http://www.jstatsoft.org/*

Generalized Additive Models for Location Scale and Shape (GAMLSS) in R

D. Mikis Stasinopoulos
London Metropolitan University

Robert A. Rigby
London Metropolitan University

Abstract

GAMLSS is a general framework for fitting regression type models where the distribution of the response variable does not have to belong to the exponential family and includes highly skew and kurtotic continuous and discrete distribution. GAMLSS allows all the parameters of the distribution of the response variable to be modelled as linear/non-linear or smooth functions of the explanatory variables. This paper starts by defining the statistical framework of GAMLSS, then describes the current implementation of GAMLSS in R and finally gives four different data examples to demonstrate how GAMLSS can be used for statistical modelling.

Keywords: Box-Cox transformation, centile estimation, cubic smoothing splines, LMS method, negative binomial, non-normal, non-parametric, overdispersion, penalized likelihood, skewness and kurtosis.

1. What is GAMLSS?

1.1. Introduction

Generalized additive models for location, scale and shape (GAMLSS) are semi-parametric regression type models. They are parametric, in that they require a parametric distribution assumption for the response variable, and "semi" in the sense that the modelling of the parameters of the distribution, as functions of explanatory variables, may involve using non-parametric smoothing functions. GAMLSS were introduced by Rigby and Stasinopoulos (2001, 2005) and Akantziliotou, Rigby, and Stasinopoulos (2002) as a way of overcoming some of the limitations associated with the popular generalized linear models, GLM, and generalized additive models, GAM (see Nelder and Wedderburn 1972; Hastie and Tibshirani 1990, respectively).

In GAMLSS the exponential family distribution assumption for the response variable (y) is relaxed and replaced by a general distribution family, including highly skew and/or kurtotic continuous and discrete distributions. The systematic part of the model is expanded to allow modelling not only of the mean (or location) but other parameters of the distribution of y as, linear and/or non-linear, parametric and/or additive non-parametric functions of explanatory variables and/or random effects. Hence GAMLSS is especially suited to modelling a response variable which does not follow an exponential family distribution, (e.g., leptokurtic or platykurtic and/or positive or negative skew response data, or overdispersed counts) or which exhibit heterogeneity, (e.g., where the scale or shape of the distribution of the response variable changes with explanatory variables(s)).

There are several R-packages that can be seen as related to the **gamlss** packages and to its R implementation. The original **gam** package (Hastie 2006), the recommenced R package **mgcv** (Wood 2001), the general smoothing splines package **gss** (Gu 2007) and the vector GAM package, **VGAM** (Yee 2007). The first three deal mainly with models for the mean from an exponential family distribution. The **VGAM** package allows the modelling from a variety of different distributions (usually up to three parameter ones) and also allows multivariate responses.

The remainder of Section 1 defines the GAMLSS model, available distributions, available additive terms and model fitting. Section 2 describes the R **gamlss** package for fitting the GAMLSS model. Section 3 gives four data examples to illustrate GAMLSS modelling.

1.2. The GAMLSS model

A GAMLSS model assumes independent observations y_i for $i = 1, 2, \ldots, n$ with probability (density) function $f(y_i|\boldsymbol{\theta}^i)$ conditional on $\boldsymbol{\theta}^i = (\theta_{1i}, \theta_{2i}, \theta_{3i}, \theta_{4i}) = (\mu_i, \sigma_i, \nu_i, \tau_i)$ a vector of four distribution parameters, each of which can be a function to the explanatory variables. We shall refer to $(\mu_i, \sigma_i, \nu_i, \tau_i)$ as the *distribution parameters*. The first two population distribution parameters μ_i and σ_i are usually characterized as location and scale parameters, while the remaining parameter(s), if any, are characterized as shape parameters, e.g., skewness and kurtosis parameters, although the model may be applied more generally to the parameters of any population distribution, and can be generalized to more than four distribution parameters.

Rigby and Stasinopoulos (2005) define the original formulation of a GAMLSS model as follows. Let $\mathbf{y}^\top = (y_1, y_2, \ldots, y_n)$ be the n length vector of the response variable. Also for $k = 1, 2, 3, 4$, let $g_k(.)$ be known monotonic link functions relating the distribution parameters to explanatory variables by

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{X}_k\boldsymbol{\beta}_k + \sum_{j=1}^{J_k} \mathbf{Z}_{jk}\boldsymbol{\gamma}_{jk}, \tag{1}$$

i.e.

$$g_1(\boldsymbol{\mu}) = \boldsymbol{\eta}_1 = \mathbf{X}_1\boldsymbol{\beta}_1 + \sum_{j=1}^{J_1} \mathbf{Z}_{j1}\boldsymbol{\gamma}_{j1}$$

$$g_2(\boldsymbol{\sigma}) = \boldsymbol{\eta}_2 = \mathbf{X}_2\boldsymbol{\beta}_2 + \sum_{j=1}^{J_2} \mathbf{Z}_{j2}\boldsymbol{\gamma}_{j2}$$

$$g_3(\boldsymbol{\nu}) = \boldsymbol{\eta}_3 = \mathbf{X}_3\boldsymbol{\beta}_3 + \sum_{j=1}^{J_3} \mathbf{Z}_{j3}\boldsymbol{\gamma}_{j3}$$
$$g_4(\boldsymbol{\tau}) = \boldsymbol{\eta}_4 = \mathbf{X}_4\boldsymbol{\beta}_4 + \sum_{j=1}^{J_4} \mathbf{Z}_{j4}\boldsymbol{\gamma}_{j4}.$$

where $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\nu}$ $\boldsymbol{\tau}$ and $\boldsymbol{\eta}_k$ are vectors of length n, $\boldsymbol{\beta}_k^\top = (\beta_{1k}, \beta_{2k}, \ldots, \beta_{J_k'k})$ is a parameter vector of length J_k', $\mathbf{X}_k$ is a fixed known design matrix of order $n \times J_k'$, $\mathbf{Z}_{jk}$ is a fixed known $n \times q_{jk}$ design matrix and $\boldsymbol{\gamma}_{jk}$ is a q_{jk} dimensional random variable which is assumed to be distributed as $\boldsymbol{\gamma}_{jk} \sim N_{q_{jk}}(\mathbf{0}, \mathbf{G}_{jk}^{-1})$, where $\mathbf{G}_{jk}^{-1}$ is the (generalized) inverse of a $q_{jk} \times q_{jk}$ symmetric matrix $\mathbf{G}_{jk} = \mathbf{G}_{jk}(\boldsymbol{\lambda}_{jk})$ which may depend on a vector of hyperparameters $\boldsymbol{\lambda}_{jk}$, and where if $\mathbf{G}_{jk}$ is singular then $\boldsymbol{\gamma}_{jk}$ is understood to have an improper prior density function proportional to $\exp\left(-\frac{1}{2}\boldsymbol{\gamma}_{jk}^\top \mathbf{G}_{jk}\boldsymbol{\gamma}_{jk}\right)$.

The model in (1) allows the user to model each distribution parameter as a linear function of explanatory variables and/or as linear functions of stochastic variables (random effects). Note that seldom will all distribution parameters need to be modelled using explanatory variables.

There are several important sub-models of GAMLSS. For example for readers familiar with smoothing, the following GAMLSS sub-model formulation may be more familiar. Let $\mathbf{Z}_{jk} = \mathbf{I}_n$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix, and $\boldsymbol{\gamma}_{jk} = \mathbf{h}_{jk} = h_{jk}(\mathbf{x}_{jk})$ for all combinations of j and k in (1), then we have the *semi-parametric additive* formulation of GAMLSS given by

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{X}_k\boldsymbol{\beta}_k + \sum_{j=1}^{J_k} h_{jk}(\mathbf{x}_{jk}) \tag{2}$$

where to abbreviate the notation use $\boldsymbol{\theta}_k$ for $k = 1, 2, 3, 4$ to represent the distribution parameter vectors $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$, and where $\mathbf{x}_{jk}$ for $j = 1, 2, \ldots, J_k$ are also vectors of length n. The function h_{jk} is an unknown function of the explanatory variable X_{jk} and $\mathbf{h}_{jk} = h_{jk}(\mathbf{x}_{jk})$ is the vector which evaluates the function h_{jk} at $\mathbf{x}_{jk}$. If there are no additive terms in any of the distribution parameters we have the simple *parametric linear* GAMLSS model,

$$g_1(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = \mathbf{X}_k\boldsymbol{\beta}_k \tag{3}$$

Model (2) can be extended to allow non-linear parametric terms to be included in the model for μ, σ, ν and τ, as follows (see Rigby and Stasinopoulos 2006)

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = h_k(\mathbf{X}_k, \boldsymbol{\beta}_k) + \sum_{j=1}^{J_k} h_{jk}(\mathbf{x}_{jk}) \tag{4}$$

where h_k for $k = 1, 2, 3, 4$ are non-linear functions and $\mathbf{X_k}$ is a known design matrix of order $n \times J_k''$. We shall refer to the model in (4) as the *non-linear semi-parametric additive* GAMLSS model. If, for $k = 1, 2, 3, 4$, $J_k = 0$, that is, if for all distribution parameters we do not have additive terms, then model (4) is reduced to a *non-linear parametric* GAMLSS model.

$$g_k(\boldsymbol{\theta}_k) = \boldsymbol{\eta}_k = h_k(\mathbf{X}_k, \boldsymbol{\beta}_k). \tag{5}$$

If, in addition, $h_k(\mathbf{X}_k, \boldsymbol{\beta}_k) = \mathbf{X}_k^\top\boldsymbol{\beta}_k$ for $i = 1, 2, \ldots, n$ and $k = 1, 2, 3, 4$ then (5) reduces to the linear parametric model (3). Note that some of the terms in each $h_k(\mathbf{X_k}, \boldsymbol{\beta_k})$ may be linear,

in which case the GAMLSS model is a combination of linear and non-linear parametric terms. We shall refer to any combination of models (3) or (5) as a *parametric* GAMLSS model.

The parametric vectors $\boldsymbol{\beta}_k$ and the random effects parameters $\boldsymbol{\gamma}_{jk}$, for $j = 1, 2, \ldots, J_k$ and $k = 1, 2, 3, 4$ are estimated within the GAMLSS framework (for fixed values of the smoothing hyper-parameters λ_{jk}'s) by maximising a penalized likelihood function ℓ_p given by

$$\ell_p = \ell - \frac{1}{2}\sum_{k=1}^{p}\sum_{j=1}^{J_k}\lambda_{jk}\boldsymbol{\gamma}_{jk}'\mathbf{G}_{jk}\boldsymbol{\gamma}_{jk} \tag{6}$$

where $\ell = \sum_{i=1}^{n}\log f(y_i|\boldsymbol{\theta}^i)$ is the log likelihood function. More details on how the penalized log likelihood ℓ_p is maximized are given in Section 1.5. For parametric GAMLSS model (3) or (5), ℓ_p reduces to ℓ, and the β_k for $k = 1, 2, 3, 4$ are estimated by maximizing the likelihood function ℓ. The available distributions and the different additive terms in the current GAMLSS implementation in R are given in Sections 1.3 and 1.4 respectively. The R function to fit a GAMLSS model is `gamlss()` in the package **gamlss** which will be described in more detail in Section 2.

1.3. Available distributions in GAMLSS

The form of the distribution assumed for the response variable y, $f(y_i|\mu_i, \sigma_i, \nu_i, \tau_i)$, can be very general. The only restriction that the R implementation of GAMLSS has is that the function $\log f(y_i|\mu_i, \sigma_i, \nu_i, \tau_i)$ and its first (and optionally expected second and cross) derivatives with respect to each of the parameters of $\boldsymbol{\theta}$ must be computable. Explicit derivatives are preferable but numerical derivatives can be used.

Table 1 shows a variety of one, two, three and four parameter families of continuous distributions implemented in our current software version. Table 2 shows the discrete distributions. We shall refer to the distributions in Tables 1 and 2 as the `gamlss.family` distributions, a name to coincide with the R object created by the package **gamlss**. Johnson, Kotz, and Balakrishnan (1994, 1995); Johnson, Kotz, and Kemp (2005) are the classical reference books for most of the distributions in Tables 1 and 2. The BCCG distribution in Table 1 is the Box-Cox transformation model used by Cole and Green (1992) (also known as the LMS method of centile estimation). The BCPE and BCT distributions, described in Rigby and Stasinopoulos (2004, 2006) respectively, generalize the BCCG distribution to allow modelling of both skewness and kurtosis. For some of the distributions shown in Tables 1 and 2 more that one parameterization has been implemented. For example, the two parameter Weibull distribution can be parameterized as $f(y|\mu, \sigma) = (\sigma y^{\sigma-1}/\mu^{\sigma})\exp\{-(y/\mu)^{\sigma}\}$, denoted as WEI, or as $f(y|\mu, \sigma) = \sigma\mu y^{\sigma-1}e^{-\mu y^{\sigma}}$, denoted as WEI2, or as $f(y|\mu, \sigma) = (\sigma/\beta)(y/\beta)^{\sigma-1}\exp\{-(y/\beta)^{\sigma}\}$ denoted as WEI3, for $\beta = \mu/[\Gamma(1/\sigma) + 1]$. Note that the second parameterization WEI2 is suited to proportional hazard (PH) models. In the WEI3 parameterization, parameter μ is equal to the mean of y. The choice of parameterization depends upon the particular problem, but some parameterizations are computationally preferable to others in the sense that maximization of the likelihood function is easier. This usually happens when the parameters μ, σ, ν and τ are orthogonal or almost orthogonal. For interpretation purposes we favour parameterizations where the parameter μ is a location parameter (mean, median or mode). The specific parameterizations used in the `gamlss.family` distributions are given in the appendix of Stasinopoulos, Rigby, and Akantziliotou (2006).

Distributions	R Name	μ	σ	ν	τ
beta	BE()	logit	logit	-	-
beta inflated (at 0)	BEOI()	logit	log	logit	-
beta inflated (at 1)	BEZI()	logit	log	logit	-
beta inflated (at 0 and 1)	BEINF()	logit	logit	log	log
Box-Cox Cole and Green	BCCG()	identity	log	identity	-
Box-Cox power exponential	BCPE()	identity	log	identity	log
Box-Cox-t	BCT()	identity	log	identity	log
exponential	EXP()	log	-	-	-
exponential Gaussian	exGAUS()	identity	log	log	-
exponential gen. beta type 2	EGB2()	identity	identity	log	log
gamma	GA()	log	log	-	-
generalized beta type 1	GB1()	logit	logit	log	log
generalized beta type 2	GB2()	log	identity	log	log
generalized gamma	GG()	log	log	identity	-
generalized inverse Gaussian	GIG()	log	log	identity	-
generalized y	GT()	identity	log	log	log
Gumbel	GU()	identity	log	-	-
inverse Gaussian	IG()	log	log	-	-
Johnson's SU (μ the mean)	JSU()	identity	log	identity	log
Johnson's original SU	JSUo()	identity	log	identity	log
logistic	LO()	identity	log	-	-
log normal	LOGNO()	log	log	-	-
log normal (Box-Cox)	LNO()	log	log	fixed	-
NET	NET()	identity	log	fixed	fixed
normal	NO()	identity	log	-	-
normal family	NOF()	identity	log	identity	-
power exponential	PE()	identity	log	log	-
reverse Gumbel	RG()	identity	log	-	-
skew power exponential type 1	SEP1()	identity	log	identity	log
skew power exponential type 2	SEP2()	identity	log	identity	log
skew power exponential type 3	SEP3()	identity	log	log	log
skew power exponential type 4	SEP4()	identity	log	log	log
shash	SHASH()	identity	log	log	log
skew t type 1	ST1()	identity	log	identity	log
skew t type 2	ST2()	identity	log	identity	log
skew t type 3	ST3()	identity	log	log	log
skew t type 4	ST4()	identity	log	log	log
skew t type 5	ST5()	identity	log	identity	log
t Family	TF()	identity	log	log	-
Weibull	WEI()	log	log	-	-
Weibull (PH)	WEI2()	log	log	-	-
Weibull (μ the mean)	WEI3()	log	log	-	-
zero adjusted IG	ZAIG()	log	log	logit	-

Table 1: Continuous distributions implemented within the **gamlss** packages (with default link functions).

Distributions	R Name	μ	σ	ν
beta binomial	`BB()`	`logit`	`log`	-
binomial	`BI()`	`logit`	-	-
Delaporte	`DEL()`	`log`	`log`	`logit`
Negative Binomial type I	`NBI()`	`log`	`log`	-
Negative Binomial type II	`NBII()`	`log`	`log`	-
Poisson	`PO()`	`log`	-	-
Poisson inverse Gaussian	`PIG()`	`log`	`log`	-
Sichel	`SI()`	`log`	`log`	`identity`
Sichel (μ the mean)	`SICHEL()`	`log`	`log`	`identity`
zero inflated poisson	`ZIP()`	`log`	`logit`	-
zero inflated poisson (μ the mean)	`ZIP2()`	`log`	`logit`	-

Table 2: Discrete distributions implemented within the **gamlss** packages (with default link functions).

All of the distributions in Tables 1 and 2 have `d`, `p`, `q` and `r` functions corresponding respectively to the probability density function (pdf), the cumulative distribution function (cdf), the quantiles (i.e., inverse cdf) and random value generating functions. For example, the gamma distribution has the functions `dGA`, `pGA`, `qGA` and `rGA`. In addition each distribution has a *fitting* function which helps the fitting procedure by providing link functions, first and (exact of approximate) expected second derivatives, starting values etc. All fitting functions have as arguments the link functions for the distribution parameters. For example, the fitting function for the gamma distribution is called `GA` with arguments `mu.link` and `sigma.link`. The default link functions for all `gamlss.family` distributions are shown in columns 3–6 of Tables 1 and 2. The function `show.link()` can be used to identify which are the available links for the distribution parameter within each of the `gamlss.family`. Available link functions can be the usual `glm()` link functions plus `logshifted`, `logitshifted` and `own`. The `own` option allows the user to define his/her own link function, for an example see the help file on the function `make.link.gamlss()`.

There are several ways to extend the `gamlss.family` distributions. This can be achieved by

- creating a new `gamlss.family` distribution
- truncating an existing `gamlss.family`
- using a censored version of an existing `gamlss.family`
- mixing different `gamlss.family` distributions to create a new finite mixture distribution.

To create a new `gamlss.family` distribution is relatively simple, if the pdf function of the distribution can be evaluated easily. To do that, find a file of a current `gamlss.family` distribution, (having the same number of distribution parameters) and amend accordingly. For more details, on how this can be done, see Stasinopoulos *et al.* (2006, Section 4.2).

Truncating existing `gamlss.family` distributions can be achieved by using the add-on package **gamlss.tr**. The function `gen.trun()`, within the **gamlss.tr** package, can take any `gamlss.family`

distribution and generate the `d`, `p`, `q`, `r` and fitting R functions for the specified truncated distribution. The truncation can be left, right or in both tails of the range of the response y variable.

The package **gamlss.cens** is designed for the situation where the response variable is censored or, more generally, it has been observed in an interval form, e.g., (3, 10] an interval from 3 to 10 (including only the right end point 10). The function `gen.cens()` will take any `gamlss.family` distribution and create a new function which can fit a response of "interval" type. Note that for "interval" response variables the usual likelihood function for independent response variables defined as

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(y_i|\boldsymbol{\theta}) \tag{7}$$

changes to

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} [F(y_{2i}|\boldsymbol{\theta}) - F(y_{1i}|\boldsymbol{\theta})] \tag{8}$$

where $F(y)$ is the cumulative distribution function and (y_{1i}, y_{2i}) is the observed interval.

Finite mixtures of `gamlss.family` distributions can be fitted using the package **gamlss.mx**. A finite mixture of `gamlss.family` distributions will have the form

$$f_Y(y|\boldsymbol{\psi}) \quad = \quad \sum_{k=1}^{K} \pi_k f_k(y|\boldsymbol{\theta}_k) \tag{9}$$

where $f_k(y|\boldsymbol{\theta}_k)$ is the probability (density) function of y for component k, and $0 \leq \pi_k \leq 1$ is the prior (or mixing) probability of component k, for $k = 1, 2, \ldots, K$. Also $\sum_{k=1}^{K} \pi_k = 1$ and $\boldsymbol{\psi} = (\boldsymbol{\theta}, \boldsymbol{\pi})$ where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k)$ and $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_K)$. Any combination of (continuous or discrete) `gamlss.family` distributions can be used. The model in this case is fitted using the EM algorithm. The component probability (density) functions may have different parameters (fitted using the function `gamlssMX()`) or may have parameters in common (fitted using the function `gamlssNP()`). In the former case, the mixing probabilities may also be modelled using explanatory variables and the finite mixture may have a zero component (e.g., zero inflated negative binomial etc.). Both functions `gamlssMX()`) and `gamlssNP()` are in the add on package **gamlss.mx**.

1.4. Available additive terms in GAMLSS

Equation (1) allows the user to model all the distribution parameters μ, σ, ν and τ as linear parametric and/or non-linear parametric and/or non-parametric (smooth) function of the explanatory variables and/or random effects terms. For modelling linear functions the Wilkinson and Rogers (1973) notation as applied for model formulae in the S language by Chambers and Hastie (1992) can be used. It is the model formulae notation used in the fit of linear models, `lm()`, and generalized lineal models, `glm()`, see for example Venables and Ripley (2002, Section 6.2). For fitting non-linear or non-parametric (smooth) functions or random effects terms, an additive term function has to be fitted. Parametric non-linear models can be also fitted using the function `nlgamlss()` of the add-on package **gamlss.nl**.

Additive terms	R Name
Cubic splines	`cs()`
Varying coefficient	`vc()`
Penalized splines	`ps()`
`loess`	`lo()`
Fractional polynomials	`fp()`
Power polynomials	`pp()`
non-linear fit	`nl()`
Random effects	`random()`
Random effects	`ra()`
Random coefficient	`rc()`

Table 3: Additive terms implemented within the **gamlss** packages.

Table 3 shows the additive term functions implemented in the current R implementation of GAMLSS. Note that all available additive terms names are stored in the list `.gamlss.sm.list`.

The cubic spline function `cs()` is based on the `smooth.spline()` function of R and can be used for univariate smoothing. Cubic splines are covered extensively in the literature (Reinsch 1967; Green and Silverman 1994; Hastie and Tibshirani 1990, Chapter 2). They assume in model (2) that the functions $h(t)$ are arbitrary twice continuously differentiable functions and we maximize a penalized log likelihood, given by ℓ subject to penalty terms of the form $\lambda \int_{-\infty}^{\infty} \left[h''(t)\right]^2 dt$. The solution for the maximizing functions $h(t)$ are all natural cubic splines, and hence can be expressed as linear combinations of their natural cubic spline basis functions (de Boor 1978). In `cs()` each distinct x-value is a knot.

The varying coefficient terms were introduced by Hastie and Tibshirani (1993) to accommodate a special type of interaction between explanatory variables. This interaction takes the form of $\beta(r)x$, that is the linear coefficient of the explanatory variable x is changing smoothly according to another explanatory variable r. In some applications r will be time. In general r should be a continuous variable, while x can be either continuous or categorical. In the current GAMLSS implementation x has to be continuous or a two level factor with levels 0 and 1.

Penalized splines were introduced by Eilers and Marx (1996). Penalized Splines (or P-splines) are piecewise polynomials defined by B-spline basis functions in the explanatory variable, where the coefficients of the basis functions are penalized to guarantee sufficient smoothness (see Eilers and Marx 1996). More precisely consider the model $\boldsymbol{\theta} = \mathbf{Z}(\mathbf{x})\boldsymbol{\gamma}$ where $\boldsymbol{\theta}$ can be any distributional parameter in a GAMLSS model, $\mathbf{Z}(\mathbf{x})$ is $n \times q$ basis design matrix for the explanatory variable $\mathbf{x}$ defined at q-different knots mostly within the range of $\mathbf{x}$ and $\boldsymbol{\gamma}$ is a $q \times 1$ vector of coefficients which have some stochastic restrictions imposed by the fact that $\mathbf{D}\boldsymbol{\gamma} \sim N(\mathbf{0}, \lambda^{-1}\mathbf{I})$ or equivalently by $\boldsymbol{\gamma} \sim N(\mathbf{0}, \lambda^{-1}\mathbf{K}^{-})$ where $\mathbf{K} = \mathbf{D}^{\top}\mathbf{D}$. The matrix $\mathbf{D}$ is a $(q-r) \times q$ matrix giving rth differences of the q-dimensional vector γ. So to define a penalized spline we need: i) q the number of knots in the x-axis defined by argument `ps.intervals` (and of course where to put them; `ps()` uses equal spaces in the x-axis), ii) the degree of the piecewise polynomial used in the B-spline basis so we can define $\mathbf{X}$, defined by argument `degree` iii) r the order of differences in the $\mathbf{D}$ matrix indicating the type of the penalty imposed in the the coefficients of the B-spline basis functions, defined by argument *order*

and iv) the amount of smoothing required defined either by the desired equivalent degrees of freedom defined by argument `df` (or alternative by the smoothing parameter defined by argument `lambda`). The `ps()` function in **gamlss** is based on an S-PLUS function of Marx (2003).

The function `lo()` allows the user to use a `loess` fit in a GAMLSS formula. A `loess` fit is a polynomial (surface) curve determined by one or more explanatory (continuous) variables, which are fitted locally (see Cleveland, Grosse, and Shyu 1993). The implementation of the `lo()` function is very similar to the function with the same name in the S-PLUS implementation of `gam`. However **gamlss** `lo()` function uses the R `loess()` function as its engine and this creates some minor differences between the two `lo()` even when the same model is fitted. `lo()` is the only function currently available in **gamlss** which allows smoothing in more than one explanatory (continuous) variables.

The `fp()` function is an implementation of the fractional polynomials introduced by Royston and Altman (1994). The functions involved in `fp()` and `bfp()` are loosely based on the fractional polynomials function `fracpoly()` for S-PLUS given by Ambler (1999). The function `bfp` generates the correct design matrix for fitting a power polynomial of the type $b_0 + b_1 x^{p_1} + b_2 x^{p_2} + ... + b_k x^{p_k}$. For given powers $p_1, p_2, ..., p_k$, given as the argument `powers` in `bfp()`, the function can be used to fit power polynomials in the same way as the functions `poly()` or `bs()` of the package `splines` are used to fit orthogonal or piecewise polynomials respectively. The function `fp()` (which uses `bfp()`) works as an additive smoother term in **gamlss**. It is used to fit the best fractional polynomials among a specific set of power values. Its argument `npoly` determines whether one, two or three fractional polynomials should used in the fitting. For a fixed number `npoly` the algorithm looks for the best fitting fractional polynomials in the list `c(-2, -1, -0.5, 0, 0.5, 1, 2, 3)`. Note that `npoly=3` is rather slow since it fits all possible 3-way combinations at each backfitting iteration.

The power polynomial function `pp()` is an experimental function and is designed for the situation in which the model is in the form $b_0 + b_1 x^{p_1} + b_2 x^{p_2}$ with powers p_1, p_2 to be estimated non-linearly by the data. Initial values for the non-linear parameters p_1, p_2 have to be supplied.

The function `nl()` exists in the add-on package **gamlss.nl** designed for fitting non-linear parametric models within GAMLSS. It provides a way of fitting non-linear terms together with linear or smoothing terns in the same model. The function takes a non-linear object, (created by the function `nl.obs`), and uses the R `nlm()` function within the backfitting cycle of `gamlss()`. The success of this procedure depends on the starting values of the non-linear parameters (which must be provided by the user). No starting values are required for the other, e.g., linear terms, of the model.

The function `random()` allows the fitted values for a factor (categorical) predictor to be shrunk towards the overall mean, where the amount of shrinking depends either on the parameter λ, or on the equivalent degrees of freedom (df). This function is similar to the `random()` function in the **gam** package of Hastie (2006) documented in Chambers and Hastie (1992). The function `ra()` is similar to the function `random()` but its fitting procedure is based on augmented least squares, a fact that makes `ra()` more general, but also slower to fit, than `random()`. The random coefficient function `rc()` is experimental. Note that the "random effects" functions, `random()`, `ra()` and `rc()` are used to estimate the random effect γ's *given* the hyperparameters λ's. In order to obtain estimates for the hyperparameters, methods

discussed in Rigby and Stasinopoulos (2005, Appendix A) can be used. Alternatively, for models only requiring a single random effect in one distribution parameter only, the function `gamlssNP()` of the package **gamlss.mx**, which uses Gaussian quadrature, can be used.

The `gamlss()` function uses the same type of additive backfitting algorithm implemented in the `gam()` function of the R package **gam** (Hastie 2006). Note that the function `gam()` implementation in the R recommended package **mgcv** (Wood 2001) does not use backfitting. The reason that we use backfitting here that it is easier to extend the algorithm so new additive terms can be included.

Each new additive term in the `gamlss()` requires two new functions. The first one, (the one that is seen by the user) is the one which defines the additive term and sets the additional required design matrices for the linear part of the model. The names of the existing additive functions are shown in the second column of Table 3. For example `cs(x)` defines a cubic smoothing spline function for the continuous explanatory variable `x`. It is used during the definition of the design matrix for the appropriate distribution parameter and it adds a linear term for `x` in the design matrix. The second function is the one that actually performs the additive backfitting algorithm. This function is called `gamlss.name()` where the `name` is one of the names in column two of Table 3. For example the function `gamlss.cs()` performs the backfitting for cubic splines. New additive terms can be implemented by defining those two functions and adding the new names in the `.gamlss.sm.list` list.

The general policy when backfitting is used in `gamlss()` is to include the linear part of an additive term in the appropriate linear term design matrix. For example, in the cubic spline function `cs()` the explanatory variable say x is put in the linear design matrix of the appropriate distribution parameter and the smoothing function is fitted as a deviation from this linear part. This is equivalent of fitting a modified backfitting algorithm, see Hastie and Tibshirani (1990). In other additive functions where the linear part is not needed (or defined) a column on zeros is put in the design matrix. For example, this is the case when the fractional polynomials additive term `fp()` is used.

If the user wishes to create a new additive term, care should be taken on how the degrees of freedom of the model are defined. The degrees of freedom for the (smoothing) additive terms are usually taken to be the extra degrees of freedom on top of the linear fit. For example to fit a single smoothing cubic spline term for say x with 5 total degrees of freedom, `cs(x,df=3)` should be used since already 2 degrees of freedom have been used for the fitting of the constant and the linear part of the explanatory variable x. This is different from the `s()` function of the **gam** package which uses `s(x,df=4)`, assuming that only the constant term has been fitted separately. After a GAMLSS model containing additive (smoothing) terms is used to fit a specific distribution parameter the following components are (usually) saved for further use. In the output below replace `mu` with `sigma`, `nu` or `"tau"` if a distribution parameter other that `mu` is involved.

`mu.s`: a matrix, each column containing the fitted values of the smoothers used to model the specific parameter. For example given a fitted model say `mod1`, then `mod1$mu.s` would access the additive terms fitted for `mu`.

`mu.var`: a matrix containing the estimated variances of the smoothers.

`mu.df`: a vector containing the extra degrees of freedom used to fit the smoothers.

`mu.lambda`: a vector containing the smoothing parameters (or random effects hyperparameters).

`mu.coefSmo`: a list containing coefficients or other components from the additive smooth fitting.

1.5. The GAMLSS algorithms

There are two basic algorithms used for maximizing the penalized likelihood given in (6). The first, the CG algorithm, is a generalization of the Cole and Green (1992) algorithm—and uses the first and (expected or approximated) second and cross derivatives of the likelihood function with respect to the distribution parameters $\boldsymbol{\theta} = (\mu, \sigma, \nu, \tau)$ for a four parameter distribution. Note that we have dropped the subscripts here to simplify the notation. However for many population probability (density) functions, $f(y|\boldsymbol{\theta})$, the parameters $\boldsymbol{\theta}$ are information orthogonal (since the expected values of the cross derivatives of the likelihood function are zero), e.g., location and scale models and dispersion family models, or approximately so. In this case the simpler RS algorithm, which is a generalization of the algorithm used by Rigby and Stasinopoulos (1996a,b) for fitting mean and dispersion additive models (MADAM), and does not use the cross derivatives, is more suited. The parameters $\boldsymbol{\theta} = (\mu, \sigma)$ are fully information orthogonal for distributions `NBI`, `GA`, `IG`, `LO` and `NO` only in Table 1. Nevertheless, the RS algorithm has been successfully used for fitting all distributions in Tables 1 and 2, although occasionally it can be slow to converge. Note also that the RS algorithm is not a special case of the CG algorithm.

The object of the algorithms is to maximize the penalized likelihood function ℓ_p, given by (6), for fixed hyperparameters $\boldsymbol{\lambda}$. For fully parametric models, (3) or (5), the algorithms maximize the likelihood function ℓ. The algorithms are implemented in the option `method` in the function `gamlss()` where a combination of both algorithms is also allowed. The major advantages of the algorithms are i) the modular fitting procedure (allowing different model diagnostics for each distribution parameter); ii) easy addition of extra distributions; iii) easy addition of extra additive terms; and iv) easily found starting values, requiring initial values for the $\boldsymbol{\theta} = (\mu, \sigma, \nu, \tau)$ rather than for the $\boldsymbol{\beta}$ parameters. The algorithms have generally been found to be stable and fast using very simple starting values (e.g., constants) for the $\boldsymbol{\theta}$ parameters. The function `nlgamlss()` in the package **gamlss.nl** provides a third algorithm for fitting parametric linear or non-linear GAMLSS models as in equations (3) or (5) respectively. However the algorithm needs starting values for all the $\boldsymbol{\beta}$ parameters, rather than $\boldsymbol{\theta} = (\mu, \sigma, \nu, \tau)$, which can be difficult for the user to choose.

Clearly, for a specific data set and model, the (penalized) likelihood can potentially have multiple local maxima. This is investigated using different starting values and has generally not been found to be a problem in the data sets analyzed, possibly due to the relatively large sample sizes used.

Singularities in the likelihood function similar to the ones reported by Crisp and Burridge (1994) can potentially occur in specific cases within the GAMLSS framework, especially when the sample size is small. The problem can be alleviated by appropriate restrictions on the scale parameter (penalizing it for going close to zero).

2. The R packages

2.1. The different packages

The GAMLSS software is implemented in a series of packages in the R language (R Development Core Team 2007) and it is available from the Comprehensive R Archive Network at http://CRAN.R-project.org/. The GAMLSS software currently comprises six different packages:

1. the original **gamlss** package for fitting GAMLSS
2. the **gamlss.cens** package for fitting censored (interval) response variables.
3. the **gamlss.dist** package for additional new distributions
4. the **gamlss.mx** package for fitting finite mixture distributions.
5. the **gamlss.nl** package for fitting non-linear models
6. the **gamlss.tr** package for fitting truncated distributions.

Many `gamlss.family` distributions are implemented in the add on package **gamlss.dist**. In this article we concentrate in the original **gamlss** package, and **gamlss.dist**. The add-on packages will be dealt with separately.

2.2. The different functions

The main function of the original **gamlss** package is `gamlss()`. This function is used to fit a GAMLSS model and consequently to create a `gamlss` object in R. Examples of how the `gamlss()` function can be used will be given in Section 3. Here we list all the available functions within the **gamlss** package by putting them in different groups depending on their functionality. More information about the arguments of any function can be found using `?`, e.g., `?gamlss`.

The following functions are used for fitting or updating a model: `gamlss()`, `refit()`,`update()`, and `histDist()`. Note that the `histDist()` is designed for fitting a parametric distribution to data where no explanatory variables exist.

The functions which extract information from the fitted model are: `AIC()`, `GAIC()`, `coef()`, `deviance()`, `extractAIC()`, `fitted()`, `formula()`, `fv()`, `logLik()`, `lp()`, `lpred()`, `model.frame()`, `model.matrix()`, `predict()`, `print()`, `summary()`, `terms()`, `residuals()` and `vcov()`. Note that saved residuals are the normalized (randomized) quantile residuals from a fitted GAMLSS model, (see Dunn and Smyth 1996, for a definition).

Note that some of the functions above are distribution parameter dependent. That is, these functions have an extra argument `what`, which can be used to specify which of the distribution parameters values are required, i.e., `"mu"`, `"sigma"`, `"nu"` or `tau`. For example `fitted(m1, what="sigma")` would give the fitted values for the σ parameter from model `m1`.

Functions which can be used for selecting a model are: `addterm()`, `dropterm()`, `find.hyper()`, `gamlss.scope()`, `stepGAIC()`, `stepGAIC.CH()`, `stepGAIC.VR()` and `VGD()`.

Functions used for plotting or diagnostics are: `plot()`, `par.plot()`, `pdf.plot()`, `Q.stats()`, `prof.dev()`, `prof.term()`, `rqres.plot()`, `show.link()`, `term.plot()` and `wp()`.

Functions created specially for centile estimation which can be applied if only one explanatory variable is involved are: `centiles()`, `centiles.com()`, `centiles.split()`, `centiles.pred()`, `fitted.plot()`.

The following two functions are used in the definition of a new `gamlss.family` distribution so the casual user does not need them: `make.link.gamlss()`, `checklink()`.

More documentation of the above functions can be found in the **gamlss** R help files by typing a question mark and the appropriate function, e.g., `?centiles`. In the next section we will use a variety of examples to demonstrate the **gamlss** package.

3. Examples

3.1. Introduction

This section gives four examples of using the GAMLSS framework for statistical modelling. In the example in Section 3.2 different distributions are fitted to univariate count data. The example in Section 3.3 illustrates different additive terms of a single explanatory variable used to model the parameters of a continuous response variable distribution. The example in Section 3.4 is a regression situation example where selecting explanatory variables is required. The final example in Section 3.5 demonstrates estimation of smooth centile curves. Note that all the examples here are used to demonstrate the capability of the GAMLSS framework and **gamlss** package and not to answer substantive questions in the data.

3.2. The lice data

The following data come from Williams (1944) and they are frequencies (f) of prisoners with number of head lice (y), for Hindu male prisoners in Cannamore, South India, 1937–1939. Here we fit four discrete distributions the Poisson (`PO`), the negative binomial type I (`NBI`), the Poisson inverse Gaussian (`PIG`) and the Sichel (`SICHEL`). Note that we are using the frequencies as `weights`. More on how to use the `weights` argument correctly can be found in Stasinopoulos *et al.* (2006, Section 3.2.1). The argument `trace=FALSE` in the `gamlss()` function is used to suppress the output at each iteration. The argument `method=mixed(10,50)` used here, for the Sichel distribution, is for speeding the convergence. It instructs `gamlss()` to use the `RS()` algorithm for the first 10 iterations (to stabilize the fitting process) and then (if it has not converged yet) to switch to the `CG()` algorithm and continue with up to 50 iterations. `CG()` converges faster close to the maximum when the distribution parameters are highly non-orthogonal at the maximum.

```
R> library("gamlss.dist")
R> data("lice")
R> mPO <- gamlss(head ~ 1, data = lice, family = PO, weights = freq,
+    trace = FALSE)
R> mNBI <- gamlss(head ~ 1, data = lice, family = NBI, weights = freq,
+    trace = FALSE)
```

```
R> mPIG <- gamlss(head ~ 1, data = lice, family = PIG, weights = freq,
+    trace = FALSE)
R> mSI <- gamlss(head ~ 1, data = lice, family = SICHEL, weights = freq,
+    method = mixed(10, 50), trace = FALSE)
R> AIC(mPO, mNBI, mPIG, mSI)

     df       AIC
mSI   3  4646.198
mNBI  2  4653.687
mPIG  2  4756.275
mPO   1 29174.823
```

From the AIC we conclude that the Sichel model is explaining the data best. The summary of the final fitted model is shown below.

```
R> summary(mSI)

******************************************************************
Family:  c("SICHEL", "Sichel")

Call:
gamlss(formula = head ~ 1, family = SICHEL, data = lice, weights = freq,
    method = mixed(10, 50), trace = FALSE)

Fitting method: mixed(10, 50)

------------------------------------------------------------------
Mu link function:  log
Mu Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept)     1.927     0.07915    24.35  8.38e-105

------------------------------------------------------------------
Sigma link function:  log
Sigma Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept)     4.863      0.2095    23.21  4.739e-97

------------------------------------------------------------------
Nu link function:  identity
Nu Coefficients:
              Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  0.0007871     0.01570  0.05013      0.96

------------------------------------------------------------------
No. of observations in the fit:  1083
Degrees of Freedom for the fit:  3
```

```
      Residual Deg. of Freedom:  1080
                      at cycle:  5

Global Deviance:     4640.198
            AIC:     4646.198
            SBC:     4661.16
*******************************************************************
```

Hence the fitted SICHEL model for the head lice data is given by $y \sim \texttt{SICHEL}(\hat{\mu}, \hat{\sigma}, \hat{\nu})$ where $\hat{\mu} = \exp(1.927) = 6.869$ and $\hat{\sigma} = \exp(4.863) = 129.4$ and $\hat{\nu} = 0.0007871$ almost identical to zero. The profile global deviance plot for the parameter ν is shown in Figure 1. The figure is created using the code `prof.dev(mSI, which="nu", min=-.12, max=.1, step=.01)` which also produces a 95% confidence interval for ν as $(-0.08532, 0.08867)$.

Figure 2 shows a plot of the fitted SICHEL model created by the following R commands. Note that starting from the already fitted model, mSI using the argument `start.from=mSI` speeds the process.

```
R> mSI <- histDist(lice$head, "SICHEL", freq = lice$freq, xmax = 10,
+    main = "Sichel distribution", start.from = mSI, xlim = c(0, 8.75),
+    trace = FALSE)
```

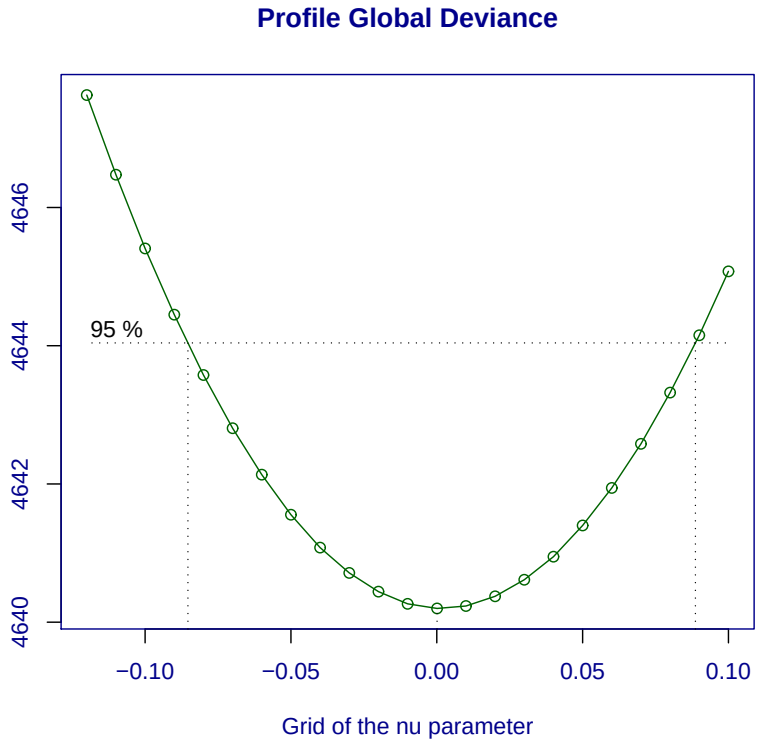

Figure 1: The profile deviance plot for ν from the fitted model mSI.

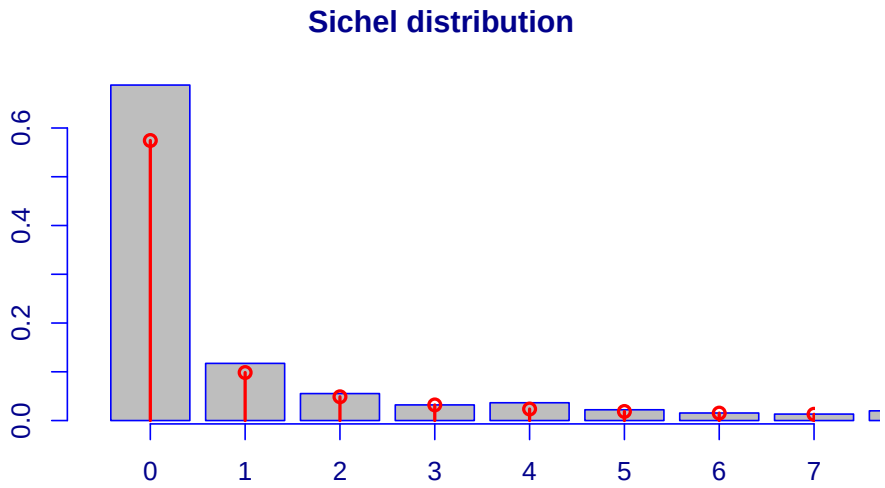

Figure 2: Sample distribution of the lice data (grey blocks) with the fitted probabilities for the Sichel distribution respectively (red bars).

3.3. The CD4 data

The following data are given by Wade and Ades (1994) and they refer to CD4 counts (cd4) from uninfected children born to HIV-1 mothers and the age in years of the child. Here we input and plot the data in Figure 3. This is a simple regression example with only one explanatory variable, the age, which is a continuous variable. The response while, strictly speaking is a count, is sufficiently large for us to treat it at this stage as a continuous response variable.

```
R> library("gamlss.dist")
R> data("CD4")
R> plot(cd4 ~ age, data = CD4)
```

There are several striking features in this specific set of data in Figure 3. The first has to do with the relationship between the mean of cd4 and age. It is hard to see from the plot whether this relationship is linear or not. The second has to do with the heterogeneity of variance in the response variable cd4. It appears that the variation in cd4 is decreasing with age. The final problem has to do with the distribution of cd4 given the age. Is this distribution normal? It is hard to tell from the figure but probably we will need a more flexible distribution. Traditionally, problems of this kind were dealt with by a transformation in the response variable or a transformation in both in the response and the explanatory variable(s). One could hope that this would possibly correct for some or all the above problems simultaneously. Figure 4 shows plots where several transformations for cd4 and age were tried. It is hard to see how we can improve the situation.

```
R> op <- par(mfrow = c(3, 4), mar = par("mar") + c(0, 1, 0, 0),
```

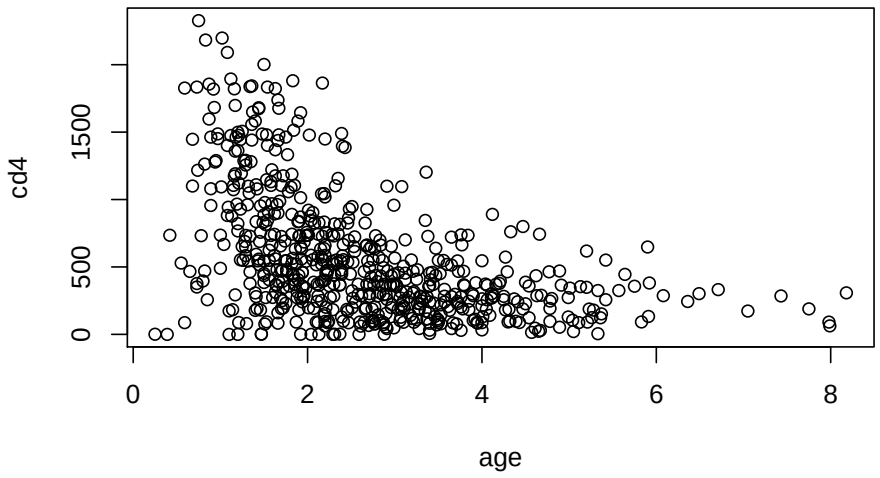

Figure 3: The plot of the CD4 data.

```
+    pch = "+", cex = 0.45, cex.lab = 1.6, cex.axis = 1.6)
R> page <- c("age^-0.5", "log(age)", "age^.5", "age")
R> pcd4 <- c("cd4^-0.5", "log(cd4+0.1)", "cd4^.5")
R> for (i in 1:3) {
+    yy <- with(CD4, eval(parse(text = pcd4[i])))
+    for (j in 1:4) {
+        xx <- with(CD4, eval(parse(text = page[j])))
+        plot(yy ~ xx, xlab = page[j], ylab = pcd4[i])
+    }
+ }
R> par(op)
```

Within the GAMLSS framework we can deal with these problems one at the time. First we start with the relationship between the mean of cd4 and age. We will fit orthogonal polynomials of different orders to the data and choose the best using an GAIC criterion. At the moment we fit a constant variance and a default normal distribution.

```
R> con <- gamlss.control(trace = FALSE)
R> m1 <- gamlss(cd4 ~ age, sigma.fo = ~1, data = CD4, control = con)
R> m2 <- gamlss(cd4 ~ poly(age, 2), sigma.fo = ~1, data = CD4, control = con)
R> m3 <- gamlss(cd4 ~ poly(age, 3), sigma.fo = ~1, data = CD4, control = con)
...
R> m8 <- gamlss(cd4 ~ poly(age, 8), sigma.fo = ~1, data = CD4, control = con)
```

First we compare the model using the Akaike Information criterion (AIC) which has penalty $k = 2$ for each parameter in the model, (the default value in the function GAIC()). Next we compare the models using Schwatz Bayesian Criterion (SBC) which uses penalty $k = \log(n)$.

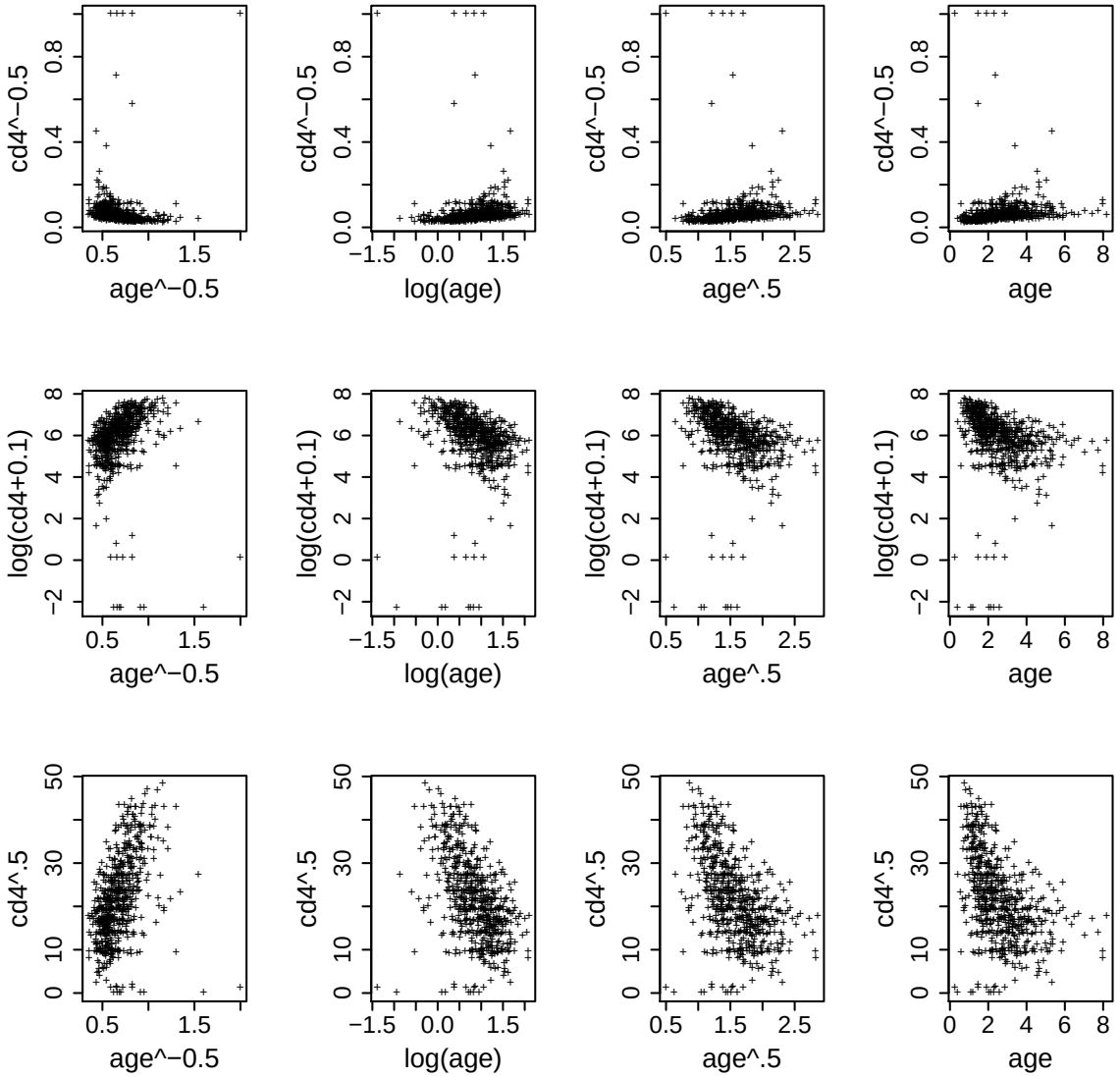

Figure 4: The CD4 data with various transformations for cd4 and age.

```
R> GAIC(m1, m2, m3, m4, m5, m7, m8)
```

```
   df       AIC
m7  9 8963.263
m8 10 8963.874
m5  7 8977.383
m4  6 8988.105
m3  5 8993.351
m2  4 8995.636
m1  3 9044.145
```

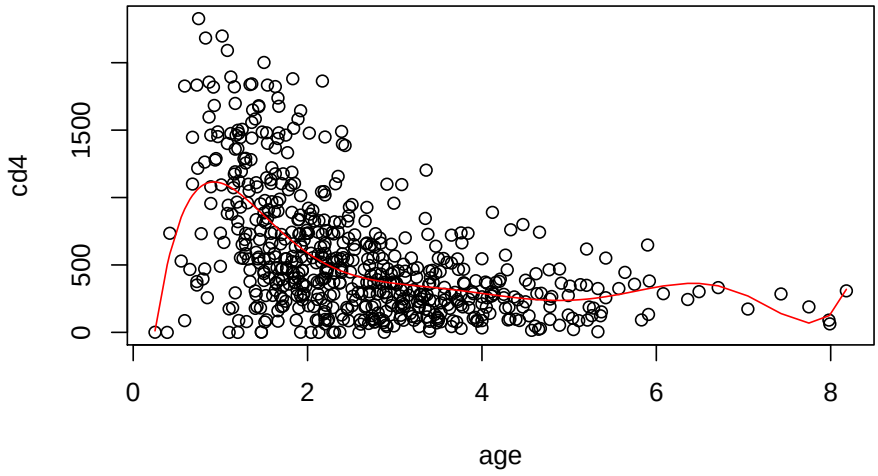

Figure 5: The CD4 data and the fitted values using polynomial of degree 7 in age.

```
R> GAIC(m1, m2, m3, m4, m5, m7, m8, k = log(length(CD4$age)))

   df      AIC
m7  9 9002.969
m8 10 9007.992
m5  7 9008.266
m2  4 9013.284
m4  6 9014.576
m3  5 9015.410
m1  3 9057.380

R> plot(cd4 ~ age, data = CD4)
R> lines(CD4$age[order(CD4$age)], fitted(m7)[order(CD4$age)], col = "red")
```

Remarkably with both AIC and SBC model m7, with a polynomial of degree 7, comes as the best model. Unfortunately the fitted values for the mean of cd4 shown together with the data in Figure 5 look rather unconvincing. The line is too wobbly at the end of the age range, trying to be very close to the data. This is a typical behavior of polynomial fitting.

Now we will try alternatives methods, two parametric, using *fractional polynomials* and piecewise polynomials, and one non-parametric smoothing method using *cubic splines*. We start first with the fractional polynomials. Fractional polynomials were introduced by Royston and Altman (1994). The function fp() which we are going to use to fit them works in gamlss() as an additive smoother. It can be used to fit the best (fractional) polynomials within a specific set of possible power values. Its argument npoly determines whether one, two or three fractional polynomials will be used in the fitting. For example with npoly=3 the following polynomials functions are fitted to the data $\beta_0 + \beta_1 x^{p_1} + \beta_2 x^{p_2} + \beta_3 x^{p_3}$ where p_j, for $j = 1, 2, 3$

can take any values within the set $(-2, -1, -0.5, 0, 0.5, 1, 2, 3)$. If two powers, p_j's, happen to be identical then the two terms $\beta_{1j}x^{p_j}$ and $\beta_{2j}x^{p_j}\log(x)$ are fitted instead. Similarly if three powers p_j's are identical the terms fitted are $\beta_{1j}x^{p_j}$, $\beta_{2j}x^{p_j}\log(x)$ and $\beta_{3j}x^{p_j}[\log(x)]^2$. Here we fit fractional polynomials with one, two and three terms respectively and we choose the best using GAIC.

```
R> m1f <- gamlss(cd4 ~ fp(age, 1), sigma.fo = ~1, data = CD4, control = con)
R> m2f <- gamlss(cd4 ~ fp(age, 2), sigma.fo = ~1, data = CD4, control = con)
R> m3f <- gamlss(cd4 ~ fp(age, 3), sigma.fo = ~1, data = CD4, control = con)
R> GAIC(m1f, m2f, m3f)

    df      AIC
m3f  8 8966.375
m2f  6 8978.469
m1f  4 9015.321

R> GAIC(m1f, m2f, m3f, k = log(length(CD4$age)))

    df      AIC
m3f  8 9001.669
m2f  6 9004.940
m1f  4 9032.968

R> m3f

Family:  c("NO", "Normal")
Fitting method: RS()

Call:  gamlss(formula = cd4 ~ fp(age, 3), sigma.formula = ~1, data = CD4,
    control = con)

Mu Coefficients:
(Intercept)   fp(age, 3)
      557.5           NA
Sigma Coefficients:
(Intercept)
       5.93

 Degrees of Freedom for the fit: 8 Residual Deg. of Freedom   601
Global Deviance:     8950.37
            AIC:     8966.37
            SBC:     9001.67

R> m3f$mu.coefSmo

[[1]]
[[1]]$coef
```

```
      one
-599.2970 1116.7924 1776.1937  698.6097

[[1]]$power
[1] -2 -2 -2

[[1]]$varcoeff
[1]  2016.238  4316.743 22835.146  7521.417
```

```
R> plot(cd4 ~ age, data = CD4)
R> lines(CD4$age[order(CD4$age)], fitted(m1f)[order(CD4$age)], col = "blue")
R> lines(CD4$age[order(CD4$age)], fitted(m2f)[order(CD4$age)], col = "green")
R> lines(CD4$age[order(CD4$age)], fitted(m3f)[order(CD4$age)], col = "red")
```

Both AIC and BSC favour the model `m3f` with three fractional polynomial terms. Note that by printing `m3f` the model for μ gives a value of 557.5 for the "Intercept" and NULL for the coefficient for `fp(age, 3)`. This is because within the backfitting the constant is fitted first and then the fractional polynomial is fitted to the partial residuals of the constant model. As a consequence the constant is fitted twice. The proper coefficients and the power transformations of the fractional polynomials can be obtained using the `mu.coefSmo` component of the `gamlss` fitted object. For the `CD4` data all powers happens to be -2 indicating that the following terms are fitted in the model, age^{-2}, $age^{-2}\log(age)$ and $age^{-2}\left[\log(age)\right]^2$. Hence the fitted model `m3f` is given by $cd4 \sim \texttt{NO}(\hat{\mu}, \hat{\sigma})$, where $\hat{\mu} = 557.5 - 599.3 + 1116.8\, age^{-2} + 17776.2\, age^{-2}\log(age) + 698.6\, age^{-2}\left[\log(age)^2\right]$. Figure 6 shows the best fitted models using one, two or three fractional polynomial terms. The situation remains unconvincing. None of the models seem to fit particular well.

Next we will fit piecewise polynomials using the R function `bs()`. We try different degrees of freedom (effectively different number of knots) and we choose the best model using AIC and SBC.

```
R> m2b <- gamlss(cd4 ~ bs(age), data = CD4, trace = FALSE)
R> m3b <- gamlss(cd4 ~ bs(age, df = 3), data = CD4, trace = FALSE)
R> m4b <- gamlss(cd4 ~ bs(age, df = 4), data = CD4, trace = FALSE)
R> m5b <- gamlss(cd4 ~ bs(age, df = 5), data = CD4, trace = FALSE)
R> m6b <- gamlss(cd4 ~ bs(age, df = 6), data = CD4, trace = FALSE)
R> m7b <- gamlss(cd4 ~ bs(age, df = 7), data = CD4, trace = FALSE)
R> m8b <- gamlss(cd4 ~ bs(age, df = 8), data = CD4, trace = FALSE)
R> GAIC(m2b, m3b, m4b, m5b, m6b, m7b, m8b)
```

```
    df      AIC
m7b  9 8959.519
m6b  8 8960.353
m8b 10 8961.073
m5b  7 8964.022
m4b  6 8977.475
m2b  5 8993.351
m3b  5 8993.351
```

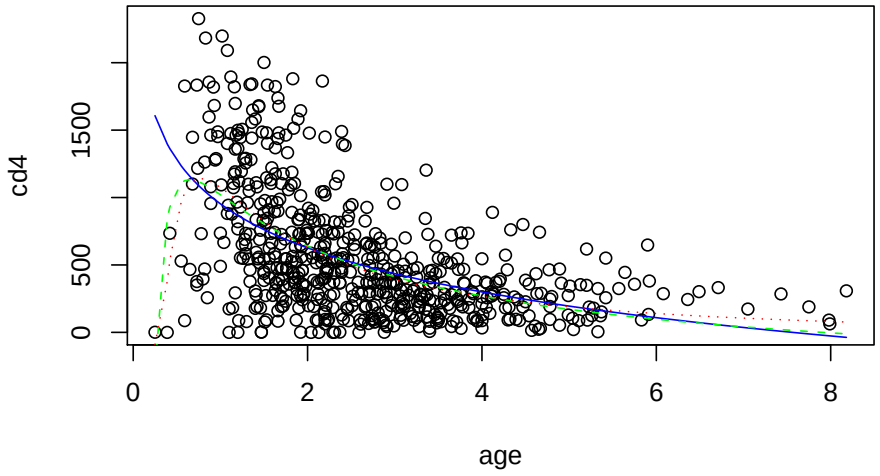

Figure 6: The CD4 data and the best fitting fractional polynomials in age with one (solid), two (dashed) and three (dotted) terms, respectively.

```
R> GAIC(m2b, m3b, m4b, m5b, m6b, m7b, m8b, k = log(length(CD4$age)))
```

```
    df      AIC
m5b  7 8994.904
m6b  8 8995.648
m7b  9 8999.225
m4b  6 9003.946
m8b 10 9005.191
m2b  5 9015.410
m3b  5 9015.410
```

The best model with AIC uses 7 degrees of freedom while with SBC 5. Figure 7 shows the fitted models using 5, 6 and 7 degrees of freedom for the polynomial terms in age.

We will proceed by fitting smoothing cubic splines to the data. In smoothing splines the problem is how to choose a sensible value for the smoothing parameter λ. The smoothing parameter is a function of the effective degrees of freedom, so we will use the following procedure: we will use the optim() function in R to find the model with an the optimal (effective) degrees of freedom according to an GAIC. Again we do not commit ourselves to what penalty we should use in the GAIC but we will try both AIC and SBC.

```
R> fn <- function(p) AIC(gamlss(cd4 ~ cs(age, df = p[1]), data = CD4,
+    trace = FALSE), k = 2)
R> opAIC <- optim(par = c(3), fn, method = "L-BFGS-B", lower = c(1),
+    upper = c(15))
R> fn <- function(p) AIC(gamlss(cd4 ~ cs(age, df = p[1]), data = CD4,
```

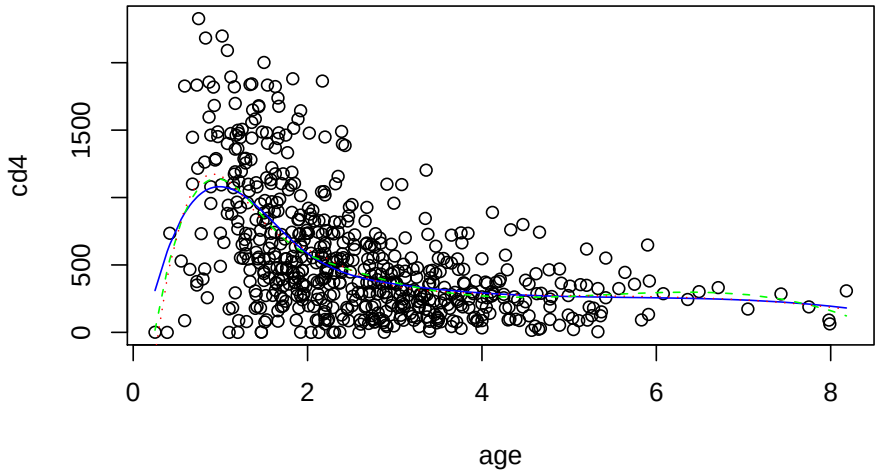

Figure 7: The CD4 data and the best fitting piecewise polynomials in age with 5 (solid), 6 (dashed) and 7 (dotted), degrees of freedom, respectively.

```
+    trace = FALSE), k = log(length(CD4$age)))
R> opSBC <- optim(par = c(3), fn, method = "L-BFGS-B", lower = c(1),
+    upper = c(15))
R> opAIC$par

[1] 10.85157

R> opSBC$par

[1] 1.854689

R> maic <- gamlss(cd4 ~ cs(age, df = 10.85), data = CD4, trace = FALSE)
R> msbc <- gamlss(cd4 ~ cs(age, df = 1.85), data = CD4, trace = FALSE)
```

According to AIC the best model is the one with degrees of freedom $10.85 \approx 11$. This model seems to overfit the data as can been seen in Figure 8, (green continuous line). This is typical behaviour of AIC when it is used in this context. Note that 11 degrees of freedom in the fit refers to the extra degrees of freedom after the constant and the linear part is fitted to the model, so the overall degrees of freedom are 13. The best model using SBC has $1.854 \approx 2$ degrees of freedom for smoothing (i.e., 4 overall) is shown in Figure 8 (blue dashed line). It fits well for most of the observations but not at small values of age. It appears that we need a model with smoothing degrees of freedom for the cubic spline with a value between 2 and 11 (i.e., 4 and 13 overall).

Given that the smooth cubic spline model for μ appears reasonable, we proceed by looking at a suitable models for $\log \sigma$ (since the default link function for σ for the normal NO distribution

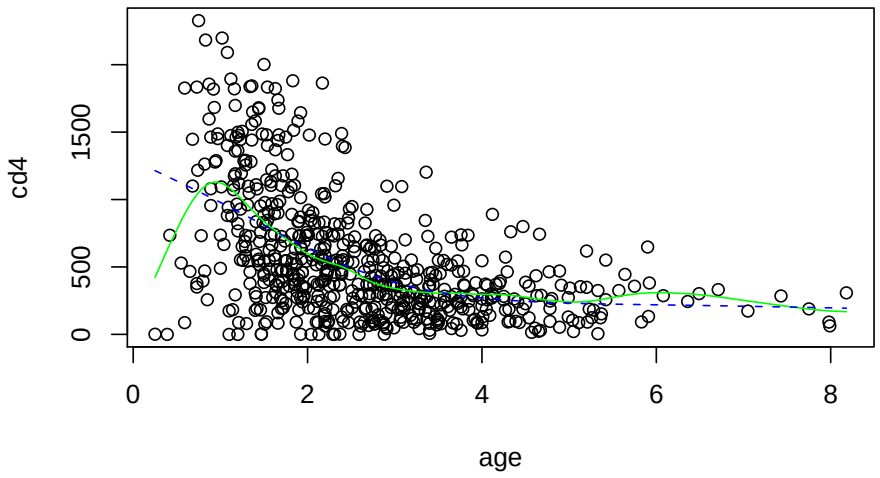

Figure 8: The `CD4` data and two different cubic spline fits in `age`, with four, (solid), and twelve, (dashed), total effective degrees of freedom in the fit.

is a log link). We try a smooth cubic splines model for both terms. The following code will give us the best choice of degrees of freedom for both μ and $\log(\sigma)$ according to AIC. Note the fit of model `m1` is used as the starting values.

```
R> m1 <- gamlss(cd4 ~ cs(age, df = 10), sigma.fo = ~cs(age, df = 2),
+    data = CD4, trace = FALSE)
R> fn <- function(p) AIC(gamlss(cd4 ~ cs(age, df = p[1]), sigma.fo = ~cs(age,
+    p[2]), data = CD4, trace = FALSE, start.from = m1), k = 2)
R> opAIC <- optim(par = c(8, 3), fn, method = "L-BFGS-B", lower = c(1,
+    1), upper = c(15, 15))
R> opAIC$par

[1] 3.717336 1.808830
```

The resulting degrees of freedom for μ and $\log(\sigma)$ are $(3.72, 1.81)$. The degrees of freedom for μ are lower than expected from the previous analysis. It appears that by picking a suitable model for σ the model for μ is less complicated. Rerunning the code for SBC results in estimated degrees of freedom $(2.55, 0.93)$. Note that the `lower` argument in `optim()` had to change to `lower=c(0.1,0.1)` allowing the degrees of freedom to be lower that 1. We fit now the two models.

```
R> m42 <- gamlss(cd4 ~ cs(age, df = 3.72), sigma.fo = ~cs(age, df = 1.81),
+    data = CD4, trace = FALSE)
R> m31 <- gamlss(cd4 ~ cs(age, df = 2.55), sigma.fo = ~cs(age, df = 0.93),
+    data = CD4, trace = FALSE)
```

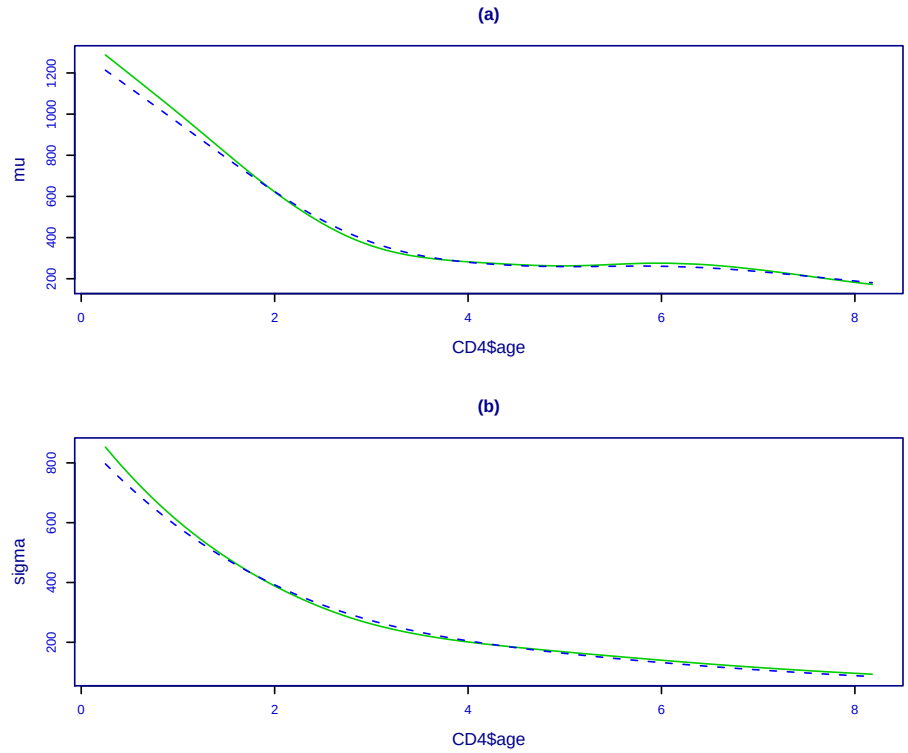

Figure 9: A plot of the fitted μ and σ values against `age` for models `m42` (in solid green) and `m31` (in dashed blue).

Figure 9 shows the fitted values for both the models. The plot is obtained using the command `fitted.plot(m42,m31, x=CD4$age, line.type=TRUE)`. The function `fitted.plot()` is appropriate when only one explanatory variable is fitted to the data. The models are almost identical apart from early `age` where the SBC model has a slightly lower mean and standard deviation of `cd4` than the AIC. The validation generalized deviance function `VGD()` provides an alternative way of tuning the degrees of freedom in a smoothing situation. It is suitable for large sets of data where we can afford to use part of the data for fitting the model (training) and part for validation. Here we demonstrate how it can be used.

```
R> set.seed(1234)
R> rSample6040 <- sample(2, length(CD4$cd4), replace = T, prob = c(0.6,
+    0.4))
R> fn <- function(p) VGD(cd4 ~ cs(age, df = p[1]), sigma.fo = ~cs(age,
+    df = p[2]), data = CD4, rand = rSample6040)
R> op <- optim(par = c(3, 1), fn, method = "L-BFGS-B", lower = c(1,
+    1), upper = c(10, 10))
R> op$par

[1] 4.779947 1.376534
```

The resulting degrees of freedom $(4.78, 1.38)$ in this instance are not very different from the ones we obtained from AIC and SBC methods.

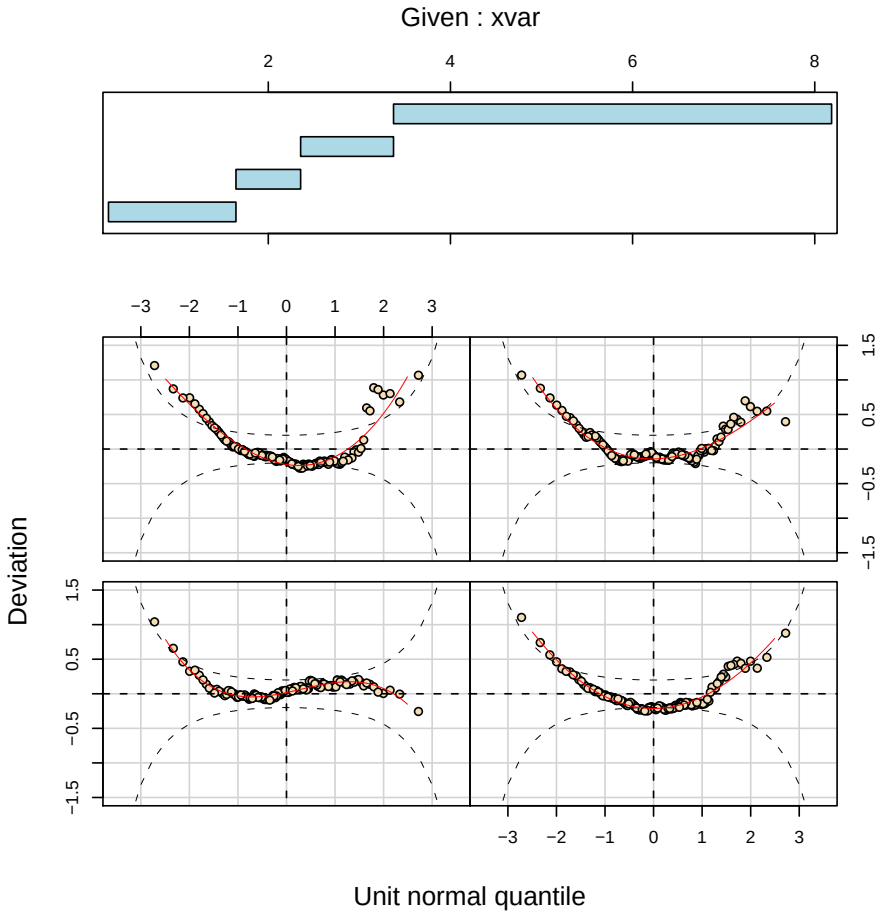

Figure 10: A worm plot of the residuals from models `m42`.

Figure 10 shows a worm plots from the residuals of model `m42`. Worm plots were introduced by van Buuren and Fredriks (2001) and they are covered in more detail in Section 3.5. The important point here is that quadratic and cubic shapes in a worm plot indicate the presence of skewness and kurtosis respectively in the residuals (within the corresponding range of the explanatory variable, i.e., `CD4$age`). That is, the normal distribution fitted so far to the data is not appropriate. Now we try to identify a suitable distribution for the response variable. There are zeros in the response so unless we shift them to the right by a small amount, we must model it with distributions accepting zeros. Here we try the t (`TF`), power exponential (`PE`), skew exponential power type 3 (`SEP3`) and sinh arcsinh (`SHASH`) distributions. We use the `update` function and start from the normal `m42` model.

```
R> library("gamlss.dist")
R> m42TF <- update(m42, family = TF)
R> m42PE <- update(m42, family = PE)
R> m42SEP3 <- update(m42, family = SEP3, method = mixed(30, 100))
R> m42SHASH <- update(m42, family = SHASH, method = mixed(20, 100))
R> GAIC(m42, m42TF, m42PE, m42SEP3, m42SHASH)
```

```
                df      AIC
m42SEP3  11.531173 8690.531
m42SHASH 11.530125 8703.417
m42TF    10.531045 8785.149
m42PE    10.531061 8790.081
m42       9.529738 8790.437
```

```
R> GAIC(m42, m42TF, m42PE, m42SEP3, m42SHASH, k = log(length(CD4$age)))
```

```
                df      AIC
m42SEP3  11.531173 8741.405
m42SHASH 11.530125 8754.286
m42TF    10.531045 8831.611
m42       9.529738 8832.481
m42PE    10.531061 8836.542
```

The SEP3 distribution seems that fit this data best according to both the AIC and SBC criteria. It can be shown that by adding a linear model to the ν models i.e., `nu.fo=~age` improves the AIC but not the SBC.

3.4. The third party claims

The data used here are provided by Gillian Heller and can be found in de Jong and Heller (2007). They are third party insurance data. Third party is a compulsory insurance for vehicle owners in Australia. It insures vehicle owners against injury caused to other drivers, passengers or pedestrians, as a result of an accident. This data set records the number of third party claims, `Claims`, in a twelve month period between 1984–1986 in each of 176 geographical areas (local government areas, `LGA`) in New South Wales, Australia. Areas are grouped into thirteen statistical divisions (`SD`). Other recorded variables are the number of accidents, `Accidents`, the number of people killed or injured (`KI`), population density (`Pop_density`) and population (`Population`) with all variables classified according to area. In most of the analysis here we will use the log values for `Population`, `Accidents`, `KI` and `Pop_density` which are denoted as `L_Population`, `L_Accidents`, `L_KI` and `L_Popdensity` respectively. Figure 11 shows the numbers of claims against the rest of the continuous variables in the data.

```
R> library"(gamlss.dist")
R> data("LGAclaims")
R> with(LGAclaims, plot(data.frame(Claims, L_Popdensity, L_KI, L_Accidents,
+    L_Population)))
```

We start with a model for μ including all the explanatory variables. To check overdispersion we compare the Poisson distribution model (`PO`) against the negative binomial type I model (`NBI`). Since the reduction in deviance between those two models is enormous (4606.169) with only one extra degree of freedom, for the rest of the chapter we will use the negative binomial distribution model in order to choose terms for the μ and (possibly) for the σ models.

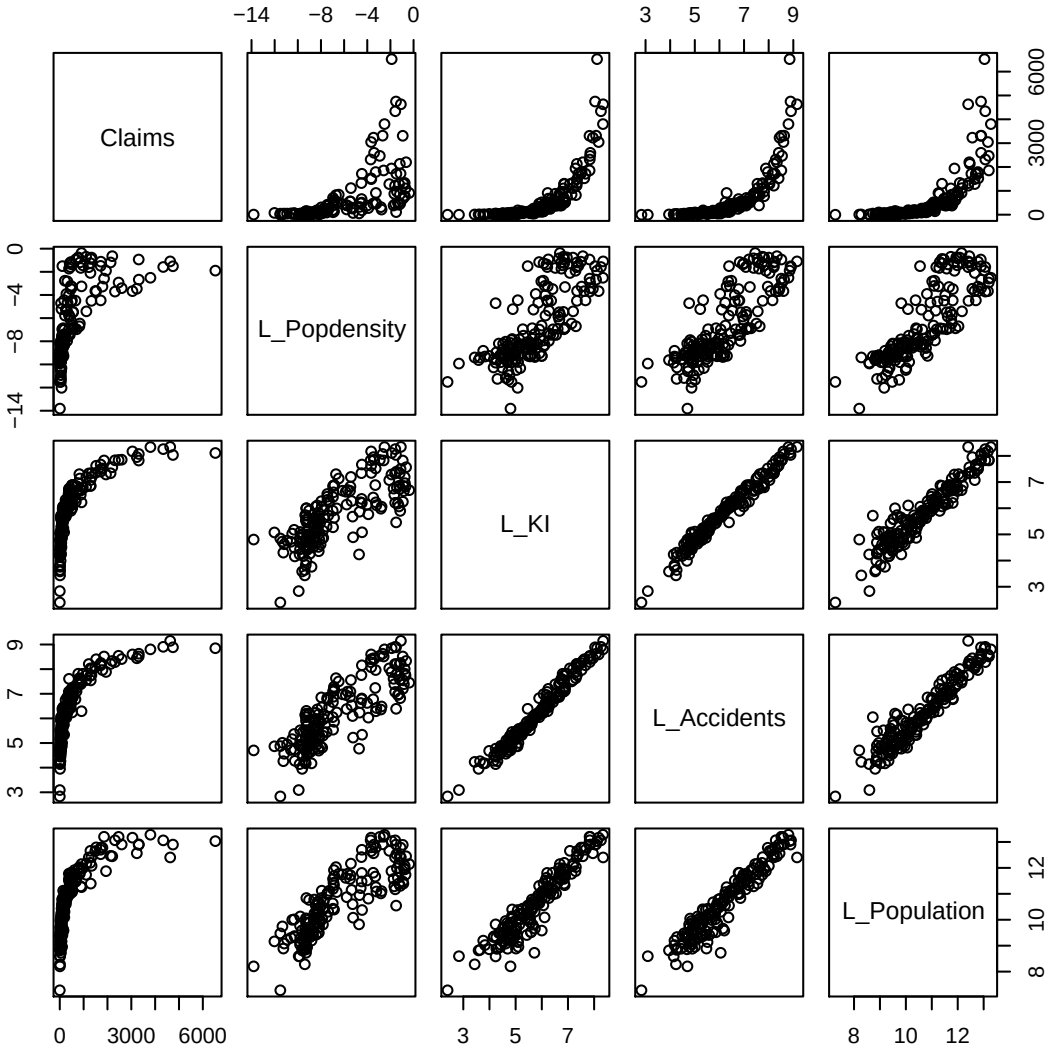

Figure 11: The plot of the third party insurance data.

```
R> m0 <- gamlss(Claims ~ factor(SD) + L_Popdensity + L_KI + L_Accidents +
+    L_Population, data = LGAclaims, family = PO)

GAMLSS-RS iteration 1: Global Deviance = 6487.73
GAMLSS-RS iteration 2: Global Deviance = 6487.73

R> m1 <- gamlss(Claims ~ factor(SD) + L_Popdensity + L_KI + L_Accidents +
+    L_Population, data = LGAclaims, family = NBI)

GAMLSS-RS iteration 1: Global Deviance = 1883.942
GAMLSS-RS iteration 2: Global Deviance = 1881.561
GAMLSS-RS iteration 3: Global Deviance = 1881.561
```

```
R> deviance(m0) - deviance(m1)

[1] 4606.169
```

Selection of variables

There are five functions within the **gamlss** package to assist with selecting explanatory variable terms. The first two are the functions `addterm()` and `dropterm()` which allow the addition or removal of a term in a model respectively. Those two functions are building blocks for the functions `stepGAIC.VR()` and `stepGAIC.CH()` suitable for stepwise selection of models. Both functions perform the stepwise model selection using a generalized Akaike information criterion. The function `stepGAIC.VR()` is based on the function `stepAIC()` given in the package **MASS** of Venables and Ripley (2002), where more details and examples of the function can be found, with the additional property that it allows selection of terms for any selected distribution parameter. The function `stepGAIC.CH()` is based on the S-PLUS function `step.gam()` (see Chambers and Hastie 1992, for more information) and is more suited to models with smoothing additive terms in them. Again the function `stepGAIC.CH()` is generalized here so it can be used for any distribution parameter within the **gamlss** packages. The main difference between `stepGAIC.VR()` and `stepGAIC.CH()` lies in the use of the `scope` argument. The function `stepGAIC()` combines the two functions by having a extra argument `additive` which when set to `TRUE` the `stepGAIC.CH()` is used, otherwise the `stepGAIC.VR()` is used. `stepGAIC.VR()` is the default.

The functions `addterm()` and `dropterm()` are generic functions with their original definitions defined at the package MASS of Venables and Ripley (2002). This package has to be attached, (i.e., `library("MASS")`), first before their method for classes `gamlss` can be used. The functions `stepGAIC()`, `stepGAIC.VR()` and `stepGAIC.CH()` can be used without attaching MASS. We shall now use the `dropterm()` to check if model `m1` can be simplified by dropping any of the existing terms in μ and the function `addterm()` to check whether two way interactions of the existing terms are needed. chunk 4

```
R> library("MASS")
R> mD <- dropterm(m1, test = "Chisq")
R> mD

Single term deletions for
mu

Model:
Claims ~ factor(SD) + L_Popdensity + L_KI + L_Accidents + L_Population
             Df     AIC     LRT   Pr(Chi)
<none>          1917.56
factor(SD)   12 1931.45   37.89  0.000160 ***
L_Popdensity  1 1970.84   55.28 1.045e-13 ***
L_KI          1 1961.00   45.44 1.576e-11 ***
L_Accidents   1 1920.63    5.07  0.024362 *
L_Population  1 1923.02    7.46  0.006313 **
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R> mA <- addterm(m1, scope = ~(factor(SD) + L_Popdensity + L_KI +
+    L_Accidents + L_Population)^2, test = "Chisq")
R> mA

Single term additions for
mu

Model:
Claims ~ factor(SD) + L_Popdensity + L_KI + L_Accidents + L_Population
                          Df     AIC     LRT Pr(Chi)
<none>                       1917.56
factor(SD):L_Popdensity   12 1927.46   14.10 0.29424
factor(SD):L_KI           12 1921.44   20.12 0.06485 .
factor(SD):L_Accidents    12 1923.62   17.94 0.11764
factor(SD):L_Population   12 1923.94   17.62 0.12759
L_Popdensity:L_KI          1 1919.46    0.10 0.74753
L_Popdensity:L_Accidents   1 1919.25    0.32 0.57410
L_Popdensity:L_Population  1 1918.21    1.36 0.24430
L_KI:L_Accidents           1 1919.51    0.05 0.82555
L_KI:L_Population          1 1919.29    0.27 0.60173
L_Accidents:L_Population   1 1919.28    0.28 0.59547
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the Chi square tests, no terms can be left out and no two way interaction is needed. Since we established that adding or dropping terms in μ is not beneficial there is no point using `stepGAIC.VR()` for modelling the μ parameter with linear terms. Instead we will use `stepGAIC.CH()` trying to establish if smoothing terms are needed in the μ model. The function `gamlss.scope()`—similar to the function `gam.scope()` in the **gam** package (Hastie 2006)—is used here to create the different models to be fitted.

```
R> gs <- gamlss.scope(model.frame(Claims ~ factor(SD) + L_Popdensity +
+    L_KI + L_Accidents + L_Population, data = LGAclaims))
R> gs

$`factor(SD)`
~1 + factor(SD)

$L_Popdensity
~1 + L_Popdensity + cs(L_Popdensity)

$L_KI
~1 + L_KI + cs(L_KI)
```

```
$L_Accidents
~1 + L_Accidents + cs(L_Accidents)

$L_Population
~1 + L_Population + cs(L_Population)

R> m2 <- stepGAIC.CH(m1, scope = gs, k = 2)

Distribution parameter:  mu
Start:  Claims ~ factor(SD) + L_Popdensity + L_KI + L_Accidents +
    L_Population; AIC= 1917.561
...
...
Trial:  Claims ~  factor(SD) + cs(L_Popdensity) + L_KI + L_Accidents +
 cs(L_Population); AIC= 1918.366

R> m2$anova

          From               To        Df  Deviance Resid. Df Resid. Dev
1                                      NA        NA  158.0000   1881.561
2  L_Accidents  cs(L_Accidents) -3.000838 -9.596529  154.9992   1871.964
3 L_Population cs(L_Population) -3.000782 -6.965547  151.9984   1864.999
4 L_Popdensity cs(L_Popdensity) -3.000727 -6.540878  148.9977   1858.458
       AIC
1 1917.561
2 1913.966
3 1913.002
4 1912.463

R> formula(m2, "mu")

Claims ~ factor(SD) + cs(L_Popdensity) + L_KI + cs(L_Accidents) +
    cs(L_Population)
```

The resulting `gamlss` object has an extra component `anova` which summarizes the selection process. The best model includes smoothing terms for `L_Popdensity`, `L_Accidents` and `L_Population` but not for `L_KI`. Plotting the smoothing additive functions can be achieved using the function `term.plot()`, see Figure 12.

```
R> op <- par(mfrow = c(3, 2))
R> term.plot(m2, se = T, partial = T)
R> par(op)
```

Given that we have established a good model for μ, we proceed to find a good model for σ. We start first with linear terms but we exclude the factor `SD` since some of the levels of the factor have very few observations.

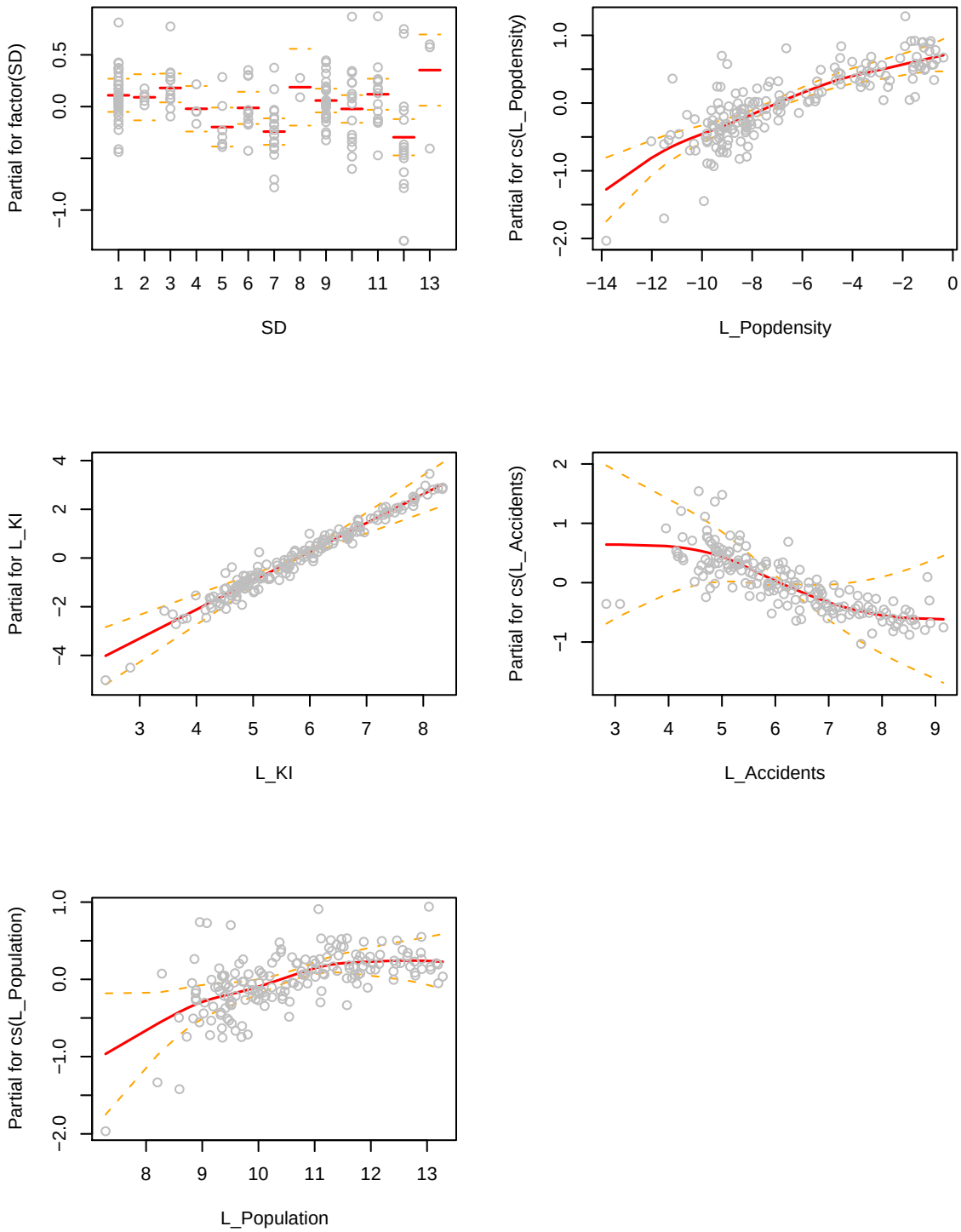

Figure 12: The additive terms plot for the μ model.

```
R> m11 <- stepGAIC.VR(m2, scope = ~L_Popdensity + L_KI + L_Accidents +
+    L_Population, what = "sigma", k = 2)

Distribution parameter:  sigma
Start:  AIC= 1912.46
 ~1
...
...
Final
sigma
 Model:
~L_Population + L_Popdensity + L_KI


            Step        Df  Deviance Resid. Df Resid. Dev      AIC
1                                     148.9977   1858.458 1912.463
2 + L_Population 0.9983966 8.4984558  147.9993   1849.960 1905.961
3  + L_Accidents 0.9999946 8.5717312  146.9993   1841.388 1899.389
4 + L_Popdensity 0.9999033 3.0910247  145.9994   1838.297 1898.298
5         + L_KI 1.0000618 2.2042417  144.9993   1836.093 1898.094
6  - L_Accidents 1.0000190 0.2617682  145.9993   1836.354 1896.356
```

Note that the argument `what` is used here to determine which distribution parameter is to be modelled. Here variables `L_Population`, `L_KI` and `L_Accidents` were found important in explaining the σ parameter. The model chosen using AIC appears over complicated. Maybe a higher penalty for GAIC would be more appropriate here.

3.5. Head circumference data

In this example we demonstrate the use of the **gamlss** package to constructing centile curves. GAMLSS was adopted by the World Health Organization for constructing the world standard child growth curves (see WHO Multicentre Growth Reference Study Group 2006).

The Fourth Dutch Growth Study (Fredriks, van Buuren, Burgmeijer, Meulmeester, Beuker, Brugman, Roede, Verloove-Vanhorick, and Wit 2000a; Fredriks, van Buuren, Wit, and Verloove-Vanhorick 2000b) is a cross-sectional study that measures growth and development of the Dutch population between the ages 0 and 22 years. The study measured, among other variables, height, weight, head circumference and age for 7482 males and 7018 females.

Here the head circumference (y) of the males is analyzed with explanatory variable $x = age^{\xi}$, the transformed age. There are 7040 observations, as there were 442 missing values for head circumference. The data are plotted in Figure 16. The data were previously analyzed by van Buuren and Fredriks (2001) who found strong evidence of kurtosis which they were unable to model. The data were subsequently analyzed by Rigby and Stasinopoulos (2006) using a `BCT` distribution to model the kurtosis.

Given $X = x$, Y is modelled here by a Box-Cox t distribution, $\texttt{BCT}(\mu, \sigma, \nu, \tau)$, where the parameters μ, σ, ν, and τ are modelled, using a special case of the GAMLSS model (2), as smooth non-parametric functions of x, i.e., $Y \sim \texttt{BCT}(\mu, \sigma, \nu, \tau)$ where

$$g_1(\mu) = h_1(x)$$

$$g_2(\sigma) = h_2(x) \qquad (10)$$
$$g_3(\nu) = h_3(x)$$
$$g_4(\tau) = h_4(x)$$

and, for $k = 1, 2, 3, 4$, $g_k(.)$ are known monotonic link functions, and $h_k(x)$ are smooth non-parametric functions of x.

The model selection procedure comprised of choosing link functions $g_k(.)$, for $k = 1, 2, 3, 4$, ξ in the transformation for age, $x = age^{\xi}$, and the total (effective) degrees of freedom for the smooth non-parametric cubic spline functions $h_k(x)$ for $k = 1, 2, 3, 4$, denoted df_μ, df_σ, df_ν and df_τ respectively. Identity link functions were chosen for μ and ν, while log link functions were chosen for σ and τ (to ensure $\sigma > 0$ and $\tau > 0$).

An automatic procedure, the function `find.hyper()` based on the numerical optimization function `optim` in R (R Development Core Team 2007) was used by Rigby and Stasinopoulos (2006) to minimize the $GAIC(\sharp) = -2\hat{\ell} + \sharp df$, (where $\hat{\ell}$ is the maximized log likelihood function, $\sharp$ is the penalty and df is the total effective degrees of freedom used in the model), over the five hyperparameters df_μ, df_σ, df_ν, df_τ and ξ in the BCT model. The chosen hyperparameters for five different values of the penalty $\sharp$ in GAIC($\sharp$) were shown in Table 3 of Rigby and Stasinopoulos (2006). [Note that in general the $GAIC(\sharp)$ can potentially have multiple local minima (especially for low values of $\sharp$) and so the automatic procedure should be run with different starting values to ensure a global minimum has been found.]

For example the following R code can be used to find the chosen hyperparameters corresponding to penalty $\sharp = 2$. The penalty $\sharp$ is specified by the `find.hyper()` argument `penalty`. Note also the `c.spar` argument in the cubic spline function `cs()` which is necessary in this case to make sure that the degrees of freedom for smoothing is able to take small values (see the comments on the help file for `cs()`). The five hyperparameters df_μ, df_σ, df_ν, df_τ and ξ are represented by `p[1]` to `p[5]` in the code below, while the argument `par` specifies initial values for the five parameters and `lower` specifies their lower bounds. See details of the function `optim()` in R for the other arguments.

```
R> library("gamlss")
R> data("db")
R> mod1 <- quote(gamlss(head~cs(nage, df = p[1]), sigma.fo = ~cs(nage, p[2]),
+    nu.fo = ~cs(nage, p[3], c.spar=c(-1.5,2.5)),
+    tau.fo = ~cs(nage, p[4], c.spar=c(-1.5,2.5)),
+    data = db, family = BCT,
+    control = gamlss.control(trace=FALSE)))
R> op <- find.hyper(model=mod1, other=quote(nage<-age^p[5]),
+    par = c(10,2,2,2,0.25),
+    lower = c(0.1,0.1,0.1,0.1,0.001),
+    steps = c(0.1,0.1,0.1,0.1,0.2),
+    factr = 2e9, parscale=c(1,1,1,1,0.035), penalty=2 )
```

The procedure takes a long time (approximately one hour!). The final chosen values of the five hyperparameters and the final value of GAIC ($\sharp$) are obtained by the components `op$par` and `op$value` respectively.

```
par 10 2 2 2 0.25 crit= 26792.24 with pen= 2
par 10.1 2 2 2 0.25 crit= 26792.06 with pen= 2
par 9.9 2 2 2 0.25 crit= 26792.43 with pen= 2
par 10 2.1 2 2 0.25 crit= 26792.01 with pen= 2
...
...
par 18.43638 2.679676 0.9969013 6.73205 0.08739384 crit= 26780.56 with pen= 2
par 18.43638 2.679676 0.9969013 6.83205 0.09439384 crit= 26780.57 with pen= 2
par 18.43638 2.679676 0.9969013 6.83205 0.08039384 crit= 26780.56 with pen= 2
R> op
$par
[1] 18.43637964  2.67967596  0.99690134  6.83204939  0.08739384

$value
[1] 26780.56

$counts
function gradient
      13       13

$convergence
[1] 0
$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Note that the degrees of freedom reported in each of the first four components of `op$par` in the output do not include the constant and the linear term, so 2 degrees of freedom have to be added to each value above to give the total degrees of freedom for each distribution parameter. For example the total degrees of freedom for μ is $2 + 18.44 = 20.44$. If the automatic procedure results in a value of 0.1 for the extra degrees of freedom for a particular parameter, that is, the lower boundary of the search, then a further search has to be done to check if the model can be simplified further to either just a linear term or just a constant term for that parameter.

A penalty value of 3 was chosen by Rigby and Stasinopoulos (2006) resulting in selected hyperparameters $(df_\mu, df_\sigma, df_\nu, df_\tau, \xi) = (12.3, 5.7, 2, 2, 0.33)$ minimizing GAIC(3).

Here for comparison we select the hyperparameters using a validation data set. The data is split into 60% training and 40% validation data. For each specific set of hyperparameters, model (10) is fitted to the training data and the resulting validation global deviance $VGD = -2\hat{\ell}_v$ where $\hat{\ell}_v$ is the log-likelihood of the validation data given the fitted training data model. The VGD is then minimized over the hyperparameters using the numeric optimization function `optim()`. The code we use is as follows:

```
R> library("gamlss")
R> data("db")
R> set.seed(101)
R> rand <- sample(2, length(db$head),replace=T, prob=c(0.6, 0.4))
R> table(rand)/length(db$head)
```

```
rand
        1         2
0.5940341 0.4059659

R> dbp <- db
R> dbp$agepower <- db$age^0.33
R> mBCT<- gamlss(head~cs(agepower,df=10.3), sigma.fo=~cs(agepower,df=3.7),
+    nu.fo=~agepower, tau.fo=~agepower, family=BCT, data=dbp)

GAMLSS-RS iteration 1: Global Deviance = 26916.28
...
GAMLSS-RS iteration 10: Global Deviance = 26745.67

R> mu.s <- fitted(mBCT, "mu")[rand == 1]
R> sigma.s <- fitted(mBCT, "sigma")[rand == 1]
R> nu.s <- fitted(mBCT, "nu")[rand == 1]
R> tau.s <- fitted(mBCT, "tau")[rand ==1 ]
R> fnBCT <- function(p) {
+    db$agepower <- db$age^p[1]
+    vgd <- VGD(head~cs(agepower,df=p[2],c.spar=list(-1.5, 2.5)),
+        sigma.fo=~cs(agepower,df=p[3],c.spar=list(-1.5, 2.5)),
+        nu.fo=~agepower,  tau.fo=~agepower,
+        family=BCT, data=db, rand=rand, mu.start=mu.s,
+        sigma.start=sigma.s, nu.start=nu.s, tau.start=tau.s)
+    cat("p=", p, " and vgd=", vgd, "\n")
+    vgd }
R> op <- optim(par=c(.33, 12, 5.7), fnBCT, method="L-BFGS-B",
+    lower=c(.01,1,1), upper=c(.5, 20, 20))
```

Note that we have not attempted to model the parameters ν and τ using smoothing cubic splines, but we just fit linear terms as in the chosen model of Rigby and Stasinopoulos (2006). Hence the resulting hyperparameters and degrees of freedoms were $(df_\mu, df_\sigma, df_\nu, df_\tau, \xi) = (15.77, 8.05, 2, 2, 0.28)$. Below we fit this model chosen by minimizing VGD to the *full* data set (i.e., training and validation data combined).

```
R> db$agepower <- db$age^0.28
R> mBCT <- gamlss(head~cs(agepower,df=13.77), sigma.fo=~cs(agepower,df=6.05),
+    nu.fo=~agepower, tau.fo=~agepower, family=BCT, data=db)

GAMLSS-RS iteration 1: Global Deviance = 26902.78
...
GAMLSS-RS iteration 9: Global Deviance = 26732.04
```

Figure 13, obtained using `fitted.plot(mBCT, x=db$age)`, shows the fitted models for μ, σ, ν, and τ for the model chosen by minimizing VGD.

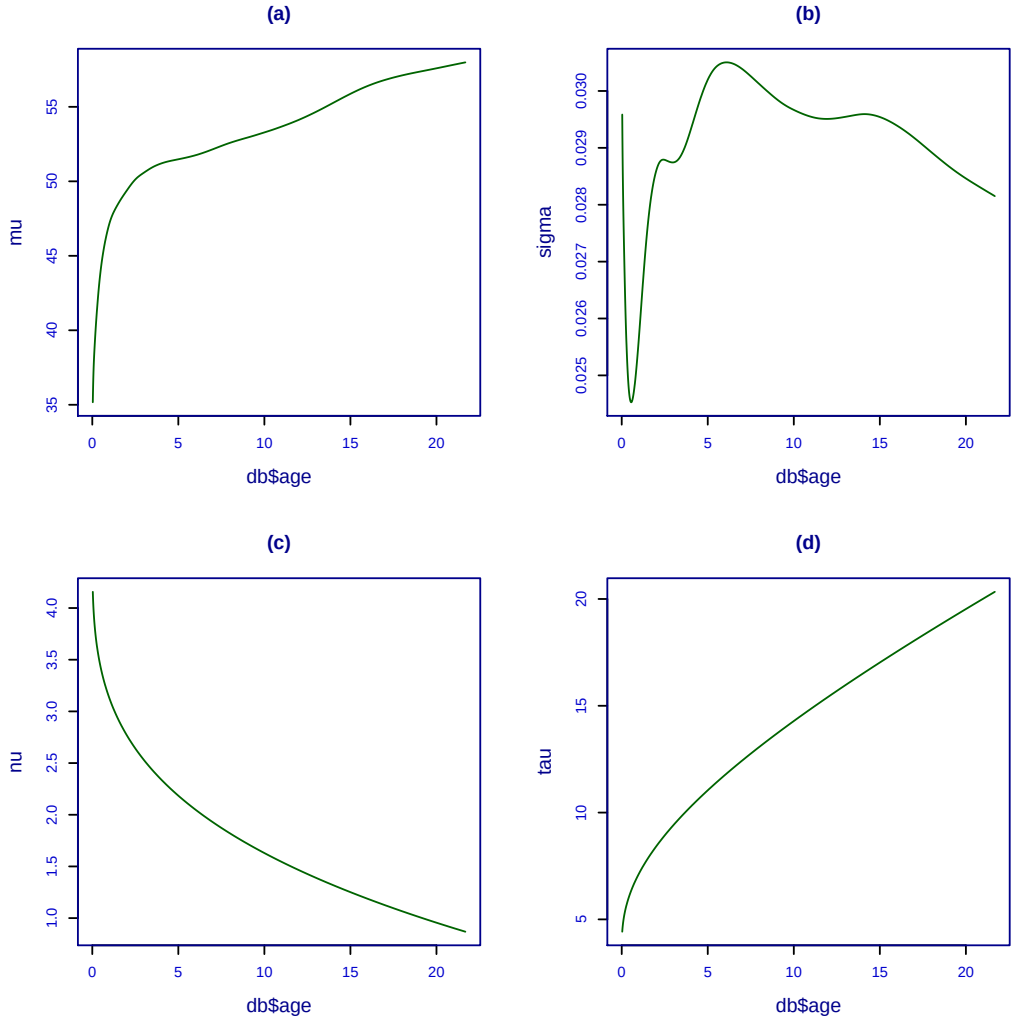

Figure 13: The fitted parameters against age for the "best" BCT model with hyperparameters chosen to minimize the validation global deviance (VGD) (a) μ (b) σ (c) ν (d) τ.

Figure 14 displays the (normalized quantile) residuals, from model $BCT(15.77, 8.05, 2, 2, 0.28)$. Panels (a) and (b) plot the residuals against the fitted values of μ and against age respectively, while panels (c) and (d) provide a kernel density estimate and normal QQ plot for them respectively. The residuals appear random but the QQ plot shows seven extreme outliers (0.1% of the data) in the upper tail of the distribution of y. Nevertheless the Box-Cox t distribution model provides a reasonable fit to the data. Below gives the commands for obtaining Figure 14.

```
R> newpar <- par(mfrow = c(2, 2), mar = par("mar") + c(0, 1, 0,0),
+     col.axis = "blue4", col = "blue4", col.main = "blue4",
+     col.lab = "blue4", pch = "+", cex = 0.45, cex.lab = 1.2,
+     cex.axis = 1, cex.main = 1.2)
R> plot(mBCT, xvar = db$age, par = newpar)
```

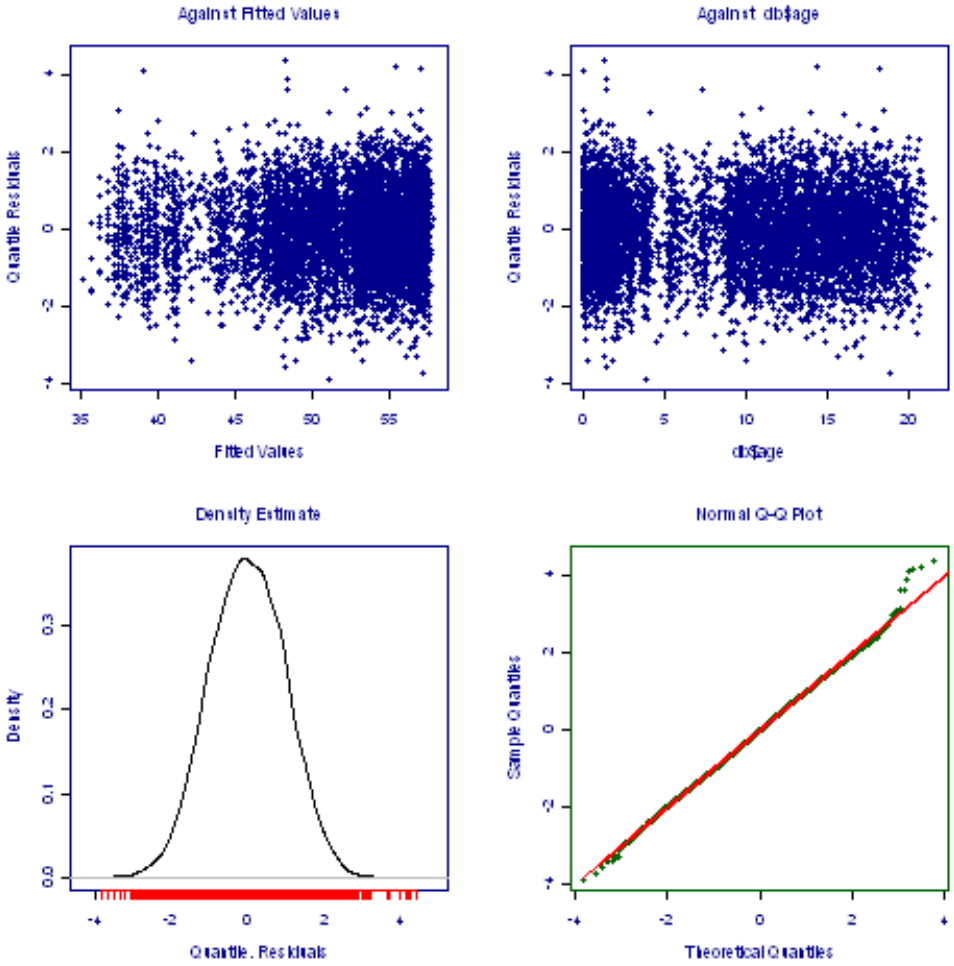

Figure 14: The residuals from model $BCT(15.77, 8.05, 2, 2, 0.28)$. (a) against fitted values of μ (b) against age (c) kernel density estimate (d) normal QQ plot.

```
*******************************************************************
            Summary of the Quantile Residuals
                        mean   =  -0.0003786376
                    variance   =  1.000140
              coef. of skewness  =  0.008874363
              coef. of kurtosis  =  3.056934
Filliben correlation coefficient  =  0.999539
*******************************************************************

R> par(newpar)
```

Figure 15 displays detailed diagnostic plots for the residuals using a worm plot developed by van Buuren and Fredriks (2001). In this plot the range of age is split into 20 contiguous non-overlapping intervals with equal numbers of cases. The 20 age ranges are displayed in horizontal steps in the chart above the worm plot in Figure 15. A detrended normal QQ plot

of the residuals in each interval is then displayed. Ten outliers are omitted from the worm plot as their deviations lie outside the deviation range used in the plots. The worm plot allows detection of inadequacies in the model fit within specific ranges of age. From Figure 15, the de-trended QQ plots show adequate fits to the data within most of the 20 age ranges, with only occasional minor inadequacies. van Buuren and Fredriks (2001) proposed fitting cubic models to each of the de-trended QQ plots, with the resulting constant, linear, quadratic and cubic coefficients, $\hat{b}_0, \hat{b}_1, \hat{b}_2$ and $\hat{b}_3$ respectively, indicating differences between the empirical and model residual mean, variance, skewness and kurtosis respectively, within the age range in the QQ plot. They summarize their interpretations in their Table II. For model diagnosis, they categorize absolute values of $\hat{b}_0, \hat{b}_1, \hat{b}_2$ and $\hat{b}_3$ in excess of threshold values, 0.10, 0.10, 0.05 and 0.03 respectively, as misfits.

The commands below produce the worm plot in Figure 15 together with the number of points missing from each of the 20 detrended Q-Q plots (from the bottom left to top right of Figure 15). The 20 age ranges used are given by `$classes`, while `$coef` gives the coefficients $\hat{b}_0$, $\hat{b}_1$, $\hat{b}_2$, and $\hat{b}_3$, of van Buuren and Fredriks (2001).

```
R> a<- wp(mBCT, xvar = db$age, n.inter = 20, ylim.worm = 0.6, cex = 0.3,
+    pch = 20)

number of missing points from plot= 3  out of  374
...
number of missing points from plot= 1  out of  351

R> a
$classes
        [,1]   [,2]
 [1,]  0.025  0.185
 ...
[20,] 19.075 21.685

$coef
               [,1]         [,2]           [,3]           [,4]
 [1,] -0.057187844  0.017469571  3.411925e-02 -0.0040514944
 ...
[20,] -0.024159686  0.027327238 -5.999749e-03 -0.0177428163
```

van Buuren and Fredriks (2001) reported results for the male head circumference data using the LMS model with a 're-scale transformation' of age which stretches periods of rapid growth and compresses periods of lower growth in y to provide a uniform growth rate on the transformed age scale (see Cole, Freeman, and Preece 1998, for details). Following this complex transformation of age, they chose 9 degrees of freedom for μ, 5 for σ and a constant value $\nu = 1$. However, they reported a total of 16 violations in the resulting worm plot coefficients from their chosen fitted model (i.e., values of $\hat{b}$'s in excess of their threshold values), indicating that the model does not adequately fit the data within many specific age ranges. In contrast, there are no violations in the worm coefficients from the fitted model $BCT(15.77, 8.05, 2, 2, 0.28)$, indicating an adequate fit to the data within age ranges.

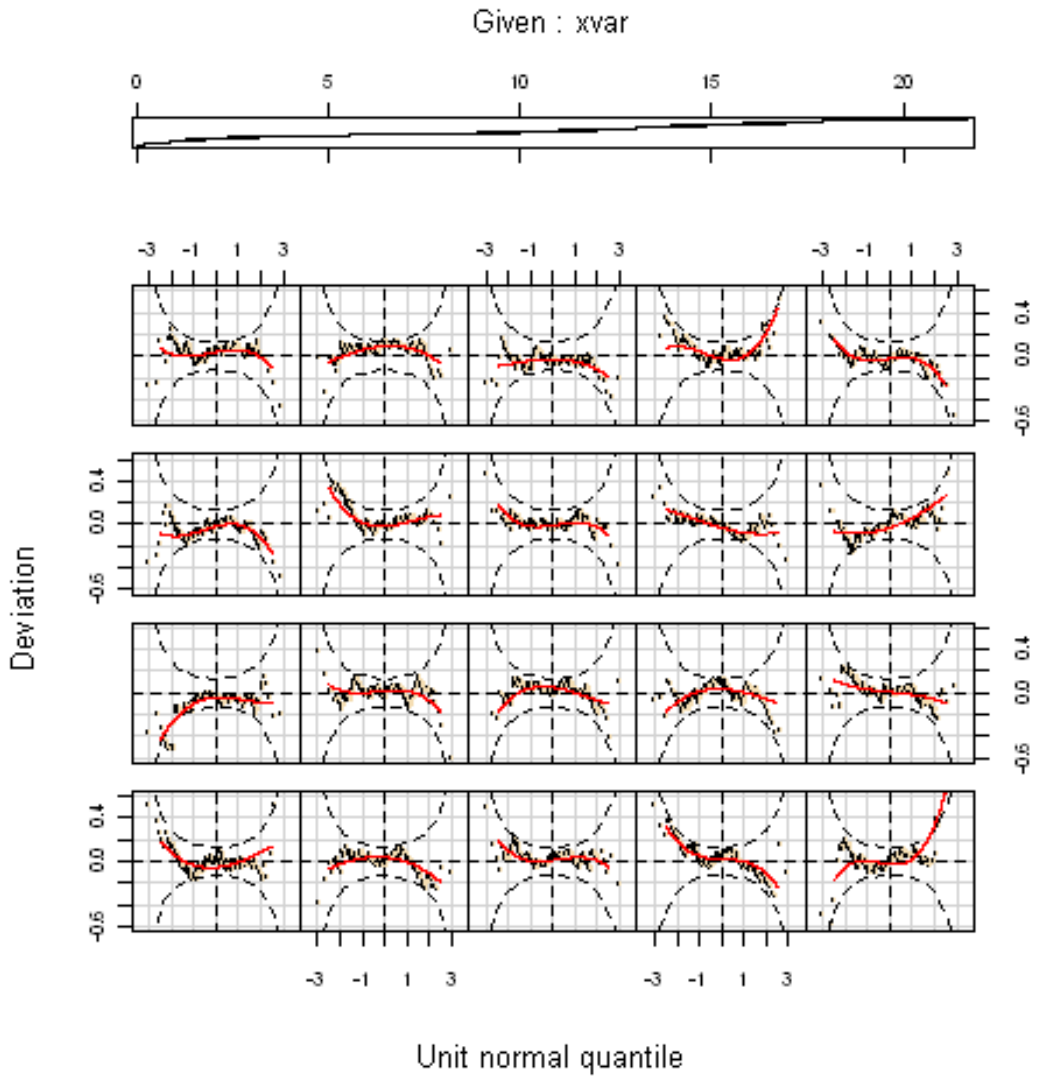

Figure 15: Worm plot of the residuals from model $BCT(15.77, 8.05, 2, 2, 0.28)$. The 20 detrended QQ plots read in rows, from bottom left to top right plot, correspond to 20 age ranges (displayed in steps above the worm plot from 0 to 22 years).

The fit within age groups can be further investigated by calculating Q statistics for testing normality of the residuals within age groups (Royston and Wright 2000). The application of Q statistics to centile data was also discussed by Rigby and Stasinopoulos (2004).

Let G be the number of age groups and let $\{r_{gi}, i = 1, 2, .., n_i\}$ be the residuals in age group g, for $g = 1, 2, .., G$. Statistics $\mathsf{Z}_{g1}, \mathsf{Z}_{g2}, \mathsf{Z}_{g3}, \mathsf{Z}_{g4}$ are calculated from the residuals in group g to test whether the residuals in group g have population mean 0, variance 1, skewness 0 and kurtosis 3. See Royston and Wright (2000) for their definition and Rigby and Stasinopoulos (2004) for an application to centile data.

The Q statistics of Royston and Wright (2000) are then calculated by $Q_j = \sum_{g=1}^{G} \mathsf{Z}_{gj}^2$ for $j = 1, 2, 3, 4$. Royston and Wright (2000) discuss approximate distributions for the Q statistics

under the null hypothesis that the true residuals are normally distributed. The resulting significance levels should be regarded as providing a guide to model inadequacy, rather than exact formal test results.

'Significant' Q_1, Q_2, Q_3 or Q_4 statistics indicate possible inadequacies in the models for parameters μ, σ, ν and τ respectively, which may be overcome by increasing the degrees of freedom in the model for the particular parameter. Q-statistics can be obtained using the `Q.stats()` function. The results below suggest possible inadequacy in the chosen model (particularly for ν and τ).

```
R> Q.stats(mBCT, xvar = db$age, n.inter = 20)
```

```
                        Z1         Z2         Z3         Z4    AgostinoK2    N
0.02500 to 0.18499 -0.4484003  0.261715  1.7067101  0.8872568  3.7000843  374
0.18499 to 0.46499  0.1840299 -0.484436 -1.1752911  0.1132373  1.3941318  334
0.46499 to 0.87499  0.3685602 -0.235589  0.4005546 -1.1005652  1.3716879  351
0.87499 to 1.255    0.4392752 -1.498977  0.1873113 -1.5442777  2.4198793  358
1.255 to 1.755      0.2217602  1.918012  2.3170652  3.0642958 14.7587003  352
1.755 to 2.355     -1.4535468  1.042338 -1.5839443  0.7935995  3.1386799  344
2.355 to 3.155      0.1568195 -0.293398 -0.4419538 -0.6724331  0.6474895  355
3.155 to 5.535      0.3904703 -0.171900 -1.4829916  1.1527648  3.5281311  348
5.535 to 8.905      0.1774860  0.179760 -1.0866496  0.6702548  1.6300489  352
8.905 to 9.995      0.1466325 -0.806457 -0.0210452 -0.2453329  0.0606311  353
9.995 to 11.015    -0.7650576  0.319668 -1.0899948 -0.9988486  2.1857873  353
11.015 to 11.975    0.3233806 -0.395298  1.5364611 -0.6661094  2.8044145  350
11.975 to 12.995   -0.0142166 -0.277707  0.4057170 -1.2407151  1.7039803  355
12.995 to 13.925   -0.3719043 -1.342941  0.5006712  0.2489784  0.3126619  353
13.925 to 14.845   -0.1652504  1.824631  1.0214185  0.7920882  1.6706996  350
14.845 to 15.835    0.5482230  0.159326 -0.3981177 -0.8198388  0.8306335  353
15.835 to 16.895    1.2623821  0.181071 -1.0792911 -0.0886510  1.1727282  349
16.895 to 17.915   -0.9114567 -0.256590 -0.8270233 -0.1165021  0.6975404  352
17.915 to 19.075    0.3389135  0.395489  2.0835231  2.2136206  9.2411850  353
19.075 to 21.685   -0.5646241 -0.629562 -0.2334364 -1.6327868  2.7204855  351
TOTAL Q stats       6.9700513 14.314279 27.3534790 28.6361020 55.9895811 7040
df for Q stats      4.2276359 15.474309 18.0000000 18.0000000 36.0000000    0
p-val for Q stats   0.1552353  0.537060  0.0726000  0.0530118  0.0179523    0
```

The Z_{gj} statistic when squared provides the contribution from age group g to the statistic Q_j, and hence helps identify which age groups are causing the Q_j statistic to be significant and therefore in which age groups the model is inadequate.

Provided the number of groups G is sufficiently large relative to the degrees of freedom in the model for the parameter, then the Z_{gj} values should have approximately standard normal distributions under the null hypothesis that the true residuals are standard normally distributed. We suggest as a rough guide values of $|\mathsf{Z}_{gj}|$ greater than 2 be considered as indicative of significant inadequacies in the model. See Rigby and Stasinopoulos (2004) for an application of Z statistics to centile data. The residuals in the age range (1.255, 1.755) years are highly positively skewed and kurtotic indicating possible inadequacies of the ν and

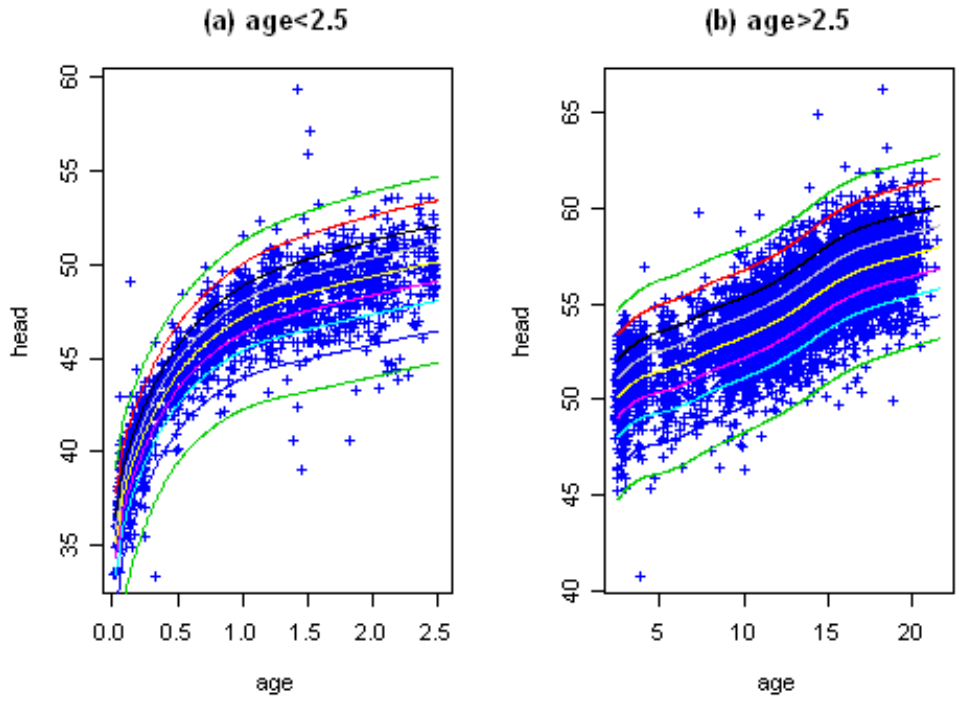

Figure 16: Observed head circumference with nine fitted model centile curves (0.4, 2, 10, 25, 50, 75, 90, 98, 99.6), from model $BCT(15.77, 8.05, 2, 2, 0.28)$, against age: (a) 0–2.5 years (b) 2.5–22 years

τ models in this range or perhaps the presence of outliers in head circumference in this age range.

The following command centiles.split obtains the centiles curves given in Figure 16 for head circumference against age, split at age= 2.5 years (defined by `c(2.5)` in the command). The output below compares the sample proportion below each centile curve for each of the two age ranges, i.e., below age 2.5 years and above age 2.5 years. They agree reasonably well with the model proportions given by the first column.

```
R> centiles.split(mBCT, xvar = db$age, c(2.5), ylab = "HEAD", xlab = "AGE",
+  bg= "transparent")
```

```
     0.03 to 2.5 2.5 to 21.68
0.4    0.4599816    0.4110152
2      1.8859246    1.7673654
10    10.0735971    9.8438142
25    26.2649494   25.2979860
50    50.2299908   50.0205508
75    73.7810488   74.4143033
90    90.0643974   90.0739827
```

```
98     98.2980681   98.2120838
99.6   99.4480221   99.7739416
```

Figure 16 provides nine fitted model centile curves, defined by (11) below, for head circumference for model $BCT(15.77, 8.05, 2, 2, 0.28)$, with centiles $100\alpha = 0.4$, 2, 10, 25, 50, 75, 90, 98, 99.6. For each α, the centile curve, y_α against x, is obtained by finding the fitted values $(\hat{\mu}, \hat{\sigma}, \hat{\nu}, \hat{\tau})$ for each x (over a range of values of x) and substituting the values into

$$y_\alpha = \begin{cases} \mu[1 + \sigma\nu t_{\tau,\alpha}]^{1/\nu} & \text{if } \nu \neq 0 \\ \mu \exp[\sigma t_{\tau,\alpha}] & \text{if } \nu = 0, \end{cases} \tag{11}$$

where $t_{\tau,\alpha}$ is the 100α centile of t_τ, a standard t distribution with degrees of freedom parameter τ. Strictly the exact formula for y_α is given in Rigby and Stasinopoulos (2006, Appendix A). The resulting centiles are plotted separately for age range 0 to 2.5 years and for age range 2.5 to 22 years in Figure 16, for clarity of presentation.

Finally, in order to investigate the effect of the extreme outliers, the 14 most extreme observations (7 from the upper and 7 from the lower tail, were removed and the model was refitted. The fit to the data were *substantially* improved, leading to improved Q statistics and improved centile estimates. Hence the 14 outliers were causing a distortion of the fitted model and of the (detrended) QQ plot of the residuals resulting in a distortion of the centile estimates. The centile percentages can be adjusted for the 0.1% of cases removed from each tail.

4. Conclusions

GAMLSS is a general framework for univariate regression type statistical problems. It allows flexibility in specifying the distribution of the response variable including highly skew and/or kurtotic distributions and also allows all the distribution parameters to be modelled flexibly as functions of explanatory variables. New distributions can be added easily. The use of (modified) backfitting in the fitting algorithm makes the addition of new additive smoothing terms easy. The fitting algorithm is fast enough to allow the rapid exploration of very large and complex data sets. For medium to large size data, GAMLSS allows flexibility in statistical modelling far beyond other currently available methods. While flexibility allows more realistic assumptions about the data in hand, it has the drawback that model selection becomes more difficult for the simple reason that there are more models to select from. More work needs to be done here.

Acknowledgments

The authors would like to thank Calliope Akantziliotou for her contribution in the earlier R development of GAMLSS. We would also like to thank all the R contributors whose packages we used to develop the different **gamlss** packages. More specifically Trevor Hastie for **gam**, (Hastie 2006), Bill Venables and Brian Ripley for **MASS**, (Venables and Ripley 2002), Jim Lindsey for **rmutil**, (Lindsey 2007) and Gordon Smyth for the function `dglm()` in the package **dglm**, (Dunn and Smyth 2006). We would also like to acknowledge the work of Tim Cole and Peter Green from which the CG algorithm in the GAMLSS framework and in the `gamlss()` function was developed.

References

Akantziliotou C, Rigby RA, Stasinopoulos DM (2002). "The R Implementation of Generalized Additive Models for Location, Scale and Shape." In M Stasinopoulos, G Touloumi (eds.), "Statistical Modelling in Society: Proceedings of the 17th International Workshop on Statistical Modelling," pp. 75–83. Chania, Greece.

Ambler G (1999). `fracpoly()`: *Fractional Polynomial Model*. S-PLUS, URL http://lib.stat.cmu.edu/S/fracpoly/.

Chambers JM, Hastie TJ (1992). *Statistical Models in S*. Chapman & Hall, London.

Cleveland WS, Grosse E, Shyu M (1993). "Local Regression Models." In J Chambers, T Hastie (eds.), "Statistical Modelling in S," pp. 309–376. Chapman and Hall: New York.

Cole TJ, Freeman JV, Preece MA (1998). "British 1990 Growth Reference Centiles for Weight, Height, Body Mass Index and Head Circumference Fitted by Maximum Penalized Likelihood." *Statistics in Medicine*, **17**, 407–429.

Cole TJ, Green PJ (1992). "Smoothing Reference Centile Curves: The LMS Method and Penalized Likelihood." *Statistics in Medicine*, **11**, 1305–1319.

Crisp A, Burridge J (1994). "A Note on Nonregular Likelihood Functions in Heteroscedastic Regression Models." *Biometrika*, **81**, 585–587.

de Boor C (1978). *A Practical Guide to Splines*. Springer-Verlag, New York.

de Jong P, Heller GZ (2007). *Generalized Linear Models for Insurance Data*. Cambridge University Press.

Dunn PK, Smyth GK (1996). "Randomised Quantile Residuals." *Journal of Computational and Graphical Statistics.*, **5**, 236–244.

Dunn PK, Smyth GK (2006). **dglm**: *Double Generalized Linear Models*. R package version 1.3, URL http://CRAN.R-project.org/.

Eilers PHC, Marx BD (1996). "Flexible Smoothing with B-splines and Penalties." *Statistical Science*, **11**, 89–121.

Fredriks AM, van Buuren S, Burgmeijer R, Meulmeester J, Beuker R, Brugman E, Roede M, Verloove-Vanhorick S, Wit JM (2000a). "Continuing Positive Secular Change in The Netherlands, 1955-1997." *Pediatric Research*, **47**, 316–323.

Fredriks AM, van Buuren S, Wit J, Verloove-Vanhorick SP (2000b). "Body Index Measurements in 1996-7 Compared with 1980." *Archives of Childhood Diseases*, **82**, 107–112.

Green PJ, Silverman BW (1994). *Nonparametric Regression and Generalized Linear Models*. Chapman and Hall, London.

Gu C (2007). **gss**: *General Smoothing Splines*. R package version 1.0-0, URL http://CRAN.R-project.org/.

Hastie T (2006). **gam***: Generalized Additive Models*. R package version 0.98, URL http://CRAN.R-project.org/.

Hastie TJ, Tibshirani RJ (1990). *Generalized Additive Models*. Chapman and Hall, London.

Hastie TJ, Tibshirani RJ (1993). "Varying Coefficient Models." *Journal of the Royal Statistical Society B*, **55**, 757–796.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions, Volume I*. Wiley, New York, 2nd edition.

Johnson NL, Kotz S, Balakrishnan N (1995). *Continuous Univariate Distributions, Volume II*. Wiley, New York, 2nd edition.

Johnson NL, Kotz S, Kemp AW (2005). *Univariate Discrete Distributions*. Wiley, New York, third edition.

Lindsey J (2007). **rmutil***: Utilities for Nonlinear Regression and Repeated Measurements Models*. R package version 1.0, URL http://popgen.unimaas.nl/~jlindsey/rcode.html.

Marx B (2003). `ps()`*: P-Spline Code for GAMs and Univariate GLM Smoothing*. S-PLUS code, URL http://www.stat.lsu.edu/faculty/marx/ps.txt.

Nelder JA, Wedderburn RWM (1972). "Generalized Linear Models." *Journal of the Royal Statistical Society A*, **135**, 370–384.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Reinsch C (1967). "Smoothing by Spline Functions." *Numerische Mathematik*, **10**, 177–183.

Rigby RA, Stasinopoulos DM (1996a). "A Semi-parametric Additive Model for Variance Heterogeneity." *Statistal Computing*, **6**, 57–65.

Rigby RA, Stasinopoulos DM (1996b). "Mean and Dispersion Additive Models." In W Härdle, MG Schimek (eds.), "Statistical Theory and Computational Aspects of Smoothing," pp. 215–230. Physica, Heidelberg.

Rigby RA, Stasinopoulos DM (2001). "The GAMLSS project: a Flexible Approach to Statistical Modelling." In B Klein, L Korsholm (eds.), "New Trends in Statistical Modelling: Proceedings of the 16th International Workshop on Statistical Modelling," pp. 249–256. Odense, Denmark.

Rigby RA, Stasinopoulos DM (2004). "Smooth Centile Curves for Skew and Kurtotic data Modelled Using the Box-Cox Power Exponential Distribution." *Statistics in Medicine*, **23**, 3053–3076.

Rigby RA, Stasinopoulos DM (2005). "Generalized Additive Models for Location, Scale and Shape." *Applied Statistics*, **54**, 507–554.

Rigby RA, Stasinopoulos DM (2006). "Using the Box-Cox t Distribution in GAMLSS to Model Skewness and Kurtosis." *Statistical Modelling*, **6**, 209–229.

Royston P, Altman DG (1994). "Regression Using Fractional Polynomials of Continuous Covariates: Parsimonious Parametric Modelling." *Applied Statistics*, **43**, 429–467.

Royston P, Wright EM (2000). "Goodness-of-Fit Statistics for Age-specific Reference Intervals." *Statistics in Medicine*, **19**, 2943–2962.

Stasinopoulos DM, Rigby RA, Akantziliotou C (2006). "Instructions on How to Use the **gamlss** Package in R." *Technical Report 01/06*, STORM Research Centre, London Metropolitan University, London. URL http://www.gamlss.com/.

van Buuren S, Fredriks M (2001). "Worm Plot: A Simple Diagnostic Device for Modelling Growth Reference Curves." *Statistics in Medicine*, **20**, 1259–1277.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer-Verlag, 4th edition. ISBN 0-387-98825-4, URL http://www.stats.ox.ac.uk/pub/MASS4/.

Wade AM, Ades AE (1994). "Age-related Reference Ranges : Significance Tests for Models and Confidence Intervals for Centiles." *Statistics in Medicine*, **13**, 2359–2367.

WHO Multicentre Growth Reference Study Group (2006). *WHO Child Growth Standards: Methods and Development*. World Health Organization, Geneva, Switzerland.

Wilkinson GN, Rogers CE (1973). "Symbolic Description of Factorial Models for Analysis of Variance." *Applied Statistics*, **22**, 392–399.

Williams CB (1944). "Some Applications of the Logarithmic Series and the Index of Diversity to Ecological Problems." *Journal of Ecology*, **32**, 1–44.

Wood SN (2001). "**mgcv**: GAMs and Generalised Ridge Regression for R." *R News*, **1**(2), 20–25. URL http://CRAN.R-project.org/doc/Rnews/.

Yee TW (2007). **VGAM***: Vector Generalized Linear and Additive Models*. R package version 0.7-5, URL http://www.stat.auckland.ac.nz/~yee/VGAM/.

Affiliation:

D. Mikis Stasinopoulos
STORM
London Metropolitan University
London, United Kingdom
E-mail: d.stasinopoulos@londonmet.ac.uk
URL: http://www.gamlss.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 23, Issue 7 *Submitted:* 2007-07-19
December 2007 *Accepted:* 2007-11-28