GAMLSS in action

Annotated examples

of working with

R and GAMLSS

Philip H. Quanjer[1]
Sanja Stanojevic[2,3]
Janet Stocks[2]
Tim J. Cole[4]

[1] Department of Pulmonary Diseases and Department of Paediatrics, Erasmus Medical Centre, Erasmus University, Rotterdam, the Netherlands
[2] Portex Respiratory Unit, UCL Institute of Child Health, London, UK,
[3] Child Health Evaluative Sciences, The Hospital for Sick Children, Toronto, Canada
[4] MRC Centre of Epidemiology for Child Health, UCL Institute of Child Health, London, UK

January 16, 2012

TABLE OF CONTENTS

Introduction

Aim

This set of worked examples is designed to help people work with `R` to develop spirometric prediction equations using `GAMLSS`. The examples can be used for other measurements with similar distributional properties. Please note that our examples represent one way to fit gamlss models; dependent on the outcome and the distributions different models may be appropriate. Even for spirometry, different sample sizes and narrow age ranges may reveal that simpler models, that do not require splines, are satisfactory.

About R

The `R` environment is an integrated suite of software for data facilities and graphical display. "`R` is a free software environment t for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS."

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. http://www.r-project.org/

For a comprehensive introduction to the use of `R` see:

W.N. Venables, B.D. Ripley. Modern Applied Statistics with S. Fourth Edition. Springer. ISBN 0-387-95457-0, 2002. http://www.stats.ox.ac.uk/pub/MASS4/

For the `R` scripting language see:

W.N. Venables, D.M. Smith and the R Development Core Team. An introduction to `R`. http://cran.r-project.org/doc/manuals/R-intro.pdf

About GAMLSS

"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." http://gamlss.org/

Rigby RA, Stasinopoulos DM. Generalized additive models for location, scale and shape (with Discussion). *Applied Statistics* 2005; **54**: 507–544.

About the dataset

The worked examples use a dataset consisting of

- FEV_1, a spirometric measure defined as the volume of air that can be expelled in one second from the lung during a forced expiration after a full inspiration,
- Age (years)
- Height (cm)
- Sex (1=male, 2=female)
- Centre: data were collated from 4 centres.

FEV_1 varies with height and age; an unusually low value adjusted for height and age may indicate lung disease. The worked examples show how to use `GAMLSS` to model the relationship between the dependent (FEV_1) and explanatory (height, age, *etc.*) variables.

Notation

Scripts appear in red, the output from `R` in blue, `objects`, `functions` and `commands` in monospaced typewriter font. `R` uses the > command prompt, but this has not been reproduced in this manuscript. Comments preceded by # are disregarded by `R`.

Installation

Installation involves two steps:

- R software
- Required packages for different jobs.

Installing R

Visit http://cran.r-project.org/, download the version of R that fits your operating system (Linux, MacOS X or Windows) and install it.

Installing packages

For the present analyses we will need the package **gamlss**.
Open R, click Packages > Install Packages > Choose location closest you > Click ok > Choose - gamlss
Click ok.

Updating R and packages

R and its packages are frequently updated. Therefore check regularly for updates of R at http://cran.r-project.org/. Packages can be easily updated as follows:
Packages > Update packages.
If updates are available and a CRAN mirror has been defined, their names will be displayed in a window, and they can be installed.

Two notes of caution

- It is recommended that you run R as an Administrator.
- R is case sensitive. Therefore, after defining an object ALL, later reference to *e.g.* all or All imply different objects and may generate an error message.

Regression models and GAMLSS

Regression analysis

Simple linear regression models take the form

$$Y_i = \beta_0 + \beta_1 \cdot x_{1i} + ... + \beta_p \cdot x_{pi} + \varepsilon_i$$

It assumes that errors associated with each term are independently and identically distributed, with zero mean and constant variance. The least squares solution, and the assumption that errors are normally distributed, forms the basis for testing the significance of $\beta_{1..n}$. The residual error term is orthogonal with respect to the predicted value.

More recently generalised linear models, generalised additive models, and generalized linear mixed models have become more widely used. In these the normal distribution for Y_i is replaced by an exponential family of distributions (of which the normal is a special case), and a link function relates μ_i, the mean of Y_i, to the linear predictor.

Generalized additive models for location, scale and shape (GAMLSS) extend the above. In GAMLSS the exponential family distribution assumption for the response variable (Y) is replaced by a general distribution family that can model both skewness and kurtosis. Thus, GAMLSS offers general linear predictors for all the distribution parameters μ (location, mean, median, mu), σ (scale, sigma, variability), ν (shape, nu, skewness) and τ (shape, tau, kurtosis), and a choice of error distributions.

Distributions

There are more than 60 different distributions available in the current implementation of GAMLSS (http://gamlss.org/images/stories/papers/Distributions-2010-onlyThetable.pdf) which allow up to four moments of a distribution (μ=`mu`, `sigma`=σ, `nu`=ν, and `tau`=τ) to be modelled. As argued above we are definitely interested in being able to model `mu` and `sigma` (mean and coefficient of variation). The Box-Cox-Cole-Green[1] (`BCCG`) distribution is suitable for all combinations of the distributional parameters within their range (*i.e.* `mu` >0, `sigma` >0, `nu` =(-Inf,+Inf)). We will explore the need to also model skewness (`nu`, using the `BCCG` distribution) and/or kurtosis (`tau`), using the `BCPE` distribution model. We shall also explore the normal distribution (`NO`). The default link functions for these models are as follows:

Distribution	R family	μ	σ	ν	τ
normal	`NO()`	identity	log	-	-
Box-Cox Cole and Green	`BCCG()`	identity	log	identity	-
Box-Cox power exponential	`BCPE()`	identity	log	identity	log

To model spirometry, where age and height act multiplicatively, the link for μ needs to be the log rather than the identity (see Cole *et al.* 2009[2]).

[1] Cole TJ, Green PJ. Smoothing reference centile curves: the LMS method and penalized likelihood. *Stat Med* 1992; 11: 1305-1319.

[2] Cole TJ, Stanojevic S, Stocks J, Coates AL, Hankinson JL, Wade AM. Age- and size related reference ranges: A case study of spirometry through childhood and adulthood. *Statist Med* 2009; 28: 880-898.

Smoothing splines

Complex effects of explanatory variables on the dependent variable can be modelled using splines, which allow the dependent variable to vary smoothly as a function of an explanatory variable. GAMLSS offers the cubic smoothing spline

```
cs(x, df = n, spar = NULL, c.spar = NULL)
```

and the penalized B-spline

```
pb(x, df = NULL, lambda = NULL, control = pb.control(...), ...)
```

The cubic spline is dealt with in the appendix. We will develop models using `pb()`, the standard in GAMLSS. This function automatically finds the optimum `df` for `mu`, `sigma` and `nu`. As shown later, FEV_1 varies in a complex manner with age. Using splines the effect of age can be described by a linear coefficient (fixed effect) and an age spline. An example of the latter is shown in Figure 1, illustrating how the linear age coefficient is 'adjusted' by an age-specific beta spline, allowing a smoothed representation of the non-linear effect of age on FEV_1.

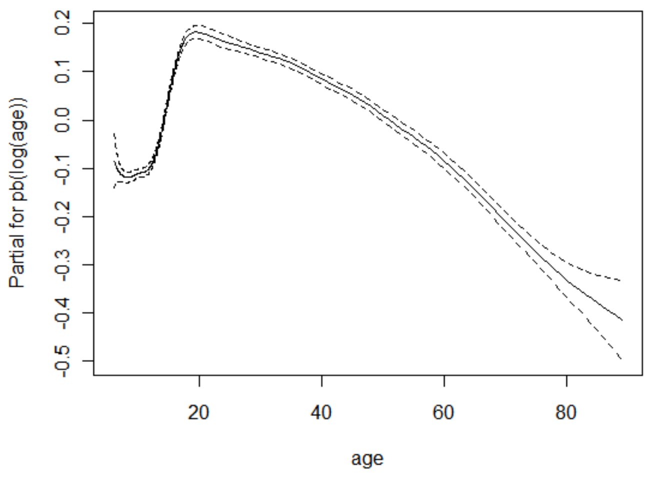

Figure 1

A detailed treatise on GAMLSS can be found in:
Rigby B, Stasinopoulos M. A flexible regression approach using GAMLSS in R.
http://studweb.north.londonmet.ac.uk/~stasinom/papers/book-2010-Athens.pdf

Quantile regression

Quantile regression (QR) is a powerful technique for estimating quantile curves such as the 5th centile. It avoids making distributional assumptions, on the grounds that the distribution cannot be assumed, and as such it is fundamentally different from the GAMLSS approach which relies on an underlying distribution. The absence of a distribution with QR means that it provides centiles not z-scores, and the centiles can be applied only to measurements within the range of the reference group. So, pathological values outside the range cannot be handled at all. This is a serious disadvantage, as it excludes from consideration the very group of patients most in need of clinical support. For this reason we have not considered QR here.

Exploring the dataset

First remove (`rm`) saved workspace from a previous R session. To learn more about a function type ? before its name, *e.g.* `?rm`.

```
rm(list=ls(all=TRUE))
```

Load the required package libraries.

```
library(gamlss)
```

Read the comma separated file with the data, and create the data frame 'ALL'. Note the double forward slash to denote folders. First define the working directory with `setwd()`.

```
setwd("D:\\LungFunction\\GAMLSS-for-statisticians")
ALL <- read.csv("data.csv")
```

Find the number of records and variables in `ALL`.

```
dim(ALL)
[1] 12829     6
```

12,829 records, 6 variables.

Find the variable names.

```
names(ALL)
[1] "ID"     "sex"    "age"    "height" "fev"    "centre"
```

`summary()` summarises the distribution of variables in ALL.

```
summary(ALL)
               ID     sex     age  height     fev  centre
 Min.           1       1   5.862     102   0.561       1
 1st Qu.     3208       1   11.65   149.2    2.14       2
 Median      6415       2      24   161.6    2.89       2
 Mean        6660   1.554   29.61     159   2.976   2.032
 3rd Qu.     9622       2      43   170.8    3.66       2
 Max.       75572       2      94   206.5    6.97       4
```

`table()` tabulates ALL by sex: 1=males, 2=females.

```
table(ALL$sex)
   1    2
5723 7106
```

Use `cut()` to create grouped variables for age and height, with bin widths of 2 years and 5 cm respectively. The function `cutbin()` achieves this and provides mid-bin value labels for each group that are multiples of the bin width.

```
cutbin <- function(x, k) {
      # x = variable, k = bin width
      # mid-bin labels are multiples of k
      k2 <- k / 2
      b <- seq(floor(min((x - k2) / k)) * k, ceiling(max((x - k2) / k)) * k,
k)
      cut(x, breaks=b + k2, labels=b[-1])
}
ALL$ageintrvl <- cutbin(ALL$age, 2)
ALL$heightintrvl <- cutbin(ALL$height, 5)
```

Split the data frame by sex by selecting the rows: `mm` for males, `ff` for females. Note the double = to denote equality. The square bracket notation is [rows, columns], and if either rows or columns is omitted all are selected.

Please note, here we use only males for the example.

```
mm <- ALL[ALL$sex == 1, ]
ff <- ALL[ALL$sex == 2, ]
```

Plot FEV$_1$ as a function of age in males. First define the dimensions of the plot using `windows()` - a new graphics device is created each time.

```
windows(6,5)
plot(mm$age, mm$fev)
```

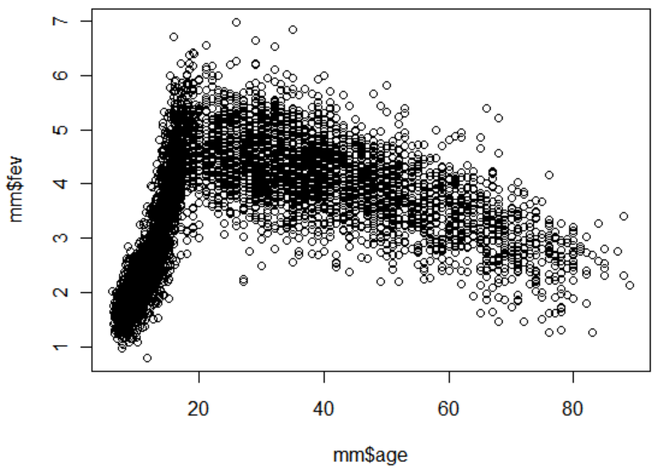

Figure 2

Plot FEV$_1$ as a function of age in males with boxplots, using `ageintrvl` rather than `age`:

```
plot(mm$ageintrvl, mm$fev)
```

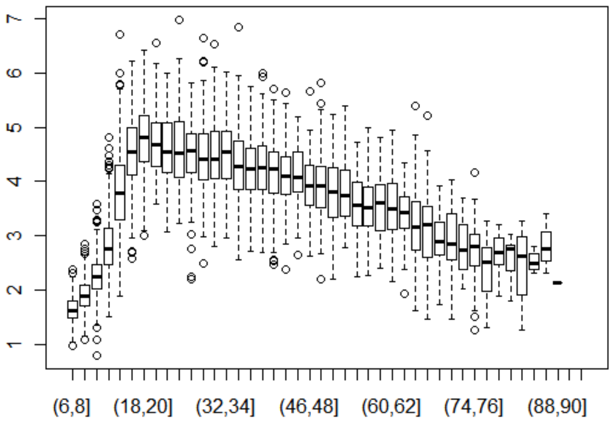

Figure 3

Enhance the plot by labelling the axes, making the y-axis labels horizontal and increasing the font size. `cex` stands for character expansion factor, so `cex` =1.3 expands the characters by 30%; `las` stands for label of axis style, `lab` stands for label (type ?title).

```
plot(mm$ageintrvl, mm$fev, xlab="Age interval (yr)", ylab="FEV1 (L)",
    main="Males", las = 1, cex.main=1.1, cex.lab=1.3, cex.axis = 1.1)
```

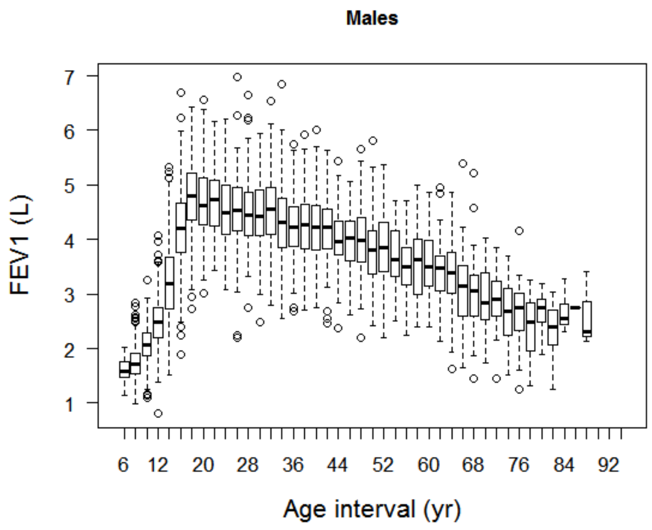

Figure 4

A curvilinear relationship, where the scatter increases with the level of FEV_1.

Look at the relationship of FEV_1 with standing height.

```
summary(mm$heightintrvl)
```

Plot FEV_1 versus height in males with boxplots.

```
plot(mm$heightintrvl, mm$fev, xlab="Age interval (yr)", ylab="FEV1 (L)",
    main="Males", las = 1, lwd=1, cex.main=1.1, cex.lab=1.3, cex.axis = 1.1)
```

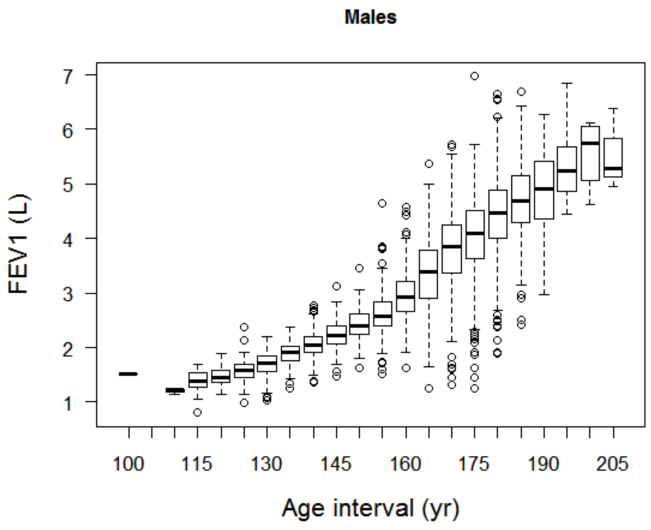

Figure 5

Inspect the age distribution in males.

```
plot(mm$ageintrvl, xlab="Age interval (yr)", ylab="Number of subjects",
    main="Males", cex.lab=1.1, las=1)
```

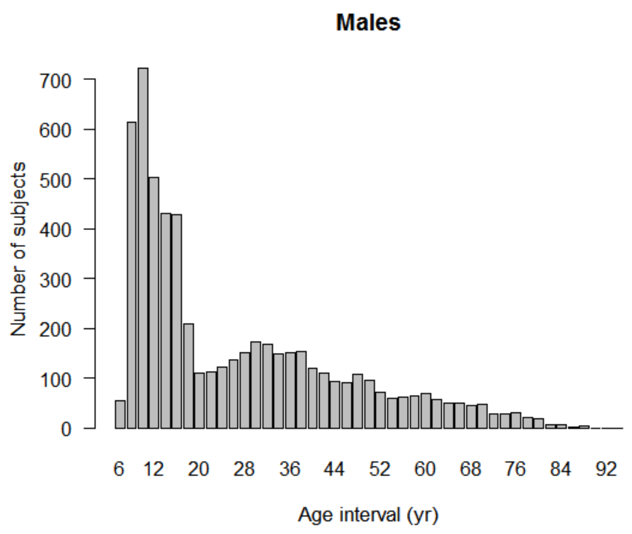

Figure 6

In the Appendix we show how to display the age distributions of both females and males.

Getting to work

Considerations for modelling pulmonary function data

As shown in the previous section, in this cross-sectional population sample there is an increase in FEV_1 through childhood until about age 20, followed by a decline with age in adulthood. The relationship between FEV_1 and age or height is not linear, and the scatter about the average not constant. These patterns need to be accounted for in the model.

Biological considerations

Humans of different body size have a scaling problem that affects all organs. The scaling must be such that the metabolic demands are met; hence all body functions should be compatible with life, and people with different body size should function equivalently. From the study of mammals of widely different sizes we know that volume scales as m^1, where m = body mass[3,4]. We might therefore be tempted to model the FEV_1 as linearly proportional to body mass or volume, *i.e*. length raised to the third power. However, this scaling applies across species but not within species, where regression effects make it inappropriate. In clinical medicine it is therefore not appropriate to predict pulmonary function using body mass or length[3].

It has been shown that within mammal species body mass scales as the square of length: $m \sim l^2$ [1,2]. In fact, this forms the basis for normalizing body mass for body size by the use of the Quetelet index[5,6], nowadays called body mass index: BMI = m/l^2, where m = body mass in kg, and l is standing height in m. So it makes sense to model lung volume (V) as a function of height squared: $V \sim l^2$. Thus one might expect a linear model with log transformed indices of the form: $\log(V) = a + k \cdot \log(\text{height}) + \varepsilon$, and $k \cong 2$.

Growth and ageing

The above allometric relationship holds for adult mammals. However, during childhood body proportions change; in particular the legs become relatively longer, thereby constituting a larger percentage of standing height. This alters the relationship between height and pulmonary function during growth.

In addition Figures 1 and 2 show complex age trends in pulmonary function during childhood, adulthood, and the transition between them. It follows that age as a spline curve should form part of the model.

In the example dataset age ranges from 6-94 years. Childhood and adolescence cover only about 14 years, which makes it hard for the software to detect trends in the period. This can be remedied by making the two periods more equal by transforming age (A), either by log transformation or more generally with the power function age^p where $p < 1$ (logs are equivalent to $p = 0$).

[3] McMahon TA. Size and shape in biology. Elastic criteria impose limits on biological proportions, and consequently on metabolic rates. *Science* 1973; 179, 1201-1204.

[4] West GB, Brown JH, Enquist BJ. A general model for the origin of allometric scaling laws in biology. *Science* 1997; 276: 122-126.

[5] Quetelet A. Recherches sur le poids de l'homme aux différent âges. Nouveaux Mémoire de l'Academie Royale des Sciences et Belles-Lettres de Bruxelles. 1832, t. VII.

[6] Quetelet A. Sur l'homme et le développement de ses facultés. Essai de physique social. Paris, Bachelier, Imprimeur-Libraire, 1835, t. 1.

Best fitting model

We want the simplest, most parsimonious model. To that end we compare different models using the Schwarz Bayesian Criterion SBC = deviance + df·log *n*, where the fitted deviance is −2·log likelihood, *n* is the sample size and df is the total model degrees of freedom. Over-fitting is avoided by selecting the model with the smallest SBC.

We start as described in the introduction: read the data file, create the age and height intervals, then create the object `mm` which contains data on males only. Note that the function `cutbin()` needs to have been created.

```
setwd("D:\\LungFunction\\GAMLSS-for-statisticians")
ALL <- read.csv("data.csv")
ALL$ageintrvl <- cutbin(ALL$age, 2)
ALL$heightintrvl <- cutbin(ALL$height, 5)
summary(ALL)
mm <- ALL[ALL$sex == 1, ]
```

What model to choose? As argued above, we expect volumes like the FEV_1 to be a power function of height, implying log transformation of both volume and height. We also argued that log(age) makes the child and adult periods more comparable. Thus we think that the model

$$\log(\text{index}) = f(\log(\text{height}), \log(\text{age}))$$

is biologically plausible. In the simplest form this translates into

```
mm.fev <- gamlss(fev ~ log(height) + log(age),family = BCCG(mu.link = "log"),
    data=mm)
```

where ~ indicates function of. This is a conventional linear equation, where `mu.link = log` indicates that the predicted mean (the M in LMS, mu, μ) relates to the log transformed FEV_1, and `data=mm` signifies that the analysis relates to object `mm`. This approach is deficient in many respects, as we shall demonstrate.
Let us first look at the data in more detail by studying log-log plots.
Opening a graphical device for each new graph using the command `windows()` offers the advantage that successive plots can be compared. Graphical devices are numbered, so one can issue several commands such as `dev.prev()`, `dev.next()`, `dev.set(which=k)`, `dev.off(k)`; `graphics.off()` terminates all graphics devices, except the null device.

```
windows(6,5)
```

Generate a log-log plot.

```
plot(mm$height, mm$fev, log="xy", las=1, xlab="log height", ylab="log FEV1")
```

Generate a grid, if you wish.

```
abline(v=seq(100,200,20), lty=3)
abline(h=seq(1,9,1), lty=3)
```

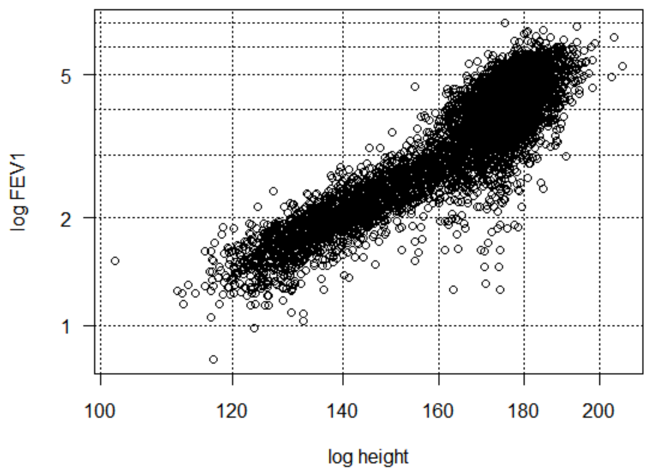

Figure 7

There seems to be a slight change in the slope at about height 165 cm. The trend is slightly obscured by the increased variability. So let us generate a log-log box plot for a clearer picture.

```
range(log(mm$age))
[1] 1.811414 4.488636
```

```
range(log(mm$height))
[1] 4.624973 5.330300
```

Divide log(age) and log(height) into intervals.

```
mm$logageintrvl <- cutbin(log(mm$age), 0.2)
mm$logheightintrvl <- cutbin(log(mm$height), 0.05)
windows(6,5)
plot(mm$logheightintrvl, log(mm$fev), las=1, xlab="log height interval",
    ylab="log FEV1")
```

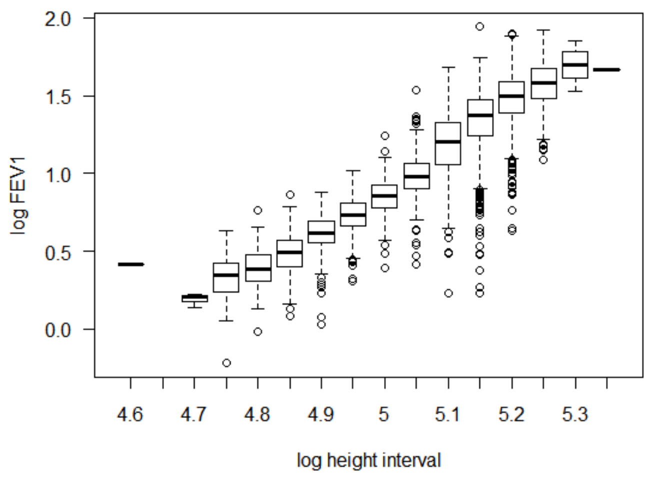

Figure 8

A fairly linear relationship, but there seems to be a step upwards at 165 cm (log(165) = 5.1). Log transformation does not stabilise the variance over the entire age range.

```
windows(6,5)
plot(mm$logageintrvl, log(mm$fev), las=1, xlab="log age interval", ylab="log
    FEV1", main=”Males”)
```

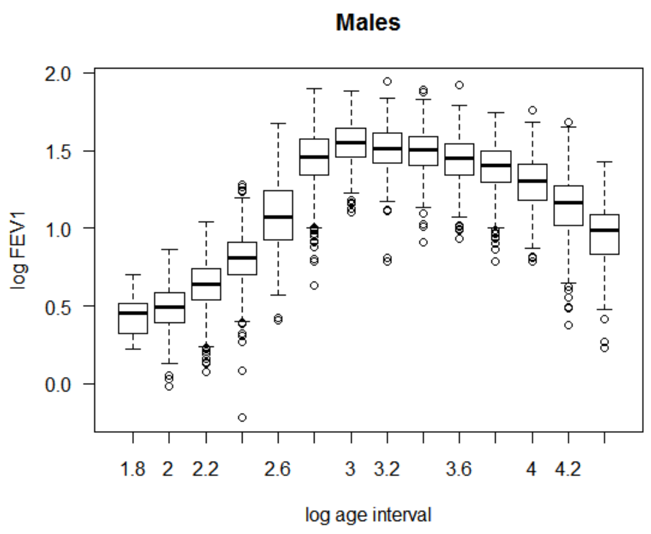

Figure 9

Curvilinear increase until about age 12 (log(12) ≅ 2.5), then a stepwise increase towards adulthood, followed by a plateau until about age 30 (log(30) = 3.4), and then an accelerating decline. The variability in FEV_1 increases sharply at age 12, the start of the growth spurt in boys. The growth spurt does not commence at the same time in all individuals, so that the scatter in height, and hence in FEV_1, increases.

It follows that

- one height exponent for all ages seems to make sense (see Cole *et al*. 2009[7]),
- log transformation of age makes the age range in children and adults about the same,
- the scatter (the S in LMS, sigma, σ) needs to be modelled so that residuals are normally distributed,
- a smoothing spline is required to properly model age-related lung growth and subsequent decline.

Which model

Please note that any age transformation should be the same for `mu`, `sigma` and `nu`:

```
mm.fev <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)),
    nu.fo =~log(age), family = BCCG(mu.link = "log"), data=mm)
```

In the case of a normal distribution `nu=1`, so let us force that value as follows:

```
mm.fev2 <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)),
    nu.fix=T, nu.start=1, family = BCCG(mu.link = "log"), data=mm)
```

By typing `summary(mm.fev)` or `summary(mm.fev2)` one obtains the summary statistics for each object, and can then compare the SBCs. However, that is cumbersome, particularly if there are many objects. Instead we use `GAIC()`. Instead of the generic AIC() function, where the penalty function *k* defaults to `k=2`, we use a heavier penalty function (which protects against over-fitting) with `k = log(length(mm$fev))`, which gives us the SBC. Don't be led astray by the fact that the output says `AIC`, you are shown the SBC of each object. As the

[7] Cole TJ, Stanojevic S, Stocks J, Coates AL, Hankinson JL, Wade AM. Age- and size related reference ranges: A case study of spirometry through childhood and adulthood. *Statist Med* 2009; 28: 880-898.

number of males is 5,723, each extra degree of freedom penalises the SBC by log(5723) = 8.65 units.

```
GAIC(mm.fev, mm.fev2, k=log(length(mm$fev)))
                df      AIC
mm.fev2 22.21772 5536.89
mm.fev  24.14602 5546.97
```

GAIC is usually applied to a series of models, and it's useful to be able to specify the model names using a pattern rather than listing them all. The routine `GAICpatt()` generalises GAIC to allow this. To use it first source it

```
GAICpatt <- function(patt, k=2) eval(parse(text=paste('GAIC(',
    paste(ls(envir=parent.frame(), patt=patt), collapse=','),', k=', k, ')',
    sep='')))
```

To use `GAICpatt()` specify the required model names using a "grep" pattern, the example here being a name starting with '`mm`' and followed by at least one character. This identifies `mm.fev` and `mm.fev2` but excludes `mm`. Also define k in the same way as for `GAIC()`.

```
GAICpatt('^mm.', k=log(length(mm$fev)))
                df      AIC
mm.fev2 22.21772 5536.89
mm.fev  24.14602 5546.97
```

`mm.fev2` is the more parsimonious model. Let us look at the summary statistics:

```
summary(mm.fev)
Family: c("BCCG", "Box-Cox-Cole-Green")

Call: gamlss(formula = fev ~ log(height) + pb(log(age)), sigma.formula =
     ~pb(log(age)), nu.formula= ~log(age), family = BCCG(mu.link =
     "log"), data = mm)

Fitting method: RS()

--------------------------------------------------------------------
Mu link function: log
Mu Coefficients:
               Estimate    Std. Error    t value     Pr(>|t|)
(Intercept)   -10.54905      0.092709    -113.79    0.000e+00
log(height)     2.27557      0.019882     114.45    0.000e+00
pb(log(age))    0.04322      0.003708      11.66    4.718e-31
--------------------------------------------------------------------
Sigma link function: log
Sigma     Coefficients:
               Estimate    Std. Error    t value     Pr(>|t|)
(Intercept)     -2.4682       0.04180    -59.053    0.000e+00
pb(log(age))     0.1177       0.01341      8.774    2.242e-18
--------------------------------------------------------------------
Nu link function: identity
Nu Coefficients:
               Estimate    Std. Error    t value     Pr(>|t|)
(Intercept)      0.5798        0.3624      1.600       0.1097
log(age)         0.1829        0.1114      1.641       0.1008
--------------------------------------------------------------------
No. Of observations in the fit:            5723
```

```
Degrees of Freedom for the fit:         24.14602
    Residual Deg. Of Freedom:           5698.854
    At cycle:          8

Global Deviance:               5338.052
    AIC:                       5386.345
    SBC:                        5546.97
```

Note: `method RS()`. The current algorithms for GAMLSS are RS(), CG() and mixed(). *i.e.* method=RS() will use the Rigby and Stasinopoulos algorithm, method=CG() will use the Cole and Green algorithm and mixed(2,10) will use the RS algorithm twice before switching to the Cole and Green algorithm for up to 10 extra iterations. The default for method is `RS()` . All methods end up with essentially the same fitted model.

```
summary(mm.fev2)
Family: c("BCCG", "Box-Cox-Cole-Green")

Call: gamlss(formula= fev ~ log(height) + pb(log(age)), sigma.formula =
    ~pb(log(age)), family = BCCG(mu.link       ="log"), data = mm, nu.start =
    1, nu.fix = T)

Fitting method: RS()
-------------------------------------------------------------------
Mu link function: log
Mu Coefficients:
                Estimate   Std. Error    t value     Pr(>|t|)
(Intercept)    -10.56310     0.093333    -113.18    0.000e+00
log(height)      2.27937     0.020035     113.77    0.000e+00
pb(log(age))     0.04112     0.003767      10.92    1.764e-27
-------------------------------------------------------------------
Sigma link function: log
Sigma Coefficients:
                Estimate   Std. Error    t value     Pr(>|t|)
(Intercept)      -2.4830      0.04180    -59.407    0.000e+00
pb(log(age))      0.1237      0.01341      9.221    4.041e-20
-------------------------------------------------------------------
Nu parameter is fixed
Nu = 1
-------------------------------------------------------------------
No. of observations in the fit:             5723
Degrees of Freedom for the fit:         22.21772
     Residual Deg. of Freedom:          5700.782
     at cycle: 5

Global Deviance:               5344.657
 AIC:                          5389.092
 SBC:                           5536.89
```

We will discuss later how to judge whether there is a good fit to the data. At this stage suffice it to say that model `mm.fev2` is to be preferred: the simplest and most parsimonious model, it requires 2 df less than `mm.fev` and produces a slightly lower `SBC`. In `mm.fev` age does not seem to contribute any information

This is about as good as it gets. The following command provides a wealth of information: The following command gives a white background

```
op <- par(mfrow = c(2, 2), mar = par("mar") + c(0, 1, 0, 0), bg = "white")
plot(mm.fev2, xvar=mm$ageintrvl, par=op)
```

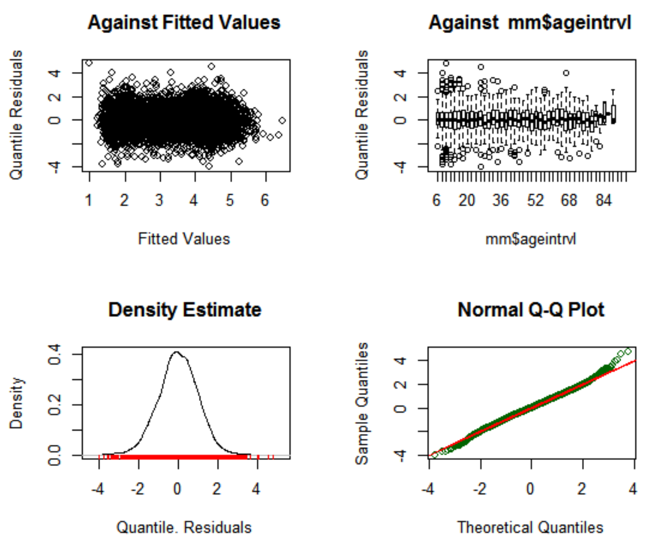

Figure 10

The quantile (standardised) residuals are evenly distributed over predicted values (upper left). There is some age dependency of variability (upper right). The frequency density plot (lower left) is compatible with a normal distribution with some skewness to the right, which is confirmed in the Q-Q plot (lower right). The distribution appears to be good between Z-scores -3 and +3, which covers the 99.7 percent confidence interval of the mean; for practical purposes this is a satisfactory fit to the data. For safety's sake, let us inspect the range of residuals:

```
range(resid(mm.fev2))
[1] -4.006300  4.785629
```

It is a matter of judgement, but we are inclined to remove data only if the Z-score is <-5 or >+5. In this example no need to worry.

The function `centiles()` also provides useful insight how well the fitted distribution agrees with a normal distribution. The function is appropriate for one continuous explanatory variable in the model. Remember that in the models `fev` was log transformed, and a function of log(height). In this case the problem can therefore be overcome by calculating `fev` divided by height^k, where k is the coefficient for log(height) in `mm.fev` or `mm.fev2`; this leaves only one explanatory variable, *i.e.* log(age). The intercept is the first, k is the second linear regression coefficient (value ~2.28). In practice the two estimates are very similar, and the second one is used.

```
mm$fevhtk <- mm$fev / (mm$height / 100) ^ mm.fev$mu.coeff[2]
mm.corr1 <- gamlss(fevhtk ~ pb(log(age)), sigma.fo =~ pb(log(age)), nu.fo
    =~log(age), family = BCCG(mu.link = "log"), data=mm)
mm.corr2 <- gamlss(fevhtk ~ pb(log(age)), sigma.fo =~ pb(log(age)),
    nu.fix=TRUE, nu.start=1, family = BCCG(mu.link = "log"), data=mm)
```

We can now inspect the age dependence of height-corrected FEV_1 by comparing the distribution of centiles:

```
centiles.com(mm.corr1,mm.corr2,xvar=mm$age)
********  Model 1 ********
% of cases below  0.4 centile is  0.5591473
% of cases below  10 centile is  9.837498
% of cases below  50 centile is  50.32326
% of cases below  90 centile is  90.73912
% of cases below  99.6 centile is  99.33601
********  Model 2 ********
% of cases below  0.4 centile is  0.6989341
% of cases below  10 centile is  9.959811
% of cases below  50 centile is  49.9039
% of cases below  90 centile is  90.80902
% of cases below  99.6 centile is  99.42338
```

For practical purposes this fit is acceptable, even at the extremes. Based on the SBC there was a preference for object mm.fev2. The figure illustrates the centile curves (0.4, 2, 5, 50, 95, 98 and 99.6 centile) for this model. Note the use of `expression()` to obtain the subscripts and superscripts in the y-axis label.

```
centiles(mm.corr2, xvar=mm$age, cent=c(0.4, 2, 5, 50, 95, 98, 99.6),
    col.centiles=c("green","blue","red","black","red","blue","green"),
    legend=FALSE, ylab=expression(FEV[1]/height^k~~(L/m^k)), xlab="(Age (yr)",
    main="", points=TRUE, pch=20, col="gray", plot=TRUE, lwd=2, las=1)
```

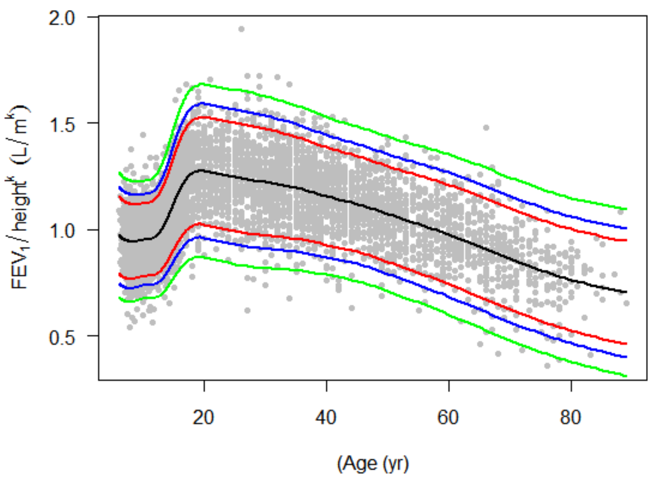

Figure 11

Finally a look at the 'worm plot' (`wp()`).Worm plots[8], *i.e.* de-trended Q-Q plots, also provide information about the goodness of fit. The figure below shows the deviations as a function of age, subdivided over 20 cells.

[8] van Buuren S, Fredriks M. Worm plot: simple diagnostic device for modelling growth reference curves. *Stat Med* 2001; 20, 1259–1277.

```
wp(mm.fev2, xvar=mm$age, n.inter=20)
```

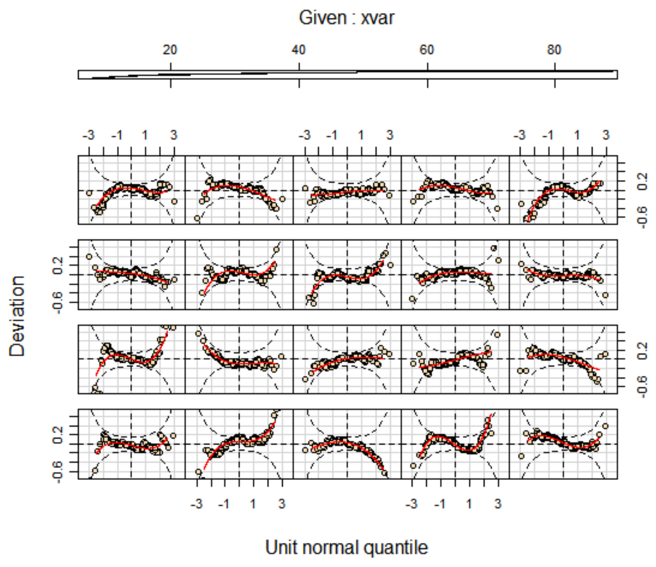

Figure 12

Each of the 20 panels shows the 95 percent confidence interval. As data become scarcer, the interval becomes broader towards the extremes, so in the tails larger differences between theoretical and empirical quantiles are acceptable. In general the mean values of empirical and theoretical quantiles agree well, and with some exceptions the graphs have a horizontal trend, suggesting a normal distribution and acceptable fit.

Other distributions

With the `BCPE` distribution one can also model kurtosis.

```
mm.bcpe <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)),
    nu.fo =~log(age), tau.fo=~log(age),family = BCPE(mu.link = "log"), data=mm)
```

This model requires `df=26` and produces an intermediate `SBC`: 5536.619; the coefficient for tau is not significant (see `summary(mm.bcpe)`), suggesting that there is no platy- or leptokurtosis.

For practical purposes there is no relevant improvement in SBC. Use `plot(mm.bcpe)`, and you will notice a very slight improvement in the skewness. But any improvement is primarily beyond +3 and -3 z-scores, which is unimportant in clinical practice.

Finally the normal distribution:

```
mm.no <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)),
    family = NO(mu.link = "log"), data=mm)
```

This produces an `SBC`: 5553.863, slightly larger than that for `mm.fev2` (5536.89), obtained with the `BCCG` distribution. This confirms the earlier finding that with the `BCCG` distribution `nu` = 1, *i.e.* a normal distribution.

Conclusion: the `BCCG` distribution is well suited for our analyses.

We argued that log transformation of height and FEV$_1$ was called for, and modelled accordingly. Let us put this to the test. First no transformation of the FEV$_1$ (`mu.link = "identity"`).

```
ca <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)), nu.fo
    =~log(age), family = BCCG(mu.link = "identity"), data=mm)
Error in dBCCG(y, mu = mu, sigma = sigma, nu = nu, log = TRUE) :
  mu must be positive
```

Assuming normality, *i.e.* `nu = 1.`

```
cb <- gamlss(fev ~ log(height) + pb(log(age)), sigma.fo =~ pb(log(age)),
    nu.start = 1, nu.fix = T, family = BCCG(mu.link = "identity"), data=mm)
Error in dBCCG(y, mu = mu, sigma = sigma, nu = nu, log = TRUE) :
  mu must be positive
```

No transformation of height.

```
cx <- gamlss(fev ~ height + pb(log(age)), sigma.fo =~ pb(log(age)), nu.fo
    =~log(age), family = BCCG(mu.link = "log"), data=mm)
GAIC(cx,mm.fev, k=log(length(mm$fev)))
               df          AIC
mm.fev2  24.14602     5546.970
cx       23.92572     5599.032
```

Two models cannot be fit, and model `cx` performs poorly. Changing the `df` led to the same error messages for `ca` and `cb`. There is no point in using untransformed height or FEV$_1$.

Regression terms

One can plot regression terms against their predictors, with standard errors:

```
term.plot(mm.fev2, what="mu", se=TRUE, partial=FALSE, col.term="black",
    col.se="black", las=1)
```

Figure 13 depicts the age spline for the predicted value (`mu`), Figure 14 the contribution of the age spline for variability (`sigma`). Please note that in GAMLSS the effect of age is presented as a linear term plus the age spline. The dotted lines represent the 95 percent confidence interval.

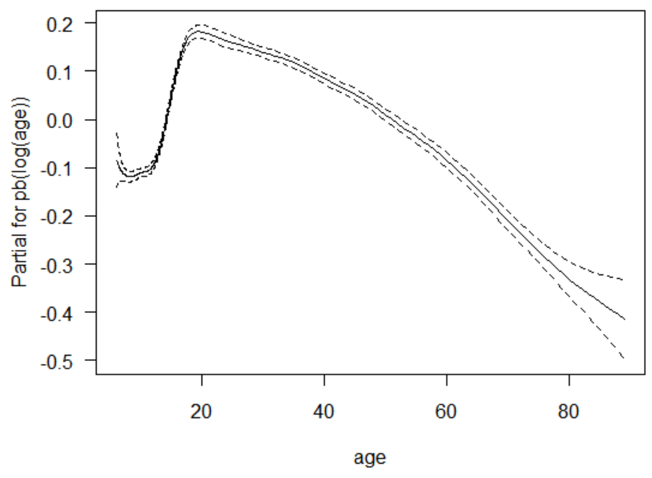

Figure 13

```
term.plot(mm.fev2, what="sigma", se=TRUE, partial=FALSE, col.term="black",
    col.se="black", las=1)
```

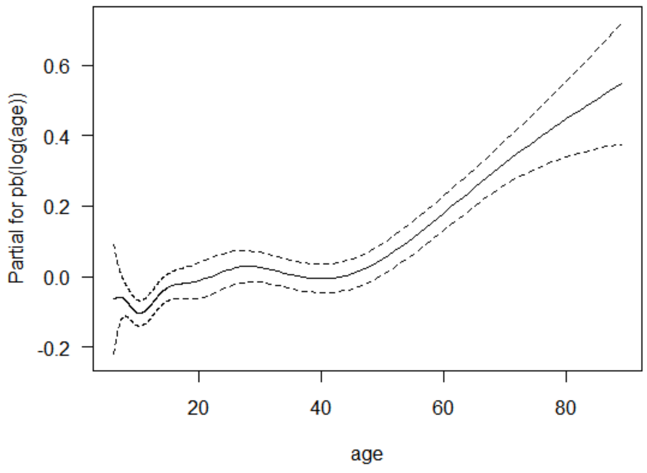

Figure 14

```
term.plot(mm.fev2, what="nu", se=TRUE, partial=FALSE, col.term="black",
    col.se="black", las=1)
Error in seq.default(len = p) : argument 'length.out' must be of length 1
```

This instruction was submitted unthinkingly. In `mm.fev2` we forced nu to be 1, so there is no reason to expect any partial residuals.

Generating predicted values

Once the model has been decided upon, it is a simple matter to generate predicted values. Remember that `fev` was log transformed with the log link, hence exponentiation is required to obtain the predicted values:

```
mm$predictfev <- exp(predict(mm.fev2))
```

Similarly one can generate z-scores for each measured value:

```
mm$zfev <- resid(mm.fev2)
```

In many cases one would like to show the predicted value as a function of age, but in that case one will have to substitute a typical height for age. We generate that as follows, using a simpler model than for FEV$_1$:

```
mm.height <- gamlss(height ~ pb(log(age)), sigma.fo =~pb(log(age)), family =
    NO, data=mm)
```

We must add (age-specific average) height to age, otherwise calculating the predicted value for FEV$_1$ will not work, as it is based on height and age.

```
p <- data.frame(age=6:94)
p$height <- predict(mm.height, newdata=p, data=mm)
plot(p$age, p$height, las=1, xlab="Age (yr)", ylab="Predicted height (cm)",
    type="l", lwd=3)
```

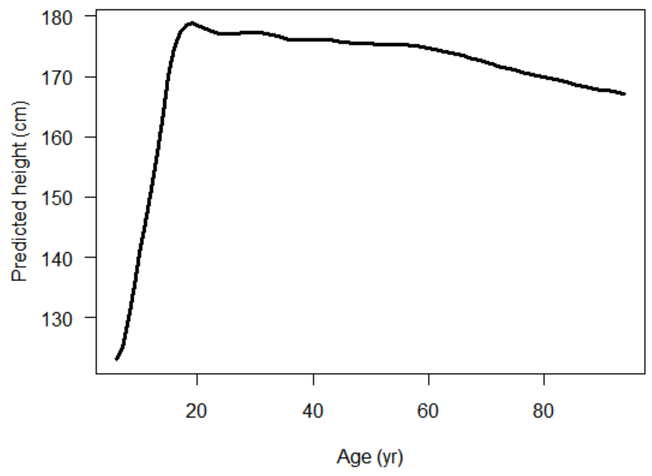

Figure 15

We can now use `p` to predict the FEV$_1$:

```
mm.predfev <- exp(predict(mm.fev2, newdata=p, data=mm))
plot(p$age, mm.predfev, las=1, xlab="Age (yr)", ylab="Predicted FEV1 (L)",
    type="l", lwd=3)
```

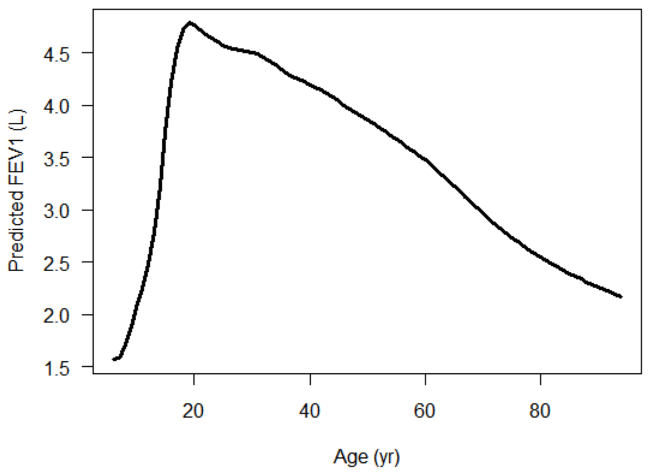

Figure 16

The following displays the coefficient of variation as a function of age (again we must exponentiate, as the link for `sigma` was log):

```
mm.cov <- exp(predict(mm.fev2, what="sigma", newdata=p, data=mm))
plot(p$age, mm.cov, las=1, xlab="Age (yr)", ylab="Coeff. of variation",
    type="l", lwd=3)
```

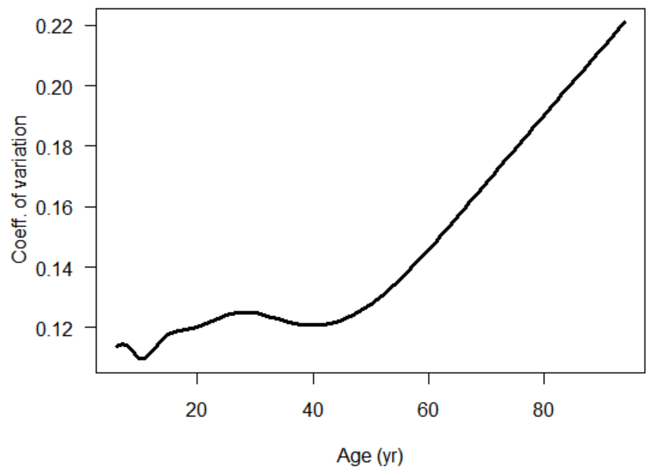

Figure 17

Implementing the prediction equations in software

Since we have decided to stick with model `mm.fev2`, let us look at the output of

```
summary(mm.fev2)
```

```
Mu link function:  log
Mu Coefficients:        Estimate
 (Intercept)           -10.56310
log(height)              2.27937
pb(log(age))             0.04112

Sigma link function:  log
Sigma Coefficients:     Estimate
 (Intercept)             -2.4830
pb(log(age))              0.1237

Nu link function:  identity
Nu parameter is fixed
Nu = 1
```

Therefore:

mu = FEV_1 = exp(-10.56310 + 2.27937·log(height) + 0.04112·log(age) + mu-spline)
sigma = CoV = exp(-2.4830+ 0.1237·log(age) + sigma-spline)
nu = 1

The contribution of the splines varies with age. In practice one needs to retrieve that information and store it in a look-up table. This information can be retrieved as follows. First generate a list for age at 0.25 year intervals; this will allow calculating predicted values with sufficient accuracy (see www.lungfunction.org/implementingequations.html).

```
p <- data.frame(age=seq(6,94,0.25))
```

We must include height in the list, as the prediction equation was based on height and age. The value of height is irrelevant, since we are only interested in the age spline component. We select a value of 1: when substituted in the equation, log(height) = 0.

```
p$height <- 1
```

Use the new list to predict mu and sigma.

```
M <- predict(mm.fev2, newdata=p, data=mm)
S <- predict(mm.fev2, what="sigma", newdata=p, data=mm)
```

Subtracting the components defined by the linear terms leaves us with the contribution from the age spline.

```
p$Mspline <- M - (mm.fev2$mu.coeff[1] + mm.fev2$mu.coeff[3] * log(p$age))
p$Sspline <- S - (mm.fev2$sigma.coeff[1] + mm.fev2$sigma.coeff[2] *
    log(p$age))
```

Please note that there was no spline for `nu`. As we are storing a look-up table with contributions from a spline, we will record Lspline = 0

```
p$Lspline <- 0
```

We only need age and the contributions from the splines, so let us remove height.

```
p$height <- NULL
```

Write the resulting file to disc.

```
write.csv(p, file="LookupTable.csv")
```

Out of curiosity we can look at the Mspline as a function of age:

```
windows(6,5)
plot(p$age, p$Mspline, xlab="Age (yr)", ylab="Mspline", type="l", lwd=3,
    las=1)
abline(h=0,lty=3)
```

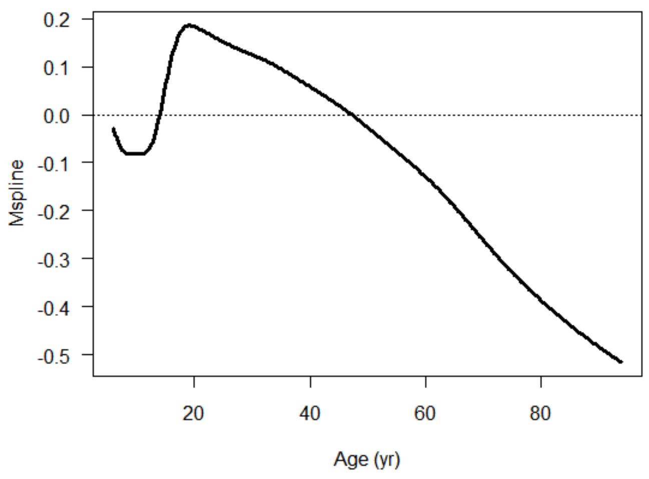

Figure 18

Now the same for the Sspline (coefficient of variation):

```
plot(p$age, p$Sspline, xlab="Age (yr)", ylab="Sspline", type="l", lwd=3,
    las=1)
abline(h=0,lty=3)
```

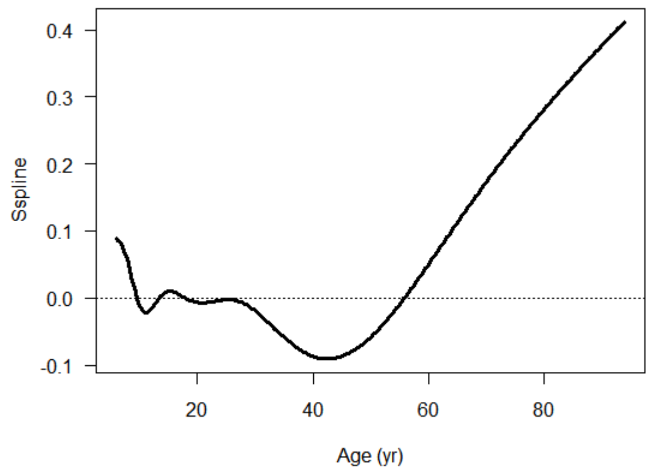

Figure 19

These figures demonstrate how the linear age coefficients are 'adjusted' by the spline function.

Differences between centres

The collated dataset that we have analysed derives from 4 centres. One potential concern is that there are systematic differences between centres. We can analyse differences between centres (predicted mean, variability) by including them in the analysis. First we redefine centre as a factor.

```
mm$centre <- factor(mm$centre)
mm.centre <- gamlss(fev ~ log(height) + pb(log(age)) + centre, sigma.fo =~
    pb(log(age)) + centre, nu.fix=TRUE, nu.start=1, family = BCCG(mu.link =
    "log"), data=mm)
GAIC(mm.fev2, mm.centre, k=log(length(mm$fev)))
                df       AIC
mm.centre 27.85963 5488.92
mm.fev2   22.21772 5536.89
```

The `SBC` improves by 48 units, at the expense of 5½ extra degrees of freedom.

What about the linear coefficients for `mu` and `sigma`?

```
summary(mm.centre)
Mu link function: log
Mu Coefficients:
                Estimate Std. Error    t value    Pr(>|t|)
(Intercept)    -10.350977    0.104291      -99.251    0.000e+00
log(height)      2.235264    0.021781      102.624    0.000e+00
pb(log(age))     0.055127    0.004052      13.606     1.636e-41
centre1         -0.040922    0.006924      -5.910     3.611e-09
centre2         -0.035810    0.006211      -5.766     8.545e-09
centre3         -0.007542    0.007196      -1.048     2.946e-01
-------------------------------------------------------------------
Sigma link function:  log
Sigma Coefficients:
                Estimate Std. Error     t value       Pr(>|t|)
(Intercept)      -2.48834      0.05619   -44.2878     0.000e+00
pb(log(age))      0.08226      0.01552     5.3000     1.201e-07
```

```
centre1            0.08315      0.04469     1.8607    6.284e-02
centre2            0.17747      0.03954     4.4887    7.306e-06
centre3           -0.02924      0.04529    -0.6456    5.185e-01
-------------------------------------------------------------------
Nu parameter is fixed
Nu = 1
-------------------------------------------------------------------
No. of observations in the fit: 5723
Degrees of Freedom for the fit:27.85963
    Residual Deg. of Freedom: 5695.14
    At cycle:             6
Global Deviance: 5247.872
    AIC:          5303.591
    SBC:           5488.92
```

Conclusion

The coefficients compare each centre with centre 4, which acts as the baseline. Due to FEV_1 being natural log transformed, the coefficients can be multiplied by 100 and treated as percentage differences. Data from centres 3 and 4 are broadly similar. Centres 1 and 2 have a 3.6-4.0% smaller mean FEV_1 than centre 4, and 8-18% larger variability.

Do the age ranges overlap in the various centres? Calculate and save the ranges for later.

```
ar <- with(mm, tapply(age, centre, function(x) round(range(x))))
ar
$`1`
[1]  8 80

$`2`
[1]  7 89

$`3`
[1]  6 13

$`4`
[1] 12 19
```

Centres 1 and 2 cover a wide age range, centre 3 school children, centre 4 adolescents. Display and save the predicted values respecting the age range in the various centres. First establish the overall range of predicted values.

```
p <- data.frame(age=c(6:94))
p$height <- predict(mm.height, newdata=p, data=mm)
p$predfev <- exp(predict(mm.fev2, newdata=p, data=mm))
windows(6,5)
plot(range(p$age), range(p$predfev), las=1, xlab="Age (yr)", ylab="Predicted
    FEV1 (L)", type="n", bty="l")
```

Then for each centre create the predicted values over the relevant age range, plot them, and save the data frames in p1, p2 etc.

```
for (i in 1:4) {
    p <- data.frame(age=seq(ar[[i]][1], ar[[i]][2], 1))
    p$height <- predict(mm.height, newdata=p, data=mm)
    p$centre <- i
    p$predfev <- exp(predict(mm.centre, newdata=p, data=mm))
    with(p, lines(age, predfev, col=c("black", "blue", "green", "red")[i]))
    assign(paste('p', i, sep=''), p)
}
```

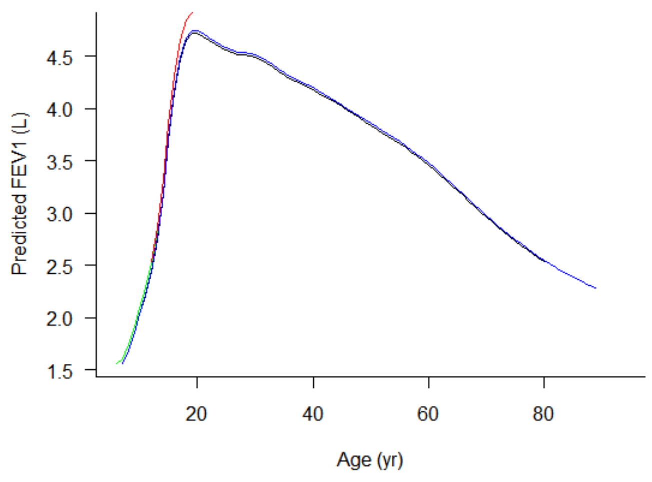

Figure 20

The differences between centres can be due to many reasons: different population sample, adherence to protocol, trained or untrained subjects, quality control, training of lung function technicians, *etc*. In this case it seems that children and adolescents have a slightly higher FEV$_1$ for age and height than adults. Should you be able to contact the different centres you would learn that the children and adolescents had performed spirometry on many occasions and were therefore well trained, unlike the adults who were naïve with respect to spirometry. One can depict the difference in variability between centres in the same way as above.

Finally, clean up:

```
dev.off()
q()
```

APPENDIX

Cubic splines versus penalised beta splines

There are two alternative ways to fit splines: cubic splines `cs()` and penalised beta splines `pb()`. With `cs()` one needs to specify the required degrees of freedom (`df`) for the spline, while `pb()` estimates the `df` itself. The procedure for `cs()` is to start with *e.g.* `df`=0 (no smoothing) and successively increase the `df` until the SBC is minimised. With very large datasets (>10,000 observations) you may receive the warning 'The output df are different from the input, change the control.spar'. In that case increase the upper limit in `c.spar`, for example `c.spar=c(-1.5, 2.5)` or `c.spar=list(-1.5, 2.5)`. These parameters for `c.spar` (the default is `c.spar=c(-1.5, 2)`) were obtained empirically: too narrow or too wide ranges produce warnings; using different data you may need different values. Type `?cs` for more information.

Occasionally you may find that for small `df`, increasing the `df` does not reduce the SBC. You may be tempted to choose the model with the smallest `df`, but it may be only a local minimum. Therefore it is a good idea to increase `df` to, say, up to 10 for `mu`, to avoid stopping at a local minimum.

As you will see, estimating the optimum `df` for μ, σ and ν can be quite time consuming, so you may be tempted to start with a fairly high `df`. We do not recommend that, as a minimum SBC may occur with fewer extra degrees of freedom.

Let us now examine models which use `cs()` and the `BCCG` distribution We must find the optimum `df` for `mu`, then for `sigma`, and finally for `nu`.

```
rm(list=ls(all=TRUE))
library(gamlss)
setwd("D:\\LungFunction\\GAMLSS-for-statisticians")
ALL <- read.csv("data.csv")
cutbin <- function(x, k) {
     # x = variable, k = bin width
     # mid-bin labels are multiples of k
     k2 <- k / 2
     b <- seq(floor(min((x - k2) / k)) * k, ceiling(max((x - k2) / k)) * k, k)
     cut(x, breaks=b + k2, labels=b[-1])
}
ALL$ageintrvl <- cutbin(ALL$age, 2)
```

Select data on males.

```
mm <-ALL[ALL$sex == 1, ]
```

The first model is a simple linear one as the degrees of freedom for the age spline is 0 (a more formal notation would be …….. + cs(log(age),df=0, ……….)

```
a0 <- gamlss(fev ~ log(height) + cs(log(age),0), family = BCCG(mu.link =
    "log"), data=mm)
GAMLSS-RS iteration 1: Global Deviance = 6143.252
GAMLSS-RS iteration 2: Global Deviance = 6149.307
Error in RS() : The global deviance is increasing
 Try different steps for the parameters or the model maybe inappropriate
In addition: There were 50 or more warnings (use warnings() to see the first
    50)
```

Type `warnings()` and you will see the message

```
Warning messages:
1: In gamlss.cs(data[["cs(log(age), 0)"]], z, w, spar = NULL,  ... :
  The output df 3.71242338858787  are different for the input 2
change the control.spar
```

As suggested above change the script into

```
a0 <- gamlss(fev ~ log(height) + cs(log(age),0, c.spar=c(-1.5, 2.5)), family =
    BCCG(mu.link = "log"), data=mm)
Warning message:
In RS() : Algorithm RS has not yet converged
```

The above parameters for `c.spar` were obtained empirically and are valid for this dataset: narrower or wider ranges produce warnings.

Another error message. The default number of convergence cycles is 20, and is defined in `gamlss.control()`. Increase the number of iterations:

```
con <- gamlss.control(n.cyc=200)
```

If you want to speed up calculations you might relax the convergence criterion. The default value is 0.001, but you might want to use:
`con <- gamlss.control(n.cyc=200, c.crit=0.01)` and return to the default value when you are finished choosing your model.

```
a0 <- gamlss(fev ~ log(height) + cs(log(age),0, c.spar=c(-1.5, 2.5)),
    control=con, family = BCCG(mu.link = "log"), data=mm)
..............
GAMLSS-RS iteration 26: Global Deviance = 7614.611
GAMLSS-RS iteration 27: Global Deviance = 7621.273
Error in RS() : The global deviance is increasing
 Try different steps for the parameters or the model maybe inappropriate
```

We previously saw the relationship between log(age) and log(fev) (Figures 1 and 4), and there was no reason to expect a satisfactory fit from a linear equation (df=0 for log(age)).. So let us skip `a0`. Since it is such a waste of time having to re-analyse after warnings, let us systematically use `c.spar()` and `con`.

```
a1 <- gamlss(fev ~ log(height) + cs(log(age),1, c.spar=c(-1.5, 2.5)), family =
    BCCG(mu.link = "log"), control=con, data=mm)
GAMLSS-RS iteration 1: Global Deviance = 6670.791
GAMLSS-RS iteration 2: Global Deviance = 6678.086
Error in RS() : The global deviance is increasing
 Try different steps for the parameters or the model maybe inappropriate
```

Let us skip `a1` as well.

Rather than writing out these models one by one, we use a script that fits models a2 to a11 as follows, where the number defines the `df`:

```
for (i in 2:11) assign(paste('a',i,sep=’’), gamlss(fev ~ log(height) +
    cs(log(age), i, c.spar=c(-1.5, 2.5)), family = BCCG(mu.link = "log"),
    control=con, data=mm))
```

One might prefer to use the function `update()`. In this case we start with `a3` as follows:

```
for (i in 3:11) assign(paste('a', i, sep=''),
    update(get(paste('a', i-1, sep='')), . ~ . + cs(log(age), i, c.spar=c(-1.5,
    2.5))))
```

By typing `summary(a2)` ... `summary(a11)` one obtains the summary statistics for each object, and can then compare the SBCs. However, that is cumbersome. Instead use `GAIC()` (see main text) where `k = log(length(mm$fev))`, which gives us the SBC. Even though the output says `AIC`, you are shown the SBC of each object.

```
GAIC(a2,a3,a4,a5,a6,a7,a8,a9,a10,a11, k=log(length(mm$fev)))
    df          AIC
a9  14.001614 5627.366
a8  13.001850 5628.062
a10 14.999733 5629.479
a11 15.997457 5633.261
a7  11.998841 5633.570
a6  11.001072 5647.482
a5  10.001030 5677.134
a4   9.000944 5736.570
a3   7.999327 5849.945
a2   6.999460 6085.134
```

This command can be generated automatically by including all objects whose names are a pattern starting with ‘a’ and followed by a number. The function `GAICpatt()` does this.

```
GAICpatt <- function(patt, k=2) eval(parse(text=paste('GAIC(',
    paste(ls(envir=parent.frame(), patt=patt), collapse=','),',', k=', k, ')',
    sep='')))
GAICpatt('^a[0-9]', k=log(length(mm$fev)))
```

Model `a9` comes out best (`df`=9 for the age spline). Hence proceed and model `sigma`, finding the optimal `df` by fitting models `b1` to `b5`.

```
for (i in 1:5) assign(paste('b', i, sep=''), gamlss(fev ~ log(height) +
    cs(log(age), 9, c.spar=c(-1.5, 2.5)), sigma.fo =~ cs(log(age), i, c.spar =
    c(-1.5, 2.5)), family = BCCG(mu.link = "log"), control=con, data=mm))
```

Compare SBC in models whose filenames start with ‘a’ or ‘b’ followed by a number.

```
GAICpatt('^[ab][0-9]', k=log(length(mm$fev)))
    df          AIC
b3  18.000553 5522.220
b2  17.000971 5523.755
b4  19.002460 5524.361
b5  20.002591 5528.499
b1  15.001517 5541.735
a9  14.001614 5627.366
a8  13.001850 5628.062
a10 14.999733 5629.479
a11 15.997457 5633.261
a7  11.998841 5633.570
a6  11.001072 5647.482
a5  10.001030 5677.134
a4   9.000944 5736.570
a3   7.999327 5849.945
a2   6.999460 6085.134
```

So `df`=3 is optimal for the `sigma` spline.

If the data are normally distributed, `nu`=1. We can force GAMLSS to accept that value via `nu.fix=T, nu.start=1`, and see whether that provides a good fit.

```
c0 <- gamlss(fev ~ log(height) + cs(log(age),9, c.spar=c(-1.5, 2.5)), sigma.fo
    =~ cs(log(age),3, c.spar=c(-1.5, 2.5)), nu.fix=T, nu.start=1, family =
    BCCG(mu.link = "log"), control=con, data=mm)
```

We proceed modelling `nu` as a function of log(age) by fitting models `c1` to `c4`.

```
for (i in 1:4) assign(paste('c', i, sep=''), gamlss(fev ~ log(height) +
    cs(log(age), 9, c.spar=c(-1.5, 2.5)), sigma.fo =~ cs(log(age), 3, c.spar =
    c(-1.5, 2.5)),       nu.fo =~cs(log(age), i,  c.spar=c(-1.5, 2.5)), family
    = BCCG(mu.link = "log"),  control=con, data=mm))
```

Compare SBC in models whose filenames start with 'a', 'b' or 'c' followed by a number.

```
GAICpatt('^[abc][0-9]', k=log(length(mm$fev)))
    df         AIC
c0  17.000841 5517.539
b3  18.000843 5522.220
b2  17.000971 5523.755
b4  19.002460 5524.361
b1  16.001091 5528.302
b5  20.002591 5528.499
c1  20.001229 5536.086
c2  21.000421 5544.218
c3  22.000169 5551.867
c4  23.001535 5559.014
a9  14.001614 5627.366
a8  13.001850 5628.062
a10 14.999733 5629.479
a11 15.997457 5633.261
a7  11.998841 5633.570
a6  11.001072 5647.482
a5  10.001030 5677.134
a4   9.000944 5736.570
a3   7.999327 5849.945
a2   6.999460 6085.134
```

Object `c0`, with `nu=1`, comes out as the best fitting model, judged by the SBC, signifying that the distribution was normal. But does `c0` provide a good fit?

Let us look at the coefficients of the equation:

```
summary(c0)
Family:  c("BCCG", "Box-Cox-Cole-Green")

Call:  gamlss(formula = fev ~ log(height) + cs(log(age), 9, c.spar = c(-1.5,
     2.5)), sigma.formula = ~cs(log(age), 3, c.spar = c(-1.5,
     2.5)), family = BCCG(mu.link = "log"), data = mm, nu.start = 1,
    nu.fix = T)

Fitting method: RS()
-------------------------------------------------------------------
Mu link function:  log
Mu Coefficients:
                 Estimate     Std. Error      t value     Pr(>|t|)
(Intercept)     -10.66928       0.092949     -114.79    0.000e+00
log(height)       2.30200       0.019952      115.38    0.000e+00
cs(log(age), 9, c.spar = c(-1.5, 2.5))
                  0.03817       0.003765       10.14    5.883e-24
-------------------------------------------------------------------
Sigma link function:  log
Sigma Coefficients:
                 Estimate     Std. Error      t value     Pr(>|t|)
(Intercept)       -2.4816        0.04180     -59.374    0.000e+00
cs(log(age), 3, c.spar = c(-1.5, 2.5))
```

```
                  0.1239      0.01341      9.241   3.359e-20
-------------------------------------------------------------------
Nu parameter is fixed
Nu =  1
-------------------------------------------------------------------
No. of observations in the fit:  5723
Degrees of Freedom for the fit:  17.00084
      Residual Deg. of Freedom:  5705.999
                      at cycle:  5

Global Deviance:     5370.443
            AIC:     5404.445
            SBC:     5517.539
*******************************************************************
Warning messages:
1: In vcov.gamlss(object, "all") :
  addive terms exists in the mu formula standard errors for the linear terms
    maybe are not appropriate
2: In vcov.gamlss(object, "all") :
  addive terms exists in the  sigma  formula standard errors for the linear
    terms maybe are not appropriate
3: In summary.gamlss(c0) :
  summary: vcov has failed, option qr is used instead
```

All coefficients are highly significant. As shown in the above output, in general we should interpret the probabilities with caution; the analyses comprise linear and additive terms, so the estimated standard errors may not be appropriate. How to interpret the age coefficient for `mu` will be addressed later.

The following command delivers a wealth of information.

The following script is the default in plot.gamlss and produces a beige background

```
# op <- par(mfrow = c(2, 2), mar = par("mar") + c(0, 1, 0, 0), col.axis =
    "blue4", col.main = "blue4", col.lab = "blue4", col = "darkgreen", bg =
    "beige")
```

This script produces a white background

```
op <- par(mfrow = c(2, 2), mar = par("mar") + c(0, 1, 0, 0), bg = "white")
plot(c0, xvar=mm$ageintrvl, par=op)
```

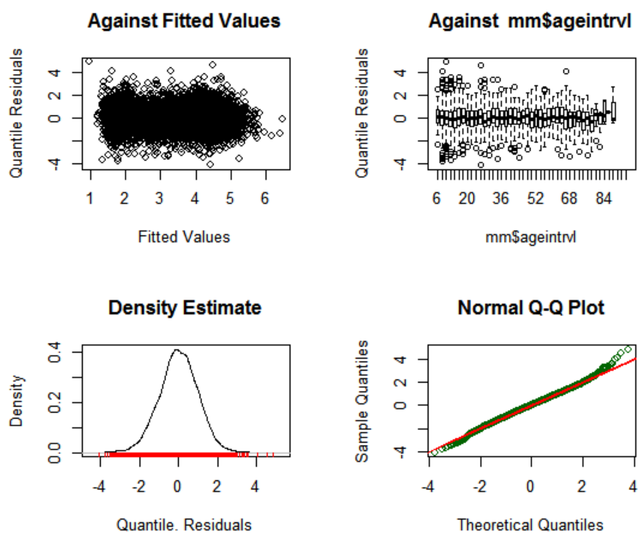

Figure 21

The quantile residuals as a function of the fitted values disclose that the quantile (standardised) residuals are evenly distributed over the predicted values (upper left). There is some age dependency of variability (upper right). The frequency density plot (lower left) is compatible with a normal distribution with some skewness to the right, which is confirmed in the Q-Q plot (lower right). The distribution appears to be very good (*i.e.* the Q-Q plot is linear) between z-scores -3 and +3, which covers the 99.7 percent confidence interval of the mean; for practical purposes this is a satisfactory fit to the data.

The analyses are sensitive to outliers, so it is always good to look for them. The Q-Q plot in the figure above is informative, but do inspect the range:

```
range(resid(c0))
 [1] -4.104653  4.856759
```

Worm plots also provide information about the goodness of fit. The figure below shows the deviations as a function of age, subdivided over 20 cells.

```
wp(c0, xvar=mm$age, n.inter=20)
```

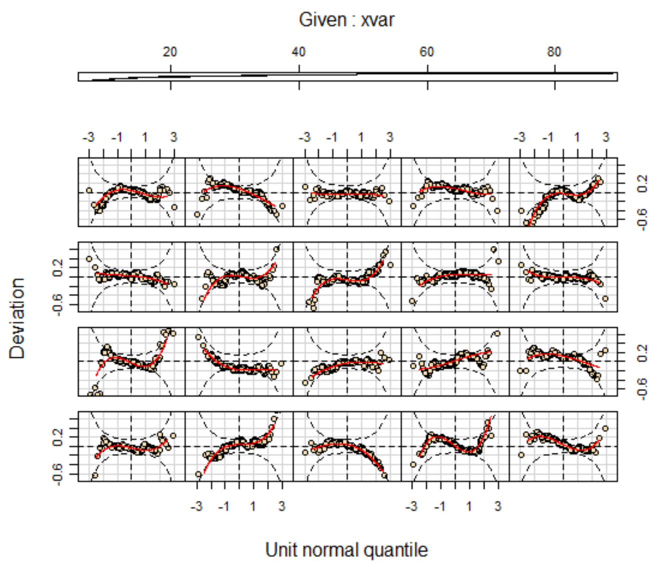

Figure 22

Each of the 20 panels shows the 95 percent confidence interval. In general the mean values of empirical and theoretical quantiles agree well, and with few exceptions have a horizontal trend, suggesting a normal distribution and good fit. Repeat this exercise for height:

```
wp(c0, xvar=mm$height, n.inter=15)
```

We have now run the analyses with `cs()` and `pb()`. Here is a comparison of the SBCs:

```
GAIC(c0, c1, c2, c3, c4, mm.fev2, k = log(length(mm$fev)))
                  df        AIC
      c0      17.00084   5517.539
      c1      19.00084   5528.102
      mm.fev2 22.21772   5536.890
      c2      21.00135   5544.226
      c3      22.00133   5551.875
      c4      23.00169   5559.015
```

Object `c0`, obtained with `cs()`, comes out best.

Quit:

```
q()
```

Conclusion: On the basis of this evidence the cubic spline produces a somewhat more parsimonious model than the penalised B-spline. However, `cs()` is far more cumbersome and time-consuming to use, as illustrated above. Also, the difference in the SBC is not dramatic. Finally, `pb()` is the current spline model in GAMLSS. Given these practical considerations, on balance `pb()` is to be preferred as the default.

Graph of two age distributions

We produced a graph of the age distribution in males (Figure 6, page 11). Often one would prefer to simultaneously display the distribution for males and females. Here is how to do that. Read the comma separated file with the data, and create the data frame 'ALL'.

```
ALL <- read.csv("data.csv")
```

Create a data frame containing the frequencies at 2 year age intervals by sex. Note this uses the script `cutbin()` defined earlier.

```
freq <- with(ALL, table(cutbin(age, 2), sex))
agedistr <- data.frame(cbind(age=as.numeric(rownames(freq)), freq))
```

Now plot the axes for the graph.

```
with(agedistr, plot(range(age), range(freq), type="n", bty="n", xlab="Age
    interval (yr)", ylab="Number of subjects", lwd=2, cex.lab=1.3,
    cex.axis=1.1, xaxp=c(0,100,10), yaxp=c(0,800,4), las=1))
```

Draw vertical lines for the frequencies by sex, thickness 4, offset ±0.5 and using different colours. Consult ?`par` under 'Color specification' for more information.

```
for (i in 1:2) lines(agedistr$age+c(-.5,.5)[i], agedistr[,i+1], lend="square",
    lwd=4, type="h", col=c("black","gray")[i])
```

Add a legend to explain the columns. Use `c()` for grouping arguments.

```
legend("right", c("Males      (N=5,723)", "Females (N=7,106)") , bty = "n", col
    = c("black", "gray"), lty = "solid", lwd=4, cex = 1.1)
```

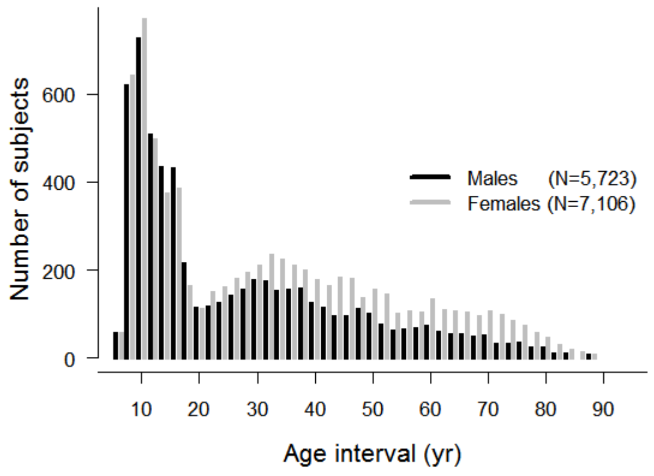

Figure 23