GAMLSS practicals for the Graz short course

Mikis Stasinopoulos

November 5, 2016

1 Day 1 Morning

1.1 Practical 1: A simple example using the gamlss packages

The following is an example from Chapter 2 of the book "Flexible Regression and Smoothing: Using GAMLSS in R.

Familiarize with the **gamlss** functions and packages by repeating the commands given below.

The `gamlss()` function allows modelling of up to four parameters in a distribution family, which are conventionally called μ, σ, ν and τ. Here we give a simple demonstration using the `film90` data set.

R data file: `film90` in package **gamlss.data** of dimension 4015×4.
variables
- **`lnosc`** : the log of the number of screens in which the film was played
- **`lboopen`** : the log of box office opening week revenues
- **`lborev1`** : the log of box office revenues after the first week (the response variable which has been randomized)
- **`dist`** : a factor indicating whether the distributor of the film was an "Independent" or a "Major" distributor

purpose: to demonstrate the fitting of a simple regression model in the **gamlss** package.

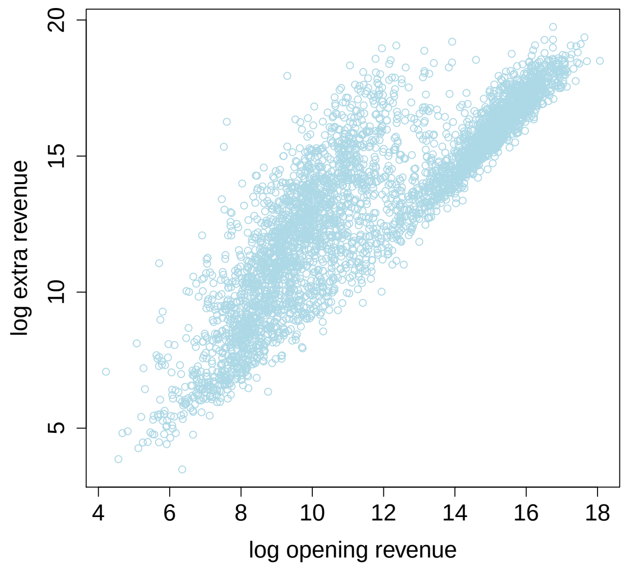
The original data were analysed in Voudouris et al. [2012], where more information about the data and the purpose of the original study can be found. Here for demonstrating some of the features of **gamlss** we analysed only two variables: `lborev1` as the response variable, and `lboopen` as an explanatory variable.

We start by plotting the data in Figure 1. Two key features are suggested: (i) the relationship between the response and the explanatory variable is nonlinear, and (ii) the shape of the response variable distribution changes for different levels of the explanatory variable. As we will see in Section 1.1.11, a GAMLSS model has the flexibility to model these features.

```r
library(gamlss)
data(film90)
plot(lborev1~lboopen, data=film90, col="lightblue",
```

Figure 1

```
     xlab="log opening revenue", ylab="log extra revenue")
```



R code on page 1

Figure 1: Scatterplot of the `film90` revenues

1.1.1 Fitting a parametric model

Below we fit a simple linear regression model with normal errors. It is clear from Figure 2 that the model does not fit well, especially for low values of `lboopen`.

```
m <- gamlss(lborev1~lboopen, data=film90, family=NO)

## GAMLSS-RS iteration 1: Global Deviance = 15079.74
## GAMLSS-RS iteration 2: Global Deviance = 15079.74
```

```
plot(lborev1~lboopen, data=film90, col = "lightblue")
lines(fitted(m)~film90$lboopen)
```

Figure 2

The problem seems to be the linear term in `lboopen`, so next we fit a cubic polynomial. One method of fitting polynomial curves in **R** is by using the function `I()`. A different method is by using the function `poly()` which fits orthogonal polynomials (see later).

```
m00 <- gamlss(lborev1~lboopen+I(lboopen^2)+I(lboopen^3), data=film90,
                       family=NO)

## GAMLSS-RS iteration 1: Global Deviance = 14518.26
## GAMLSS-RS iteration 2: Global Deviance = 14518.26

summary(m00)
```

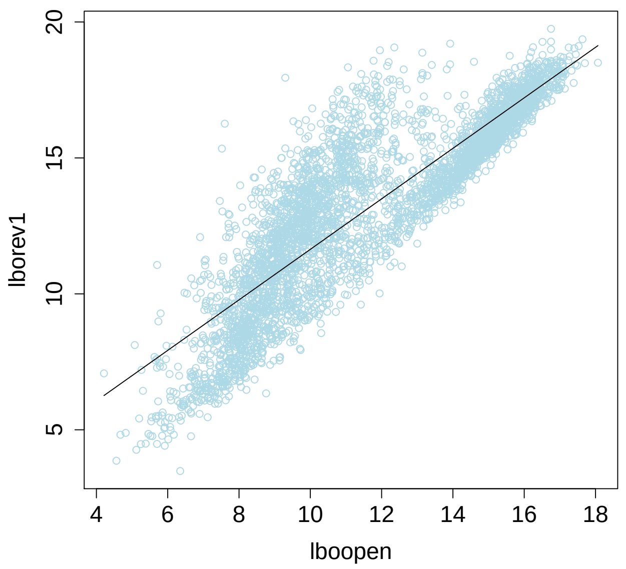


Figure 2: Scatterplot of the `film90` data with the fitted linear model for the mean.

R code on page 2

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:
## gamlss(formula = lborev1 ~ lboopen + I(lboopen^2) +
##     I(lboopen^3), family = NO, data = film90)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.232e+01  1.271e+00  -17.57   <2e-16 ***
## lboopen       7.147e+00  3.516e-01   20.32   <2e-16 ***
## I(lboopen^2) -4.966e-01  3.153e-02  -15.75   <2e-16 ***
## I(lboopen^3)  1.270e-02  9.142e-04   13.89   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.38189    0.01114   34.29   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -------------------------------------------------------------------
## No. of observations in the fit:  4031
## Degrees of Freedom for the fit:  5
##       Residual Deg. of Freedom:  4026
##                       at cycle:  2
##
## Global Deviance:     14518.26
##             AIC:     14528.26
##             SBC:     14559.77
## *******************************************************************
```

Note that for large data sets it could be more efficient (and may be essential) to calculate the polynomial terms in advance prior to using the `gamlss()` function, e.g.

```
x2<-xˆ2; x3<-xˆ3
```

and then use them within the `gamlss()` function, since the evaluation is then done only once:

```
film90 <- transform(film90, lb2=lboopenˆ2, lb3=lboopenˆ3)
m002 <- gamlss(lborev1~lboopen + lb2 + lb3, data=film90, family=NO)
```

The fitted model is displayed in Figure 3. Although the new model is an improvement, the polynomial line does not fit well for smaller values of `lboopen`. This behaviour, i.e. erratic fitting in the lower or upper end of the covariate, is very common in fitting parametric polynomial curves.

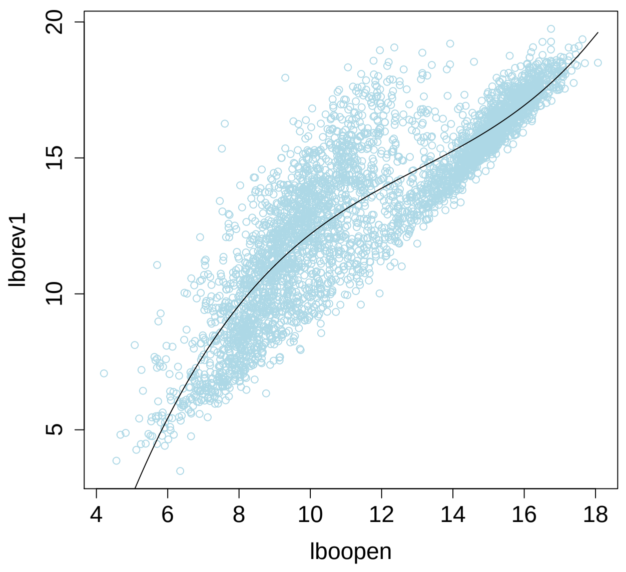
```
plot(lborev1~lboopen, col="lightblue", data=film90)
lines(fitted(m002)[order(film90$lboopen)]~
                   film90$lboopen[order(film90$lboopen)])
```

Using the notation $y =$ `lborev1` and $x =$ `lboopen`, the fitted model `m00` is given by

$$
\begin{aligned}
y &\sim \mathcal{N}(\hat{\mu}, \hat{\sigma}^2) \\
\text{where} & \\
\hat{\mu} &= \hat{\beta}_{10} + \hat{\beta}_{11}x + \hat{\beta}_{12}x^2 + \hat{\beta}_{13}x^3 \\
&= -22.320 + 7.147x - 0.497x^2 + 0.013x^3 \\
\log(\hat{\sigma}) &= 0.3819 \ ,
\end{aligned}
$$

giving $\hat{\sigma} = \exp(0.3819) = 1.465$.

The `summary()` function is useful for providing standard errors for the fitted coefficient parameters. The `summary()` function has two ways of producing standard errors: (i) `type="vcov"` (the default) and (ii) `type="qr"`. The way the standard errors are produced using the `vcov` method



R code on page 4

Figure 3: Scatterplot of the `film90` data with the fitted cubic model for the mean.

is described in detail in Section ??. It starts by defining the likelihood function at the maximum (using `gen.likelihood()`) and then obtaining the full (numerical) Hessian matrix of all the beta coefficient parameters in the model. Standard errors are obtained from the observed information matrix (the inverse of the Hessian matrix). The standard errors obtained this way are more reliable than those produced by the `qr` method, since they take into account the information about the interrelationship between the distribution parameters, i.e. μ and σ in the above example. On occasions when the above procedure fails, the standard errors are obtained from `type= "qr"`, which uses the individual fits of the distribution parameters and therefore should be used with caution. The `summary()` output gives a warning when this happens, as the standard errors produced this way do not take into the account the correlation between the estimates of the distribution parameters μ, σ, ν and τ. (In the example above the estimates of μ and σ of the normal distribution are asymptotically uncorrelated.)

Robust ("*sandwich*" or "*Huber sandwich*") standard errors can be obtained using the argument `robust=TRUE` of the `summary()` function. Robust standard errors were introduced by Huber [1967] and White [1980] and are, in general, more reliable than the usual standard errors when the variance model is suspected not to be correct (assuming the mean model is correct). The sandwich standard errors are usually (but not always) larger than the usual ones.

Next we demonstrate how `vcov()` can be used to obtain the variance-covariance matrix, the correlation matrix and the (usual and robust) standard errors of the estimated parameters:

```
# the variance-covariance matrix of the parameters
print(vcov(m00), digit=3)

##              (Intercept)   lboopen I(lboopen^2)
## (Intercept)     1.61e+00 -4.43e-01     3.90e-02
## lboopen        -4.43e-01  1.24e-01    -1.10e-02
## I(lboopen^2)    3.90e-02 -1.10e-02     9.94e-04
## I(lboopen^3)   -1.10e-03  3.15e-04    -2.87e-05
## (Intercept)     2.24e-11 -6.15e-12     5.40e-13
##                I(lboopen^3) (Intercept)
## (Intercept)     -1.10e-03    2.24e-11
## lboopen          3.15e-04   -6.15e-12
## I(lboopen^2)    -2.87e-05    5.40e-13
## I(lboopen^3)     8.36e-07   -1.53e-14
## (Intercept)     -1.53e-14    1.24e-04

# the correlation matrix
print(vcov(m00, type="cor"), digit=3)

##                (Intercept)   lboopen I(lboopen^2)
## (Intercept)     1.00e+00 -9.93e-01     9.74e-01
## lboopen        -9.93e-01  1.00e+00    -9.94e-01
## I(lboopen^2)    9.74e-01 -9.94e-01     1.00e+00
## I(lboopen^3)   -9.49e-01  9.79e-01    -9.95e-01
## (Intercept)     1.58e-09 -1.57e-09     1.54e-09
##                I(lboopen^3) (Intercept)
## (Intercept)     -9.49e-01    1.58e-09
## lboopen          9.79e-01   -1.57e-09
## I(lboopen^2)    -9.95e-01    1.54e-09
## I(lboopen^3)     1.00e+00   -1.50e-09
## (Intercept)     -1.50e-09    1.00e+00

# standard errors
print(vcov(m00, type="se"), digits=2)

##  (Intercept)      lboopen I(lboopen^2) I(lboopen^3)
##      1.27058      0.35164      0.03153      0.00091
##  (Intercept)
##      0.01114

print(vcov(m00, type="se", robust=TRUE), digits=2)

##  (Intercept)      lboopen I(lboopen^2) I(lboopen^3)
##       1.9702       0.5217       0.0446       0.0012
##  (Intercept)
##       0.0135
```

Note that in the final row and/or column of the above output, `Intercept` refers to the intercept of the predictor model for σ ($\hat{\beta}_{20}$), while the first row and/or column `Intercept` refers to the intercept of the predictor for μ ($\hat{\beta}_{10}$).

Now we fit the same model as in `m00`, but using orthogonal polynomials (see Section **??**) using function `poly()`, i.e. `poly(x,3)`:

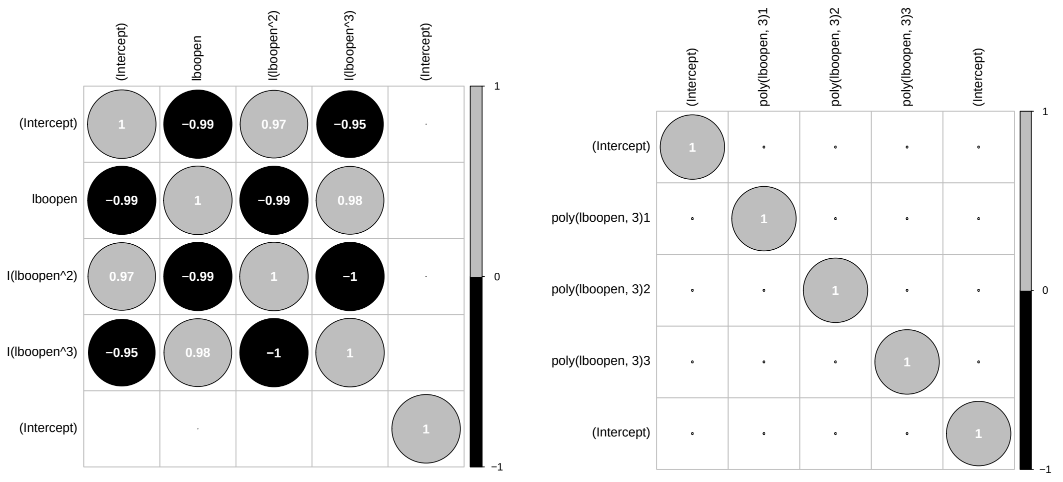
```
m0 <- gamlss(lborev1~poly(lboopen,3), data=film90, family=NO)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 14518.26
## GAMLSS-RS iteration 2: Global Deviance = 14518.26
```

It is of some interest to compare the correlations between the parameter estimates for the two fitted models `m00` and `m0`. Visual representation of the correlation coefficients can be obtained using the package **corrplot**.

```
library(corrplot)
col1 <- colorRampPalette(c("black","grey"))
corrplot(vcov(m00, type="cor"), col=col1(2), outline=TRUE,
         tl.col = "black", addCoef.col = "white")
corrplot(vcov(m0, type="cor"),   col=col1(2), outline=TRUE,
         tl.col = "black", addCoef.col = "white")
```

Figure 4



R code on page 7

Figure 4: Graphical displays of the correlation coefficient matrices for models `m00` (left) and `m0` (right)

Figure 4 shows the resulting graphical displays. Because, μ and σ in the normal distribution are information independent (i.e. asymptotically uncorrelated), the first four estimated parameters (μ model) are effectively not correlated with the fifth, the constant in the model for $\log(\sigma)$, in both models `m0` and `m00`. In addition all the parameters of the μ model for `m0` are uncorrelated because we used orthogonal polynomials, but for `m00` they are highly correlated.

1.1.2 Fitting a nonparametric smoothing model

In this section, we outline a few of the nonparametric smoothing functions implemented in GAMLSS. In particular, we discuss the `pb()` (P-splines), `cs()` (cubic splines), `lo()` (locally weighted regression) and `nn()` (neural networks) functions. For a comprehensive discussion (and list of smoothing functions within GAMLSS), see Chapter **??**.

1.1.3 P-splines

Model `m0` is a linear parametric GAMLSS model, which we have seen does not fit particularly well. Another approach is to fit a smooth term to the covariate `lboopen`. Eilers and Marx [1996] introduced nonparametric penalized smoothing splines (P-splines), which are described in Section ??. In order to fit the mean of `lborevl` with a P-spline for `lboopen`, use:

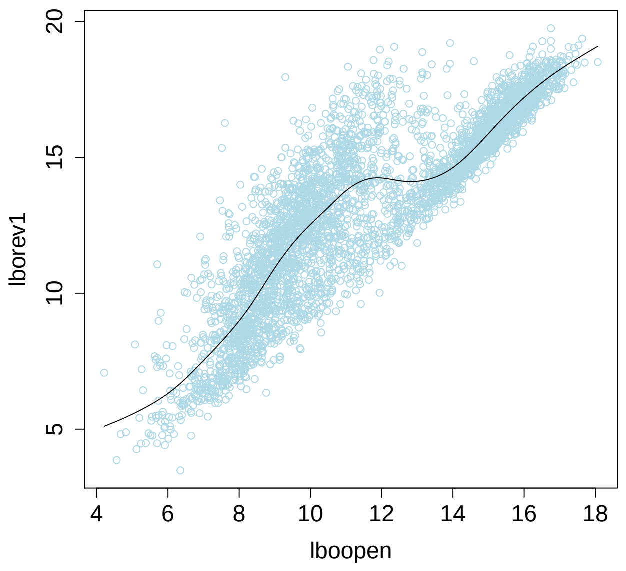
```
m1<-gamlss(lborev1~pb(lboopen), data=film90, family=NO)

## GAMLSS-RS iteration 1: Global Deviance = 14109.58
## GAMLSS-RS iteration 2: Global Deviance = 14109.58

summary(m1)

## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:
## gamlss(formula = lborev1 ~ pb(lboopen), family = NO,
##     data = film90)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.347147   0.087053   26.96   <2e-16 ***
## pb(lboopen) 0.928889   0.007149  129.93   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.33120    0.01114   29.74   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms maybe are not accurate.
## ------------------------------------------------------------------
## No. of observations in the fit:  4031
## Degrees of Freedom for the fit:  12.73672
##       Residual Deg. of Freedom:  4018.263
##                          at cycle:  2
##
## Global Deviance:     14109.58
##             AIC:     14135.05
##             SBC:     14215.32
## ******************************************************************
```

In the smoothing function `pb()` the smoothing parameter (and therefore the effective degrees of freedom) are estimated automatically using the default local maximum likelihood method described in Rigby and Stasinopoulos [2013]. Within the `pb()` function there are also alternative ways of estimating the smoothing parameter, such as the local generalized AIC (GAIC), and the local Generalized Cross Validation (GCV). See Section **??** for details.

The fitted model is displayed in Figure 5:

```
plot(lborev1~lboopen, col="lightblue", data=film90)
lines(fitted(m1)[order(film90$lboopen)]~
                  film90$lboopen[order(film90$lboopen)])
```

Figure 5



R code on page 9

Figure 5: P-splines fit: the `film90` data with the fitted smooth mean function fitted using `pb()`.

The effective degrees of freedom fitted by the `pb()` can be obtained using `edf()`:

```
edf(m1, "mu")

## Effective df for mu model
## pb(lboopen)
##    11.73672
```

One of the important things to remember when fitting a smooth nonparametric term in `gamlss()` is that the displayed coefficient of the smoothing term and its standard error (s.e.) refer only to the linear component of the term. For example the coefficient 0.9289 and its s.e. 0.0071 in the above output should be interpreted with care. They are an artefact of the way the fitting algorithm works with the `pb()` function. This is because the linear part of the smoothing is fitted together with all other linear terms (in the above case only the intercept). One should try to interpret the whole smoothing function, which can be obtained using `term.plot()`. The effect that the smoothing function has on the specific parameters can also be checked using the function `getPEF()`, which calculates the partial effect of a continuous variable given the rest of the explanatory variables are fixed at specified values. The same function can be used to obtain the first and second derivatives for the partial effects. Significance of smoothing terms is obtained using the function `drop1()`, but this may be slow for a large data set with many fitted smoothing terms.

Important: Do not try to interpret the linear coefficients or the standard errors of the smoothing terms.

Note also that when smoothing additive terms are involved in the fitting, both methods (default and robust) used in `summary` to obtained standard errors are questionable. The reason is that the way `vcov()` is implemented effectively assumes that the estimated smoothing terms were fixed at their estimated values. The functions `prof.dev()` and `prof.term()` can be used for obtaining more reliable individual parameter confidence intervals, by fixing the smoothing degrees of freedom at their previously selected values.

1.1.4 Cubic Splines

Other smoothers are also available. For details on cubic smoothing splines see Section **??**. In order to fit a nonparametric smoothing cubic spline with 10 effective degrees of freedom in addition to the constant and linear terms, use

```
m2<-gamlss(lborev1~cs(lboopen,df=10), data=film90, family=NO)
```

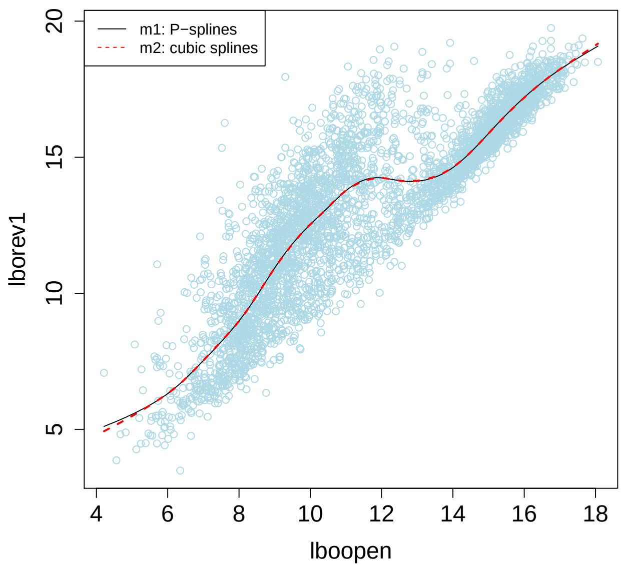
```
## GAMLSS-RS iteration 1: Global Deviance = 14107.72
## . . .
## GAMLSS-RS iteration 2: Global Deviance = 14107.72
```

The effective degrees of freedom used in the fitting of μ in the above model are 12 (one for the constant, one for the linear and 10 for smoothing). Note that the `gamlss()` notation is different from the `gam()` notation in `S-PLUS` where the equivalent model is fitted using `s(x,11)`.

The total degrees of freedom used for model `m2` is 13, i.e. 12 for μ and 1 for σ. The fitted values of μ for models `m1` and `m2` are displayed in Figure 6:

```
plot(lborev1~lboopen, col="lightblue", data=film90)
lines(fitted(m1)[order(film90$lboopen)]~
                 film90$lboopen[order(film90$lboopen)])
lines(fitted(m2)[order(film90$lboopen)]~
                 film90$lboopen[order(film90$lboopen)],
col="red", lty=2,  lwd=2)
```

Figure 6



R code on page 10

Figure 6: P-splines and cubic splines fits: plot of the `film90` data together with the fitted smooth mean functions of model `m1` fitted by `pb()` (continuous line) and model `m2` fitted by `cs()` (dashed line).

```r
legend("topleft",legend=c("m1: P-splines","m2: cubic splines"),
       lty=1:2,col=c("black","red"),cex=1)
```

1.1.5 loess

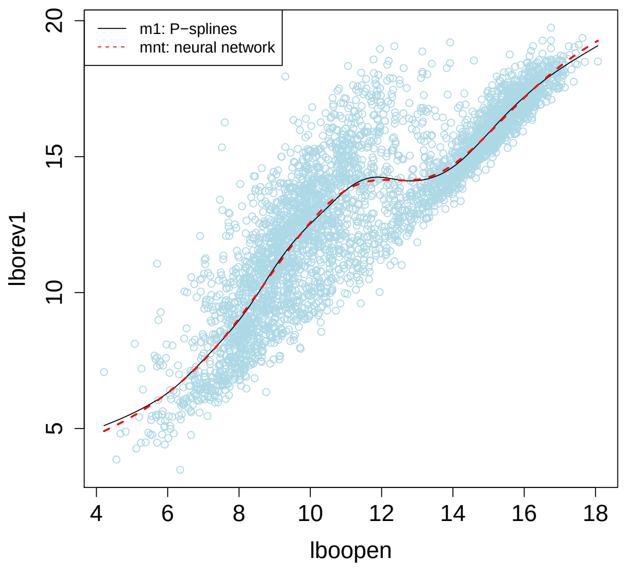
Locally weighted scatterplot smoothing [Cleveland and Devlin, 1988], or loess, is described in Section ??. Loess curves are implemented as

```r
m4 <- gamlss(lborev1~lo(~lboopen,span=.4), data=film90, family=NO)
```

1.1.6 Neural Networks

Neural networks can be considered as another type of smoother. For details see Section ??. Here a neural network smoother is fitted using an interface of **gamlss** with the **nnet** package [Venables and Ripley, 2002]. The additive function to be used with `gamlss()` is `nn()`, which is part of the package **gamlss.add**. The following example illustrates its use.

```r
library(gamlss.add)
mnt <- gamlss(lborev1~nn(~lboopen,size=20,decay=0.1), data=film90,
              family=NO)
```



R code on page 12

Figure 7: Neural network fit: a plot of the `film90` data together with the fitted smooth mean functions of model `m1` fitted by `pb()` (black continuous line) and the neural network model `mnt` fitted by `nn()` (red dashed line).

```
## GAMLSS-RS iteration 1: Global Deviance = 14186.98
## . . .
## GAMLSS-RS iteration 4: Global Deviance = 14125.05
```

This fits a neural network model with one covariate and 20 hidden variables. The `decay` argument is used for penalizing the fitted coefficients. The fitted values of models `mnt` and `m1` are displayed in Figure 7.

```
plot(lborev1~lboopen, col="lightblue", data=film90)
lines(fitted(m1)[order(film90$lboopen)]~
                 film90$lboopen[order(film90$lboopen)])
lines(fitted(mnt)[order(film90$lboopen)]~
                  film90$lboopen[order(film90$lboopen)],
      col="red", lty=2,  lwd=2)
legend("topleft",legend=c("m1: P-splines","mnt: neural network"),
       lty=1:2,col=c("black","red"),cex=1)
```

Figure 7

The function `getSmo()` is used to get more information about the fitted neural network model. This function retrieves the last fitted object within the backfitting GAMLSS algorithm (in this case a `"nnet"` object). Reserved methods such as `print()`, `summary()` or `coef()` can be used to get information for the objects. Here we retrieve its 61 coefficients. (There are 40 parameters from the relationship between the 20 hidden variables and the explanatory variable (constant and slope parameters), together with 21 parameters from the relationship between the response variable and the 20 hidden variables (constant and 20 slope parameters).)

```
coef(getSmo(mnt))
```

```
##        b->h1       i1->h1        b->h2       i1->h2        b->h3
##   0.71711189 -0.13290196   6.78268584 -0.76164048   3.08247814
## . . .
```

1.1.7 Extracting fitted values

Fitted values of the distribution parameters of a GAMLSS model (for all cases) can be obtained using the `fitted()` function. For example

```
plot(lboopen, fitted(m1,"mu"))
```

will plot the fitted values of μ distribution parameter against x (`lboopen`). The constant estimated scale parameter (the standard deviation of the normal distribution in this case) can be obtained:

```
fitted(m1,"sigma")[1]

##        1
## 1.392632
```

where `[1]` indicates the first element of the vector. The same value can be obtained using the more general function `predict()`:

```
predict(m1,what="sigma", type="response")[1]

##        1
## 1.392632
```

The function `predict()` can also be used to predict the response variable distribution parameters for both old and new data values of the explanatory variables. This is explained in Section **??**.

One of the flexibilities offered by GAMLSS is the modelling of all the distribution parameters (rather than just μ). This means that the scale and shape of the distribution can vary as a (linear or smooth) function of explanatory variables. Below, we show how to model both μ and σ of a normal response distribution. Figure 1 suggests that this flexibility of a GAMLSS model might be required.

1.1.8 Modelling both μ and σ

To model the predictors of both the mean μ and the scale parameter σ as nonparametric smoothing P-spline functions of `lboopen` (with a normal response distribution) use:

```
m3 <- gamlss(lborev1~pb(lboopen),sigma.formula=~pb(lboopen),
             data=film90, family=NO)
edfAll(m3)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 12263.21
## . . .
## GAMLSS-RS iteration 4: Global Deviance = 12263.54
## $mu
## pb(lboopen)
##     12.1442
##
## $sigma
## pb(lboopen)
##    10.67769
```

The function `edfAll()` is used to obtain the effective degrees of freedom for all parameters. These are 12.14 and 10.68 for μ and σ respectively. The fitted model for μ is displayed in Figure 8.

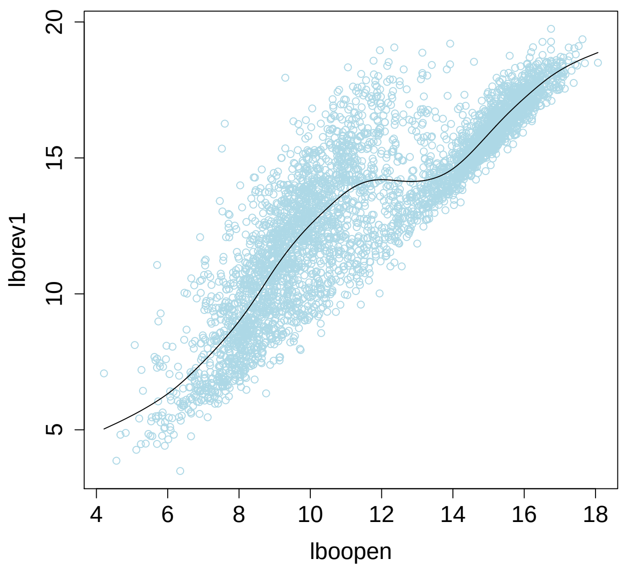
```
plot(lborev1~lboopen, col="lightblue", data=film90)
lines(fitted(m3)[order(film90$lboopen)]~
                  film90$lboopen[order(film90$lboopen)])
```

Figure 8

1.1.9 Diagnostic plots

Once a GAMLSS model is fitted, it is important to assess the adequacy of the fitted model by examining the model residuals. See Chapter **??** for more details. The function `resid()` (or `residuals()`) can be used to obtain the fitted (normalized randomized quantile) residuals of a model, referred to as residuals throughout this book. See Dunn and Smyth [1996] and Chapter **??** for more details. Residual plots are graphed using `plot()`:

Figure 9



R code on page 14

Figure 8: The `film90` data with the fitted smooth mean function of model `m3`, in which both the mean and variance models are fitted using `pb(lboopen)`.

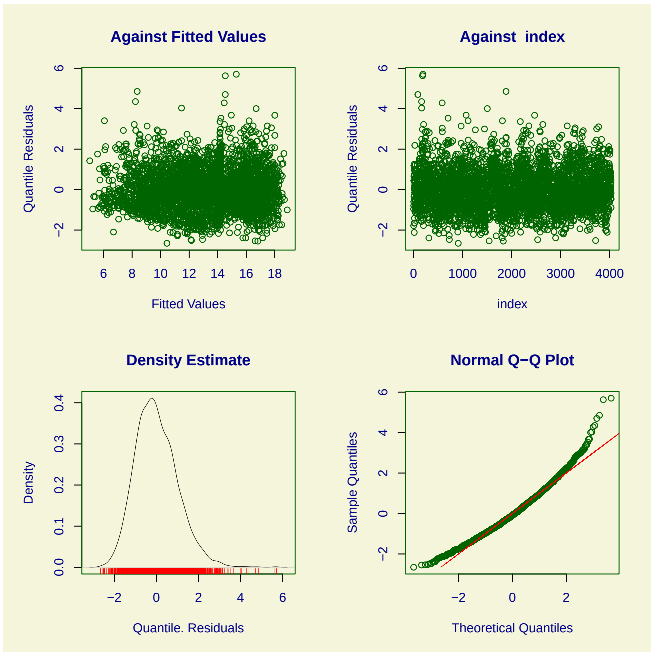
```
plot(m3)

## ******************************************************************
##       Summary of the Quantile Residuals
##                            mean   =  0.0006979142
##                        variance   =  1.000248
##                coef. of skewness   =  0.5907226
##                coef. of kurtosis   =  3.940587
## Filliben correlation coefficient   =  0.9909749
## ******************************************************************
```

Figure 9 shows plots of the residuals: (top left) against the fitted values of μ; (top right) against an index (i.e. case number); (bottom left) a nonparametric kernel density estimate; (bottom right) a normal Q-Q plot. Note that the `plot()` function does not produce additive term plots (as it does, for example, in the `gam()` function of **mgcv**). The function which does this in the **gamlss** package is `term.plot()`.

The worm plot (see Section **??**) is a de-trended normal Q-Q plot of the residuals. Model inadequacy is indicated when many points plotted lie outside the (dotted) point-wise 95% confidence bands. The worm plot is obtained using `wp()`:

Figure 10

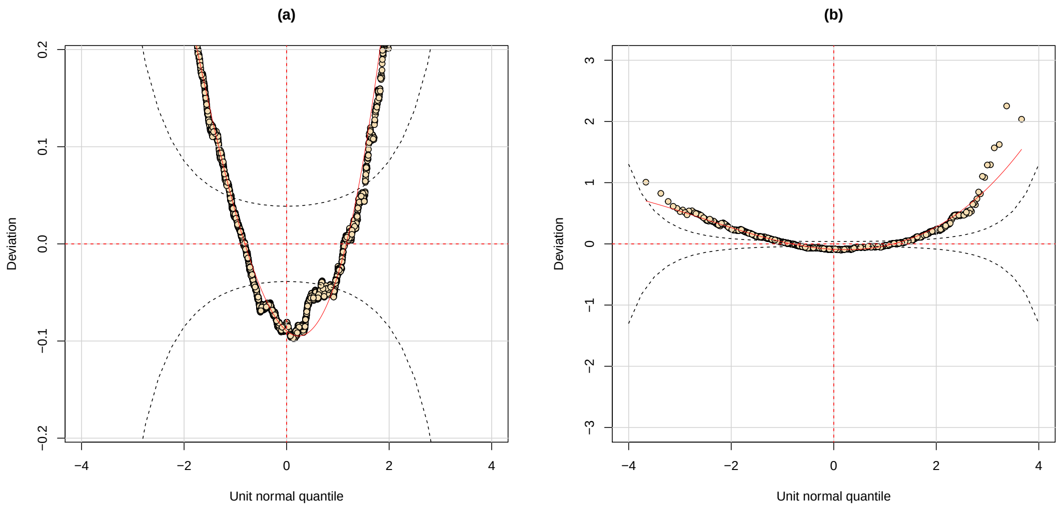
```
wp(m3)

## Warning in wp(m3): Some points are missed out
## increase the y limits using ylim.all
```



R code on page 15

Figure 9: Residual plots from the fitted normal model `m3`, using `pb(lboopen)` for both μ and $\log(\sigma)$.

```
title("(a)")
```



R code on page 16

Figure 10: Worm plots from model `m3`.

To include all points in the worm plot, change the "Deviation" axis range by increasing the value of `ylim.all` until all points are included in the plot (avoiding a warning message):

```
wp(m3, ylim.all=3)
title("(b)")
```

Figure 10

Since there is no warning message, all points have been included in the worm plot. Model inadequacy is indicated by the fact that many points lie outside the 95% confidence bands.

1.1.10 Fitting different distributions

One of the most important modelling decisions for a GAMLSS model is the choice of the distribution for the response variable. See Chapter **??** for a discussion of available distributions in GAMLSS. To use a distribution other than the normal (the default), use the `family` option of `gamlss()`. For example, to fit the Box-Cox-Cole-Green (`BCCG`), a three-parameter continuous distribution, use:

```
m5 <-gamlss(lborev1~pb(lboopen), sigma.formula=~pb(lboopen),
            nu.formula=~pb(lboopen), data=film90, family=BCCG)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 11888.56
## . . .
## GAMLSS-RS iteration 5: Global Deviance = 11809.64
```

To fit the Box-Cox power exponential (`BCPE`) distribution, a four-parameter continuous distribution:

```
m6 <-gamlss(lborev1~pb(lboopen), sigma.formula=~pb(lboopen),
            nu.formula=~pb(lboopen), tau.formula=~pb(lboopen),
            data=film90, start.from=m5, family=BCPE)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 11738.54
## . . .
## GAMLSS-RS iteration 20: Global Deviance = 11733.63
```

Note that we have used the argument `start.from=m5` to start the iterations from the previous fitted `m5` model. The details of all the distributions currently available in `gamlss()` are given in Rigby et al. [in press].

1.1.11 Selection between models

Once different models in GAMLSS have been fitted (either by using different distributions and/or smoothing terms), models may be selected by using, for example, an information criterion. See Chapter **??** for model selection techniques in GAMLSS.

For example, different models can be compared by a test based on their global deviances: GDEV $= -2\hat{\ell}$ (if they are nested), or by selecting the model with lowest generalized Akaike information criterion: GAIC $= -2\hat{\ell} + \kappa \cdot \text{df}$, where $\hat{\ell}$ is the fitted log-likelihood function and κ is a required penalty, e.g. $\kappa = 2$ for the AIC, $\kappa = \log n$ for the SBC, or $\kappa = 3.84$ (corresponding to a Chi-squared test with one degree of freedom for a single parameter). The function `deviance()` provides the global deviance of the model.

Note that the `gamlss()` global deviance is different from the deviance provided by `glm()` and `gam()`, see Section **??**. The global deviance is *exactly* minus twice the fitted log-likelihood function, *including* all constant terms in the log-likelihood. The `glm()` deviance is calculated as a deviation from the saturated model. It does not include 'constant' terms (which do not depend on the mean of distribution but do depend on the scale parameter) in the fitted log-likelihood, and so cannot be used to compare different distributions. The functions `AIC()` or `GAIC()` (which are identical) are used to obtain the generalized Akaike information criterion. For example to compare the models `m0` to `m6`:

```
GAIC(m0,m1,m2,m3,m4,m5,m6)
```

```
##          df      AIC
## m6 44.97879 11823.59
## m5 36.06436 11881.77
## m3 22.82189 12309.19
## m2 12.99817 14133.72
## m1 12.73672 14135.05
## m4 10.08556 14139.34
## m0  5.00000 14528.26
```

`GAIC()` uses default penalty $\kappa = 2$, resulting in the AIC. Hence according to the AIC model `m6` is selected as best (smallest value of AIC). To change the penalty in `GAIC()` use the argument `k`:

```
GAIC(m0,m1,m2,m3,m4,m5,m6, k=log(4031))
```

```
##          df      AIC
## m6 44.97879 12107.03
## m5 36.06436 12109.04
## m3 22.82189 12453.00
## m4 10.08556 14202.89
## m1 12.73672 14215.32
## m2 12.99817 14215.63
## m0  5.00000 14559.77
```

In this case with GAIC ($\kappa = \log n$) we have the SBC. Models selected using SBC are generally simpler than those selected using AIC. This is the case here, where model `m5` is selected.

Other model selection criteria based on training, validation and test samples are discussed on Chapter **??**.

Chosen Model

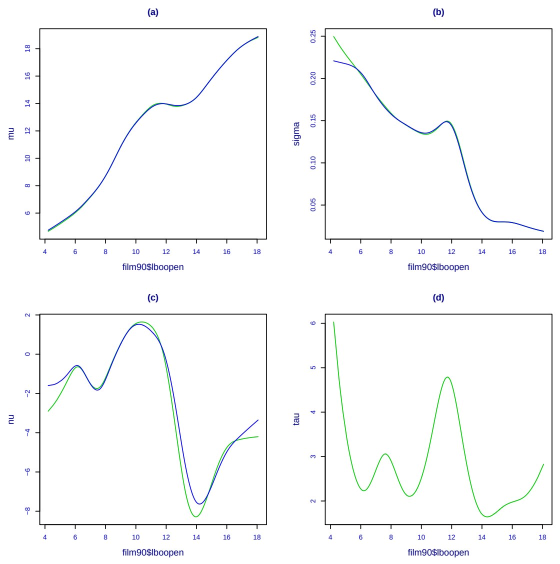
Using the AIC, model `m6` is selected with $Y =$ `lborev` $\sim$ BCPE(μ, σ, ν, τ) where each of μ, σ, ν and τ are modelled as smooth functions of $x =$ `lboopen`. The fitted smooth functions for both `m5` and `m6` models are shown in Figure 11.

Figure 11

```
fittedPlot(m5, m6, x=film90$lboopen, line.type = TRUE)
```

Since, in this example, only one explanatory variable is used in the fit, centile estimates for the fitted distribution can be shown using the functions `centiles()` or `centiles.fan()`.

Figure 12



R code on page 18

Figure 11: A plot of the smooth fitted values for all the parameters (a) μ, (b) σ, (c) ν and (d) τ from models m5 (dashed line) and m6 (continuous line). The distribution for model m5, BCCG, has only three parameters so does not appear in panel (d).

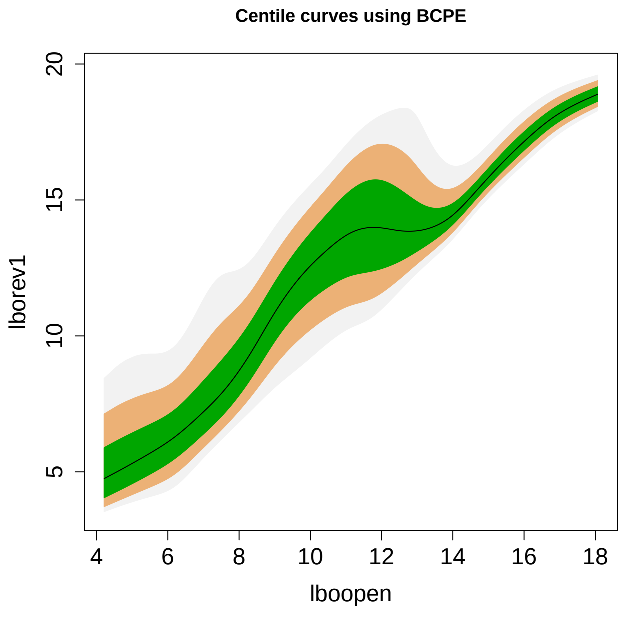
```
centiles.fan(m6,  xvar=film90$lboopen,  cent=c(3,10,25,50,75,90,97),
colors="terrain",ylab="lborev1", xlab="lboopen")
```

Figure 12 shows centile curves for lborev1 against lboopen from the fitted model m6. For example the lowest curve is the fitted 3% centile curve, defined by 3% of the values of lborev1 lying below the curve for each value of lboopen, for the fitted model m6 if it was the correct model. For more details on centile curves see Chapter ??. Figure 13 also shows how the fitted conditional distribution for the response variable lborev1 changes according to variable lboopen. The function plotSimpleGamlss() from the package **gamlss.util** is used here.

Figure 13

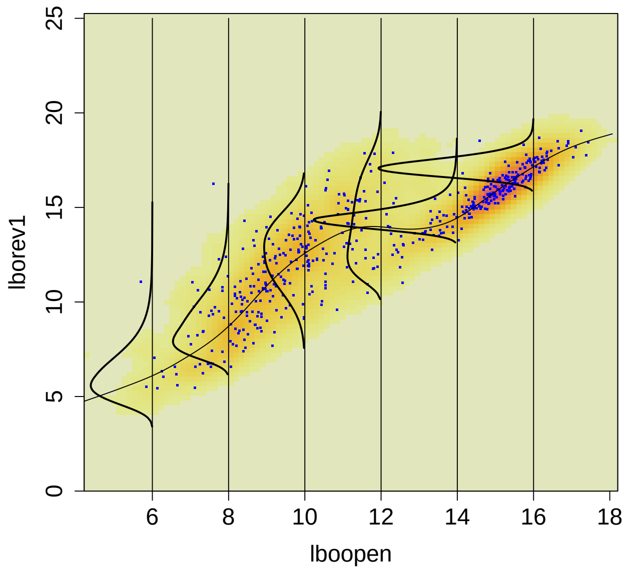
```
library(gamlss.util)
library(colorspace)
plotSimpleGamlss(lborev1,lboopen, model=m6,   data=film90,
                 x.val=seq(6,16,2), val=5, N=1000, ylim=c(0,25),
                 cols=heat_hcl(100))

## new prediction
## new prediction
## new prediction
## new prediction
```

Figure 13 highlights how the fitted conditional distribution of lborev1 changes with lboopen. This is the essence of GAMLSS modelling.



R code on page 19

Figure 12: Centile fan plot for the `m6` model showing the 3%, 10%, 25%, 50%, 75%, 90% and 97% centiles for the fitted BCPE distribution.



R code on page 19

Figure 13: Fitted conditional distribution of the response variable `lborev1`, showing how it changes for different values of the covariate `lboopen`.

Important: Within GAMLSS, the shape of the conditional distribution of the response variable can vary according to the values of the explanatory variables.

1.2 Practical 2: The `abdom` data

Information on the abdominal data is given on page **??**. Fit different response distributions and choose the 'best' model according to the GAIC criterion:

1. Load the `abdom` data and print the variable names.
2. Fit the normal distribution model, using `pb()` to fit P-spline smoothers for the predictors for μ and σ with automatic selection of smoothing parameters:
   ```
   mNO<- gamlss(y~pb(x), sigma.fo=~pb(x), data=abdom, family=NO)
   ```
3. Try fitting alternative distributions:
   (a) two-parameter distributions: `GA`, `IG`, `GU`, `RG`, `LO`,
   (b) three-parameter distributions: `PE`, `TF`, `BCCG`,
   (c) four-parameter distributions: `BCT`, `BCPE`.

   Apply `pb()` to all parameters of each distribution. Make sure to use different model names.
4. Compare the fitted models using GAIC with each of the penalties `k=2`, `k=3` and `k=log(length(abdom$y))`, e.g.
   ```
   GAIC(mNO,mGA,mIG,mGU,mRG,mLO,mPE,mTF,mBCCG,mBCT,mBCPE,k=2)
   ```
5. Check the residuals for your chosen model, say `m`, by `plot(m)` and `wp(m)`.
6. For a chosen model, say `m`, look at the total effective degrees of freedom `edfAll(m)`, plot the fitted parameters, `fittedPlot(m,x=abdom,$x)`, and plot the data by `plot(y∼x,data=abdom)`, and fitted μ against `x`, `lines(fitted(m)∼x, data=abdom)`.
7. For a chosen model, examine the centile curves using `centiles(m,abdom$x)`.

2 Day 1 Afternoon

2.1 Practical 3: Use the gamlss.demo package to plot distributions.

Use the **gamlss.demo** package to plot distributions.

```
library(gamlss.demo)
gamlss.demo()
```

Investigate how the following distributions change with their parameters:

1. Continuous distributions
   (a) Power exponential distribution (`PE`) for $-\infty < y < \infty$

    (b) Gamma distribution (`GA`) for $0 < y < \infty$
    (c) Beta distribution (`BE`) for $0 < y < 1$
2. Discrete distributions
    (a) Negative binomial type I (`NBI`) for $y = 0, 1, 2, 3, \ldots$
    (b) Beta binomial (`BB`) for $y = 0, 1, 2, 3, ..., n$
3. Mixed distributions
    (a) Zero adjusted gamma (`ZAGA`) for $0 \leq y < \infty$
    (b) Beta inflated (`BEINF`) for $0 \leq y \leq 1$

2.2 Practical 4: plotting different distributions

The **gamlss.dist** package (which is downloaded automatically with **gamlss**) contains many distributions. Typing

```
?gamlss.family
```

will show all the available distributions in the **gamlss** packages. You can also explore the shape and other properties of the distributions. For example the following code will produce the pdf, cdf, inverse cdf and a histogram of a random sample generated from a gamma distribution:

```
PPP <- par(mfrow=c(2,2))
plot(function(y) dGA(y, mu=10 ,sigma=0.3),0.1, 25) # pdf
plot(function(y) pGA(y, mu=10 ,sigma=0.3), 0.1, 25) #cdf
plot(function(y) qGA(y, mu=10 ,sigma=0.3), 0, 1) # inverse cdf
hist(rGA(100,mu=10,sigma=.3)) # randomly generated values
par(PPP)
```

Note that the first three plots above can also be produced by using the function `curve()`, for example

```
curve(dGA(x=x, mu=10, sigma=.3),0, 25)
```

To explore discrete distributions use:

```
PPP <- par(mfrow=c(2,2))
plot(function(y) dNBI(y, mu = 10, sigma =0.5 ), from=0, to=40,
     n=40+1, type="h", main="pdf", ylab="pdf(x)")
cdf <- stepfun(0:39, c(0, pNBI(0:39, mu=10, sigma=0.5 )), f = 0)
plot(cdf,main="cdf", ylab="cdf(x)", do.points=FALSE )
invcdf <-stepfun(seq(0.01,.99,length=39), qNBI(seq(0.01,.99,
                 length=40), mu=10, sigma=0.5 ), f = 0)
plot(invcdf,main="inverse cdf",ylab="inv-cdf(x)",do.points=FALSE)
tN <- table(Ni <- rNBI(1000,mu=5, sigma=0.5))
r <- barplot(tN, col='lightblue')
par(PPP)
```

Note that to find moments or to check if a distribution integrates or sums to one, the functions `integrate()` or `sum()` can be used. For example

```
integrate(function(y) dGA(y, mu=10, sigma=.1),0, Inf)
```

will check that the distribution integrates to one, and

```
integrate(function(y) y*dGA(y, mu=10, sigma=.1),0, Inf)
```

will give the mean of the distribution.

The pdf of a GAMLSS family distribution can also be plotted using the **gamlss** function `pdf.plot()`. For example

```
pdf.plot(family=GA, mu=10, sigma=c(.1,.5,1,2), min=0.01,max=20,
         step=.5)
```

will plot the pdf's of four gamma distributions GA(μ, σ), all with $\mu = 10$, but with $\sigma = 0.1, 0.5, 1$ and 2, respectively.

Try plotting other continuous distributions, e.g. `IG` (inverse Gaussian), `PE` (power exponential) and `BCT` (Box-Cox t); and discrete distributions, e.g. `NBI` (negative binomial type I) and `PIG` (Poisson inverse Gaussian). Make sure you define the values of all the parameters of the distribution.

2.3 Practical 5: Turkish stock exchange

Turkish stock exchange: the `tse` data. The data are for the eleven-year period 1 January 1988 to 31 December 1998. Continuously compounded returns in domestic currency were calculated as the first difference of the natural logarithm of the series. The objective is to fit a distribution to the Turkish stock exchange index.

R data file: `tse` in package **gamlss.data** of dimensions 2868×6.

variables

- **`year`**
- **`month`**
- **`day`**
- **`ret`** : day returns `ret[t]=ln(currency[t])-ln(currency[t-1])`
- **`currency`** : the currency exchange rate
- **`tl`** : day return `ret[t]=log10(currency[t])-log10(currency[t-1])`

purpose: to show the `gamlss` family of distributions.

1. Input the data and plot the returns sequentially using

   ```
   with(tse, plot(ret,type="l"))
   ```

2. Fit continuous distributions on $(-\infty < y < \infty)$ to `ret`. Automatically choose the best fitting distribution according to AIC. Show the AIC for the different fitted distributions.

Do any of the fits fail?

```
mbest<-fitDist(tse$ret,type="realline",k=2)
mbest
mbest$fits
mbest$fails
```

Repeat with `k=3.84` and `k=log(length(tse$ret))` (corresponding to criteria $\chi^2_{1,0.05}$ and SBC respectively).

3. For the chosen distribution, plot the fitted distribution using `histDist()`. Refit the model using `gamlss()` in order to output the parameter estimates using `summary()`.

4. An alternative approach is to manually fit each of the following distributions for `ret` using `histDist()` (and using different model names for later comparison):

   (a) two-parameter: normal `NO`(μ, σ),

   ```
   mNO<-histDist(tse$ret,"NO",nbins=30, n.cyc=100)
   ```

   (b) three-parameter: t family `TF`(μ, σ, ν) and power exponential `PE`(μ, σ, ν)

   (c) four-parameter: Johnson Su `JSU`(μ, σ, ν, τ), skew exponential power type 1 to 4, e.g. `SEP1`(μ, σ, ν, τ), skew t type 1 to 5, e.g. `ST1`(μ, σ, ν, τ) and sinh arc-sinh `SHASH`(μ, σ, ν, τ).

   (Note that `histDist()` has as default `nbins=30`, to provide a detailed histogram.)

5. Use `GAIC()` with each of the penalties $\kappa = 2, 3.84$ and $7.96 = \log(2868)$ (corresponding to criteria AIC, $\chi^2_{1,0.05}$ and SBC respectively), in order to select a distribution model. Output the parameter estimates for your chosen model using the function `summary()`.

2.4 Practical 6: The stylometric data

R data file: `stylo` in package **gamlss.data** of dimensions 64×2

variables

`word` : number of times a word appears in a single text

`freq` : frequency of the number of times a word appears in a text

purpose: to demonstrate the fitting of a truncated discrete distribution.

Note that the response variable `word` is (left) truncated at 0.

1. Load the data and plot them.

2. Create different truncated at zero count data distributions (`PO, NBII, DEL, SICHEL`), for example:

   ```
   gen.trun(par = 0, family = PO, type = "left")
   ```

3. Fit the different truncated distributions, for example:

   ```
     mPO <- gamlss(word ~ 1, weights = freq, data = stylo,
                 family = POtr, trace = FALSE)
   ```

4. Compare the distributions using GAIC.
5. Check the residuals of the chosen model using `plot()` and `wp()`.
6. Plot the fitted distributions using `histDist`.

3 Day 2 Morning

3.1 Practical 7: Victims of crime

The `VictimsOfCrime` data were introduced on page **??**.

R data file: `VictimsOfCrime` in package **gamlss.data** of dimensions 10590×2

variables

> **`reported`** : whether the crime was reported in local media ($0 =$no, $1 =$yes)
>
> **`age`** : age of the victim

purpose: to demonstrate binary data smoothing.

1. Load the data and plot `reported` against `age`.

```
data(VictimsOfCrime)
plot(reported~age, data=VictimsOfCrime,  pch="|")
```

2. Now use the different smoothers investigated in this chapter to fit smooth curves for `age`. Note that the response is binary and therefore the binomial distribution (`BI`) is used in the `family` argument. For example:

```
# P-splines
m1<- gamlss(reported~pb(age), data=VictimsOfCrime, family=BI)
```

   The smoothers include `pb`, `pbm`, `cy`, `scs`, `lo`, `nn` and `tr`.

3. Compare the results using AIC and SBC.
4. Plot the different fitted μ (probability of a crime being reported in local media) for comparison. First study the behaviour of the P-spline based curves, i.e. `pb()`, `pbm()` and `cy()`, e.g.

```
plot(reported~age, data=VictimsOfCrime,  type="n")
with(VictimsOfCrime, lines(fitted(m1)[order(age)]~
                           age[order(age)],col="red", lwd=2))
```

5. Compare the fitted curves of the P-splines and cubic splines.
6. Compare the fitted curves of the P-splines and the neural network.
7. Compare the P-splines with the decision trees fitted curves.
8. Check the residuals of model `m1`. Note that for binary responses, the function `rqres.plot()` returns multiple realizations of the residuals.

```
rqres.plot(m1, ylin.all=.6)
```

9. Obtain a multiple worm plot of the residuals.

```
wp(m1, xvar=age, n.inter=9)
```

3.2 Practical 8: The Film data analysis

The film revenue data from the 1990s were analysed in Chapter 2. The data are an anonymized and randomized version of the data used by Voudouris et al. [2012] and are used here for demonstrating some of the features of GAMLSS, and in particular for exploring smooth interactions of explanatory variables. Information about the data can be found in Section **??**.

3.2.1 Preliminary analysis

Here we demonstrate how the data can be plotted in two- and three-dimensional plots. In Figure 14 we plot the response variable against (a) the log of the number of screens and (b) the log of box office opening revenues. The major and independent distributors are represented with different symbols.

Figure 14

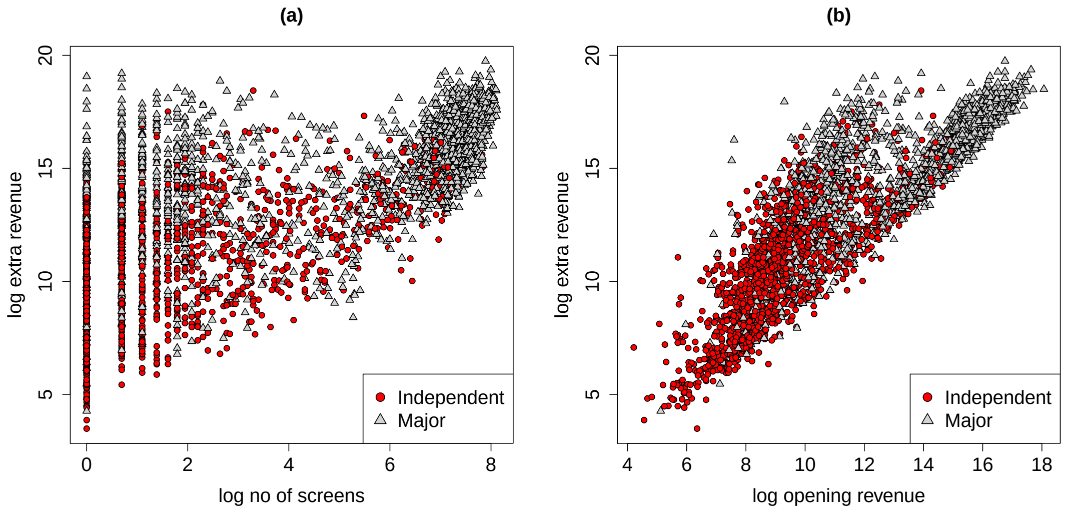
```
data(film90)
names(film90)

## [1] "lnosc"   "lboopen" "lborev1" "dist"

with(film90, plot(lnosc,lborev1,pch=c(21,24)[unclass(dist)],
  bg=c("red","lightgray")[unclass(dist)],
  xlab="log no of screens", ylab="log extra revenue", main="(a)"))
legend("bottomright",legend=c("Independent","Major"),pch=c(21,24),
  pt.bg=c("red","lightgray"),cex=1.5)
with(film90,  plot(lboopen,lborev1,pch=c(21,24)[unclass(dist)],
  bg=c("red","lightgray")[unclass(dist)],
  xlab="log opening revenue", ylab="log extra revenue", main="(b)"))
legend("bottomright",legend=c("Independent","Major"),pch=c(21,24),
  pt.bg=c("red","lightgray"),cex=1.5)
```

A good way of inspecting the data in three dimensions is with the package **rgl**. The following commands show how this can be done. The user may increase the size (by clicking and expanding the border), and rotate the figure:

```
library(rgl)
with(film90, plot3d(lboopen, lnosc, lborev1,
      col=c("red","green3")[unclass(dist)]))
```

To show a linear least squares fit to the data, the **rpanel** package may be used:

```
library(rpanel)
with(film90, rp.regression(cbind(lboopen, lnosc), lborev1))
```



R code on page 26

Figure 14: Showing (a) `lborev1` against `lnosc` (b) `lborev1` against `lboopen`.

3.2.2 Modelling the data using the normal distribution

To start the analysis we assume a normal distribution for the response variable and check whether the mean model needs:

- a simple linear interaction model for the two explanatory variables `lboopen` and `lnosc`,
- an additive smoothing model for each of `lboopen` and `lnosc` or
- a fitted smooth surface model (using a tensor product spline) for `lboopen` and `lnosc`.

We also check whether we should include or exclude the factor `dist` in the mean model. Note that in order to fit a smooth surface to the data, we use the function `ga()` which is an interface to `gam()` from the **mgcv** package [Wood, 2001]. Note that `te()` gives a tensor product spline with five knots for each variable (which may need to be increased). For more details about the interface see Section **??**.

```
library(gamlss.add)
# linear interaction model
 m1 <- gamlss(lborev1~lboopen*lnosc, data=film90, trace=FALSE)
 m2 <- gamlss(lborev1~lboopen*lnosc+dist, data=film90, trace=FALSE)
# additive model using the pb() function
 m3 <- gamlss(lborev1~pb(lboopen) +pb(lnosc), data=film90,
              trace=FALSE)
 m4 <- gamlss(lborev1~pb(lboopen) +pb(lnosc)+dist, data=film90,
              trace=FALSE)
# fitting a surface using ga()
 m5 <- gamlss(lborev1~ga(~te(lboopen,lnosc)), data=film90,
              trace=FALSE)
 m6 <- gamlss(lborev1~ga(~te(lboopen,lnosc))+dist, data=film90,
              trace=FALSE)
```

```
GAIC(m1, m2, m3, m4, m5, m6)
```

```
##          df      AIC
## m6 16.01650 11779.76
## m4 18.53520 11828.59
## m5 15.91276 11843.78
## m3 18.12674 11908.73
## m2  6.00000 12080.99
## m1  5.00000 12226.84
```

```
GAIC(m1, m2, m3, m4, m5, m6, k=log(4031))
```

```
##          df      AIC
## m6 16.01650 11880.69
## m5 15.91276 11944.06
## m4 18.53520 11945.39
## m3 18.12674 12022.96
## m2  6.00000 12118.80
## m1  5.00000 12258.35
```

The best model appears to be `m6`, which fits a surface for `lboopen` and `lnosc` and an additive term for `dist`. Unfortunately a look at its residuals reveals that the normal distribution model fits very badly. The following worm plot shows this clearly, since most of the points lie outside the approximate pointwise 95% confidence interval bands (shown as dashed elliptical curves).

```
wp(m6, ylim.all=1.1)
```

Figure 15

Note that in order to visualize the fitted surface, `plot()` or `vis.gam()` of **mgcv** may be used. The `gam` object fitted within the backfitting algorithm is saved under the name `g4$mu.coefSmo` and is retrieved using the function `getSmo()`:

```
library(mgcv)
plot(getSmo(m6))
vis.gam(getSmo(m6),theta = 0, phi = 30)
```

Figure 16

To check whether we need to model σ as a function of the explanatory variables:

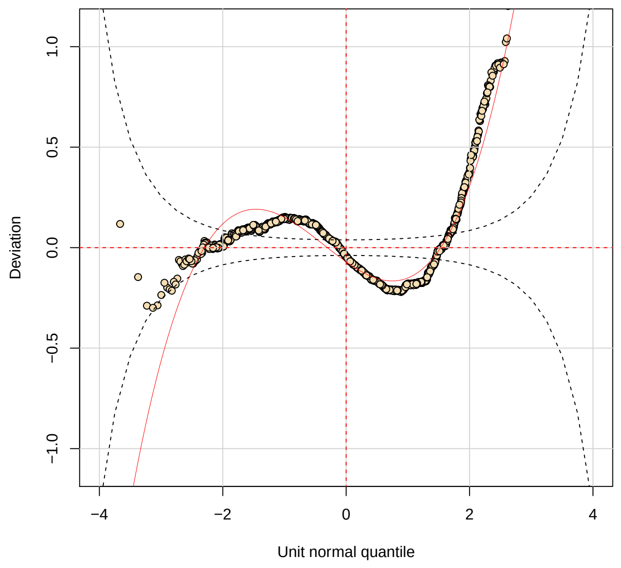
```
m7<- gamlss(lborev1~ga(~te(lboopen,lnosc))+dist,
           sigma.fo=~ga(~te(lboopen,lnosc))+dist,
           data=film90, trace=FALSE)
```

```
AIC(m6, m7)
```

```
##          df      AIC
## m7 27.64729 10043.89
## m6 16.01650 11779.76
```

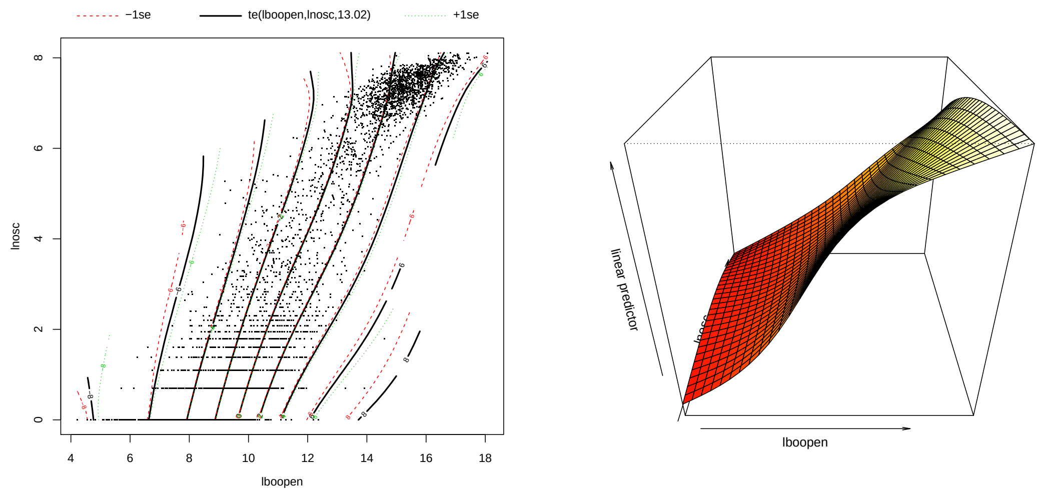
```
AIC(m6, m7, k=log(4031))
```

```
##          df      AIC
## m7 27.64729 10218.12
```



R code on page 28

Figure 15: The worm plot from the normal distribution model `m6`, in which a fitted surface was used for μ.



R code on page 28

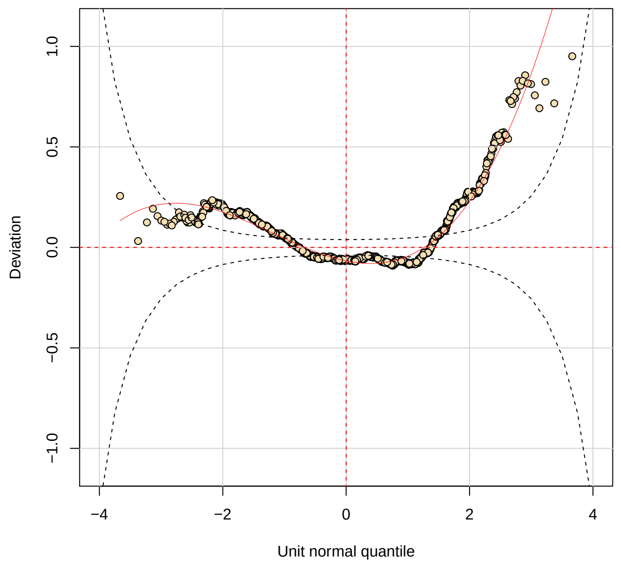
Figure 16: The fitted contour and surface plot from model `m6`.

```
## m6 16.01650 11880.69
```

We find that model `m7` is superior to `m6`, using either AIC or SBC. A worm plot of the residuals (Figure 17) is used to check the adequacy of the model. This indicates that model `m7`, while an improvement compared to `m6`, still does not adequately explain the response variable.

```
wp(m7, ylim.all=1.1)
```

Figure 17



R code on page 30

Figure 17: The worm plot from the normal distribution model `m7`, in which a fitted surface is used for both μ and σ.

3.2.3 Modelling the data using the BCPE distrbution

Next we model the response variable using the BCPE distribution [Rigby and Stasinopoulos, 2004], which is a four-parameter distribution defined on the positive real line. Model `m8` fits additive terms using `pb()`, while model `m9` fits smooth surfaces using `ga()` for all four distribution parameters.

```
m8 <- gamlss(lborev1 ~ pb(lboopen)+pb(lnosc) + dist,
         sigma.fo = ~ pb(lboopen)+pb(lnosc) + dist,
            nu.fo = ~ pb(lboopen)+pb(lnosc) + dist,
           tau.fo = ~ pb(lboopen)+pb(lnosc) + dist,
         family = BCPE, data = film90, trace=FALSE)
m9 <- gamlss(lborev1 ~ ga(~te(lboopen,lnosc)) + dist,
         sigma.fo = ~ ga(~te(lboopen,lnosc)) + dist,
            nu.fo = ~ ga(~te(lboopen,lnosc)) + dist,
           tau.fo = ~ ga(~te(lboopen,lnosc)) + dist,
         family = BCPE, data = film90, n.cyc=20, trace=FALSE)
```

```
AIC(m6, m7, m8, m9)
```

```
##          df       AIC
## m9 41.83029  9836.412
## m8 44.95828  9980.948
## m7 27.64729 10043.889
## m6 16.01650 11779.759
```

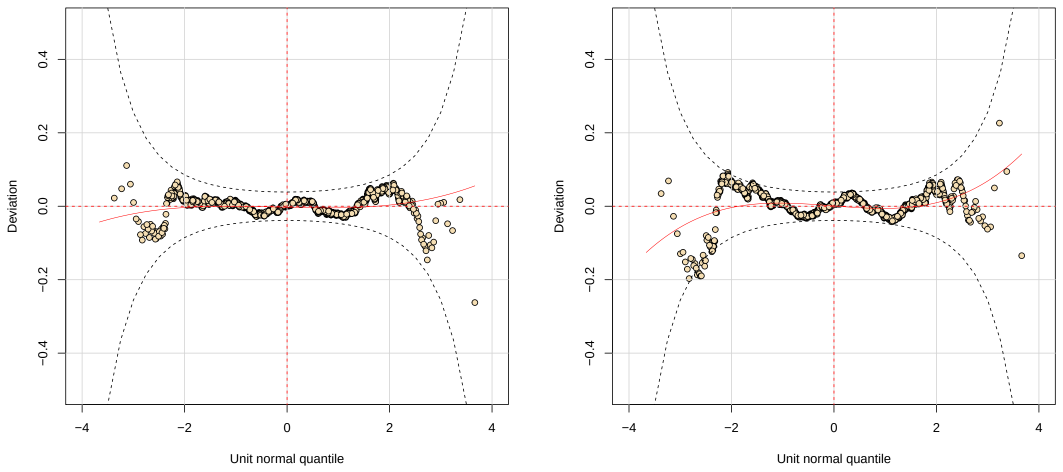
```
AIC(m6, m7, m8, m9, k=log(4031))
```

```
##          df      AIC
## m9 41.83029 10100.02
## m7 27.64729 10218.12
## m8 44.95828 10264.26
## m6 16.01650 11880.69
```

The model `m9` seems superior according to AIC and SBC, but it is more complicated (using far more degrees of freedom) and may be overfitting the data. Next we plot the worm plots for `m8` and `m9`.

```
wp(m8, ylim.all=0.5)
wp(m9, ylim.all=0.5)
```

Figure 18



R code on page 32

Figure 18: Worm plots from the BCPE distribution models. Left: m8, right: m9.

The worm plot of m8 (left panel of Figure 18) looks slightly better than that of m9 (on the right), but it is hard to decide. We can get a better idea of how the model fits in the joint ranges of the two explanatory varibles lboopen and lnosc by using a worm plot with two explanatory variables:

Figure 19

```
wp(m9, xvar=~lboopen+lnosc, ylim.worm=1)
```

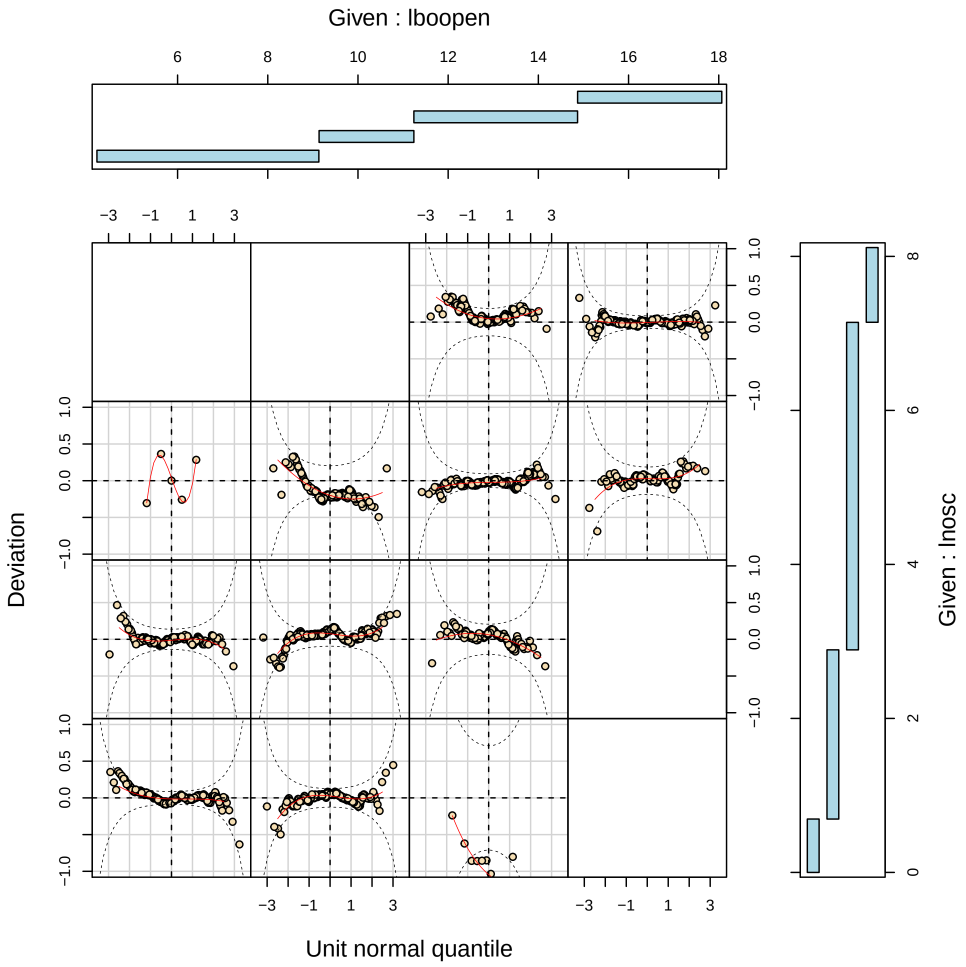
In the resulting worm plot given in Figure 19, the four columns correspond to the four ranges of lboopen displayed above the plot, and the four rows correspond to the four ranges of lnosc displayed to the right of the plot. Within the plot there are 16 individual worm plots of the residuals corresponding to the 16 joint ranges of lboopen and lnosc. Some joint ranges have no observations within them. The worm plots generally indicate an adequate fit within the joint ranges.

The fitted smooth surfaces for μ, σ, ν and τ for model m9 are plotted in Figure 20 by using the following commands:

Figure 20

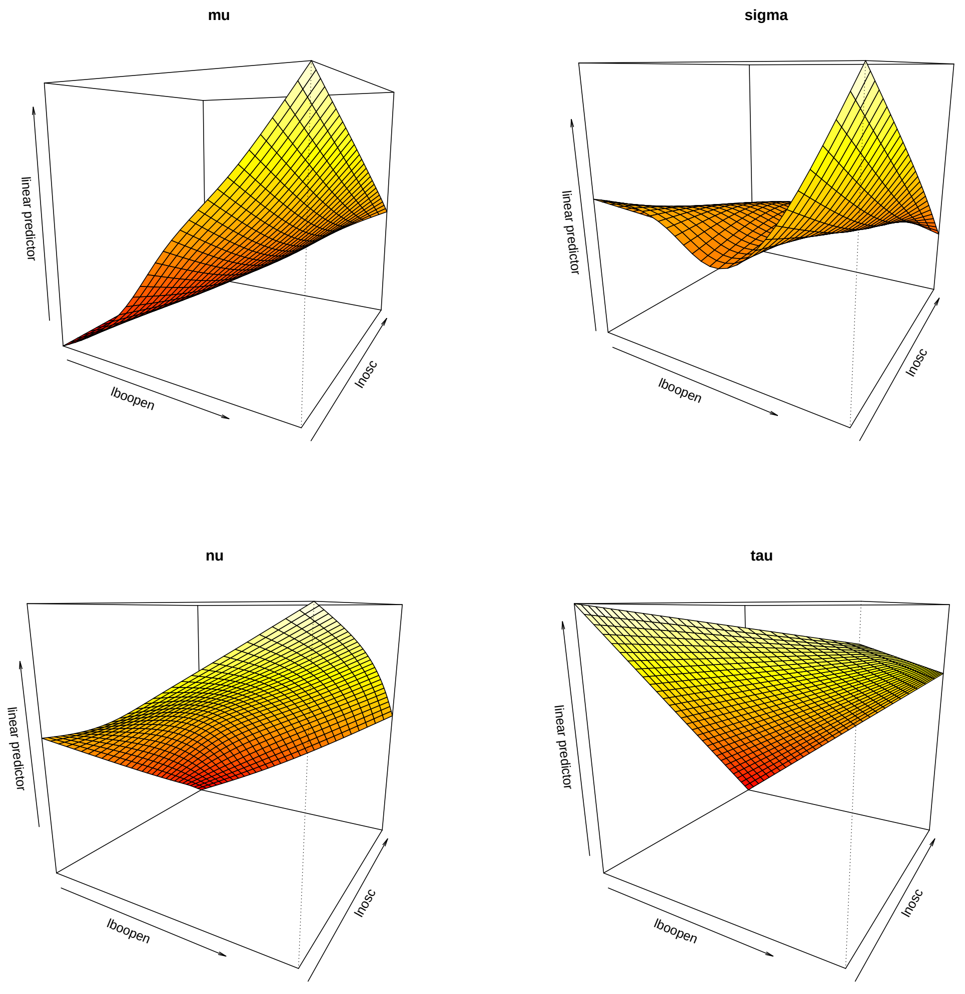
```
vis.gam(getSmo(m9,what="mu"), theta=30, phi=10)
title("mu")
vis.gam(getSmo(m9,what="sigma"), theta=30, phi=15)
title("sigma")
vis.gam(getSmo(m9,what="nu"), theta=30, phi=15)
title("nu")
vis.gam(getSmo(m9,what="tau"), theta=30, phi=15)
title("tau")
```

We leave further simpification of the model to the reader.



R code on page 32

Figure 19: The worm plot for model `m8`, by `lboopen` and `lnosc`.



R code on page 32

Figure 20: The fitted smooth surfaces for μ, σ, ν and τ of model `m9`.

4 Day 2 Afternoon

4.1 Practical 9: The LGA Claims data

The `LGAclaims` data set [de Jong and Heller, 2008] contains the number of third party claims in a twelve month period between 1984-1986 in each of 176 geographical areas (local government areas) in New South Wales, Australia. Areas are grouped into thirteen statistical divisions (`SD`). Other recorded variables are the number of accidents, the number of people killed or injured and population in each area. The number of claims (`Claims`) is analyzed as the response variable.

R data file: `LGAclaims` in package **gamlss.data** of dimensions 176×11

var **`LGA`** : local government area name

`SD` : statistical division (1,2,...,13)

`Claims` : number of third party claims

`Pop_density` : population density

`KI` : number of people killed or injured

`Accidents` : number of accidents

`Population` : population size

`L_Population` : log population

`L_Accidents` : log number of accidents

`L_KI` : log `KI`

`L_Popdensity` : log population density

purpose: to demonstrate selection of variables.

This exercse explores the use of `stepGAIC()` for the selection of terms for particular distribution parameters. Exercise **??** explores the automated function `stepGAICAll.A()` for selecting terms for all the distribution parameters.

1. Input the data and plot them.
2. Check whether a Poisson or negative binomial model for `Claims` (using the explanatory variables) is appropriate for the data.
3. The function `dropterm()` provides a single term deletion facility in `gamlss`. Check whether any of the linear terms can be deleted from the model.
4. The function `addterm()` provides the facility of adding single terms in the model. Use the function to check whether a two-way interaction is needed (from the model with the linear terms).
5. The fuction `stepGAIC()` provides a mechanism for stepwise selection of appropriate linear terms for any of the parameters of the distribution. Use it here to select a model for μ. Note that the argument `gd.tol=Inf` is crucial for some of the fitting at later stages since it prevents the algorithm from stopping if the deviance increases in any of the iterations.

6. Use the function `stepGAIC()` to select an appropriate model for σ, given the model for μ.
7. Conditional on the selected model for σ, explore if the model for μ can be simplified.
8. Plot the fitted terms for μ and σ respectively, using `term.plot()`.
9. Use diagnostics to assess whether the model residuals are supportive of the assumed model (see Chapter **??** for details).

4.2 Practical 10: The Dutch boys head circumference data

The Fourth Dutch Growth Study [Fredriks et al., 2000a,b] also recorded head circumference. In the data file `db` from **gamlss.data**, we have head circumference and age of the Dutch boys. Cases with missing values have been removed. There are 7,040 observations.

R data file: `db` in package **gamlss.data** of dimensions 7040×2

variables

`head` : head circumference in *cm*

`age` : age in years

purpose: to demonstrate centile estimation.

Familiarize with centile estimation by repeating the **R** commands given in this chapter, using the Dutch boys head circumference against age.

1. Input and plot the data.

```
data(db)
names(db)
plot(head~age,data=db)
```

2. To obtain centile curves for head circumference (`head`) against `age`, use the automated function `lms()`. This function performs the following:

   (a) first chooses an appropriate power transformation of `age`, $u = age^{\xi}$, by default `trans.x=TRUE`;

   (b) fits each of a list of families of distributions to the response variable `head`. Each parameter of a distribution is fitted locally using the P-spline smoothing function `pb()` in the transformed explanatory variable. `pb()` automatically chooses the smoothing parameter;

   (c) chooses the best distribution from the list of families according to criterion GAIC(κ).

```
m0<-lms(head,age,families=c("BCCGo","BCPEo","BCTo"),data=db,
        k=4,calibration=F, trans.x=T)
m0$family
m0$power
```

This function takes a few minutes, so read on while you wait.

Note that the best distribution family, according to GAIC(4), is stored in `m0$family`, i.e. `BCTo`. The power transformation chosen for age is stored in `m0$power`, i.e. $u = \texttt{age}^{\texttt{m0\$power}}$.

3. You can refit the chosen model using the `gamlss` function:

```
db$Tage<-(db$age)^(m0$power)
m1<-gamlss(head~pb(Tage),sigma.fo=~pb(Tage),nu.fo=~pb(Tage),
           tau.fo=~pb(Tage), family=BCTo,data=db)
```

Alternatively use

```
m2<-gamlss(head~pb(age^m0$power),sigma.fo=~pb(age^m0$power),
           nu.fo=~pb(age^m0$power), tau.fo=~pb(age^m0$power),
           family=BCTo,data=db)
GAIC(m0,m1,m2,k=4)
```

The alternative method is not recommended for large data since the calculation $\texttt{age}^{\texttt{m0\$power}}$ is performed at each iteration of the `gamlss` fitting algorithm.

4. The centile curves are given by

```
centiles(m0,xvar=db$age)
centiles.fan(m0,xvar=db$age)
```

To split the centile plot at `age` $= 3$ (in order to see the centiles for `age` < 3 more clearly):

```
centiles.split(m0,xvar=db$age,xcut.points=c(3))
```

A plot showing the distribution of head circumference (vertically) for specific values of age is given by

```
library(gamlss.util)
plotSimpleGamlss(head,age,m0,data=db,x.val=seq(5,20,5),
                 xlim=c(-3,23))
plotSimpleGamlss(head,age,m0,data=db,x.val=seq(1,22,7),
                 xlim=c(-8,23))
```

5. Look at the fitted parameters μ (the approximate median), σ (the approximate coefficient of variation), ν (the skewness parameter) and τ (the kurtosis parameter) of the `BCTo` distribution plotted against age.

```
fittedPlot(m0,x=db$age)
```

6. Check the residuals of the model to see if the model is adequate. The function `plot()` gives a QQ plot of the residuals, and `wp()` gives a worm plot. Approximately 95% of the residuals should be between the 95% pointwise interval bands in the worm plot.

```
plot(m0)
wp(m0,ylim.all=1)
```

Looking at the single worm plot, we find seven outliers in the upper tail. Now split the range of age into 16 intervals and obtain a QQ plot of the residuals within each age range. This allows the identification of any regions of age where the model is inadequate.

```
wp(m0,xvar=db$age,ylim.worm=1.5,n.inter=16)
```

Now obtain Q statistics for the 16 regions of age. This also allows the identification of any regions of age where the model is inadequate.

```
Q.stats(m0,xvar=db$age,n.inter=20)
```

7. The centiles values can be calculated and plotted for new values of age:

```
nage<-seq(0,20,0.1)
centiles.pred(m0,xname="age",xvalues=nage,plot=T,ylab="head",
              xlab="age",legend=F)
```

Also the z-scores for three new people with `(head, age)=(45,5)`, `(50,10)` and `(60,15)` respectively is obtained by:

```
newhead<- c(45,50,60)
newage<-c(5,10,15)
centiles.pred(m0,xname="age",xvalues=newage,yval=newhead,
              type="z-scores",plot=T, ylab="head",xlab="age")
```

A individual with a z-score < -2 indicates the person has an unusually low head circumference for his age, while a z-score > 2 indicates the person has an unusually high head circumference for his age.

8. Remove (or weight out) extreme outliers in head circumference (given age) as follows below.

From the worm plot of the residuals from model `m0` there are seven extreme outliers in the upper tail, with residuals greater than 3.5. There are also two cases with residuals less than -3.5. They are causing a distortion in the worm plot, resulting in a distortion in the fitted model as seen by the Q statistics and hence a distortion in the centile curves.

One solution to the distorted centile percentages (i.e. a difference between the nominal model percentages and the sample percentages below the centile curves) is to use calibration.

```
calibration(m0,xvar=db$age)
```

An alternative, possibly better, approach is to remove these outliers, i.e. seven extreme outliers in the upper tail and two in the lower tail.

```
which(resid(m0)>3.5)
which(resid(m0)< -3.5)
dbsub <- subset(db, (resid(m0)> -3.5)&(resid(m0)< 3.5))
```

Refit the model.

```
m3<-gamlss(head~pb(age^m0$power),sigma.fo=~pb(age^m0$power),
           nu.fo=~pb(age^m0$power),tau.fo=~pb(age^m0$power),
           family=BCTo,data=dbsub)
wp(m3,ylim.all=1)
```

The resulting fit to the data, worm plot and Q statistics are substantially improved. If the nine outliers are believed to be errors in the data set, then centile curves can be obtained directly from m3. However if the outliers are believed to be genuine observations, then centile curves should be obtained for the full data set db. The centile curve percentages from m3 need to be adjusted for the cases removed from each tail. To obtain centile curves for the full data set db at centiles given by cent use the following:

```
cent<- c(0.4,2,10,25,50,75,90,98,99.6)
a<- (2/7040)*100 # lower percentage removed
b<- (7/7040)*100 # upper percentage removed
newcent<-(cent-a)/(1-(a+b)/100)
centiles(m3,xvar=dbsub$age,cent=newcent, legend=FALSE)
```

4.3 Practical 11: The Global Lung Function Initiative data, males

This analysis finds centiles of a response variable dependent on two quantitative explanatory variables. The data are provided by the Global Lung Function Initiative, and are accessed at
www.ers-education.org/guidelines/global-lung-function-initiative/
statistics.aspx
The response variable is the forced expired volume (fev) and the explanatory variables are height and age.

1. (a) Input the data into data frame lung and select the males into data frame dm.

```
dm<-subset(lung, sex==1)
dim(dm)
```

   The number of male cases is $n = 5,723$.

   (b) Obtain a scatterplot of fev against height and age and both.

```
plot(fev~height,data=dm)
plot(fev~age,data=dm)
height <-dm$height
age <- dm$age
fev <- dm$fev
library(lattice)
cloud(fev~height*age)
# or more detailed
library(rgl)
plot3d(height,age,fev)
library(car)
scatter3d(dm$height,dm$age,dm$fev,xlab="height",ylab="age",
          zlab="fev",ticktype="detailed")
```

   (c) Following Cole et al. [2009] and Quanjer et al. [2012], apply a log transformation to height and age.

```
dm <- transform(da, la= log(age),lh=height)
```

2. Use stepGAICAll.A() to search for a suitable model for fev using the BCTo distribution (starting from a model m1 with constant parameters). Use a *local* SBC to choose the

effective degrees of freedom for smoothing in the smoothing functions pb. Also use a *global* SBC criterion to select terms in the stepGAICAll.A procedure. The reason for using SBC (i.e. $\kappa = \log(5723)$) is to achieve smooth centiles. A lower value of κ (e.g. $\kappa = 4$) would result in less smooth centiles but a better fit to the data, while a higher value of κ would result in even smoother centiles, but a worse fit to the data.

```
m1<-gamlss(fev~1,sigma.fo=~1,nu.fo=~1,tau.fo=~1, family=BCTo,
           data=dm,n.cyc=100)
k1<-log(5723)
m2<-stepGAICAll.A(m1,scope=list(lower=~1,upper=~pb(lh,
    method="GAIC",k=k1) + pb(la,method="GAIC",k=k1)), k=k1)
```

This will take about five minutes to complete. See the chosen model by

```
summary(m2)
```

3. (a) Refit the chosen model, but replacing lh and la by log(height) and log(age) in order to use predictAll() in (f) below.

```
m3<-gamlss(fev~pb((log(height)),method="GAIC",k=k1)+
               pb((log(age)),method="GAIC",k=k1),
     sigma.fo=~pb((log(height)),method="GAIC",k=k1)+
               pb((log(age)),method="GAIC",k=k1),
        nu.fo=~1,tau.fo=~1, family=BCTo,data=dm, n.cyc=100)
```

   (b) Amend model m3 to fit distribution BCCGo and then BCPEo and show that m3 has the lowest SBC.

   (c) Check the adequacy of model m3 using residual diagnostics.

```
plot(m3)
wp(m3,ylim.all=0.6)
wp(m3, xvar=~age, n.inter=9, ylim.worm=0.8)
wp(m3, xvar=~height, n.inter=9, ylim.worm=0.8)
wp(m3, xvar=~age+height, n.inter=4, ylim.worm=1)
Q.stats(m3,xvar=dm$height,n.inter=25)
```

   (d) Output the effective degrees of freedom (including 2 for the constant and linear terms) used for each smoothing function in model m3.

```
edfAll(m3)
```

   (e) Look at the fitted smooth functions in model m3.

```
term.plot(m3,what="mu", pages=1)
term.plot(m3,what="sigma", pages=1)
```

4. An alternative method of choosing the effective degrees of freedom for the smoothing functions is by minimizing a global SBC, instead of a local SBC in (c), using the find.hyper() function. This should use cubic splines instead of penalized splines. This takes about 60 minutes.

```
mod<-quote(gamlss(fev~cs((log(height)),df=p[1])+
   cs((log(age)),df=p[2]),sigma.fo=~cs((log(height)),
   df=p[3])+cs((log(age)),df=p[4]),nu.fo=~1,tau.fo=~1,
   family=BCTo,data=dm, control=gamlss.control(trace=FALSE,
                                               n.cyc=100)))

best<-find.hyper(model=mod,par=c(6,6,3,3),
        lower=c(0.01,0.01,0.01,0.01),
        steps=c(0.1,0.1,0.1,0.1), k=k1)
best
```

The resulting effective degrees of freedom are very similar to model `m3`.

5. (a) Now fit a model for `height` against `age`. The purpose of this is to find lower and upper centile limits (0.1% and 99.9%) of `height` for each `age` (to be used for the contour plot of the 5% centile of `fev` against `height` and `age`. in (f) below.

```
mh<-gamlss(height~pb(log(age),method="GAIC",k=k1),
           sigma.fo=~pb(log(age),method="GAIC",k=k1),
           nu.fo=~pb(log(age),method="GAIC",k=k1),
           tau.fo=~pb(log(age),method="GAIC",k=k1),
           family=BCTo, data=dm)
```

   Plot the centiles for `height` against `age` for model `mh`.

```
centiles(mh,xvar=dm$age,cent=c(0.1,0.4,2,10,25,50,75,90,
    98,99.6,99.9),ylab="height",xlab="age",legend=FALSE)
```

   (b) Now find lower (0.1%) and upper (99.9%) limits for `height` given `age`, stored in `maty[,2]` and `maty[,4]`.

```
newage<- seq(5,90,0.1)
newcent<- c(0.1,50,99.9)
maty<-centiles.pred(mh,xname="age",xvalues=newage,
                    cent=newcent,plot=TRUE)
maty[1:10,]
```

6. Construct a contour plot of the 5th centile of `fev` against `height` and `age`:

   (a) Expand a grid of values of `age` from 5 to 90 years and `height` from 100 to 210 cm to cover the limits of `height` in (e)(ii) above.

```
newdata<-expand.grid(age=seq(5,90,0.1),
                     height=seq(100,210,1))
```

   (b) Use the chosen model `m3` for `fev` to predict all the parameters μ, σ, ν and τ of the distribution `BCTo` for the values of `age` and `height` in `newdata`.

```
m3p<-predictAll(m3, newdata=newdata, type="response")
```

   (c) Calculate the 5th centile of `fev` for all cases in `newdata`.

```
fev5<-qBCPE(0.05,m3p$mu,m3p$sigma,m3p$nu,m3p$tau)
```

   (d) For all cases of `newdata` with values of `height` outside the lower (0.1%) and upper (99.9%) bounds for `height`, replace the value of `fev5` with a missing value (`NaN`).

```
lower<-rep(maty[,2],111)
upper<-rep(maty[,4],111)
fev5a<-ifelse(((newdata$height<lower)|
              (newdata$height>upper)),NaN,fev5)
```

(e) Obtain a contour plot of the 5th centile of `fev` against `height` and `age`.

```
newheight<-seq(100,210,1)
newage<-seq(5,90,0.1)
mfev5<-matrix(data=fev5a,nrow=851,ncol=111)
contour(newage,newheight,mfev5,nlevels=40,
        xlab="age(years)",ylab="height(cm)")
```

References

W. S. Cleveland and S. J. Devlin. Robust locally-weighted regression: an approach to regression analysis by local fitting. J. Am. Statist. Ass., 83:597–610, 1988.

T.J. Cole, S. Stanojevic, J. Stocks, A.L. Coates, J.L. Hankinson, and A.M. Wade. Age-and size-related reference ranges: A case study of spirometry through childhood and adulthood. Statistics in Medicine, 28(5):880–898, 2009.

P. de Jong and G. Z. Heller. Generalized Linear Models for Insurance Data. Cambridge University Press, 2008.

P. K. Dunn and G. K. Smyth. Randomized quantile residuals. J. Comput. Graph. Statist., 5: 236–244, 1996.

P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties (with comments and rejoinder). Statist. Sci, 11:89–121, 1996.

A.M. Fredriks, S. van Buuren, R.J.F. Burgmeijer, J.F. Meulmeester, R.J. Beuker, E. Brugman, M.J. Roede, S.P. Verloove-Vanhorick, and J. M. Wit. Continuing positive secular change in The Netherlands, 1955-1997. Pediatric Research, 47:316–323, 2000a.

A.M. Fredriks, S. van Buuren, J.M. Wit, and S. P. Verloove-Vanhorick. Body index measurements in 1996-7 compared with 1980. Archives of Childhood Diseases, 82:107–112, 2000b.

P. J. Huber. The behavior of maximum likelihood estimates under nonstandard conditions. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, volume 1, pages 221–233, 1967.

P. H. Quanjer, S. Stanojevic, T. J. Cole, X. Baur, G. L. Hall, B. H. Culver, P. L. Enright, J. L. Hankinson, M. S. Ip, J. Zheng, J. Stocks, and ERS Global Lung Function Initiative. Multi-ethnic reference values for spirometry for the 3-95-yr age range: the global lung function 2012 equations. The European Respiratory Journal, 40(6):1324–1343, 2012. URL `http://view.ncbi.nlm.nih.gov/pubmed/22743675`.

R. A. Rigby and D. M. Stasinopoulos. Smooth centile curves for skew and kurtotic data modelled using the Box-Cox power exponential distribution. Statistics in Medicine, 23:3053–3076, 2004.

R. A. Rigby and D. M. Stasinopoulos. Automatic smoothing parameter selection in GAMLSS with an application to centile estimation. Statistical Methods in Medical Research, 23(4): 318–332, 2013. doi: 10.1177/0962280212473302.

R. A. Rigby, D. M. Stasinopoulos, G. Z. Heller, V. Voudouris, and F. De Bastiani. Distributions for Modelling Location, Scale, and Shape: Using GAMLSS in R. Chapman and Hall, in press.

W. N. Venables and B. D. Ripley. Modern Appl. Statist. with S. Springer, 4th edition, 2002.

V. Voudouris, R. Gilchrist, R. Rigby, J. Sedgwick, and D. Stasinopoulos. Modelling skewness and kurtosis with the BCPE density in GAMLSS. Journal of Appl. Statist., 39(6):1279–1293, 2012.

H. White. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. Econometrica: Journal of the Econometric Society, pages 817–838, 1980.

S. N. Wood. mgcv: Gams and generalised ridge regression for r. R News, 1:20–25, 2001.