# GAMLSS Practicals for Semana de Estatística - Universidade Federal do Rio Grande do Norte

**Fernanda De Bastiani, Mikis D. Stasinopoulos, Robert A. Rigby, and Gillian Z. Heller**

October 21st and 22nd, 2019

## 1 The Munich rent data.

Use the code below to reproduce the analysis of the Munich rent data given in the lecture "Flexible Regression and Smoothing: Using GAMLSS in **R**".

```
library(gamlss)
PPP <- par(mfrow=c(2,2))
plot(R~Fl, data=rent, col=gray(0.7), pch=15, cex=0.5)
plot(R~A, data=rent, col=gray(0.7), pch=15, cex=0.5)
plot(R~H, data=rent, col=gray(0.7), pch=15, cex=0.5)
plot(R~loc, data=rent, col=gray(0.7), pch=15, cex=0.5)
par(PPP)
## ------------------------------------------------------------
r1 <-  gamlss(R ~ Fl+A+H+loc, family=NO,  data=rent, trace=FALSE)
l1 <- lm(R ~ Fl+A+H+loc,data=rent)
coef(r1)
coef(l1)
## ------------------------------------------------------------
fitted(r1, "sigma")[1]
summary(r1)
## ------------------------------------------------------------
Rsq(r1)
## ------------------------------------------------------------
plot(r1)
## ------------------------------------------------------------
### using gamlss
r2 <-  gamlss(R ~ Fl+A+H+loc, family=GA,  data=rent)
coef(r2)
coef(r2, "sigma")  ### extract log(sigma)
deviance(r2)
### using glm
l2 <- glm(R ~ Fl+A+H+loc, family=Gamma(link="log"),  data=rent)
coef(l2)
```

```
summary(l2)$dispersion  ### extract phi
deviance(l2)
## ------------------------------------------------------------
summary(r2)
## ------------------------------------------------------------
r22 <-  gamlss(R ~ Fl+A+H+loc, family=IG,  data=rent, trace=FALSE)
GAIC(r1, r2, r22, k=0) # GD

## ------------------------------------------------------------
plot(r2)
## ------------------------------------------------------------
r3 <-  gamlss(R ~ pb(Fl)+pb(A)+H+loc, family=GA,  data=rent,
              trace=FALSE)
AIC(r2,r3)
## ------------------------------------------------------------
summary(r3)
## ------------------------------------------------------------
drop1(r3)

## ------------------------------------------------------------
term.plot(r3, pages=1, ask=FALSE)
## ------------------------------------------------------------
wp(r3, ylim.all=.6)
## ------------------------------------------------------------
r4 <-  gamlss(R ~ pb(Fl)+pb(A)+H+loc, sigma.fo=~pb(Fl)+pb(A)+H+loc,
              family=GA, data=rent, trace=FALSE)
r5 <-  gamlss(R ~ pb(Fl)+pb(A)+H+loc, sigma.fo=~pb(Fl)+pb(A)+H+loc,
              family=IG, data=rent, trace=FALSE)
AIC(r3, r4, r5)

## ------------------------------------------------------------
term.plot(r4, pages=1, what="sigma", ask=FALSE)
## ------------------------------------------------------------
drop1(r4, what="sigma")
## ------------------------------------------------------------
wp(r4, ylim.all=.6)
## ------------------------------------------------------------
r6 <-  gamlss(R ~ pb(Fl)+pb(A)+H+loc, sigma.fo=~pb(Fl)+pb(A)+H+loc,
              nu.fo=~1, family=BCCGo,  data=rent, trace=FALSE)
r7 <-  gamlss(R ~ pb(Fl)+pb(A)+H+loc,sigma.fo=~pb(Fl)+pb(A)+H+loc,
              nu.fo=~pb(Fl)+pb(A)+H+loc, family=BCCGo,  data=rent,
              trace=FALSE)
AIC(r4, r6, r7)

## ------------------------------------------------------------
wp(r6, ylim.all=.6) ;  title("r6: BCCG(mu, sigma)")
wp(r7, ylim.all=.6) ;  title("r7:  BCCG(mu, sigma, nu)")
```

# 2 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 $\mu$, $\sigma$, $\nu$ and $\tau$. Here we give a simple demonstration using the `film90` data set.

**R data file:** `film90` in package **gamlss.data** of dimension $4015 \times 4$.
**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 **?**, 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 2.0.11, a GAMLSS model has the flexibility to model these features.

```
library(gamlss)
data(film90)
plot(lborev1~lboopen, data=film90, col="lightblue",
     xlab="log opening revenue", ylab="log extra revenue")
```

Figure 1

### 2.0.1 Fitting a parametric model

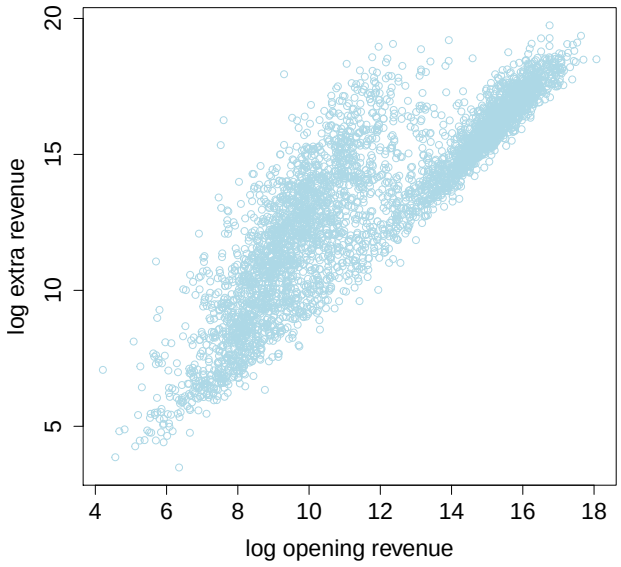
Below we fit a simple linear regression model with normal errors. It is clear from Figure 2 that the model does not fit well, especially for low values of `lboopen`.

```
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

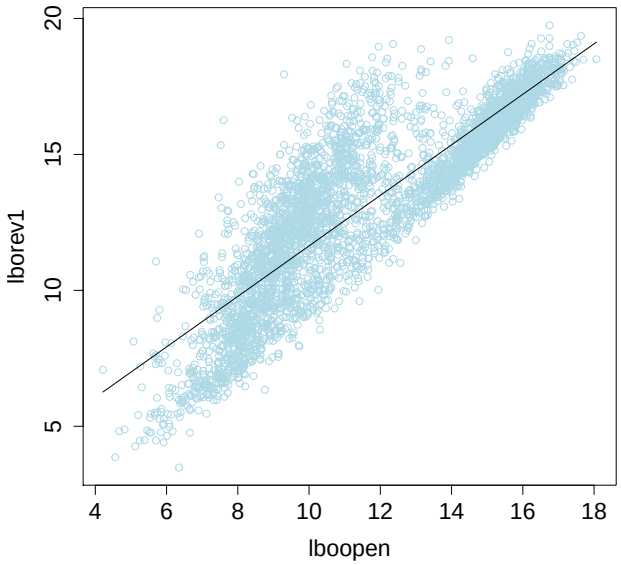



**R** code on page 3

Figure 1: Scatterplot of the `film90` revenues

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)

## ******************************************************************
## 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 ***
```




**R** code on page 3

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

```
## 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.

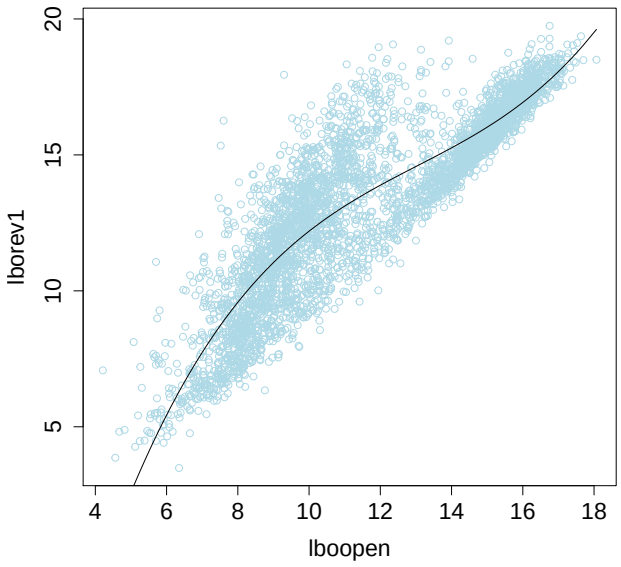
```
x2<-x^2; x3<-x^3
```

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

```
film90 <- transform(film90, lb2=lboopen^2, lb3=lboopen^3)
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)])
```




**R** code on page 6

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

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 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. $\mu$ and $\sigma$ 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 $\mu$, $\sigma$, $\nu$ and $\tau$. (In the example above the estimates of $\mu$ and $\sigma$ 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 **?** and **?** 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 $\sigma$ ($\hat{\beta}_{20}$), while the first row and/or column `Intercept` refers to the intercept of the predictor for $\mu$ ($\hat{\beta}_{10}$).

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

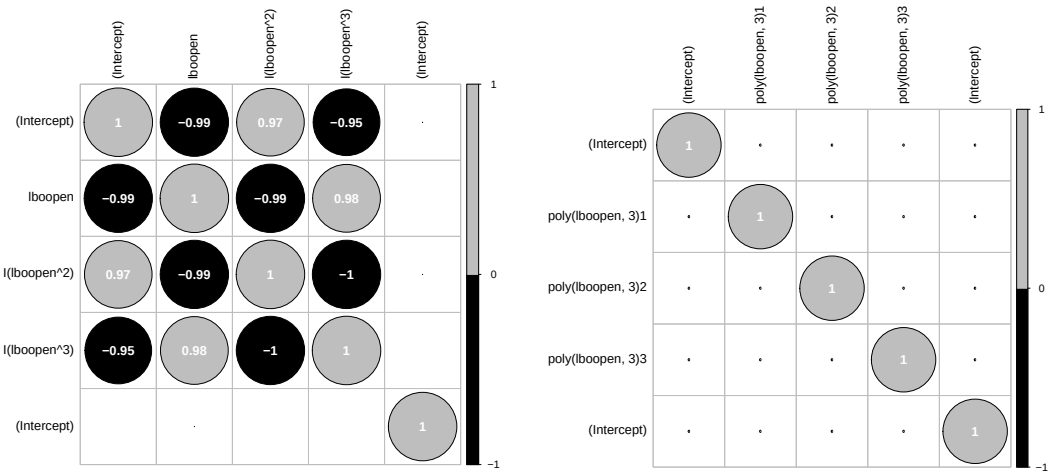
```
m0 <- gamlss(lborev1~poly(lboopen,3), data=film90, family=NO)

## 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,
```

Figure 4

```
         tl.col = "black", addCoef.col = "white")
corrplot(vcov(m0, type="cor"),   col=col1(2), outline=TRUE,
         tl.col = "black", addCoef.col = "white")
```




**R** code on page 9

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

Figure 4 shows the resulting graphical displays. Because, $\mu$ and $\sigma$ in the normal distribution are information independent (i.e. asymptotically uncorrelated), the first four estimated parameters ($\mu$ 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 $\mu$ model for `m0` are uncorrelated because we used orthogonal polynomials, but for `m00` they are highly correlated.

### 2.0.2 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 **??**.

### 2.0.3 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`. **?** introduced non-parametric 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:

```r
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
```

```r
summary(m1)
```

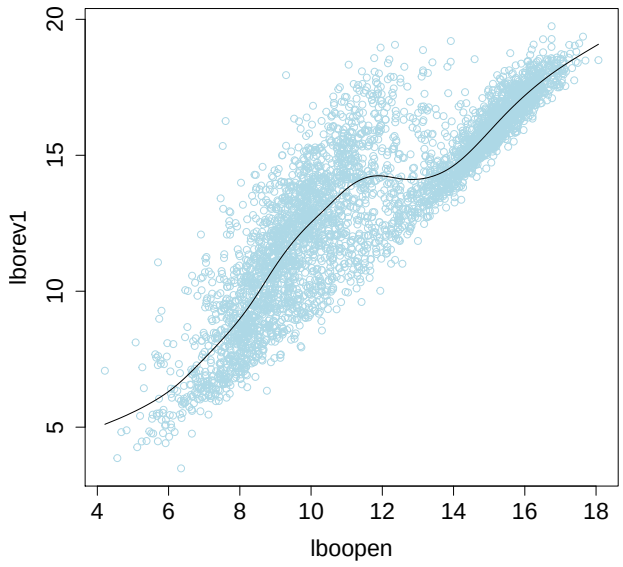
```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:
## gamlss(formula = lborev1 ~ pb(lboopen), family = NO,
##     data = film90)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link 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 **?**. 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).

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 11

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.

### 2.0.4 Cubic Splines

Other smoothers are also available. 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 $\mu$ 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)`.

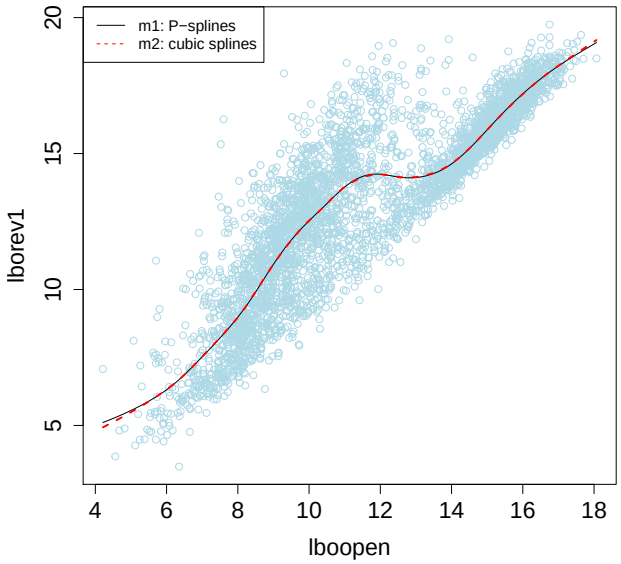
The total degrees of freedom used for model `m2` is 13, i.e. 12 for $\mu$ and 1 for $\sigma$. The fitted values of $\mu$ for models `m1` and `m2` are displayed in Figure 6:

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

Figure 6

### 2.0.5 loess

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




**R** code on page 12

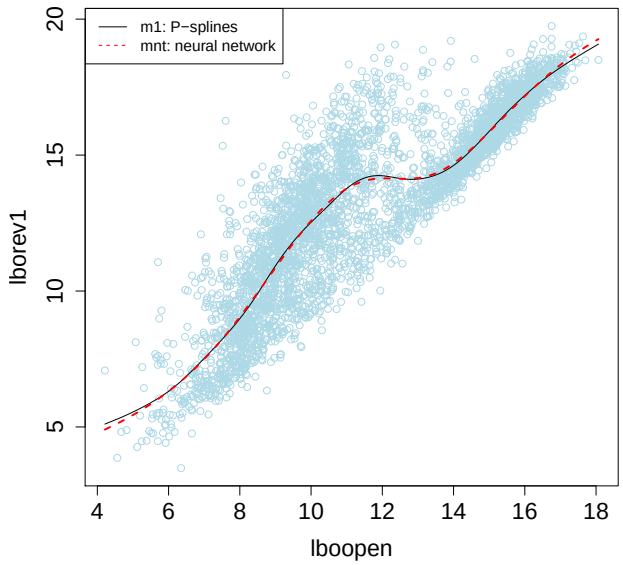
Figure 6: P-splines and cubic splines fits: plot of the `film90` data together with the fitted smooth mean functions of model `m1` fitted by `pb()` (continuous line) and model `m2` fitted by `cs()` (dashed line).

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

### 2.0.6 Neural Networks

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

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




**R** code on page 14

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.

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)
```

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
## . . .
```

### 2.0.7 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 $\mu$ 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 $\mu$). This means that the scale and shape of the distribution can vary as a

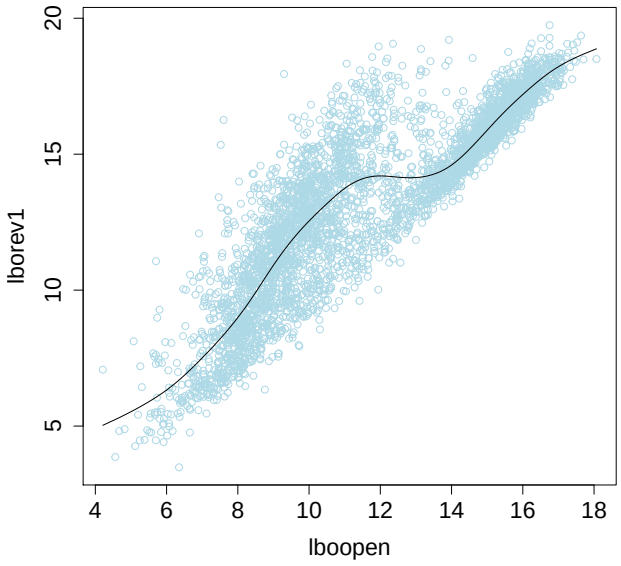
(linear or smooth) function of explanatory variables. Below, we show how to model both $\mu$ and $\sigma$ of a normal response distribution. Figure 1 suggests that this flexibility of a GAMLSS model might be required.

### 2.0.8 Modelling both $\mu$ and $\sigma$

To model the predictors of both the mean $\mu$ and the scale parameter $\sigma$ 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
```




**R** code on page 17

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)`.

The function `edfAll()` is used to obtain the effective degrees of freedom for all parameters. These are 12.14 and 10.68 for $\mu$ and $\sigma$ respectively. The fitted model for $\mu$ 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

### 2.0.9 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 **?** and Chapter **??** for more details. Residual plots are graphed using `plot()`:

```
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

Figure 9 shows plots of the residuals: (top left) against the fitted values of $\mu$; (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()`:

```
wp(m3)
```

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

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

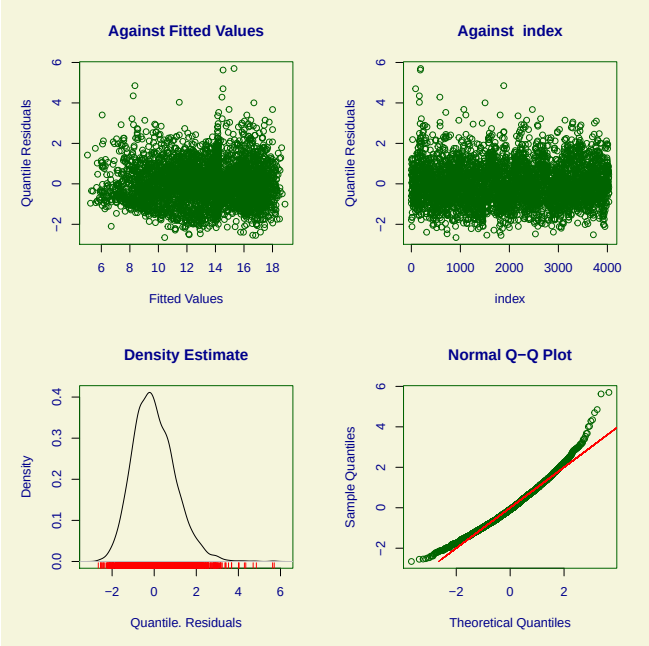
Figure 10

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)")
```

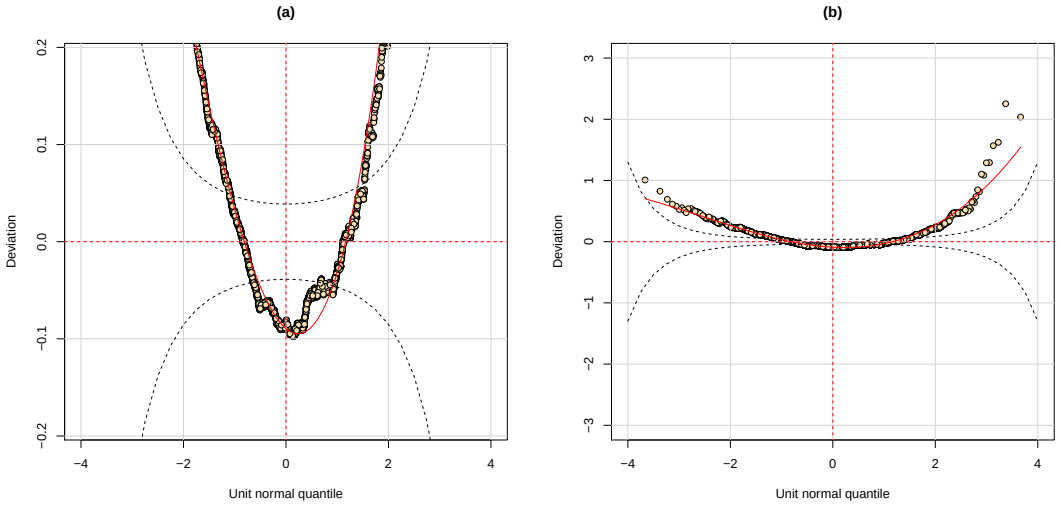
Figure 10

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.




**R** code on page 17

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




**R** code on page 17

Figure 10: Worm plots from model `m3`.

### 2.0.10 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 `?`.

The details of all the distributions currently available in `gamlss()` are given in `?`.

### 2.0.11 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 11 for model selection techniques in GAMLSS.

For example, different models can be compared by a test based on their global deviances: `GD` $= -2\hat{\ell}$ (if they are nested), or by selecting the model with lowest generalized Akaike information criterion: `GAIC` $= -2\hat{\ell} + \kappa \cdot$ `df`, where $\hat{\ell}$ is the fitted log-likelihood function and $\kappa$ 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()`,.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`: chunk 13

```
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 11.

### Chosen Model

Using the AIC, model `m6` is selected with $Y =$ `lborev` $\sim$ `BCPE`$(\mu, \sigma, \nu, \tau)$ where each of $\mu$, $\sigma$, $\nu$ and $\tau$ 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)
```

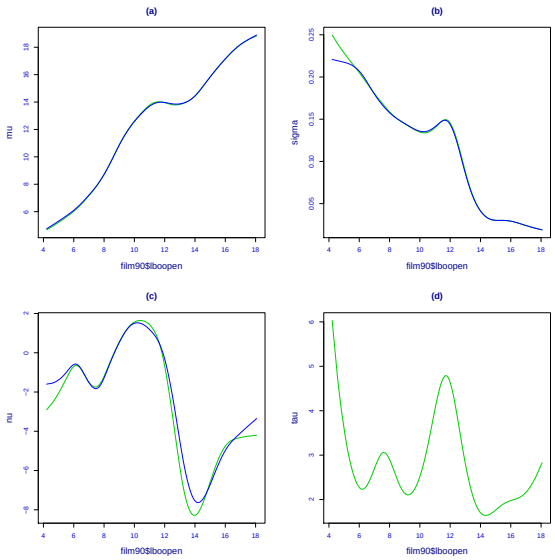
Since, in this example, only one explanatory variable is used in the fit, centile estimates for the fitted distribution can be shown using the functions `centiles()` or `centiles.fan()`.

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

Figure 12

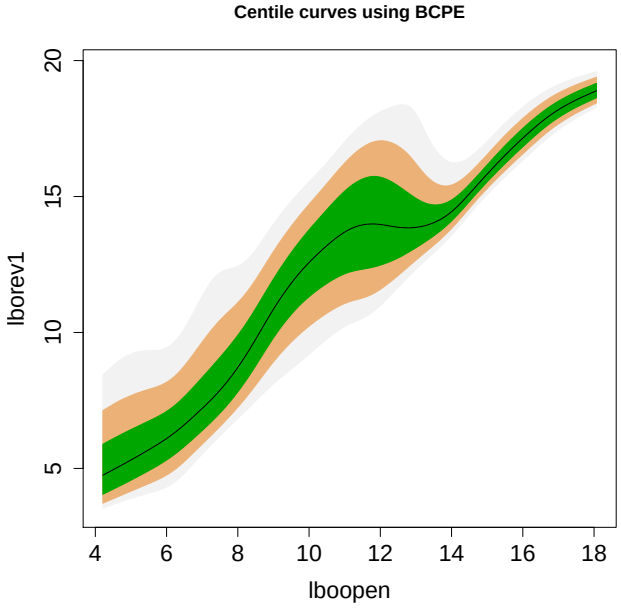
Figure 12 shows centile curves for `lborev1` against `lboopen` from the fitted model `m6`. For example the lowest curve is the fitted 3% centile curve, defined by 3% of the values of `lborev1` lying below the curve for each value of `lboopen`, for the fitted model `m6` if it was the correct model. For more details on centile curves see Chapter 13. 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




**R** code on page 20

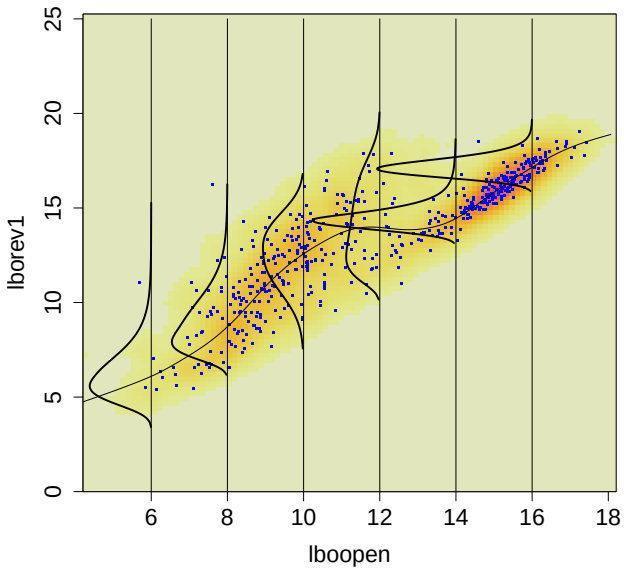
Figure 11: A plot of the smooth fitted values for all the parameters (a) $\mu$, (b) $\sigma$, (c) $\nu$ and (d) $\tau$ from models `m5` (dashed line) and `m6` (continuous line). The distribution for model `m5`, BCCG, has only three parameters so does not appear in panel (d).




**R** code on page 20

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.

```
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 way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
```




**R** code on page 22

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

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

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

# 3 The abdom data.

**R data file:** abdom in package **gamlss.data** of dimensions $610 \times 2$

**variables**

**y** : abdominal circumference

**x** : gestational age

**purpose:** to demonstrate the fitting of a simple regression type model in GAMLSS

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 $\mu$ and $\sigma$ 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 $\mu$ against x, lines(fitted(m)∼x, data=abdom).
7. For a chosen model, examine the centile curves using centiles(m,abdom$x).

# 4 The air quality data.

**The air quality data:** The data set airquality is one of the data frames available in **R** within the standard package **datasets**. It has the daily air quality measurements in New York, from May to September 1973.

**R data file:** airquality in package **datasets** of dimensions $154 \times 6$

**variables**

**Ozone** : in ppb

**Solar.R** : in lang

**Wind** : in mph

**Temp** : in F

**Month** : Month (1–12)

**Day** : Day of month (1–31)

**purpose:** to demonstrate the need for smooth functions.

(a) Here we will use `Ozone` as the response variable and `Solar.R`, `Wind` and `Temp` as explanatory variables. (We will not consider `Month` and `Day`.) The data can be plotted using:

```
data(airquality)
plot(airquality[,-c(5,6)])
```

Comment on the plot.

(b) To fit a standard regression model (i.e. with a normal distribution and constant variance) use the function `lm()`:

```
# Fit the standard linear model
air.lm <- lm(Ozone~Temp+Wind+Solar.R,data=airquality)
summary(air.lm)
```

The `summary()` provides information about the coefficients and their standard errors. To plot the fitted model terms use `termplot()`:

```
op<-par(mfrow=c(1,3))
termplot(air.lm,partial.resid=TRUE,se=T)
par(op)
```

Comment on the term plot.

(c) Check the residuals using `plot()`:

```
op<-par(mfrow=c(1,2))
plot(air.lm,which=1:2)
par(op)
```

(d) Fit the same model using the `gamlss()` function, but note that the data set `airquality` has some missing observations (i.e. `NA` values). The `gamlss()` function does not work with `NA`'s, so before fitting the model the cases with missing values have to be removed:

```
library(gamlss)
da <- na.omit(airquality)  # clear the data of NA's
mno<-gamlss(Ozone~Temp+Wind+Solar.R, data=da)  # fit the model
summary(mno)
```

Summarize the fitted `gamlss` model using `summary()`. Plot the fitted terms using the corresponding function for `gamlss` called `term.plot()`:

```
term.plot(mno, pages=1, partial=T)  # plot the fitted terms
```

(e) Check the residuals using the `plot()` and `wp()` functions:

```
plot(mno)
wp(mno)
```

Comment on the worm plot. Note the warning message that some points are missed out of the worm plot. Increase the limits in the vertical axis by using the argument `ylim.all=2` in `wp()`.

(f) Since the fitted normal distribution seems not to be correct, try to fit different distributions (e.g. gamma (`GA`), inverse Gaussian (`IG`) and Box Cox Cole and Green (`BCCGo`)) to the data. Compare them with the normal distribution using GAIC with penalty $k = 2$ (i.e. AIC).

```
# fit different distributions
mga <- gamlss(Ozone~Temp+Wind+Solar.R, data=da, family=GA)
mig <- gamlss(Ozone~Temp+Wind+Solar.R, data=da, family=IG)
mbccg <- gamlss(Ozone~Temp+Wind+Solar.R, data=da, family=BCCGo)
GAIC(mno, mga, mig, mbccg)
```

(g) For the selected distribution, fit smoothing terms, i.e `pb()`, for `Solar.R`, `Wind` and `Temp`.

```
# fit smoothers
mga1=gamlss(Ozone~pb(Temp)+pb(Wind)+pb(Solar.R),data=da,
            family=GA)
term.plot(mga1, pages=1)
plot(mga1)
wp(mga1)
```

Is the model improved according to the AIC? Use `term.plot()` output to see the fitted smooth functions for the predictor of $\mu$ for your chosen distribution. Use `plot()` and `wp()` output to check the residuals.

# 5 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 $0 \leq y \leq 1$

# 6 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 $\texttt{GA}(\mu, \sigma)$, 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.

# 7 The DAX data.

Use the code below to reproduce the analysis of the DAX data given in lecture.

```
#-----------------------------------------------------
# plot the data
dax <- EuStockMarkets[,"DAX"]
Rdax<-diff(log(dax))
plot(Rdax, col=gray(.2)); title("(a)")
library(MASS)
truehist(Rdax, col=gray(.7)); title("(b)")
#-----------------------------------------------------
# using fitDist()
f1 <- fitDist(Rdax, type="realline")
f1$fits
f1$failed
#-----------------------------------------------------
# using chooseDist()
m1 <- gamlssML(Rdax, family = NO)
t1 <- chooseDist(m1, type = "realline")
t1
# getting the order of best fits
getOrder(t1,1)[1:6]
# refit the final model
mf <- update(m1, family="GT")
# getting the coefficients
summary(mf)
# the fitted parameters
fitted(f1,"mu")[1]
fitted(f1,"sigma")[1]
fitted(f1,"nu")[1]
```

```
fitted(f1, "tau")[1]
# plot of the fitted distribution
fh<-histDist(Rdax, family=GT, nbins=30, line.col="black")
```

# 8 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 $2868 \times 6$.

**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`$(\mu, \sigma)$,

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

(b) three-parameter: t family TF$(\mu, \sigma, \nu)$ and power exponential PE$(\mu, \sigma, \nu)$

(c) four-parameter: Johnson Su JSU$(\mu, \sigma, \nu, \tau)$, skew exponential power type 1 to 4, e.g. SEP1$(\mu, \sigma, \nu, \tau)$, skew t type 1 to 5, e.g. ST1$(\mu, \sigma, \nu, \tau)$ and sinh arc-sinh SHASH$(\mu, \sigma, \nu, \tau)$.

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

5. Use `GAIC()` with each of the penalties $k = 2, 3.84$ 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()`.

# 9 Parzen snowfall data

**R data file:** `parzen` in package **gamlss.data** of dimension $63 \times 1$
**source:** **?**
**variables**
**snowfall** : the annual snowfall in Buffalo, NY (inches) from 1910 to 1972 inclusive.
**purpose:** to demonstrate the fitting of continuous distribution to a single variable.
**conclusion:** the Weibull distribution appears to fit best.

This data set is used by **?** and is also in **?**, data set 278.

**Selecting the distribution** Use the function `fitDist()` to fit distributions to the data. We are using the default value for the argument `type = "realAll"`, meaning we are using all available continuous distributions. Also we try two different information criteria: AIC and SBC.

```
data(parzen)
mod1 <- fitDist(snowfall, data=parzen, k=2)
mod2 <- fitDist(snowfall, data=parzen, k=log(dim(parzen)[1]))
mod1$fit[1:6]
mod2$fit[1:6]
```

Using both criteria, it is obvious that the best model is the one using the Weibull distribution, although several other distributions (including the normal) have similar values of AIC and SBC.

Next refit and plot the fitted model using `histDist()`, giving Figure **??**. Note that the option `density=TRUE` requests a nonparametric kernel density estimate to be superimposed on the plot.

```
m1 <-histDist(parzen$snowfall, "WEI3" , density=TRUE,
              line.col=c(1,1), line.ty=c(1,2))
```

The WEI3$(\mu, \sigma)$ distribution is the parameterization of the Weibull distribution with $\mu$ the mean.

**Checking the model**

A check of the normalized quantile residuals using a Q-Q and a worm plot (i.e. a detrended Q-Q plot) provides a guide to the adequacy of the fit. The **gamlss** package provides the functions `plot()` and `wp()` for this purpose.

```
plot(m1)
wp(m1)
```

### Testing hypotheses about the model

There are several methods to check the reliability of the fitted parameters of the distribution. Standard errors for the fitted parameters are provided by two functions: (i) `summary()` and (ii) `vcov()`. In general the values obtained should be identical, since by default `summary()` gives the standard errors obtained by `vcov`. The standard errors obtained by `vcov()` are the ones obtained by inverting the full Hessian matrix and they do take into account the correlations between the distribution parameter estimates. Note that the function `vcov()`, applied to a **gamlss** object, refits the final model one more time in order to obtain the Hessian matrix. Occasionally this could fail, in which case `summary()` will use an alternative method called `qr` and give a *warning* that `qr` is used. This uses the QR decomposition of the individual distribution parameter estimation fits. The standard errors given by the `qr` method of `summary()` are not very reliable since they are the conditional standard errors obtained by assuming that the other distribution parameters are fixed at their maximum likelihood estimates. Use the `summary()` and the `vcov()` function.

```
m1<-gamlss(snowfall~1, data=parzen, family=WEI3, trace=FALSE)
summary(m1)
vcov(m1, type="se")
```

The fitted Weibull distribution model is given by $Y_i \sim$ `WEI3`$(\hat{\mu}, \hat{\sigma})$ where $\log(\hat{\mu}) = 4.387$, $\hat{\mu} = \exp(4.387) = 80.399$; and $\log(\hat{\sigma}) = 1.344$, so $\hat{\sigma} = 3.834$. Note that $\hat{\mu}$ and $\hat{\sigma}$ are the maximum likelihood estimates of $\mu$ and $\sigma$.

The standard errors obtained are 0.0368 for $\log(\hat{\mu}) = \hat{\beta}_{01}$ and 0.0992 for $\log(\hat{\sigma}) = \hat{\beta}_{02}$ respectively, using either the `summary()` or `vcov()` functions. Note that since the Weibull fitting function `WEI3()` uses the log link for both $\mu$ and $\sigma$, the standard errors given are those for $\log(\hat{\mu}) = \hat{\beta}_{01}$ and for $\log(\hat{\sigma}) = \hat{\beta}_{02}$. For example, an approximate 95% confidence interval (CI) for $\log(\sigma) = \beta_{02}$, using the `vcov()` results, is

$$(1.344 - (1.96 \times 0.0992), 1.344 + (1.96 \times 0.0992)) = (1.150, 1.538)\,.$$

Hence an approximate 95% CI confidence interval for $\sigma$ is given by

$$(\exp(1.150), \exp(1.538)) = (3.158, 4.655)\,.$$

This 95% CI for $\sigma$ can be compared with the more reliable profile deviance 95% CI:

```
prof.dev(m1, "sigma", min=3, max=4.8, step=.01, col=1)
```

giving 95% CI (3.126, 4.617). Note that `prof.dev()` works only with **gamlss** objects.

These CIs may also be compared with the bootstrap 95% CI for $\sigma$:

```
library(boot)
set.seed(1453)
```

```
mod1<-gamlss(snowfall~1, data=parzen, family=WEI3, trace=FALSE)
funB <- function(data, i)
     {
     d<-data.frame(snowfall=data[i,])
     coef(update(mod1, data=d),"sigma")
     }
(mod1.boot<-boot(parzen, funB, R=199, parallel="multicore",
                 ncpus = 4))
boot.ci(mod1.boot, type=c("norm", "basic"))
```

There are two 95% bootstrap CIs intervals for $\sigma$, the 'normal'

$$(\exp(1.154), \exp(1.507)) = (3.171, 4.513),$$

and the 'basic"

$$(\exp(1.160), \exp(1.491)) = (3.190, 4.442).$$

More details about the `boot()` function can be found in **?**, p. 173.

# 10 The cable television data.

The `cable` data set concerns the penetration of cable television in $n = 283$ market areas in the USA. The data were collected in a mailed survey questionnaire in 1992 [**?**]. The aim of the study was to explain cable television uptake (the proportion `pen5`) as a function of area demographics.

**R data file:** `cable` in package **gamlss.data** of dimension $283 \times 6$
**source:** **?**
**variables**
- **pen5** : proportion of households having cable TV in market area
- **lin** : log median income
- **child** : percentage of households with children
- **ltv** : number of local TV stations
- **dis** : consumer satisfaction index
- **agehe** : age of cable TV headend

**purpose:** to demonstrate the fitting of a parametric distribution to the response variable `pen5` with range $(0, 1)$.
**conclusion:** For the marginal distribution, the truncated normal fits best; for the regression model, the truncated skew $t$ (`SSTtr`) is best.

1. Examine a histogram of `pen5`. What is the range of `pen5`?
   ```
   truehist(cable$pen5, nbins=20, xlim=c(0,1))
   ```
2. Using `fitDist()`, find the distribution that best fits `pen5`. Use generated and truncated distributions, as well as the explicit `gamlss.family` distributions. Display the fit of the 'best' distribution (using the default, AIC).

```
library(gamlss.tr)
# logit transformations
gen.Family("TF", "logit")
gen.Family("ST3", "logit")
gen.Family("SEP3", "logit")
# truncated distributions
gen.trun(c(0,1),"TF",type="both")
gen.trun(c(0,1),"NO",type="both")
gen.trun(c(0,1),"SST",type="both")

a <- fitDist(cable$pen5, type="real0to1", extra=c("logitTF",
          "logitST3", "logitSEP3","TFtr", "NOtr", "SSTtr"))
a$fits
histDist(cable$pen5, family=NOtr)
```

3. Now select a distribution for the regression model for `pen5`, with $\mu$ predictor `pb(lin)+ltv+agehe`.

```
m0 <- gamlss(pen5~pb(lin)+ltv+agehe, family=BE, data=cable,
             n.cyc=50)
c1 <- chooseDist(m0, type="real0to1", extra=c("logitTF",
          "logitST3", "logitSEP3","TFtr", "NOtr", "SSTtr"))
m1 <- gamlss(pen5~pb(lin)+ltv+agehe, family=SSTtr, data=cable,
             n.cyc=50)
plot(m1)
wp(m1)
```

4. Investigate whether model `m1` can be improved on by the addition or deletion of covariates in the model for $\mu$, and the addition of covariates in the model for $\sigma$.

# 11 The 2000 Presidential Election.

**?** analyse US election data, at the state level, in the 2000 Presidential Election. The response variable is the proportion of the state that voted for George Bush; and the predictors are state demographic indicators.

**R data file:** `bush2000` in package **gamlss.data** of dimension $51 \times 10$
**source:** ?
**variables**
- **state** : name of state
- **bush** : proportion of state's vote for George Bush
- **male** : percentage of population male
- **pop** : population
- **rural** : percentage of population living in rural areas
- **bpovl** : percentage of population with income below the poverty level
- **clfu** : unemployment rate (%)
- **mgt18** : percentage of male population older than 18 years
- **pgt65** : percentage of population older than 65 years
- **numgt75** : percentage of population with income > \$75K

**purpose:** to demonstrate the fitting of a parametric distribution to the response variable `bush` with range $(0, 1)$.
**conclusion:** For the marginal distribution, the truncated $t$ family fits best; for the regression model, the truncated normal is best.

- Examine a histogram of `bush`. What type of distribution is suggested?

```
truehist(bush2000$bush, nbins=20, xlim=c(0,1))
```

- Using `fitDist()`, find the distribution that best fits `bush`. Use generated and truncated distributions, as well as the explicit `gamlss.family` distributions as in question 1. Display the fit of the 'best' distribution.
- Now select a distribution for the regression model for `bush`, with $\mu$ predictor `male+log(pop)+rural+bpovl+clfu`.

```
m0 <- gamlss(bush~male+log(pop)+rural+bpovl+clfu,
             family=BE, data=bush2000, n.cyc=100)
c1 <- chooseDist(m0, type="real0to1", extra=c("logitTF",
          "logitST3", "logitSEP3","TFtr", "NOtr", "SSTtr"))
m1 <- gamlss(bush~male+log(pop)+rural+bpovl+clfu,
             family=NOtr, data=bush2000, n.cyc=100)
plot(m1)
wp(m1)
```

- Investigate whether model `m1` can be improved on by the addition or deletion of covariates in the model for $\mu$, and the addition of covariates in the model for $\sigma$.

## 12 `EuStockMarkets` data

For the rest of the returns from the `EuStockMarkets` data (that is do not use DAX):

1. Use the functions `checkMomentSK()` and `checkCentileSK()` to check the skewness and kurtosis. For example to get the UK FTSE returns use:

```
ftse <- EuStockMarkets[,"FTSE"]
Rftse<-diff(log(ftse))
```

Repeat the same to getthe Switzerland SMI and France CAC returns.

2. Fit an appropriate model to each of the returns and check the skewness and kurtosis of the fitted residuals.

# 13 Munich rent data

This example shows how to check the skewness and kurtosis of several fitted models using the same plot. We use the Munich `rent` data, which come from a survey conducted in April 1993 by Infratest Sozialforschung, in which a random sample of accommodation with new tenancy agreements or increases of rents within the last four years in Munich was selected.

**R data file:** `rent` in package **gamlss.data** of dimension $1969 \times 9$

**var** `R` : monthly net rent in Deutsche Marks (DM), i.e. the monthly rent minus calculated or estimated utility cost (response variable)

`Fl` : floor space in square meters

`A` : year of construction

1. Fit different models to the rent data.

```
r1 <- gamlss(R~pb(Fl)+pb(A), data=rent)
r2 <- gamlss(R~pb(Fl)+pb(A), data=rent, family=GA)
r3 <- gamlss(R~pb(Fl)+pb(A), data=rent, family=BCCG)
r4 <- gamlss(R~pb(Fl)+pb(A), sigma.fo=~pb(Fl)+pb(A),data=rent)
r5 <- gamlss(R~pb(Fl)+pb(A), sigma.fo=~pb(Fl)+pb(A),data=rent,
             family=GA)
r6 <- gamlss(R~pb(Fl)+pb(A), sigma.fo=~pb(Fl)+pb(A),data=rent,
             family=BCCG)
```

2. Inspect the skewness and kurtosis for all models simultaneously:

```
checkMomentSK(r1, boot=T, col.boot="yellow1")
checkMomentSK(r2, add=T, boot=T, col.bootstrap = "turquoise")
checkMomentSK(r3, add=T, boot=T, col.bootstrap = "tan")
checkMomentSK(r4, add=T, boot=T, col.bootstrap = "violet")
checkMomentSK(r5, add=T, boot=T, col.bootstrap = "whitesmoke")
checkMomentSK(r6, add=T, boot=T, col.bootstap = "wheat" )
```

3. Comment on the results.

4. Investigate the adequacy of your chosen model using multiple worm plots and Q-statistics (**?**, Chapter 12).

# 14 The stylometric data.

**R data file:** `stylo` in package **gamlss.data** of dimensions $64 \times 2$

**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`.

# 15 The fish species data.

**R data file:** `species` in package **gamlss.data** of dimension $70 \times 2$

**variables**

**`fish`** : the number of different species in 70 lakes in the world

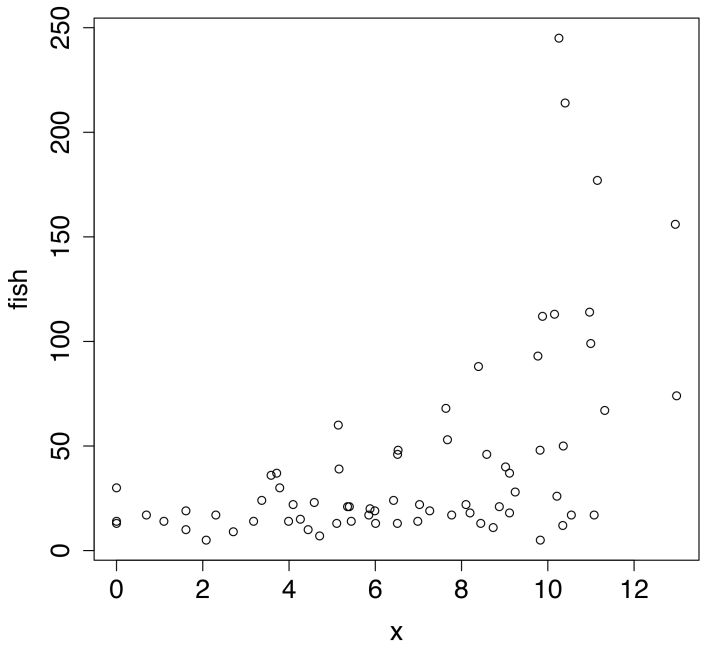
**`lake`** : the lake area

**purpose:** to demonstrate results of fitting using RS, CG and 'mixed' algorithms

The number of different fish species (`fish`) was recorded for 70 lakes of the world together with explanatory variable `x` = log lake area. Follow the analysis below:

```
library(gamlss)
data(species)
# creating the log(lake)
species <- transform(species, x=log(lake))
plot(fish~x,data=species)
```

Figure 14




**R** code on page 35

Figure 14: The fish species data.

The data were analysed by Stein and Juritz (1988) using a Poisson inverse Gaussian (PIG($\mu, \sigma$)) distribution for `fish`, with a linear model in `log(lake)` for $\log \mu$, and a constant for $\sigma$. Rigby *et al.* analysed this data set and identified the following questions that need to be answered. Note that the same questions could apply to any regression situation where the response variable is a count and `x` represents a set of explanatory variables.

- How does the mean of the response variable depend on `x`?
- Is the response variable overdispersed Poisson?
- How does the variance of the response variable depend on its mean?
- What is the conditional distribution of the response variable given `x`?
- Do the scale and shape parameters of the response variable distribution depend on `x`?

Here we will model the data using different discrete distributions and consider flexible models for the distribution parameters, where any or all of them may depend on the explanatory variable `log(lake)`. We start by fitting seven different count distributions to the data:

- Poisson (`PO`),
- double Poisson (`DPO`),
- negative binomial types I and II (`NBI`, `NBII`),
- Poisson inverse Gaussian (`PIG`),
- Delaporte (`DEL`) and

- Sichel (`SICHEL`).

We first use a linear and then a quadratic polynomial in `x=log(lake)`. The AIC of each model is printed for comparison.

```
# the count distributions
fam<-c("PO","DPO", "NBI", "NBII", "PIG", "DEL", "SICHEL")
#creating lists to keep the results
m.l<-m.q<-list()
# fitting the linear in x models
for (i in 1:7) {
m.l[[fam[i]]]<-GAIC(gamlss(fish~x,data=species, family=fam[i],
               n.cyc=60, trace=FALSE),k=2)}
# fitting the quadratic in x models
for (i in 1:7) {
m.q[[fam[i]]]<-GAIC(gamlss(fish~poly(x,2),data=species,
               family=fam[i], n.cyc=60, trace=FALSE), k=2)}
# print the AICs
unlist(m.l)
```

```
##        PO       DPO       NBI      NBII       PIG       DEL
## 1900.1562  654.1616  625.8443  647.5359  623.4632  626.2330
##    SICHEL
##  625.3923
```

```
unlist(m.q)
```

```
##        PO       DPO       NBI      NBII       PIG       DEL
## 1855.2965  655.2520  622.3173  645.0129  621.3459  623.5816
##    SICHEL
##  623.0995
```

The Poisson model has a very large AIC compared to the rest of the distributions so we can conclude that the data are overdispersed. The quadratic polynomial in `x` seems to fit better than the linear term across the different count distributions (except for `DPO`), as judged by AIC. The best model at this stage is the Poisson inverse Gaussian (`PIG`) model with a quadratic polynomial in `x`. We now compare the AIC of a `PIG` model with a P-spline smoother, instead of a quadratic polynomial, in `x`. The total effective degrees of freedom for `x` is calculated automatically using `pb(x)` by the local ML method.

```
GAIC(m.pb<-gamlss(fish~pb(x), data=species, family=PIG, trace=FALSE))
```

```
## [1] 623.4637
```

```
m.pb$mu.df
```

```
## [1] 2.000016
```

The P-spline smoothing does not seem to improve the model, so we keep the quadratic polynomial in `x`. We now model $\log(\sigma)$ as a linear function of `x` in the six remaining count distributions (after excluding the Poisson distribution which does not have a $\sigma$ parameter).

```
# redefine the list of distributions
fam<-c("DPO","NBI", "NBII", "PIG", "DEL", "SICHEL")
m.ql<-list()
for (i in 1:6) {
m.ql[[fam[i]]]<-GAIC(gamlss(fish~poly(x,2),data=species,
        sigma.fo=~x, family=fam[i], n.cyc=60, trace=FALSE))}
unlist(m.ql)
```

```
##      DPO      NBI     NBII      PIG      DEL   SICHEL
## 626.4056 614.9565 615.1250 612.3667 614.6059 613.7327
```

Modelling $\log(\sigma)$ as a linear function of `x` improves the AIC for all models. The `PIG` model is still the 'best'. Since the Sichel and the Delaporte distributions have three parameters we will try to model the predictor of the third parameter $\nu$ as a linear function of `x`. The Sichel uses the `identity` as the default link for $\nu$ while the Delaporte uses the `logit`.

```
fam<-c("DEL", "SICHEL")
m.qll<-list()
for (i in 1:2) {
m.qll[[fam[i]]]<-GAIC(gamlss(fish~poly(x,2),data=species,
                    sigma.fo=~x, nu.fo=~x, family=fam[i], n.cyc=60,
                    trace=FALSE))}
unlist(m.qll)
```

```
##      DEL   SICHEL
## 614.7376 611.6346
```

Modelling the predictor of $\nu$ as a linear function of `x` improves the Sichel model (which now has a lower AIC than the `PIG` model) but not the Delaporte model. A further simplification of the Sichel model can be achieved by dropping the linear term in `x` for the $\log(\sigma)$ model which does not contribute anything to the fit (at least according to the AIC):

```
mSI<-gamlss(fish~poly(x,2),data=species, sigma.fo=~1, nu.fo=~x,
            family=SICHEL, n.cyc=60, trace=FALSE)
GAIC(mSI)
```

```
## [1] 609.7268
```

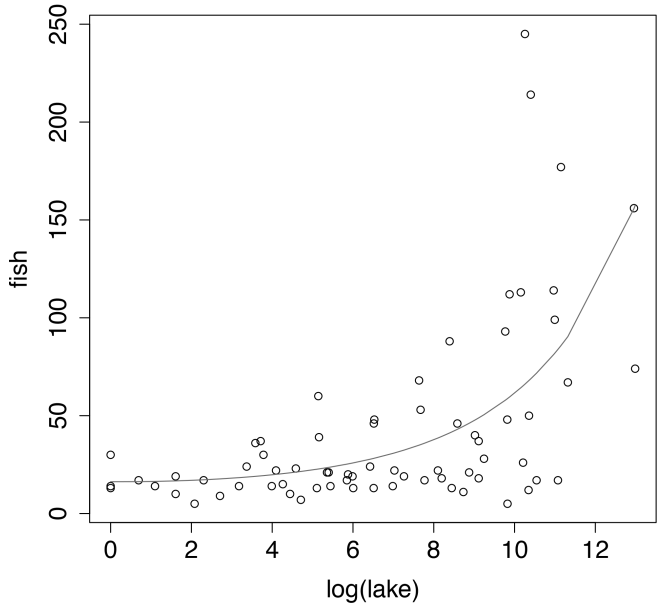
```
plot(fish~log(lake), data=species)
lines(species$x[order(species$lake)],   fitted(mSI)[order(
      species$lake)], col="red")
```

Figure 15

The fitted $\mu$ model together with the data are shown in Figure 15. Figures 16(a) and 16(b) give the fitted distribution of the number of fish species for observation 7, with lake area of 44 $\text{km}^2$, i.e. $x = \log(44) = 3.74$, and $(\hat{\mu}, \hat{\sigma}, \hat{\nu}) = (19.37, 1.44, -7.18)$, and observation 68, with lake area 9,065 $\text{km}^2$, i.e. $x = \log(9065) = 9.11$ and $(\hat{\mu}, \hat{\sigma}, \hat{\nu}) = (48.86, 1.44, -1.10)$, respectively. Note that the vertical scale is different for the two plots in Figure 16.

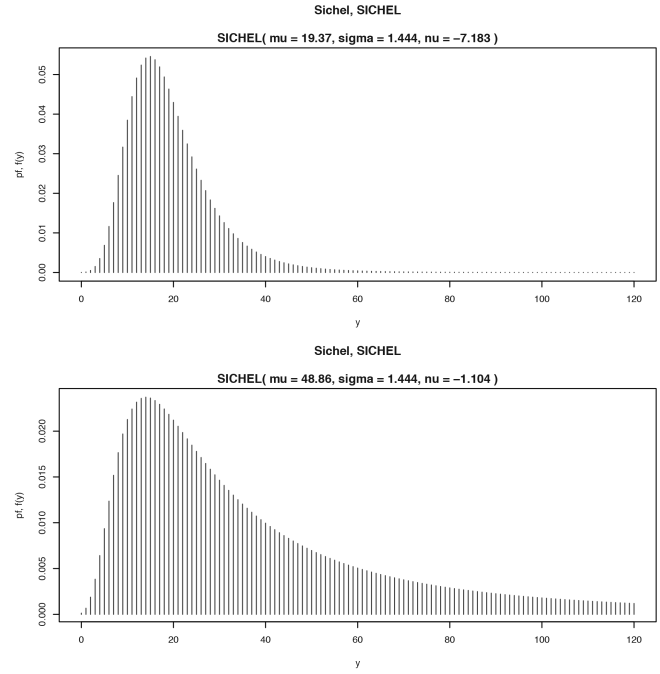
```
pdf.plot(mSI,c(7,68), min=0, max=120, step=1)
```

Table 1 (effectively Table 2 from **?**), gives GDEV, AIC and SBC for specific models fitted to the fish species data, and is used to answer the questions at the start of this section. The terms 1,




**R** code on page 38

Figure 15: Fitted mean number of fish species against log lake area.




**R** code on page 38

Figure 16: Fitted Sichel distributions for observations (a) 7 and (b) 68.

`x` and `x<2>` indicate constant, linear and quadratic terms, respectively, while the term `cs(x,3)` indicates a cubic smoothing spline with three degrees of freedom on top of the linear term `x`. Table 1 includes additional distributions to those previously fitted.

The following four paragraphs are taken from Rigby *et al.* (2008). "Comparing models 2, 3 and 4 indicates that a quadratic model for log $\mu$ is found to be adequate (while the linear and the cubic spline models were found to be inappropriate here). Comparing model 1 and 3 indicates that $Y$ has a highly overdispersed Poisson distribution. Comparing model 3 with models 5 and 6 shows that either a linear model in `x` for $\log(\sigma)$ or a different variance-mean relationship from that of the negative binomial (`NBI`) [i.e. $V[Y] = \mu + \sigma\mu^2$] is required. In particular the estimated $\nu$ parameter in the negative binomial family (`NBF`) of model 6 is $\hat{\nu} = 2.9$ suggesting a possible variance-mean relationship $V[Y] = \mu + \sigma\mu^3$. Modelling $\sigma$ in the `NBF` did not improve the fit greatly, as shown by model 7."

"A search of alternative mixed Poisson distributions included the Poisson-inverse Gaussian (`PIG`), the Sichel (`SI`) and the Delaporte (`DEL`). The models with the best AIC for each distribution were recorded" in Table 1 models 8 to 11. "A normal random effect mixture distribution was fitted" (using 20 Gaussian quadrature points) "to the Poisson and NBI conditional distributions giving models 12 and 13, i.e. Poisson-Normal and NBI-Normal, respectively. 'Non-parametric' random effects (effectively finite mixtures) (NPFM) were also fitted to Poisson and `NBI` conditional distributions giving models 14 and 15", i.e. `PO-NPFM(6)` and `NB-NPFM(2)` with 6 and 2 components, respectively. "Efron's double exponential (Poisson) distribution was fitted giving model 16" (DPO). "The best discretized continuous distribution fitted was a discrete inverse Gaussian distribution giving model 17 (`IGdisc`), again suggesting a possible cubic variance-mean relationship." Note that the Table 1 model 14 gives results for model `PO-NPFM(6)` instead of `PO-NPFM(5)` in Table 2 of Rigby *et al.* (2008).

"Overall the best model according to Akaike information criterion (AIC) is model 9, the Sichel model, followed closely by model 11, a Delaporte model. According to the Schwarz Bayesian criterion (SBC) the best model is model 17, the discretized inverse Gaussian distribution, again followed closely by model 11." In model 11, $\sigma$ was fixed to 1.

The following code reproduces the results of Table 1.

```
library(gamlss.mx)
m1 <- gamlss(fish~poly(x,2), data=species, family=PO, trace=FALSE)
m2 <- gamlss(fish~x, data=species, family=NBI, trace=FALSE)
m3 <- gamlss(fish~poly(x,2), data=species, family=NBI, trace=FALSE)
m4 <- gamlss(fish~cs(x,3), data=species, family=NBI, trace=FALSE)
m5 <- gamlss(fish~poly(x,2), sigma.fo=~x, data=species, family=NBI,
             trace=FALSE)
m6 <- gamlss(fish~poly(x,2), sigma.fo=~1, data=species, family=NBF,
            n.cyc=200, trace=FALSE)
m7 <- gamlss(fish~poly(x,2), sigma.fo=~x, data=species, family=NBF,
            n.cyc=100,  trace=FALSE)
m8 <- gamlss(fish~poly(x,2),  data=species, family=PIG, trace=FALSE)
m9 <- gamlss(fish~poly(x,2), nu.fo=~x, data=species, family=SICHEL,
            trace=FALSE)
m10 <- gamlss(fish~poly(x,2), nu.fo=~x, data=species, family=DEL,
             n.cyc=50, trace=FALSE)
m11 <- gamlss(fish~poly(x,2), nu.fo=~x, data=species, family=DEL,
```

```
          sigma.fix=TRUE, sigma.start=1, n.cyc=50, trace=FALSE)
m12 <- gamlssNP(fish~poly(x,2), data=species, mixture = "gq", K=20,
          family=PO, control=NP.control(trace=FALSE))
m13 <- gamlssNP(fish~poly(x,2), sigma.fo=~x, data=species,
          mixture = "gq", K=20, family=NBI,
          control=NP.control(trace=FALSE))
m14 <- gamlssNP(fish~poly(x,2), data=species, mixture = "np", K=6,
          tol=0.1,family=PO, control=NP.control(trace=FALSE))
m15 <- gamlssNP(fish~poly(x,2), data=species, mixture = "np", K=2,
          family=NBI, control=NP.control(trace=FALSE))
m16 <- gamlss(fish~poly(x,2), nu.fo=~x, data=species, family=DPO,
          trace=FALSE)
library(gamlss.cens)
m17 <- gamlss(Surv(fish,fish+1,type= "interval2")~x+I(x^2),
          sigma.fo=~1, data=species,
          family=cens(IG, type="interval"), trace=FALSE)
GAIC(m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14,
          m15, m16, m17)

##           df       AIC
## m9   6.00000  609.7268
## m11  5.00000  610.6493
## m17  4.00000  611.2793
## m10  6.00000  612.6593
## m5   5.00000  614.9565
## m13  6.00000  615.7281
## m6   5.00000  616.0828
## m7   6.00000  616.9229
## m8   4.00000  621.3459
## m3   4.00000  622.3173
## m14 13.00000  622.8926
## m12  4.00000  623.2455
## m15  6.00000  623.8794
## m4   5.99924  623.9083
## m2   3.00000  625.8443
## m16  4.00000  655.2520
## m1   3.00000 1855.2965

GAIC(m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14,
         m15, m16, m17, k=log(70))

##           df       AIC
## m17  4.00000  620.2733
## m11  5.00000  621.8918
## m9   6.00000  623.2178
## m10  6.00000  626.1503
## m5   5.00000  626.1990
## m6   5.00000  627.3253
## m13  6.00000  629.2191
```

Table 1: Comparison of models for the fish species data

| Model | Response distribution | $\mu$ | $\sigma$ | $\nu$ | GDEV | df | AIC | SBC |
|---|---|---|---|---|---|---|---|---|
| 1 | PO | x<2> | - | - | 1849.3 | 3 | 1855.3 | 1862.0 |
| 2 | NBI | x | 1 | - | 619.8 | 3 | 625.8 | 632.6 |
| 3 | NBI | x<2> | 1 | - | 614.3 | 4 | 622.3 | 631.3 |
| 4 | NBI | cs(x,3) | 1 | - | 611.9 | 6 | 623.9 | 637.4 |
| 5 | NBI | x<2> | x | - | 605.0 | 5 | 615.0 | 626.2 |
| 6 | NB family | x<2> | 1 | 1 | 606.1 | 5 | 616.1 | 627.3 |
| 7 | NB family | x<2> | x | 1 | 604.9 | 6 | 616.9 | 630.4 |
| 8 | PIG | x<2> | 1 | - | 613.3 | 4 | 621.3 | 630.3 |
| 9 | SICHEL | x<2> | 1 | x | 597.7 | 6 | 609.7 | 623.2 |
| 10 | DEL | x<2> | 1 | x | 600.7 | 6 | 612.7 | 626.2 |
| 11 | DEL | x<2> | - | x | 600.6 | 5 | 610.6 | 621.9 |
| 12 | PO-Normal | x<2> | 1 | - | 615.2 | 4 | 623.2 | 632.2 |
| 13 | NBI-Normal | x<2> | x | 1 | 603.7 | 6 | 615.7 | 629.2 |
| 14 | PO-NPFM(6) | x<2> | - | — | 596.9 | 13 | 622.9 | 652.1 |
| 15 | NB-NPFM(2) | x<2> | 1 | — | 611.9 | 6 | 623.9 | 637.4 |
| 16 | DPO | x<2> | x | - | 647.3 | 5 | 655.3 | 664.2 |
| 17 | IGdisc | x<2> | 1 | - | 603.3 | 4 | 611.3 | 620.3 |

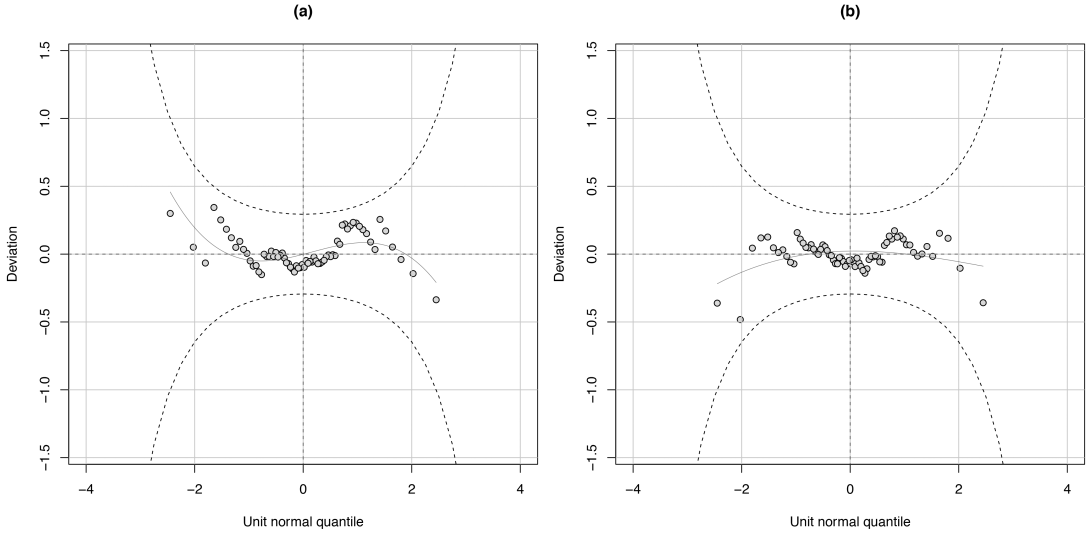
```
## m8   4.00000  630.3399
## m7   6.00000  630.4138
## m3   4.00000  631.3113
## m12  4.00000  632.2395
## m2   3.00000  632.5898
## m15  6.00000  637.3704
## m4   5.99924  637.3975
## m14 13.00000  652.1230
## m16  4.00000  664.2460
## m1   3.00000 1862.0420
```

```
wp(m9) ; title("(a)")
wp(m11); title("(b)")
```

Figure 17

The 'best' fitted models are `m9` and `m17`, as suggested by AIC and SBC, respectively, Their worm plots are shown in Figure 17, indicating that both models have adequate fits. The fitted parameters of the Sichel model `m9` are shown below. They are obtained by refitting the model using an ordinary quadratic polynomial in `x` for $\log(\mu)$, rather than the orthogonal quadratic polynomial produced by `poly(x,2)`:

```
mSI<- gamlss(fish~x+I(x^2), sigma.fo=~1, nu.fo=~x, data=species,
             family=SICHEL, trace=FALSE)
summary(mSI)

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




**R** code on page 42

Figure 17: Worm plots for the chosen model (a) `m9` using AIC and (b) `m17` using SBC.

```
## Call:
## gamlss(formula = fish ~ x + I(x^2), sigma.formula = ~1,
##     nu.formula = ~x, family = SICHEL, data = species,
##     trace = FALSE)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  log
## Mu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.788203   0.171613  16.247   <2e-16 ***
## x           -0.006376   0.066870  -0.095   0.9243
## I(x^2)       0.013957   0.005503   2.536   0.0137 *
## ---
## 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)   0.3674     0.4632   0.793    0.431
##
## ------------------------------------------------------------------
## Nu link function:  identity
## Nu Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.5009     3.1110  -3.697 0.000455 ***
## x             1.1410     0.3249   3.512 0.000822 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  70
## Degrees of Freedom for the fit:  6
##       Residual Deg. of Freedom:  64
##                       at cycle:  7
##
## Global Deviance:     597.7268
##             AIC:     609.7268
##             SBC:     623.2178
## ******************************************************************
```

# 16 Victims of crime.

The `VictimsOfCrime` data were introduced in the lecture.

**R data file:** `VictimsOfCrime` in package **gamlss.data** of dimensions $10590 \times 2$

**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 $\mu$ (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)
```