Inflated distributions on the interval $[0, 1]$

Mikis Stasinopoulos, Marco Enea,
Roberd A. Rigby, and Abu Hossain

April 29, 2017

Contents

1 Introduction

The new R package `gamlss.inf` is designed to fit inflated distributions on the interval $[0, 1]$ for a response variable. This allows the fitting of a response variable with a mixed (i.e continuous and discrete) distribution which is continuous within the interval $(0, 1)$, with additional point probabilities at either 0 or 1 or both 0 and 1. The **gamlss** package, Rigby and Stasinopoulos [2005], Stasinopoulos and Rigby [2007] already provides the inflated beta distributions, `BEINF`, `BEINF0` and `BEINF1` which allow the user to fit a beta distribution on $(0, 1)$, with extra point probabilities at 0 or 1 or both 0 and 1. The probabilities at the points 0 and 1 may depend on explanatory variables. Since the beta distribution has 2 parameters, the inflated beta, `BEINF`, (with the addition of the two probabilities at 0 and 1) has a total of 4 parameters. In practice, and for complicated data sets, the part of the response which lies on $(0, 1)$ may need more than 2 distribution parameters to be captured correctly.

There are three methods within the **gamlss** packages to obtain a more flexible distribution on the range $(0, 1)$, not including zero or one:

1. use a flexible explicit distribution within the **gamlss.dist** package, e.g. `GB1`(μ, σ, ν, τ); see Section 2.1
2. create a flexible distribution on $(0, 1)$ by transforming any continuous distribution with range $(-\infty, \infty)$ available in the `gamlss.dist` package to range $(0, 1)$, using an inverse logit transformation, through the function `gen.family()`; see Section 2.2
3. create a flexible distribution on $(0, 1)$ by truncating any continuous distribution on either $(-\infty, \infty)$ or $(0, \infty)$ available in `gamlss.dist` to range $(0, 1)$, through the function `gen.trun()`; see Section 2.3.

The new **R** package **gamlss.inf** enhances this capability of the **gamlss** packages in that a distribution on $(0, 1)$ (with up to four parameters, obtained from any of the three methods above) can be inflated with probabilities at 0 and/or 1. The overall distribution can then have up to six parameters. Let $\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}$ represent the four parameters of the the distribution defined on $(0, 1)$ and $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_1$ be parameters related to the probabilities at 0 and 1. Then the general inflated $[0, 1]$ model that the new package `gamlss.inf` can fit can be written as:

$$
\begin{aligned}
Y &\overset{\text{ind}}{\sim} \mathcal{D}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}, \boldsymbol{\xi}_0, \boldsymbol{\xi}_1) \\
\boldsymbol{\eta}_1 &= g_1(\boldsymbol{\mu}) = \mathbf{X}_1\boldsymbol{\beta}_1 + s_{11}(\mathbf{x}_{11}) + \ldots + s_{1J_1}(\mathbf{x}_{1J_1}) \\
\boldsymbol{\eta}_2 &= g_2(\boldsymbol{\sigma}) = \mathbf{X}_2\boldsymbol{\beta}_2 + s_{21}(\mathbf{x}_{21}) + \ldots + s_{2J_2}(\mathbf{x}_{2J_2}) \\
\boldsymbol{\eta}_3 &= g_3(\boldsymbol{\nu}) = \mathbf{X}_3\boldsymbol{\beta}_3 + s_{31}(\mathbf{x}_{31}) + \ldots + s_{3J_3}(\mathbf{x}_{3J_3}) \\
\boldsymbol{\eta}_4 &= g_4(\boldsymbol{\tau}) = \mathbf{X}_4\boldsymbol{\beta}_4 + s_{41}(\mathbf{x}_{41}) + \ldots + s_{4J_4}(\mathbf{x}_{4J_4}) \\
\boldsymbol{\eta}_5 &= g_5(\boldsymbol{\xi}_0) = \mathbf{X}_5\boldsymbol{\beta}_5 + s_{51}(\mathbf{x}_{51}) + \ldots + s_{5J_5}(\mathbf{x}_{5J_5}) \\
\boldsymbol{\eta}_6 &= g_6(\boldsymbol{\xi}_1) = \mathbf{X}_6\boldsymbol{\beta}_6 + s_{61}(\mathbf{x}_{61}) + \ldots + s_{6J_6}(\mathbf{x}_{6J_6})
\end{aligned}
\tag{1}
$$

where $\mathcal{D}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}, \boldsymbol{\xi}_0, \boldsymbol{\xi}_1)$ is a zero and one inflated distribution of the response variable Y, defined on $[0, 1]$ including 0 and 1, where $\mathbf{X}_k$ are the design matrices incorporating the linear additive terms in the model, $\boldsymbol{\beta}_k$ are the linear coefficient parameters and $s_{kj}(\mathbf{x}_{kj})$ represent smoothing functions for explanatory variables $\mathbf{x}_{kj}$, for $k = 1, 2, 3, 4, 5, 6$ and $j = 1, \ldots, J_k$. Note that the quantitative explanatory variables in the $\mathbf{X}$'s can be the same or different for the ones

defined in the smoothers. The vectors $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$, $\boldsymbol{\eta}_3$, $\boldsymbol{\eta}_4$ $\boldsymbol{\eta}_5$ and $\boldsymbol{\eta}_6$ are called the *predictors* of the distribution parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\nu}$, $\boldsymbol{\tau}$, $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_1$ respectively. Models of the type in equation (1) are discussed in Hossain et al. [2016a] and Hossain et al. [2016b].

2 Distributions on $(0, 1)$

2.1 Explicit distributions on $(0, 1)$

Within the `gamlss.dist` package there are currently three distributions defined on $(0, 1)$,

1. the beta distribution, `BE`, with two parameters,
2. the logit normal distribution, `LOGITNO`, with two parameters and
3. the generalised beta type 1 distribution, `GB1`, with four parameters.

2.2 Logit transform distributions on $(0, 1)$

In addition, any continuous random variable say Z defined on $(-\infty, \infty)$ can be transformed by the inverse logit transformation $Y = 1/(1+\exp(-Z))$ to a random variable Y defined on $(0, 1)$. For example, if Z is a t-family distributed variable, i.e. $Z \sim \texttt{TF}(\mu, \sigma, \nu)$, and the inverse logit transformation is applied, then $Y \sim \texttt{logitTF}(\mu, \sigma, \nu)$, i.e. a logit-$t$ family distribution on $(0, 1)$.

The following is an example on how to take a `gamlss.family` distribution on $(-\infty, \infty)$ and create a corresponding logit distribution defined on $(0, 1)$. The gamlss function `gen.Family()` of the **gamlss.dist** package generates the `d` (pdf), `p` (cdf), `q` (inverse cdf) and `r` (random generation) functions of the distribution together with the function which can be used for fitting within **gamlss**. Here first generate a logit-t distribution and plot the distribution for different values of μ, σ and ν. Note that μ, σ and ν are defined on the original t-distribution ranges $(-\infty, \infty)$ for μ and $(0, \infty)$ for σ and ν. This implies that $1/(1+\exp(-\mu))$ is not the mean of the logit distribution, `logitTF`, but its median. Also σ and ν are related to the scale and shape of the distribution. Below we use `gen.Family("TF", type="logit")` to generate a logit-t distribution and in Figure 1 we plot the distribution for different values of μ, σ and ν using the function `curve()`.

```
# generate the distribution
library(gamlss)
gen.Family("TF", type="logit")

## A  logit  family of distributions from TF has been generated
##  and saved under the names:
##  dlogitTF plogitTF qlogitTF rlogitTF logitTF

# different mu
curve(dlogitTF(x, mu=-5, sigma=1, nu=10), 0,1, ylim=c(0,3), lwd=3, lty=2, col=2)
title("(a)")
curve(dlogitTF(x, mu=-1, sigma=1, nu=10), 0,1, add=TRUE, lwd=3, lty=3, col=3)
curve(dlogitTF(x, mu=0,  sigma=1, nu=10), 0,1, add=TRUE, lwd=3, lty=4, col=4)
curve(dlogitTF(x, mu=1,  sigma=1, nu=10), 0,1, add=TRUE, lwd=3, lty=5, col=5)
```

Figure 1

```
curve(dlogitTF(x, mu=5,  sigma=1, nu=10), 0,1, add=TRUE, lwd=3, lty=6, col=6)
legend("top",
       legend=c("mu = -5","mu = -1","mu = 0","mu = 1","mu = 5"),
       lty=2:6, col=2:6, cex=1, lwd=3)
# different sigma
curve(dlogitTF(x, mu=0, sigma=.5, nu=10), 0,1, ylim=c(0,3), lwd=3, lty=2, col=2)
title(("(b)"))
curve(dlogitTF(x, mu=0, sigma=1, nu=10), 0,1, add=TRUE, lwd=3,lty=3, col=3)
curve(dlogitTF(x, mu=0, sigma=2, nu=10), 0,1, add=TRUE, lwd=3,lty=4, col=4)
curve(dlogitTF(x, mu=0, sigma=5, nu=10), 0,1, add=TRUE, lwd=3,lty=5, col=5)
legend("topleft",
       legend=c("sigma = .5","sigma = 1","sigma = 2","sigma = 5"),
       lty=2:5, col=2:5, cex=1, lwd=3)
# different nu
curve(dlogitTF(x, mu=0, sigma=1, nu=1000), 0,1, ylim=c(0,3), lwd=3, lty=2, col=2)
title("(c)")
curve(dlogitTF(x, mu=0, sigma=1, nu=10), 0,1, add=TRUE, lwd=3, lty=3, col=3)
curve(dlogitTF(x, mu=0, sigma=1, nu=5 ), 0,1, add=TRUE, lwd=3,lty=4, col=4)
curve(dlogitTF(x, mu=0, sigma=1, nu=1 ), 0,1, add=TRUE, lwd=3,lty=5, col=5)
legend("top",
       legend=c("nu = 1000","nu = 10","nu = 5","nu = 1"),
       lty=2:5, col=2:5, cex=1, lwd=3)
```

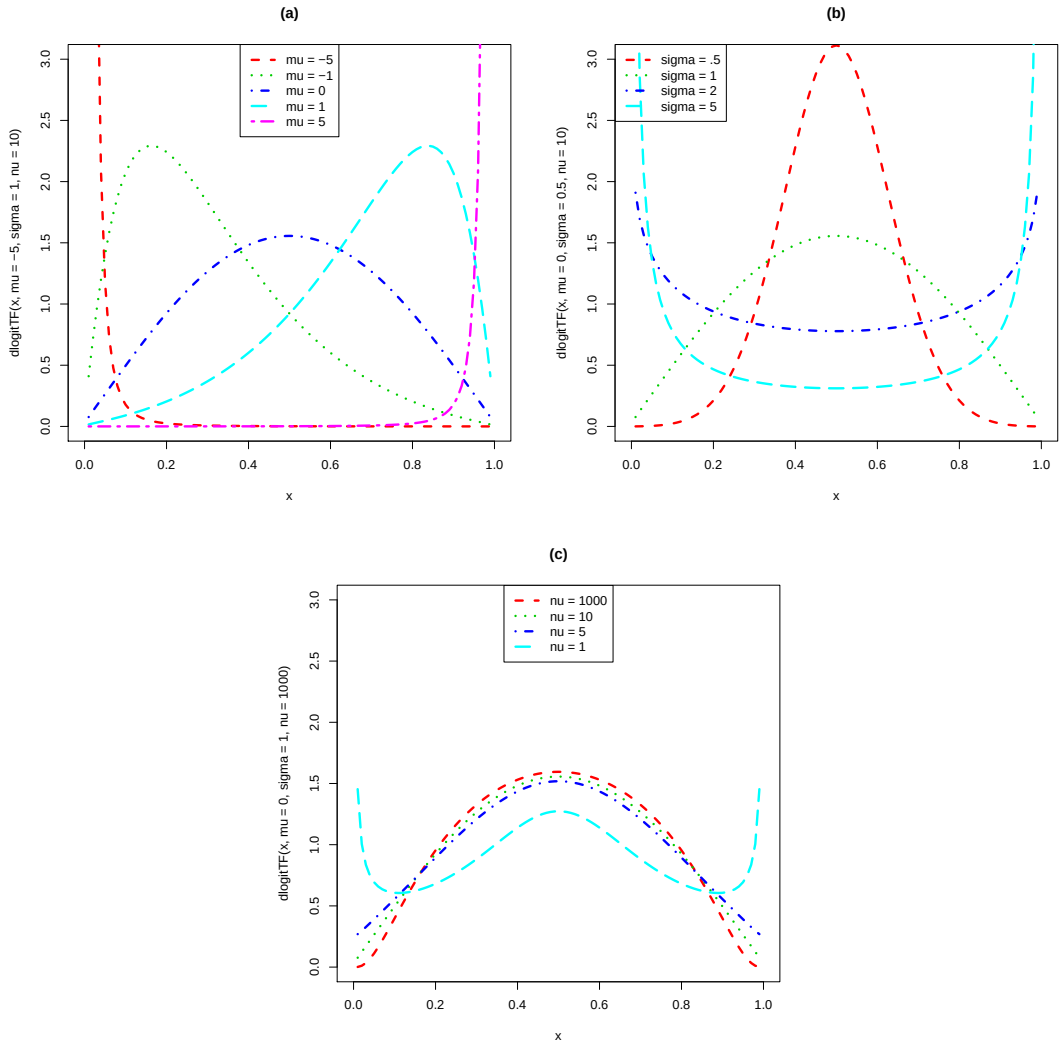
Figure 1 shows the different shapes the distribution can take. Panel (a) shows for fixed $\sigma = 1$ and $\nu = 10$ how the distribution changes for different values of $\mu = (-5, -1, 0, 1, 5)$. Panel (b) for fixed $\mu = 0$ and $\nu = 10$ varies $\sigma = (0.5, 1, 2, 5)$. Finally panel (c) fixes $\mu = 0$ and $\sigma = 1$ and varies $\nu = (1, 5, 10, 1000)$.

2.3 Truncated distributions on (0,1)

Any distribution defined on the real line $(-\infty, \infty)$ can be left truncated at 0 and right truncated at 1 to give a truncated distribution on (0,1) using the function `gen.trun()` from the `R` package **gamlss.tr**.

```
# generate the distribution
library(gamlss.tr)
gen.trun(c(0,1),"SST",type="both")

## A truncated family of distributions from SST has been generated
##  and saved under the names:
##  dSSTtr pSSTtr qSSTtr rSSTtr SSTtr
## The type of truncation is both
##  and the truncation parameter is 0 1

# different mu
curve(dSSTtr(x, mu=.1, sigma=.1, nu=1, tau=10), 0,1, lwd=3, lty=2, col=2)
curve(dSSTtr(x, mu=.5, sigma=.1, nu=1, tau=10), 0,1, lwd=3, lty=3, col=3, add=TRUE)
curve(dSSTtr(x, mu=.9, sigma=.1, nu=1, tau=10), 0,1, lwd=3, lty=4, col=4, add=TRUE)
title("(a)")
```

Figure 2



R code on page 3

Figure 1: A logit-t distribution, `logitTF`(μ, σ, τ): (a) with values $\mu = (-5, -1, 0, 1, 5)$, $\sigma = 1$ and $\nu = 10$, (b) with values $\mu = 0$, $\sigma = (0.5, 1, 2, 5)$ and $\nu = 10$ and (c) with values $\mu = 0$, $\sigma = 1$ and $\nu = (1, 5, 10, 1000)$.

```
legend("top",
       legend=c("mu = .1","mu = .5","mu = .9"),
       lty=2:4, col=2:4, cex=1, lwd=3)
# different sigma
curve(dSSTtr(x, mu=.5, sigma=.1, nu=1, tau=10), 0,1, lwd=3, lty=2, col=2)
curve(dSSTtr(x, mu=.5, sigma=.2, nu=1, tau=10), 0,1, lwd=3, lty=3, col=3, add=TRUE)
curve(dSSTtr(x, mu=.5, sigma=.5, nu=1, tau=10), 0,1, lwd=3, lty=4, col=4, add=TRUE)
title("(b)")
legend("topleft",
       legend=c("sigma = .1","sigma = .2","sigma = .5"),
       lty=2:4, col=2:4, cex=1, lwd=3)
# different nu
curve(dSSTtr(x, mu=.5, sigma=.2, nu=.1, tau=10), 0,1, lwd=3, lty=2, col=2)
curve(dSSTtr(x, mu=.5, sigma=.2, nu=1,  tau=10), 0,1, lwd=3, lty=3, col=3, add=TRUE)
curve(dSSTtr(x, mu=.5, sigma=.2, nu=2,  tau=10), 0,1, lwd=3, lty=4, col=4, add=TRUE)
title("(c)")
legend("topleft",
       legend=c("nu = .1","nu = 2","nu = .5"),
       lty=2:4, col=2:4, cex=1, lwd=3)
# different tau
curve(dSSTtr(x, mu=.5, sigma=.2, nu=1, tau=3 ), 0,1, lwd=3, lty=2, col=2)
curve(dSSTtr(x, mu=.5, sigma=.2, nu=1, tau=4) , 0,1, lwd=3, lty=3, col=3, add=TRUE)
curve(dSSTtr(x, mu=.5, sigma=.2, nu=1, tau=10), 0,1, lwd=3, lty=4, col=4, add=TRUE)
title("(d)")
legend("topleft",
       legend=c("tau = 3","tau = 4","tau = 10"),
       lty=2:4, col=2:4, cex=1, lwd=3)
```

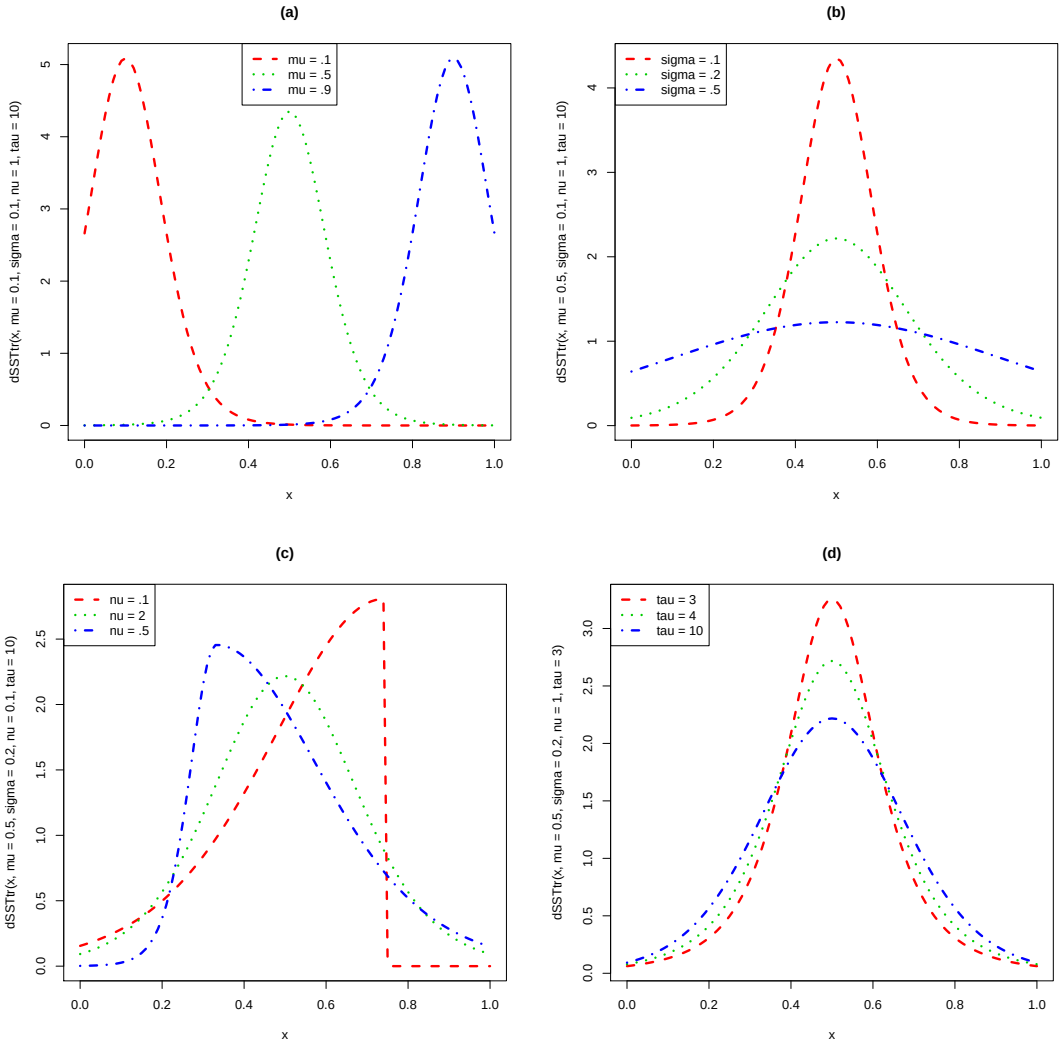
Similarly any distribution defined on the positive real line $(0, \infty)$ can be right truncated at one to give a truncated distribution on $(0, 1)$ using the function `gen.trun()`.

3 Generating inflated distributions on $[0, 1]$

Next it is shown how any continuous `gamlss.family` distribution defined on $(0, 1)$, obtained by any of the three methods of Section 2, can be extended by inflation to range $[0, 1]$ by including point probabilities at 0 and/or 1.

The function `gen.Inf0to1()` takes as an argument a `gamlss.family` distribution on $(0, 1)$ and generates an inflated version of the distribution with point probabilities at 0 and/or 1. The function has two arguments, `family` and `type.of.Inflation`. The first specifies a distribution family on $(0, 1)$, while the second specifies the type of inflation. The options are i) `"Zero"`, `"One"` and `"Zero&One"`.

The resulting mixed continuous-discrete probability (density) function (pdf) for option `"Zero&One"`



R code on page 4

Figure 2: A truncated SST distribution, $\texttt{SSTtr}(\mu, \sigma, \nu, \tau)$: (a) with values $\mu = (0.1, 0.5, 0.9)$, $\sigma = 0.1$ $\nu = 1$ and $\tau = 10$, (b) with values $\mu = 0.5$, $\sigma = (0.1, 0.2, 0.5)$, $\nu = 1$ and $\tau = 10$ (c) with values $\mu = 0.5$, $\sigma = 0.2$, $\nu = (0.1, 1, 2)$ and $\tau = 10$. (d) with values $\mu = 0.5$, $\sigma = 0.2$, $\nu = 1$ and $\tau = (3, 4, 10)$

is given by:

$$f_Y(y|\boldsymbol{\theta},\xi_0,\xi_1) = \begin{cases} p_0 & \text{if } y = 0 \\ (1-p_0-p_1) f_W(y|\boldsymbol{\theta}) & \text{if } 0 < y < 1 \\ p_1 & \text{if } y = 1 \end{cases} \tag{2}$$

for $0 \leq y \leq 1$, where $f_W(y|\boldsymbol{\theta})$ is any probability density function defined on $(0,1)$, i.e. for $0 < y < 1$, with up to four parameters in $\boldsymbol{\theta}^T = (\theta_1, \theta_2, \theta_3, \theta_4)$ and $0 < p_0 < 1$, $0 < p_1 < 1$ and $0 < p_0 + p_1 < 1$ and where $\xi_0 = p_0/p_2$, $\xi_1 = p_1/p_2$, where $p_2 = 1 - p_0 - p_1$, so $\xi_0 > 0$ and $\xi_1 > 0$. Hence

$$\begin{pmatrix} p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} \frac{\xi_0}{(1+\xi_0+\xi_1)} \\ \frac{\xi_1}{(1+\xi_0+\xi_1)} \end{pmatrix} \tag{3}$$

The default link functions for ξ_0 and ξ_1 in (2) are $\eta_5 = \log(\xi_0)$ and $\eta_6 = \log(\xi_1)$.

However for option `"Zero"` the pdf is

$$f_Y(y|\boldsymbol{\theta},\xi_0,\xi_1) = \begin{cases} \xi_0 & \text{if } y = 0 \\ (1-\xi_0) f_W(y|\boldsymbol{\theta}) & \text{if } 0 < y < 1 \end{cases} \tag{4}$$

so in this case $\xi_0 = P(Y = 0)$. The default link function for ξ_0 in (4) is $\eta_5 = \log\left[\xi_0/\left(1-\xi_0\right)\right]$.

Also for option `"One"` the pdf is

$$f_Y(y|\boldsymbol{\theta},\xi_1,\xi_1) = \begin{cases} (1-\xi_1) f_W(y|\boldsymbol{\theta}) & \text{if } 0 < y < 1 \\ \xi_1 & \text{if } y = 1 \end{cases} \tag{5}$$

so in this case $\xi_1 = P(Y = 1)$. The default link function for ξ_1 in (5) is $\eta_5 = \log\left[\xi_1/\left(1-\xi_1\right)\right]$.

Note that, in the **gamlss.inf** implementation, $\boldsymbol{\theta}$ has at most 4 parameters, $\boldsymbol{\theta}^T = (\mu, \sigma, \nu, \tau)$.

In the example below first take the skew t-family distribution, `SST`, and use the `gen.Family()` function in the **gamlss.dist** package to generate the distribution `logitSST` defined on $(0,1)$. By using the function `gen.Inf0to1()` on the new generated `logitSST` distribution, an inflated logitSST distribution, inflated at 0 and 1, is created.

```
library(gamlss.inf)
gen.Family(family="SST", type="logit")

## A  logit  family of distributions from SST has been generated
##  and saved under the names:
##  dlogitSST plogitSST qlogitSST rlogitSST logitSST

gen.Inf0to1(family="logitSST",  type.of.Inflation="Zero&One")

## A  0to1 inflated logitSST distribution has been generated
##  and saved under the names:
##  dlogitSSTInf0to1 plogitSSTInf0to1 qlogitSSTInf0to1 rlogitSSTInf0to1
##  plotlogitSSTInf0to1
```

There are five functions generated here for the `logitSST` distribution inflated at 0 and 1:

`dlogitSSTInf0to1` The pdf of the distribution, `d` function.

`plogitSSTInf0to1` The cdf of the distribution, `p` function.

`qlogitSSTInf0to1` The inverse cdf of the distribution, `q` function.

`rlogitSSTInf0to1` The random generating function of the distribution, `r` function.

`plotlogitSSTInf0to1` The function for plotting the pdf of the distribution.

4 Plotting inflated distributions on $[0, 1]$

The newly created `plotlogitSSTInf0to1()` function can be used to plot the pdf of the inflated distribution (which is a mixed continuous-discrete distribution). Figure 3 shows the use of the `plotlogitSSTInf0to1()` function. The function plots the inflated distribution function including point probabilities at zero and one. Only one plot is allowed per figure. Figure 3 shows eight different realisations of the distribution for different values of the parameters.

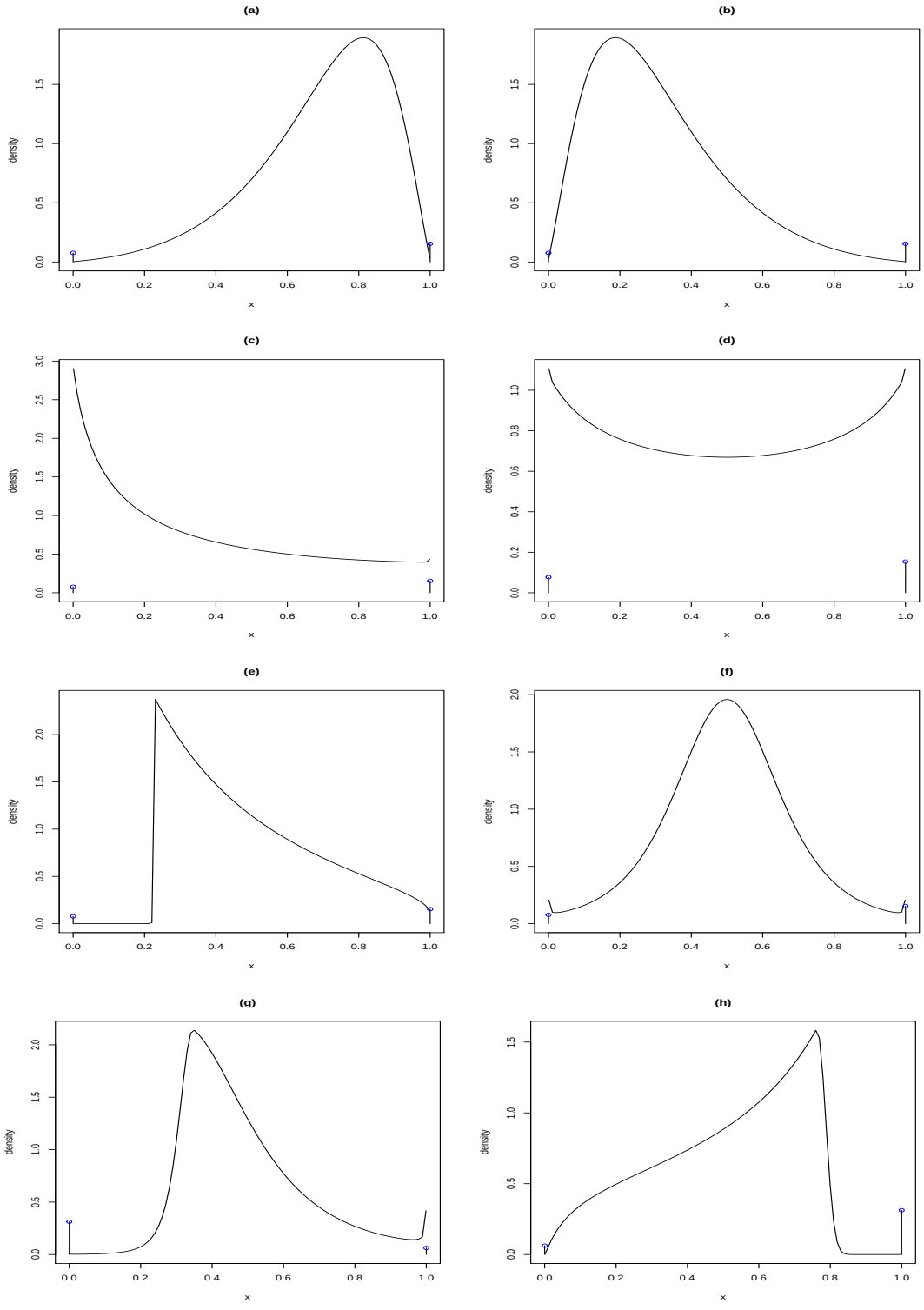
```
plotlogitSSTInf0to1(mu= 1, sigma=1, nu=1, tau=10, xi0=.1, xi1=.2); title("(a)")
plotlogitSSTInf0to1(mu=-1, sigma=1, nu=1, tau=10, xi0=.1, xi1=.2); title("(b)")
plotlogitSSTInf0to1(mu=-1, sigma=2, nu=1, tau=10, xi0=.1, xi1=.2); title("(c)")
plotlogitSSTInf0to1(mu=0,  sigma=2, nu=1, tau=10, xi0=.1, xi1=.2); title("(d)")
plotlogitSSTInf0to1(mu=0,  sigma=1, nu=10,tau=10, xi0=.1, xi1=.2); title("(e)")
plotlogitSSTInf0to1(mu=0,  sigma=1, nu=1, tau=3,  xi0=.1, xi1=.2); title("(f)")
plotlogitSSTInf0to1(mu=0,  sigma=1, nu=2, tau=3,  xi0=.5, xi1=.1); title("(g)")
plotlogitSSTInf0to1(mu=0,  sigma=1, nu=.3,tau=100,xi0=.1, xi1=.5); title("(h)")
```

Figure 3

The standard plotting functions of `R` can also be used to plot the created mixed distribution as is shown below. Figure 4 shows how the pdf, cdf, inverse cdf and randomisation functions can be displayed for different values of the distribution parameters.

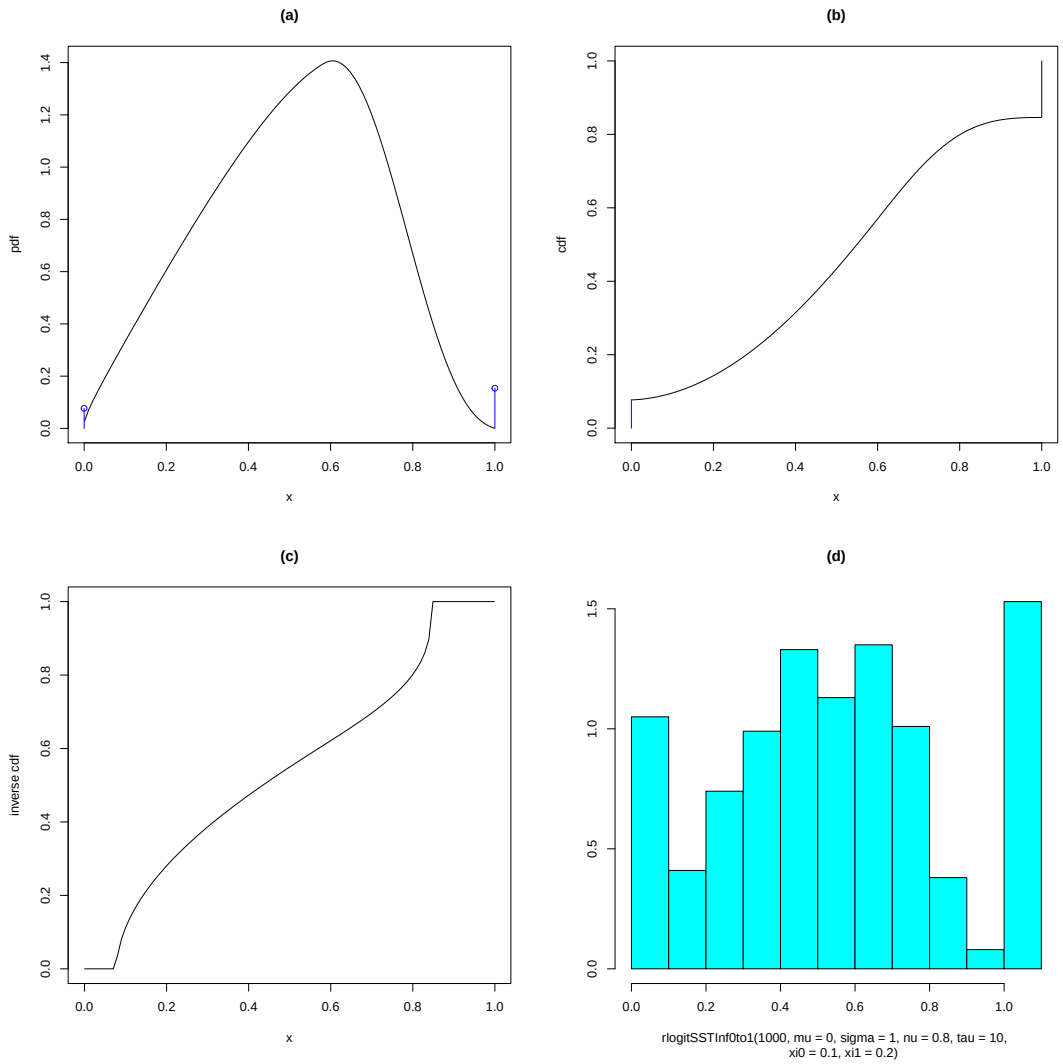
```
# plotting the pdf ------------------------------
curve(dlogitSSTInf0to1(x, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2),
      0.001,0.999, ylab="pdf",  main="(a)")
# getting the probabilities
p0 <- dlogitSSTInf0to1(x=0, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2)
p1 <- dlogitSSTInf0to1(x=1, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2)
points(c(0,1), c(p0,p1), col="blue")
lines(c(0,1), c(p0,p1), col="blue", type="h")
# plotting the cdf ------------------------------
curve(plogitSSTInf0to1(x, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2),
      0.0001,0.999, ylim=c(0,1), ylab="cdf", main="(b)")
#points(c(0), c(p0), col="blue")
lines(c(0), c(p0), col="blue", type="h")
p1 <- plogitSSTInf0to1(q=.999, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2)
lines(c(1,1),c(p1,1))
# plotting the inverse cdf ----------------------
curve(qlogitSSTInf0to1(x, mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2),
      0.0001,0.999, ylim=c(0,1), ylab="inverse cdf", main="(c)")
# plottind simulated data
truehist(rlogitSSTInf0to1(1000,mu=0, sigma=1, nu=.8, tau=10, xi0=.1, xi1=.2),
         main="(d)")
```

Figure 4



R code on page 9

Figure 3: A logit-SST distribution: (a) with values $\mu = 1$, $\sigma = 1$, $\nu = 1$, $\tau = 10$, $\xi_0 = 0.1$, and $\xi_1 = .2$ (b) with values $\mu = -1$, $\sigma = 1$, $\nu = 1$, $\tau = 10$, $\xi_0 = 0.1$, and $\xi_1 = 0.2$ (c) with values $\mu = -1$, $\sigma = 2$, $\nu = 1$, $\tau = 10$, $\xi_0 = 0.1$, and $\xi_1 = 0.2$ (d) with values $\mu = 0$, $\sigma = 2$, $\nu = 1$, $\tau = 10$, $\xi_0 = 0.1$, and $\xi_1 = 0.2$ (e) with values $\mu = 0$, $\sigma = 1$, $\nu = 10$, $\tau = 10$, $\xi_0 = 0.1$, and $\xi_1 = 0.2$ (f) with values $\mu = 0$, $\sigma = 1$, $\nu = 1$, $\tau = 3$, $\xi_0 = 0.1$, and $\xi_1 = 0.2$ (g) with values $\mu = 0$, $\sigma = 1$, $\nu = 2$, $\tau = 3$, $\xi_0 = 0.5$, and $\xi_1 = 0.1$ (e) with values $\mu = 0$, $\sigma = 1$, $\nu = 0.3$, $\tau = 100$, $\xi_0 = 0.1$, and $\xi_1 = 0.5$



R code on page 9

Figure 4: The (a) pdf (b) cdf (c) inverse pdf and (d) simulated data from a logit-SST distribution inflated at 0 and 1 with $\mu = 0$, $\sigma = 1$, $\nu = .8$, $\tau = 10$. $\xi_0 = .1$, and $\xi_1 = .2$

Note that in Figure 4(a) the probabilities p_0 and p_1 at 0 and 1, respectively, are given by equation (3), i.e. $p_0 = 0.077$ and $p_1 = 0.154$.

The next section demonstrates how to use the function `gamlssInf0to1()` in package **gamlss.inf** to fit a model which has a response variable on the interval $[0, 1]$ including 0 and/or 1.

5 Fitting a distributions on $[0, 1]$

5.1 The `gamlssInf0to1()` function

The main function for fitting a model with a response variable Y on the interval $[0, 1]$ including 0 and/or 1 is `gamlssInf0to1()`. In an inflated distribution the parameters μ, σ, ν and τ are orthogonal to the parameters ξ_0 and ξ_1 in the sense that the log-likelihood function can be factorised in two components, one containing μ, σ, ν and τ and another containing ξ_0 and ξ_1. This means that the two sets of parameters can be estimated separately. The function `gamlssInf0to1()` takes advantage of this separation and works as follows:

- It picks the argument `family` which defines a `gamlss.family` distribution defined on $(0, 1)$
- It checks the range of the response variable Y and depending on whether the range of the response variable Y is $[0, 1)$, $(0, 1]$ or $[0, 1]$, it creates an appropriate binary or multinomial response variable and it fits an appropriate GAMLSS model.
  - for $[0, 1)$ it fits a binary logistic model, using the `gamlss.family BI()`, to response $Y_1 = 1$ (if $Y = 0$) $+ 0$ (if $0 < Y < 1$)
  - for $(0, 1]$ it fits a binary logistic model, using the `gamlss.family BI()`, to response $Y_1 = 1$ (if $Y = 1$) $+ 0$ (if $0 < Y < 1$)
  - for $[0, 1]$ it fits a three level multinomial model, using the `gamlss.family MN3()` to response $Y_1 = 1$ (if $Y = 0$) $+ 2$ (if $Y = 1$) $+ 3$ (if $0 < Y < 1$)
- Fits a GAMLSS model to the data cases with Y inside $(0, 1)$ using the distribution defined by `family`, by weighting out the observations with zero and/or one.
- Creates the (normalised randomized) quantile residuals for the whole model
- Saves the output as a `gamlssinf0to1` object which is a subclass of a `gamlss object`.

The idea is that the object `gamlssinf0to1` should behave similar to a `gamlss` object. For this purpose the following S3 methods are created.

1. `fitted.gamlssinf0to1()`,
2. `coef.gamlssinf0to1()`,
3. `print.gamlssinf0to1()`,
4. `deviance.gamlssinf0to1()`,
5. `vcov.gamlssinf0to1()`,
6. `summary.gamlssinf0to1()`,
7. `predict.gamlssinf0to1()`,
8. `formula.gamlssinf0to1`.

The above methods are demonstrated in the next sections.

The function `gamlssInf0to1()` has the following arguments:

y the response variable on $[0, 1]$ (including values at zero and/or one)

mu.formula a model formula for the μ parameter

sigma.formula a model formula for the σ parameter

nu.formula a model formula for the ν parameter

tau.formula a model formula for the τ parameter

xi0.formula a model formula for the ξ_0 parameter which is related to the probability at zero

xi1.formula a model formula for the ξ_1 parameter which is related to the probability at one

data a data frame containing the variables occurring in the formula.

family any `gamlss()` distribution family defined on $(0, 1)$

weights a vector of prior weights as in `gamlss()`

trace logical, if TRUE information on model estimation will be printed during the fitting

... for extra arguments which can be passed to `gamlss()`.

Since the individual models fitted within the algorithm used in `gamlssInf0to1()` are GAMLSS models, the parameter formulae above can take any linear or additive GAMLSS terms inclining smoothers and random effects.

To demonstrate the use of the `gamlssInf0to1()` function, simulated examples are used below. In the examples there are no explanatory variables. That is, in Sections 5.3, 5.4 and 5.5 below, a response from different inflated distributions on $[0, 1)$, $(0, 1]$ and $[0, 1]$; respectively, is simulated and then a distribution is fitted to the response variable.

In Section 6 we use simulated data with one explanatory variable.

5.2 Simulating data

To compare the results obtained by the function `gamlssInf0to1()` to the ones obtained from standard `gamlss()` function, simulate data from the inflated beta distributions `BEINF0`, `BEINF1`, `BEINF` which generate data on $[0, 1)$, $(0, 1]$ and $[0, 1]$ respectively.

```
library(gamlss)     # loading gamlss package
library(gamlss.inf)
# creating  data
set.seed(324)
 y0 <- rBEINF0(1000, mu=.3, sigma=.3, nu=.15)# p0=0.13
 y1 <- rBEINF1(1000, mu=.3, sigma=.3, nu=.15)# p1=0.13
y01 <- rBEINF(1000, mu=.3, sigma=.3, nu=0.1, tau=0.2) # p0=0.769, p1=0.1538
```

The mixed continuous-discrete probability (density) function of $Y \sim \texttt{BEINF}(\mu, \sigma, \nu, \tau)$ is given by

$$f_Y(y) = \begin{cases} p_0 & \text{if } y = 0 \\ (1 - p_0 - p_1) f_W(y) & \text{if } 0 < y < 1 \\ p_1 & \text{if } y = 1 \end{cases} \tag{6}$$

for $0 \leq y \leq 1$, where $W \sim BE(\mu, \sigma)$ has a beta distribution with $0 < \mu < 1$ and $0 < \sigma < 1$ and $p_0 = \nu/(1 + \nu + \tau)$ and $p_1 = \tau/(1 + \nu + \tau)$. Hence $\nu = p_0/p_2$ and $\tau = p_1/p_2$ where

$p_2 = 1 - p_0 - p_1$. Since $0 < p_0 < 1$, $0 < p_1 < 1$ and $0 < p_0 + p_1 < 1$, hence $\nu > 0$ and $\tau > 0$. Here $f_W(y)$ is a beta, BE(μ, σ), probability density function given by

$$f_W(y) = \frac{1}{B(\alpha, \beta)} y^{\alpha-1}(1-y)^{\beta-1}$$

for $0 < y < 1$ where $\alpha = \mu(1-\sigma^2)/\sigma^2$ and $\beta = (1-\mu)(1-\sigma^2)/\sigma^2$, with $E(W) = \mu$ and $Var(W) = \sigma^2\mu(1-\mu)$. Hence in (6),

$$\begin{pmatrix} \alpha \\ \beta \\ p_0 \\ p_1 \end{pmatrix} = \begin{pmatrix} \frac{\mu(1-\sigma^2)}{\sigma^2} \\ \frac{(1-\mu)(1-\sigma^2)}{\sigma^2} \\ \frac{\nu}{(1+\nu+\tau)} \\ \frac{\tau}{(1+\nu+\tau)} \end{pmatrix}$$

Hence

$$\begin{pmatrix} \mu \\ \sigma \\ \nu \\ \tau \end{pmatrix} = \begin{pmatrix} \alpha(\alpha+\beta)^{-1} \\ (\alpha+\beta+1)^{-1/2} \\ \frac{p_0}{p_2} \\ \frac{p_1}{p_2} \end{pmatrix}$$

The default predictors are $\eta_1 = \log[\mu/(1-\mu)]$, $\eta_2 = \log[\sigma/(1-\sigma)]$, $\eta_3 = \log(\nu)$ and $\eta_4 = \log(\tau)$.

For $Y \sim$ BEINF0(μ, σ, ν) set $\tau = 0$ (and hence $p_1 = 0$) in the above probability (density) function (6). For $Y \sim$ BEINF1(μ, σ, ν) set $\nu = 0$ (and hence $p_0 = 0$) and then set $\tau = \nu$ in the above probability (density) function (6).

In all three simulated examples W has a beta distribution with $\mu = 0.3$ and $\sigma = 0.3$. For the distribution on $[0, 1)$ the probability at zero is $p_0 = \frac{\nu}{1+\nu} = 0.15/(1 + .15) = 0.1304348$. For the distribution on $(0, 1]$ the probability at one is $p_1 = \frac{\nu}{1+\nu} = 0.15/(1 + .15) = 0.1304348$. For the distribution on $[0, 1]$ the probability at zero is $p_0 = \frac{\nu}{1+\nu+\tau} = 0.1/(1 + 0.1 + 0.2) = 0.0769231$ while the probability at one is $p_1 = \frac{\tau}{1+\nu+\tau} = 0.2/(1 + 0.1 + 0.2) = 0.1538462$. The sample proportions of zeros and ones in the sample are 0.123 for $[0, 1)$, 0.127 for $(0, 1]$ and $(0.07, 0.167)$ for $[0, 1]$. Next plot the three data sets using `histdist()`.

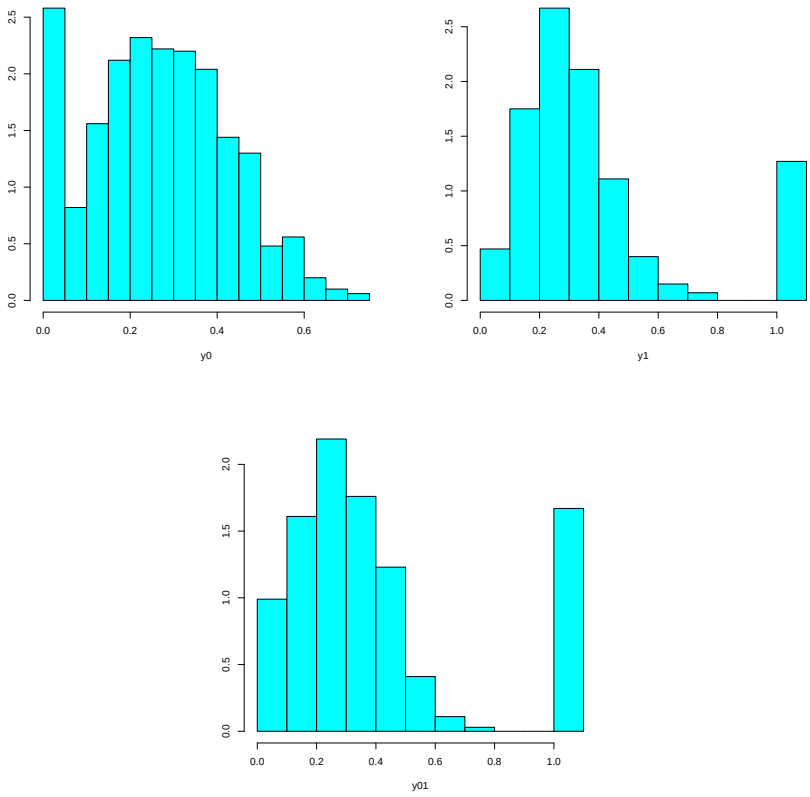
```
library(MASS)
truehist(y0)
truehist(y1)
truehist(y01)
```

Figure 5

5.3 Fitting a distributions on $[0, 1)$

Below an inflated distribution at 0 is fitted using both `gamlss()` and `gamlssInf0to1()` functions. Note that the `family` argument in `gamlssInf0to1()` takes a `gamlss.family` distribution defined on $(0, 1)$. The `trace=TRUE` argument is used in `gamlssInf0to1()` to check the convergence of the two different models fitted, one using the `BI` family and the other using the `BE`.

```
g0 <- gamlss(y0~1, family=BEINF0)
```



R code on page 14

Figure 5: Generated data using inflated beta distribution: with values $\mu = 0.3$, $\sigma = 0.3$, and $\nu = 0.15$ for the `BEINF0`(μ, σ, ν) distribution on $[0, 1)$, $\nu = 0.15$ for the `BEINF1`(μ, σ, ν) distribution on $(0, 1]$, and $\nu = 0.1$ and $\tau = 0.2$ for the `BEINF`(μ, σ, ν, τ) distribution on $[0, 1]$.

```
## GAMLSS-RS iteration 1: Global Deviance = -239.6607
## GAMLSS-RS iteration 2: Global Deviance = -307.2383
## GAMLSS-RS iteration 3: Global Deviance = -307.6252
## GAMLSS-RS iteration 4: Global Deviance = -307.6258
```

```
 t0 <- gamlssInf0to1(y=y0, mu.formula=~1, family=BE, trace=TRUE)
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 745.7199
## GAMLSS-RS iteration 2: Global Deviance = 745.7199
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -985.3807
## GAMLSS-RS iteration 2: Global Deviance = -1052.958
## GAMLSS-RS iteration 3: Global Deviance = -1053.345
## GAMLSS-RS iteration 4: Global Deviance = -1053.346
##             The Final Global Deviance  = -307.6258
```

```
AIC(g0,t0, k=0)
```

```
##    df       AIC
## t0  3 -307.6258
## g0  3 -307.6258
```

Note that the global deviance of the fitted `t0` model, using `gamlssInf0to1()`, is obtained by adding the individual deviances from the binomial and the beta model. The third fitted parameter in both models, is related to the the probability at zero. The third parameter is called `nu` (i.e. ν) in `gamlss` but `xi0` (i.e. ξ_0) in `gamlssInf0to1()`. The coefficients for the predictor η_3 for the third parameter are the same for both model as shown below.

```
coef(g0, "nu")
```

```
## (Intercept)
##   -1.964323
```

```
coef(t0, "xi0")
```

```
## (Intercept)
##   -1.964323
```

The two fitted coefficients for predictor η_3 are identical, but the fitted values for ν and ξ_0 are not the same because the parametrization used for the zero inflated distribution using `gamlssInf0to1()` is different from `gamlss()` using `BEINF0`. Next only the first element of the fitted values vector for the third parameter is displayed (since all values are identical because we fit a constant model).

```
fitted(t0, "xi0")[1]
```

```
## [1] 0.123
```

```
fitted(g0, "nu")[1]
```

```
##         1
## 0.1402509
```

The difference in the fitted values of ξ_0, (`xi0`), and ν, (`nu`), above is due to the way the

two models are parametrized, since $\xi_0 = p_0$, while $\nu = p_0/(1-p_0)$. `gamlssInf0to1()` fits a binary distribution model with a `logit` link for the binomial distribution parameter i.e. $\eta_t = \log\left[\xi_0/\left(1-\xi_0\right)\right]$. The vector `fitted(t0, "xi0")` contains the fitted probabilities at zero. In the above example predictor $\eta_t = \hat{\beta}_t = -1.964$ is the `coef(t0, "xi0")` so $\hat{\xi}_0 = 1/(1+e^{-\hat{\eta}_t}) = 1/(1+e^{-\hat{\beta}_t}) = \hat{p}_0 = 0.123$. In `gamlss()`, ν is fitted using a log link i.e. $\eta_g = \log(\nu)$. In the above example $\hat{\eta}_g = \hat{\beta}_g = -1.964$ is the `coef(g0, "nu")`, so $\hat{\nu} = e^{\hat{\eta}_g} = e^{\hat{\beta}_g} = 0.1402509$ is the fitted value for ν. In `BEINF0` ν is defined as the odds ratio i.e. $\hat{\nu} = \hat{p}_0/(1-\hat{p}_0)$ which implies that $\hat{p}_0 = \hat{\nu}/(1+\hat{\nu})$ so again $\hat{p}_0 = 0.123$. This can be confirmed by:

```
fitted(g0, "nu")[1]/(1+fitted(g0, "nu"))[1]
```

```
##     1
## 0.123
```

which is the fitted probability of observing zero. The `summary()` function makes it clear that the two models use different link functions for the third parameters ν or ξ_0.

```
summary(t0)
```

```
## ******************************************************************
## Family:  "InfBE"
##
## Call:  gamlssInf0to1(y = y0, mu.formula = ~1, family = BE, trace = TRUE)
##
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.84294    0.02203  -38.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.84659    0.02989  -28.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## xi0 link function:  logit
## xi0 Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.96432    0.09628   -20.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## ------------------------------------------------------------------
## No. of observations in the fit:  1000
## Degrees of Freedom for the fit:  3
##       Residual Deg. of Freedom:  997
##                       at cycle:
##
## Global Deviance:     -307.6258
##             AIC:     -301.6258
##             SBC:     -286.9026
## ******************************************************************
```

```
summary(g0)
```

```
## ******************************************************************
## Family:  c("BEINF0", "Beta Inflated zero")
##
## Call:  gamlss(formula = y0 ~ 1, family = BEINF0)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.84294    0.02203  -38.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.84659    0.02989  -28.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  log
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.96432    0.09628   -20.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  1000
## Degrees of Freedom for the fit:  3
##       Residual Deg. of Freedom:  997
```

```
##                       at cycle:  4
##
## Global Deviance:     -307.6258
##              AIC:    -301.6258
##              SBC:    -286.9026
## ******************************************************************
```

The variance covariance matrix for the fitted `g0` and `t0` models can be obtained as follows:

```
vcov(t0)
```

```
##              (Intercept)  (Intercept) (Intercept)
## (Intercept) 0.0004854761 0.0001292291 0.000000000
## (Intercept) 0.0001292291 0.0008936886 0.000000000
## (Intercept) 0.0000000000 0.0000000000 0.009270331
```

```
vcov(g0)
```

```
##              (Intercept)  (Intercept) (Intercept)
## (Intercept) 0.0004854761 0.0001292291 0.000000000
## (Intercept) 0.0001292291 0.0008936886 0.000000000
## (Intercept) 0.0000000000 0.0000000000 0.009270331
```

Note the three columns (and three rows) in the estimated variance-covariance matrices above correspond to the predictor parameters $\eta_{kt} = \beta_{kt}$ for $k = 1, 2, 3$ for μ, σ and ξ_0 and $\eta_{kg} = \beta_{kg}$ for $g = 1, 2, 3$ for μ, σ and ν, where subscripts t and g indicate models `t0` and `g0`; respectively.

Note that because of the partition of the likelihood function parameters μ and σ are orthogonal to ν or ξ_0.

The residuals for the two models should be identical for the non zero response values. Due to the randomization in the residuals at discrete values of the response variable (zero here), we expect differences between the two models in the residuals when the response is zero. This is demonstrated in the lower part of Figure 6 where the residuals are plotted against the observation index.

```
plot(resid(t0), pch="+")
points(resid(g0), col="red")
```

Figure 6

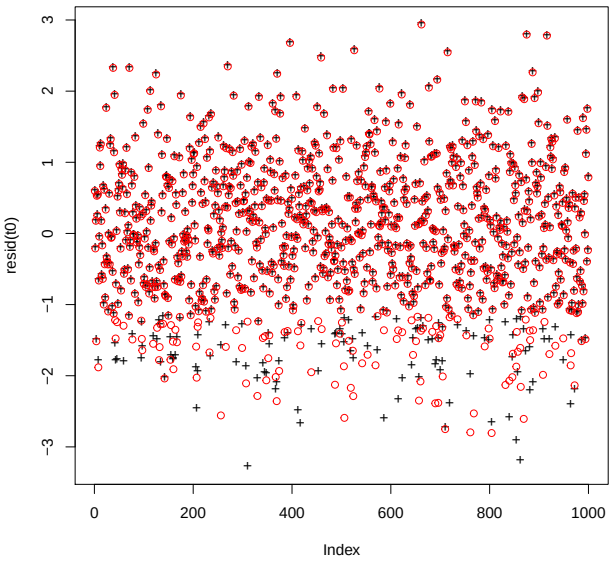
Next we will plot the fitted distribution in Figure 7. The standard `BEINF0` distribution in **gamlss.dist** has its own plotting function called `plotBEINF0()` which can be used here. For the model fitted with `gamlssInf0to1()` such a function, `plotBEInf0()`, is created using the `gen.Inf0to1()` function.

```
# generate the beta distribution inflated at 0
gen.Inf0to1("BE", type="Zero")
```

```
## A  0 inflated BE distribution has been generated
##  and saved under the names:
##  dBEInf0 pBEInf0 qBEInf0 rBEInf0
##  plotBEInf0
```

```
plotBEINF0(mu=fitted(g0, "mu")[1], sigma=fitted(g0, "sigma")[1],
           nu=fitted(g0, "nu")[1], main="(a)", ylab="density")
```

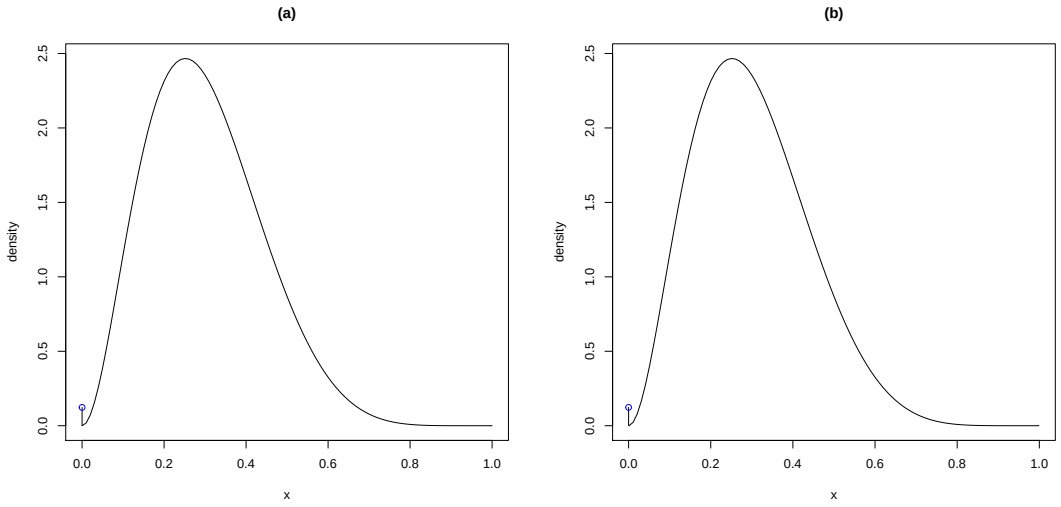
Figure 7



R code on page 19

Figure 6: Superimposed residuals from models `t0` (+) and g0 (o). Because of the randomization in the residuals of the zero values of the response variable, the values of the residuals in the lower part of the plot are not identical.

```
plotBEInf0(mu=fitted(t0, "mu")[1], sigma=fitted(t0, "sigma")[1],
           xi0=fitted(t0, "xi0")[1]) ; title("(b)")
```



R code on page 19

Figure 7: The fitted distribution using (a) `gamlss()` and (b) `gamlssInf0to1()`

The fitted distributions are identical.

5.4 Fitting a distributions on $(0, 1]$

Now the data inflated at 1 is analyzed.

```
 g1 <- gamlss(y1~1, family=BEINF1)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -264.3046
## GAMLSS-RS iteration 2: Global Deviance = -343.1028
## GAMLSS-RS iteration 3: Global Deviance = -343.6308
## GAMLSS-RS iteration 4: Global Deviance = -343.6318
```

```
 t1 <- gamlssInf0to1(y=y1, mu.formula=~1, family=BE)
AIC(g1,t1, k=0)
```

```
##    df       AIC
## t1  3 -343.6318
## g1  3 -343.6318
```

The third fitted parameter in both models, is related to p_1 the probability at one. It is called ν in `gamlss` and ξ_1 in `gamlssInf0to1()`, where $\xi_1 = p_1$ and $\nu = p_1/(1 - p_1)$.

```
coef(g1, "nu")
```

```
## (Intercept)
##   -1.927748
```

```
coef(t1, "xi1")
```

```
## (Intercept)
##   -1.927748
```

Again the two fitted coefficients in the predictors $\eta_t = \log\left[\xi_1/\left(1-\xi_1\right)\right] = \beta_t$ and $\eta_g = \log\left(\nu\right) = \beta_g$ are identical, but the fitted values for ξ_1 and ν are different because of the different link functions.

```
fitted(t1, "xi1")[1]
```

```
## [1] 0.127
```

```
fitted(g1, "nu")[1]
```

```
##         1
## 0.1454754
```

The vector `fitted(t1, "xi1")` contains the fitted probabilities at one. For example let $\hat{\beta}_{t1}$ be the `coef(t1, "xi1")` then $\hat{\xi}_1 = 1/(1+e^{-\hat{\beta}_{t1}}) = \hat{p}_1 = 0.127$. In `gamlss`, ν is fitted using the log link. Let $\hat{\beta}_{g1}$ be the `coef(g1, "nu")` so $\hat{\nu} = e^{\hat{\beta}_{g1}} = 0.1454754$. In `BEINF1` ν is defined as the odds ratio for example $\hat{\nu} = \hat{p}_1/(1-\hat{p}_1)$ which implies that $\hat{p}_1 = \hat{\nu}/(1+\hat{\nu})$ so again $\hat{p}_0 = 0.127$. This can be confirmed by:

```
fitted(g1, "nu")[1]/(1+fitted(g1, "nu"))[1]
```

```
##     1
## 0.127
```

which is the fitted probability of observing one. The `summary()` function makes it clear that the two models use different link functions for the third parameters ν or ξ_1.

```
summary(t1)
```

```
## ******************************************************************
## Family:  "InfBE"
##
## Call:  gamlssInf0to1(y = y1, mu.formula = ~1, family = BE)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.88074    0.02172  -40.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -0.88122    0.02987   -29.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## xi1 link function:  logit
## xi1 Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.92775    0.09497   -20.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  1000
## Degrees of Freedom for the fit:  3
##       Residual Deg. of Freedom:  997
##                       at cycle:
##
## Global Deviance:     -343.6318
##             AIC:     -337.6318
##             SBC:     -322.9085
## ******************************************************************
```

```
summary(g1)
```

```
## ******************************************************************
## Family:  c("BEINF1", "Beta Inflated one")
##
## Call:  gamlss(formula = y1 ~ 1, family = BEINF1)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.88074    0.02172  -40.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.88122    0.02987   -29.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
```

```
## Nu link function:  log
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.92775    0.09497   -20.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  1000
## Degrees of Freedom for the fit:  3
##       Residual Deg. of Freedom:  997
##                       at cycle:  4
##
## Global Deviance:     -343.6318
##             AIC:     -337.6318
##             SBC:     -322.9085
## ******************************************************************
```

The variance covariance matrix for the fitted `g1` and `t1` models can be obtained as follows:

```
vcov(t1)
```

```
##              (Intercept)  (Intercept) (Intercept)
## (Intercept) 0.0004719565 0.0001297286  0.00000000
## (Intercept) 0.0001297286 0.0008923410  0.00000000
## (Intercept) 0.0000000000 0.0000000000  0.00901949
```

```
vcov(g1)
```

```
##              (Intercept)  (Intercept) (Intercept)
## (Intercept) 0.0004719565 0.0001297286  0.00000000
## (Intercept) 0.0001297286 0.0008923410  0.00000000
## (Intercept) 0.0000000000 0.0000000000  0.00901949
```

The fitted distributions can be plotted as follows:

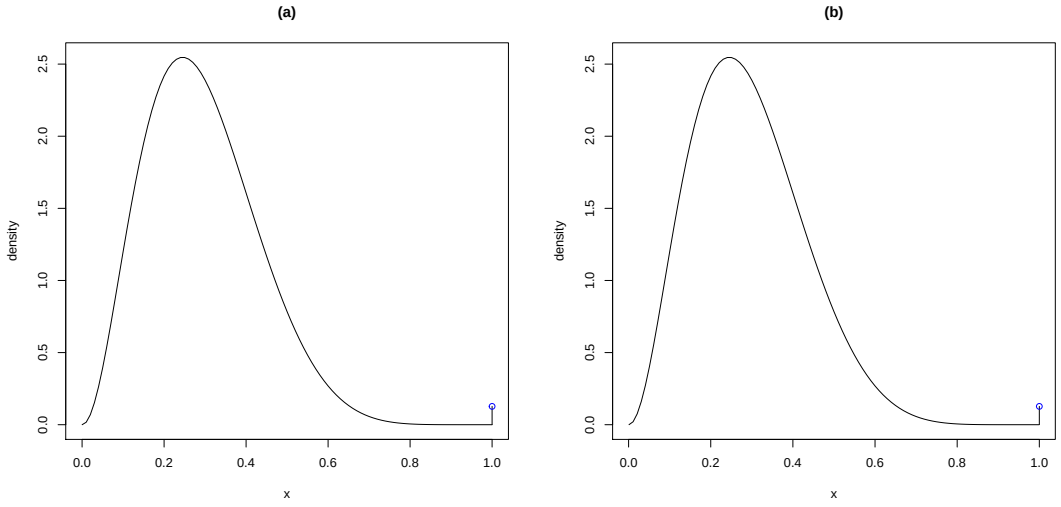
```
# generate the
gen.Inf0to1("BE", type="One")
```

```
## A  1 inflated BE distribution has been generated
##  and saved under the names:
##  dBEInf1 pBEInf1 qBEInf1 rBEInf1
##  plotBEInf1
```

```
plotBEINF1(mu=fitted(g1, "mu")[1], sigma=fitted(g1, "sigma")[1],
           nu=fitted(g1, "nu")[1], main="(a)", ylab="density")
plotBEInf1(mu=fitted(t1, "mu")[1], sigma=fitted(t1, "sigma")[1],
           xi1=fitted(t1, "xi1")[1]) ; title("(b)")
```

Figure 8

The fitted distributions are identical.



R code on page 24

Figure 8: The fitted distribution using (a) `gamlss()` and (b) `gamlssInf0to1()`

5.5 Fitting a distributions on $[0, 1]$

Now an inflated distribution on 0 and 1 is fitted using both `gamlss()` and `gamlssInf0to1()` functions

```
 g01 <- gamlss(y01~1, family=BEINF)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 471.2145
## GAMLSS-RS iteration 2: Global Deviance = 401.5136
## GAMLSS-RS iteration 3: Global Deviance = 401.1247
## GAMLSS-RS iteration 4: Global Deviance = 401.1241
```

```
 t01 <- gamlssInf0to1(y=y01, mu.formula=~1, family=BE)
AIC(g01,t01, k=0)
```

```
##     df      AIC
## g01  4 401.1241
## t01  4 401.1241
```

Note that in `gamlssInf0to()` it was not needed to specify that the distribution was on $[0, 1]$ because the function detected whether there are any zero and one values in the response variable and acts accordingly. The third and fourth fitted parameters in both models are related to the probabilities at zero and one. They are called ν and τ in `gamlss` but ξ_0 and ξ_1 in `gamlssInf0to1()`.

```
coef(g01, "nu")
```

```
## (Intercept)
##   -2.388763
```

```
coef(t01, "xi0")
```

```
## (Intercept)
##   -2.388747
```

```
coef(g01, "tau")
```

```
## (Intercept)
##   -1.519264
```

```
coef(t01, "xi1")
```

```
## (Intercept)
##   -1.519263
```

The fitted coefficients are (almost) identical for ν and ξ_0 and also for τ and ξ_1. Now look at the fitted values.

```
fitted(t01, "xi0")[1]
```

```
##          1
## 0.09174455
```

```
fitted(g01, "nu")[1]
```

```
##          1
## 0.09174314
```

```
fitted(t01, "xi1")[1]
```

```
##         1
## 0.2188732
```

```
fitted(g01, "tau")[1]
```

```
##         1
## 0.2188729
```

Note that contrary to the models with only zero or only one in the response variable, the models on $[0, 1]$ use the same parametrization so the fitted values are identical. In fact here the parameters are related to the probabilities at zero and one as

$$\xi_0 = \nu = \frac{p_0}{(1 - p_0 - p_1)}$$

and

$$\xi_1 = \tau = \frac{p_1}{(1 - p_0 - p_1)}$$

so

$$p_0 = \frac{\xi_0}{(1 + \xi_0 + \xi_1)} = \frac{\nu}{(1 + \nu + \tau)}$$

and

$$p_1 = \frac{\xi_1}{(1 + \xi_0 + \xi_1)} = \frac{\tau}{(1 + \nu + \tau)}.$$

This can be verified by:

```
# probability for y=0
fitted(g01, "nu")[1]/(1+fitted(g01, "nu")+fitted(g01, "tau"))[1]
```

```
##          1
## 0.07000002
```

```
fitted(t01, "xi0")[1]/(1+fitted(t01, "xi0")+fitted(t01, "xi1"))[1]
```

```
##         1
## 0.070001
```

```
# probability for y=1
fitted(g01, "tau")[1]/(1+fitted(g01, "nu")+fitted(g01, "tau"))[1]
```

```
##     1
## 0.167
```

```
fitted(t01, "xi1")[1]/(1+fitted(t01, "xi0")+fitted(t01, "xi1"))[1]
```

```
##     1
## 0.167
```

The `summary()` produces the same results for both models, so only `t01` is represented here.

```
summary(t01)
```

```
## ******************************************************************
## Family:  "InfBE"
##
## Call:  gamlssInf0to1(y = y01, mu.formula = ~1, family = BE)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.8484     0.0226  -37.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.91148    0.03182  -28.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## xi0 link function:  log
## xi0 Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.3887     0.1249  -19.13   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -------------------------------------------------------------------
## xi1 link function:  log
## xi1 Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.51926    0.08543  -17.78   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -------------------------------------------------------------------
## No. of observations in the fit:  1000
## Degrees of Freedom for the fit:  4
##       Residual Deg. of Freedom:  996
##                       at cycle:
##
## Global Deviance:     401.1241
##             AIC:     409.1241
##             SBC:     428.7551
## *******************************************************************
```

The variance covariance matrix for the fitted `g01` and `t01` models is for all practical purposes identical.

```
vcov(t01)
```

```
##              (Intercept)  (Intercept) (Intercept) (Intercept)
## (Intercept) 0.0005107378 0.0001350223 0.000000000 0.000000000
## (Intercept) 0.0001350223 0.0010125764 0.000000000 0.000000000
## (Intercept) 0.0000000000 0.0000000000 0.015596125 0.001310618
## (Intercept) 0.0000000000 0.0000000000 0.001310618 0.007298641
```

```
vcov(g01)
```

```
##               (Intercept)   (Intercept)   (Intercept)  (Intercept)
## (Intercept)  5.107378e-04  1.350223e-04 -6.855538e-14 1.073091e-14
## (Intercept)  1.350223e-04  1.012576e-03 -1.218317e-13 1.907021e-14
## (Intercept) -6.855538e-14 -1.218317e-13  1.559632e-02 1.310616e-03
## (Intercept)  1.073091e-14  1.907021e-14  1.310616e-03 7.298640e-03
```

Because of the randomization of the residuals when the response variable takes values at zero and one, the residuals differ when the response variable is at those values, as is shown in Figure 9 where the residuals are plotted against the observation index.

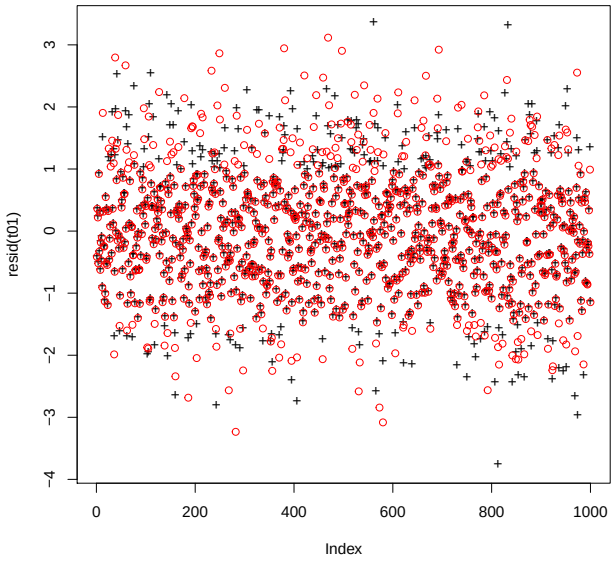
```
plot(resid(t01), pch="+")
points(resid(g01), col="red")
```

Figure 9

The fitted distributions are plotted next.

```
# generate the
gen.Inf0to1("BE", type="Zero&One")
```

Figure 10



R code on page 28

Figure 9: Superimposed residuals form models `t01` (+) and g01 (o). Because of the randomization the values differ when the response is at zero and one

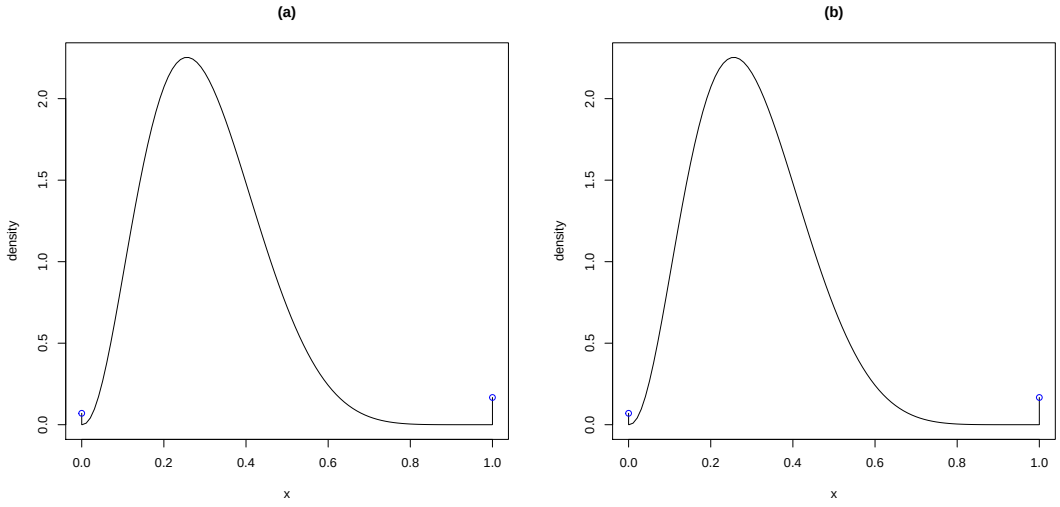
```
## A  0to1 inflated BE distribution has been generated
##  and saved under the names:
##  dBEInf0to1 pBEInf0to1 qBEInf0to1 rBEInf0to1
##  plotBEInf0to1
plotBEINF(mu=fitted(g01, "mu")[1], sigma=fitted(g01, "sigma")[1],
          nu=fitted(g01, "nu")[1], tau=fitted(g01, "tau")[1],
          main="(a)", ylab="density")
plotBEInf0to1(mu=fitted(t01, "mu")[1], sigma=fitted(t01, "sigma")[1],
           xi0=fitted(t01, "xi0")[1],  xi1=fitted(t01, "xi1")[1])
title("(b)")
```

The fitted distributions are identical.

6 Fitting a regression model

6.1 Simulating regression models on $[0, 1]$

We first generate the values for the explanatory variable x and then create four different functions of x for the parameters μ, σ, ν and τ, respectively, of the beta inflated distributions (`BEINF0`, `BEINF1` and `BEINF`). Since we would like the simulated data to look as a real data example, we will simulate from some saved functions from a real data analysis. Figure 11 displays



R code on page 28

Figure 10: The fitted distribution using (a) `gamlss()` and (b) `gamlssInf0to1()`

those functions.

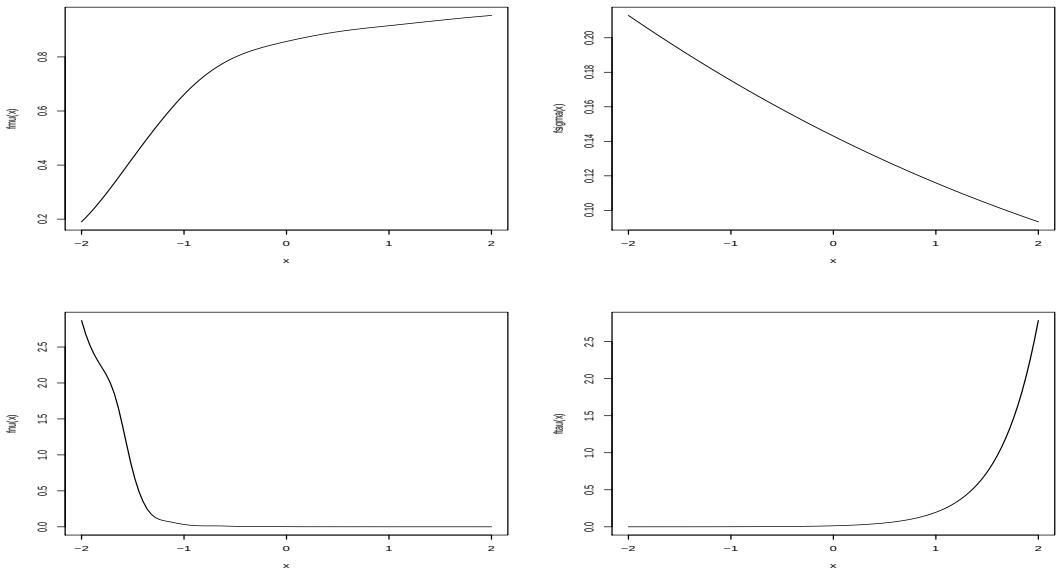
```
# generating x ----------
set.seed(3210)
x <- (runif(1000)*4)-2
range(x)

## [1] -1.995186  1.999197

data(sda)
fmu <- splinefun(sda$x, sda$mu)
curve(fmu, -2,2)
fsigma <- splinefun(sda$x, sda$sigma)
curve(fsigma, -2,2)
fnu <- splinefun(sda$x, sda$nu)
curve(fnu, -2,2)
ftau <- splinefun(sda$x, sda$tau)
curve(ftau, -2,2)
```

Figure 11

Next we generate three different response variables, `y0`, `y1` and `y01`, from a beta inflated distribution with values defined on $[0, 1)$, $(0, 1]$ and $[0, 1]$ respectively. Figure 12 shows the three different response variables y against x.

```
# generating x ----------
set.seed(1234)
 y0 <- rBEINF0(1000, mu=fmu(x), sigma=fsigma(x), nu=fnu(x))
 y1 <- rBEINF1(1000, mu=fmu(x), sigma=fsigma(x), nu=ftau(x))
y01 <- rBEINF(1000,  mu=fmu(x), sigma=fsigma(x), nu=fnu(x), tau=ftau(x))
# plotting the y's
plot(x,y0, col="darkgray")
```

Figure 12



R code on page 30

Figure 11: Showing the four functions used for the simulation of the data in this section. From top left to bottom right we have the functions for μ, σ, ν and τ.

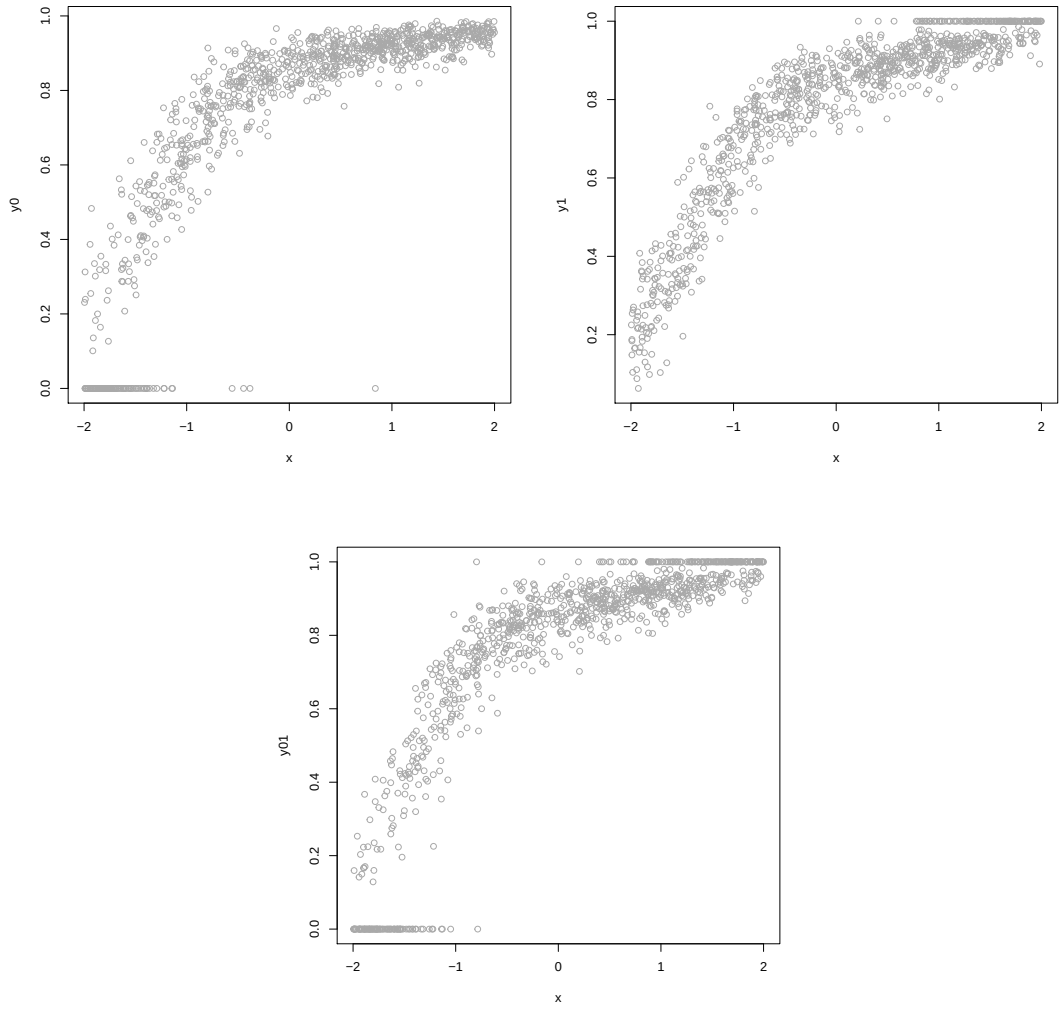
```
plot(x,y1, col="darkgray")
plot(x,y01, col="darkgray")
```

6.2 Fitting a regression model on $[0, 1)$

We start with the data shown in the top left of Figure 12. We will fit a beta inflated at zero distribution using the functions `gamlss()`. The `gamlss.dist` package has two functions for using the beta inflated at zero distribution: `BEINF0` and `BEZI`. Their parametrization is slightly different. The same model will be also fitted using `gamlssIng0to1()` function. Since in general the type of relationship existing between the parameters and the explanatory variable is unknown we will use smooth functions for x. In the following code we used the P-spline smother implemented in the additive term function `pb()`.

```
g0p <- gamlss(y0~pb(x), sigma.fo=~pb(x), nu.fo=~pb(x), family=BEINF0)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -1826.759
## GAMLSS-RS iteration 2: Global Deviance = -2508.671
## GAMLSS-RS iteration 3: Global Deviance = -2799.86
## GAMLSS-RS iteration 4: Global Deviance = -2845.529
## GAMLSS-RS iteration 5: Global Deviance = -2846.434
## GAMLSS-RS iteration 6: Global Deviance = -2846.404
## GAMLSS-RS iteration 7: Global Deviance = -2846.388
## GAMLSS-RS iteration 8: Global Deviance = -2846.387
## GAMLSS-RS iteration 9: Global Deviance = -2846.386
```



R code on page 30

Figure 12: Showing the response variable against the single explanatory variable x for the three simulated data sets from beta inflated distributions (`BEINF0`, `BEINF1` and `BEINF`). The response variables are `y0`, `y1` and `y01`, with values defined on $[0, 1)$, $(0, 1]$ and $[0, 1]$, respectively.

```
g0p1 <- gamlss(y0~pb(x), sigma.fo=~pb(x), nu.fo=~pb(x), family=BEZI)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -1076.408
## GAMLSS-RS iteration 2: Global Deviance = -1935.656
## GAMLSS-RS iteration 3: Global Deviance = -2505.354
## GAMLSS-RS iteration 4: Global Deviance = -2783.475
## GAMLSS-RS iteration 5: Global Deviance = -2842.939
## GAMLSS-RS iteration 6: Global Deviance = -2846.26
## GAMLSS-RS iteration 7: Global Deviance = -2846.208
## GAMLSS-RS iteration 8: Global Deviance = -2846.192
## GAMLSS-RS iteration 9: Global Deviance = -2846.191
## GAMLSS-RS iteration 10: Global Deviance = -2846.191
```

```
 t0p <- gamlssInf0to1(y=y0, mu.fo=~pb(x), sigma.fo=~pb(x),
                 xi0.fo=~pb(x), family="BE", trace=TRUE)
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 277.5741
## GAMLSS-RS iteration 2: Global Deviance = 278.5096
## GAMLSS-RS iteration 3: Global Deviance = 278.6246
## GAMLSS-RS iteration 4: Global Deviance = 278.6555
## GAMLSS-RS iteration 5: Global Deviance = 278.6632
## GAMLSS-RS iteration 6: Global Deviance = 278.6652
## GAMLSS-RS iteration 7: Global Deviance = 278.6656
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -2104.597
## GAMLSS-RS iteration 2: Global Deviance = -2787.287
## GAMLSS-RS iteration 3: Global Deviance = -3078.498
## GAMLSS-RS iteration 4: Global Deviance = -3124.185
## GAMLSS-RS iteration 5: Global Deviance = -3125.097
## GAMLSS-RS iteration 6: Global Deviance = -3125.069
## GAMLSS-RS iteration 7: Global Deviance = -3125.054
## GAMLSS-RS iteration 8: Global Deviance = -3125.052
## GAMLSS-RS iteration 9: Global Deviance = -3125.052
##             The Final Global Deviance  = -2846.386
```

```
AIC(g0p,g0p1, t0p)
```

```
##            df       AIC
## g0p1 13.26330 -2819.664
## g0p  13.39814 -2819.590
## t0p  13.39823 -2819.590
```

```
summary(t0p)
```

```
## ******************************************************************
## Family:  "InfBE"
##
## Call:
## gamlssInf0to1(y = y0, mu.formula = ~pb(x), sigma.formula = ~pb(x),
##     xi0.formula = ~pb(x), family = "BE", trace = TRUE)
```

```
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  logit
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.52210    0.01373  110.86   <2e-16 ***
## pb(x)        0.93845    0.01322   71.01   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  logit
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.77953    0.02739 -64.977   <2e-16 ***
## pb(x)       -0.24929    0.02500  -9.971   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## xi0 link function:  logit
## xi0 Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.2907     0.4914  -8.731  < 2e-16 ***
## pb(x)        -1.8306     0.3177  -5.762 1.11e-08 ***
## ---
## 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:  1000
## Degrees of Freedom for the fit:  13.39823
##       Residual Deg. of Freedom:  986.6018
##                       at cycle:
##
## Global Deviance:     -2846.386
##             AIC:     -2819.59
##             SBC:     -2753.835
## ******************************************************************
```

The three models have similar deviances as we would expect. Do not try to interpret the coefficients and the standard errors of the smoothing functions in the summary table. They are for the linear term in the variable x and are here as a consequence of how the model is fitted

within the `gamlss()` algorithm. They do not indicate the coefficients and the standard errors of the smoothing functions. To check whether the smoothing function as a whole is significant use the function `drop1()`. In our case, since we are dealing with simulated data, we can actually plot both the true functions and the fitted functions against x. Note that the fitted values for the parameters μ and σ are identical in both fitted models. The third fitted distribution parameter, $\hat{\nu}$ or $\hat{\xi}_9$ is different for the two fitted models. In model `t0p` the fitted values of ξ_0 are probabilities of being zero p_0, since $\xi_0 = p_0$, while in model `g0p` the fitted values for ν are odds since $\nu = p_0/(1-p_0)$. Figure 13 plots the true and fitted functions (for models `g0p` [or `t0p`]) for μ, σ and ν [or $\xi_0/(1-\xi_0)$].

```
# mu
curve(fmu, -2,2, main="mu", lwd=3)
lines(fitted(g0p)[order(x)]~x[order(x)], col="red",   lty=2, lwd=3)
lines(fitted(t0p, "mu")[order(x)]~x[order(x)], col="blue",   lty=2, lwd=3)
# sigma
curve(fsigma, -2,2, main="sigma", lwd=3)
lines(fitted(g0p, "sigma")[order(x)]~x[order(x)], col="red",   lty=2, lwd=3)
lines(fitted(t0p, "sigma")[order(x)]~x[order(x)], col="blue",   lty=2, lwd=3)
# nu
curve(fnu, -2,2, main="nu", lwd=3)
lines(fitted(g0p, "nu")[order(x)]~x[order(x)], col="red",   lty=2, lwd=3)
fp <- fitted(t0p, "xi0")
fn <- fp/(1-fp)
lines(fn[order(x)]~x[order(x)], col="blue",   lty=2, lwd=3)
```

Figure 13

Note that the `term.plot()` function is not working for models fitted through the `gamlssInf0to1()` function. The `term.plotInf0to1()` can be used instead. Figure 14 shoes the fitted additive terms using the `gamlss` `BEINF0` fit on the left and the `gamlssInf0to1` fit on the right. The plots are identical.

```
# generating x ----------
term.plot(g0p);title("gamlss")
term.plotInf0to1(t0p,parameter="mu");title("gamlssInf0to1")
term.plot(g0p, "sigma")
term.plotInf0to1(t0p,parameter="sigma")
term.plot(g0p, "nu")
term.plotInf0to1(t0p,parameter="xi0")
```

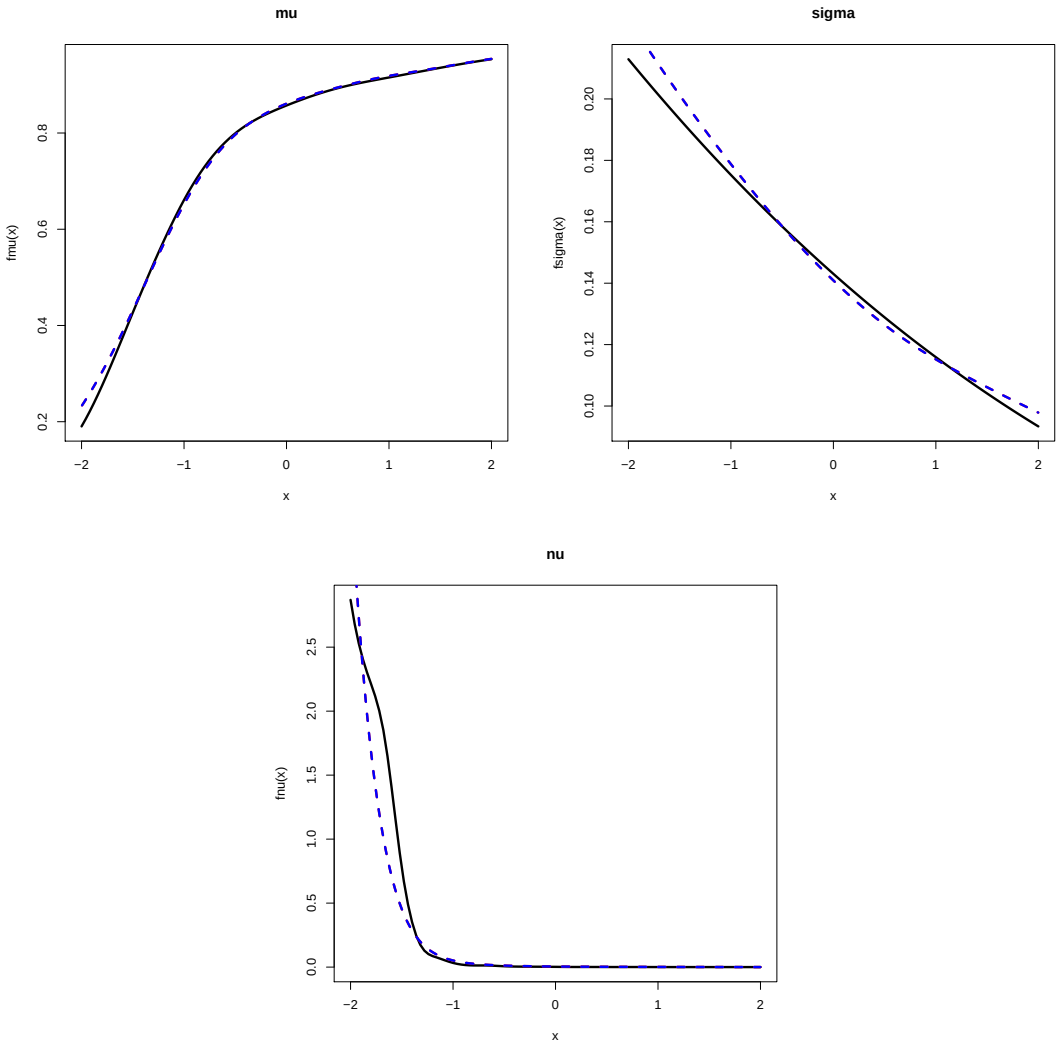
Figure 14

Note that diagnostics plots like the residual plot, `plot()`, the worm plot, `wp()`, and the Q-statistics plot, `Q.stats()`, are all working with a `gamlssInf0to1` object. Also in the case in which only one explanatory variable exists, the function `centile.Inf0to1()` can be used to plot centile curves, see Figure 15.

```
plot(t0p)
```

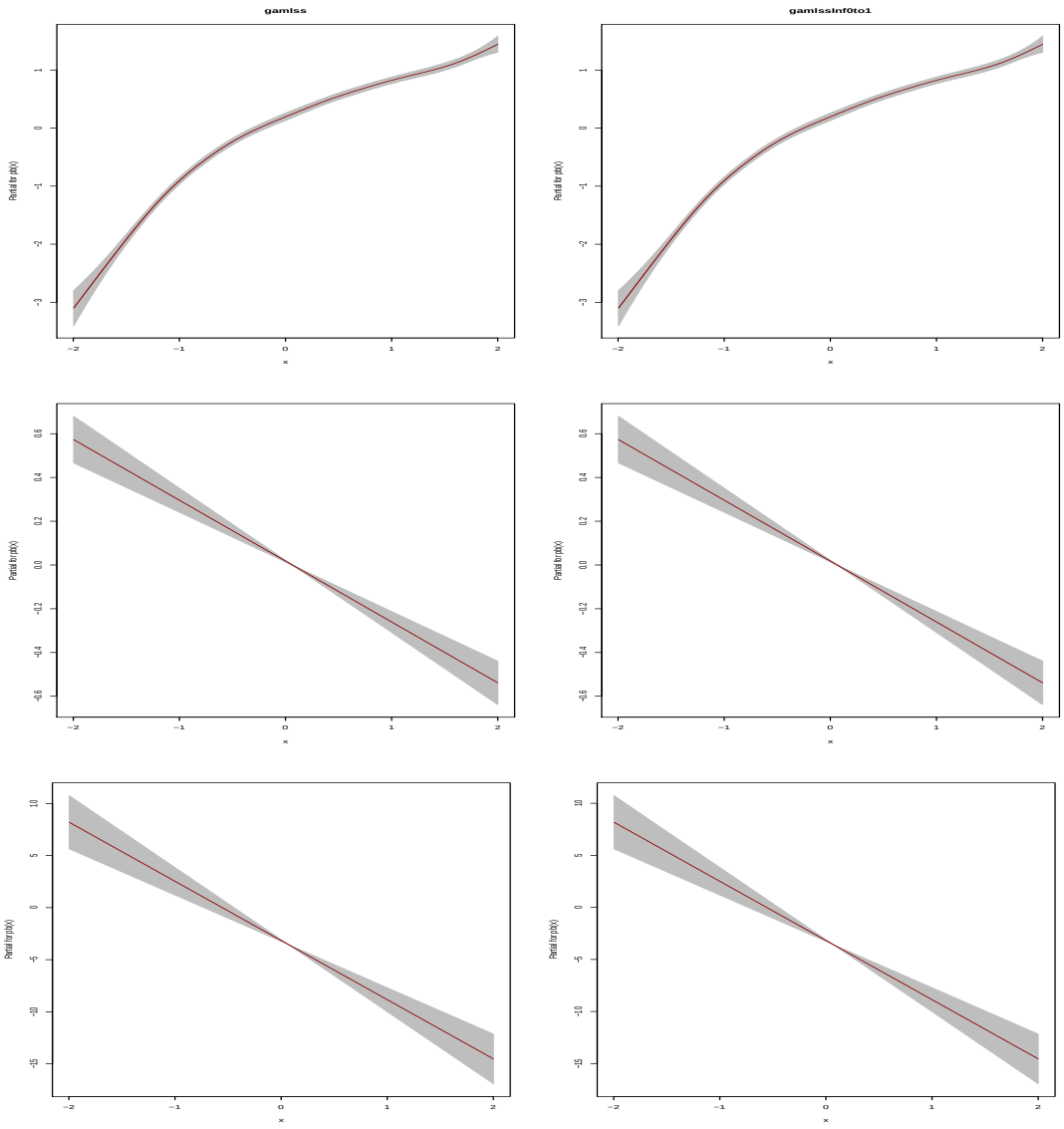
```
## ******************************************************************
##   Summary of the Randomised Quantile Residuals
##                            mean   =  0.009216538
##                        variance   =  0.9955378
##                coef. of skewness  =  0.01056115
##                coef. of kurtosis  =  3.054254
```

Figure 15



R code on page 35

Figure 13: Showing the function from which the data were simulated from together with the fitted smooth function for μ, σ and ν [or $\xi_0/(1-\xi_0)$]. The solid black line is the true function. The dashed line is the fitted values from both models `g0p` and `t0p` since they are identical.



R code on page 35

Figure 14: Showing the fitted additive predictors for μ, σ and ν (or ξ_0) and their approximate 95% confidence bands for models fitted using the function `gamlss()` on the left and using the function `gamlssInf0to1()` on the right.

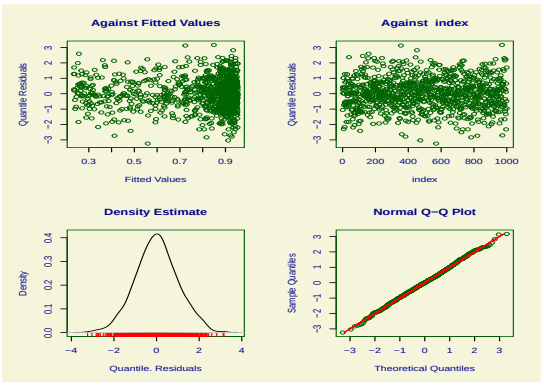
```
## Filliben correlation coefficient  =  0.9994537
## ---------------------------------------------------------------
```

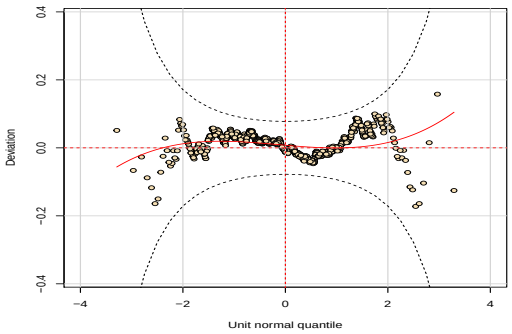
```
wp(t0p)
#par(mar = c(0,1,0,1))
Q.stats(t0p, xvar=x)
```

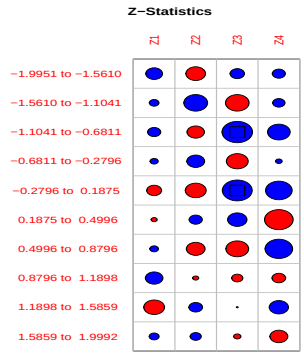
```
##                             Z1          Z2           Z3          Z4
## -1.9951 to -1.5610  0.64460452 -0.88922346  0.474012315  0.39101311
## -1.5610 to -1.1041  0.21257491  1.29206569 -1.280238090  0.34594471
## -1.1041 to -0.6811  0.39730531 -0.69702195  2.134276198  1.17349835
## -0.6811 to -0.2796  0.15060631  0.72668969 -1.090940296  0.09858706
## -0.2796 to  0.1875 -0.51100979 -0.99917838  2.047660372  1.62634003
##  0.1875 to  0.4996 -0.08303713  0.39489446  0.858454864 -1.88716607
##  0.4996 to  0.8796  0.17006367 -0.80424372 -1.188046016  1.74570047
##  0.8796 to  1.1898  0.70570965 -0.08142732 -0.283852346 -0.42599972
##  1.1898 to  1.5859 -0.98965419  0.43040708 -0.006917952  0.84419299
##  1.5859 to  1.9992  0.22449057  0.28433908 -0.120591925 -0.73788560
## TOTAL Q stats       2.46602222  5.54790604 14.045456374 12.35184730
## df for Q stats      3.11764466  8.08165435 10.000000000 10.00000000
## p-val for Q stats   0.50296796  0.70548969  0.170928282  0.26219852
##                    AgostinoK2    N
## -1.9951 to -1.5610  0.3775789  100
## -1.5610 to -1.1041  1.7586873  100
## -1.1041 to -0.6811  5.9322333  100
## -0.6811 to -0.2796  1.1998701  100
## -0.2796 to  0.1875  6.8378949  100
##  0.1875 to  0.4996  4.2983405  100
##  0.4996 to  0.8796  4.4589235  100
##  0.8796 to  1.1898  0.2620479  100
##  1.1898 to  1.5859  0.7127097  100
##  1.5859 to  1.9992  0.5590176  100
## TOTAL Q stats      26.3973037 1000
## df for Q stats     20.0000000    0
## p-val for Q stats   0.1530875    0
```

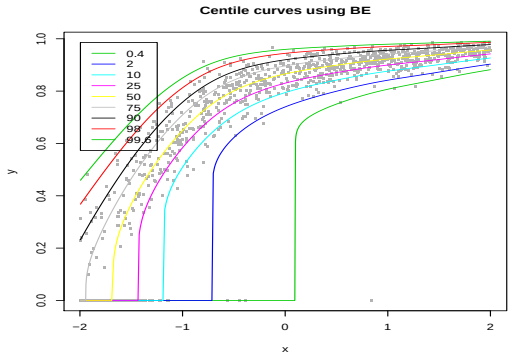
```
#par(mar = c(0,0,1,0))n
centiles.Inf0to1(t0p, xvar=x)
```

```
## % of cases below  0.4 centile is  8.8
## % of cases below  2 centile is  9.6
## % of cases below  10 centile is  15.5
## % of cases below  25 centile is  28.1
## % of cases below  50 centile is  52.2
## % of cases below  75 centile is  75.8
## % of cases below  90 centile is  89.5
## % of cases below  98 centile is  97.5
## % of cases below  99.6 centile is  99.7
```









R code on page 35

Figure 15: Showing the diagnostic plots created by `plot()`, `wp()`, and `Q.stats()`, and centile curves produced by `centile.Inf0to1()`.

6.3 Fitting alternative inflated distributions on $[0,1)$

6.3.1 Inflated logit family distributions on $[0,1)$

The function `gamlssInf0to1()` can be used with a logit transformed zero to one distribution or with a truncated zero to one distribution. The following code shows how an inflated logit transformed zero to one distribution can be fitted:

```
# generate a logit transformed distribution from 0 to 1
gen.Family("SST", "logit")

## A  logit  family of distributions from SST has been generated
##  and saved under the names:
##  dlogitSST plogitSST qlogitSST rlogitSST logitSST

# fit the model
t0sst <- gamlssInf0to1(y=y0, mu.fo=~pb(x), sigma.fo=~pb(x),
                  xi0.fo=~pb(x), family="logitSST")
```

6.3.2 Inflated truncated distributions on $[0,1)$

Here we demonstrate how an inflated truncated zero to one distribution can be fitted.

```
# generate a truncated distribution from 0 to 1
library(gamlss.tr)
gen.trun(c(0,1),"SST",type="both")

## A truncated family of distributions from SST has been generated
##  and saved under the names:
##  dSSTtr pSSTtr qSSTtr rSSTtr SSTtr
## The type of truncation is both
##  and the truncation parameter is 0 1

# fit the model
t0ssttr <- gamlssInf0to1(y=y0, mu.fo=~pb(x), sigma.fo=~pb(x),
                      xi0.fo=~pb(x), family="SSTtr")
GAIC(g0p, t0p, t0sst,t0ssttr )

##                df       AIC
## g0p     13.398137 -2819.590
## t0p     13.398227 -2819.590
## t0sst   16.115402 -2812.341
## t0ssttr  9.853711 -2173.978
```

As we would generally expect, since we have generated the data using the beta distribution inflated at 0, the two extra fitted models performed worst according to criterion AIC.

Note that `logitSST`(μ,σ,ν,τ) and `SSTtr`(μ,σ,ν,τ) distributions have four parameters so their corresponding inflated at 0 distributions have five parameters (including the extra $\xi_0 = p_0$). Hence models `t0sst` and `t0ssttr` above can include models for parameters ν and τ, e.g, `nu.fo=~pb(x)` and `tau.fo=~pb(x)`, as well models for μ, ν and ξ_0.

6.3.3 Generalized Tobit model distributions on $[0, 1)$

In general for a restricted values response variable, that is, having a distribution with a restricted range, the Tobit model (which requires a survival analysis response variable), can be appropriate. Here we show how this model can be fitted within `gamlss`. Note though that for Tobit model the probability at zero is **not** modelled independently as a function of explanatory variables but is equal to the probability of being censored below zero. For more details see Hossain et al. [2016a] and Hossain et al. [2016b]. Below we fit a Tobit normal and a Tobit `SST` model both left censored at 0 to provide a point probability at 0.

```
library(survival)
y0surv<- Surv(y0, y0!=0, type="left")
# creating the distribution
library(gamlss.cens)
# Gaussian
gen.cens("NO", type="left")

## A censored family of distributions from NO has been generated
##  and saved under the names:
##  dNOlc pNOlc qNOlc NOlc
## The type of censoring is left

# SST distribution
gen.cens("SST", type="left")

## A censored family of distributions from SST has been generated
##  and saved under the names:
##  dSSTlc pSSTlc qSSTlc SSTlc
## The type of censoring is left

# fitting the  model
# Tobit model
s0no <- gamlss( y0surv ~ pb(x), sigma.formula=~pb(x),
                family=NOlc)

## GAMLSS-RS iteration 1: Global Deviance = -1923.728
## GAMLSS-RS iteration 2: Global Deviance = -1923.348
## GAMLSS-RS iteration 3: Global Deviance = -1923.369
## GAMLSS-RS iteration 4: Global Deviance = -1923.371
## GAMLSS-RS iteration 5: Global Deviance = -1923.372

# generalized Tobit
s0sst <- gamlss( y0surv ~ pb(x), sigma.formula=~pb(x),
                family=SSTlc)

## GAMLSS-RS iteration 1: Global Deviance = -1356.215
## GAMLSS-RS iteration 2: Global Deviance = -1809.66
## GAMLSS-RS iteration 3: Global Deviance = -1878.17
## GAMLSS-RS iteration 4: Global Deviance = -1921.45
## GAMLSS-RS iteration 5: Global Deviance = -1944.657
## GAMLSS-RS iteration 6: Global Deviance = -1956.981
## GAMLSS-RS iteration 7: Global Deviance = -1963.848
```

```
## GAMLSS-RS iteration 8: Global Deviance = -1967.505
## GAMLSS-RS iteration 9: Global Deviance = -1969.736
## GAMLSS-RS iteration 10: Global Deviance = -1970.959
## GAMLSS-RS iteration 11: Global Deviance = -1971.647
## GAMLSS-RS iteration 12: Global Deviance = -1972.008
## GAMLSS-RS iteration 13: Global Deviance = -1972.177
## GAMLSS-RS iteration 14: Global Deviance = -1972.232
## GAMLSS-RS iteration 15: Global Deviance = -1972.229
## GAMLSS-RS iteration 16: Global Deviance = -1972.169
## GAMLSS-RS iteration 17: Global Deviance = -1972.127
## GAMLSS-RS iteration 18: Global Deviance = -1972.083
## GAMLSS-RS iteration 19: Global Deviance = -1972.044
## GAMLSS-RS iteration 20: Global Deviance = -1972.009
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
GAIC(g0p, t0p, t0sst,t0ssttr, s0no, s0sst )
```

```
##                df       AIC
## g0p     13.398137 -2819.590
## t0p     13.398227 -2819.590
## t0sst   16.115402 -2812.341
## t0ssttr  9.853711 -2173.978
## s0sst    6.267237 -1959.475
## s0no    12.274395 -1898.823
```

Note that the model fitting for `s0sst` has not quite converged. To obtain convergence use the argument `n.cyc` but be warned that it needs 885 iteration to converge. The Tobit models do not perform as well as the inflated models in this case, which is not surprising since the data were simulated from a beta inflated distribution, `BEINF0`(μ, σ, ν). The `SSTlc`(μ, σ, ν, τ) distribution has four parameters. Hence model `s0sst` can include models for parameters ν and τ.

6.4 Fitting a regression model on $(0, 1]$

To model the data used on the right top side of Figure 12, we will use the same type of models as in the previous section, but this time inflated at one. The models are:

- the `BEINF1` distribution using `gamlss()`
- the `BEOI` distribution using `gamlss()`
- the beta inflated using `gamlssInf0to1()`
- the inflated `logitSST` using `gamlssInf0to1()`
- the inflated truncated `SST` using `gamlssInf0to1()`
- Tobit `NO` using `gamlss()`
- Tobit `SST` using `gamlss()`

```
# BEINF1
g1p <- gamlss(y1~pb(x), sigma.fo=~pb(x), nu.fo=~pb(x), family=BEINF1)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -1533.075
## GAMLSS-RS iteration 2: Global Deviance = -2151.917
## GAMLSS-RS iteration 3: Global Deviance = -2346.826
## GAMLSS-RS iteration 4: Global Deviance = -2371.192
## GAMLSS-RS iteration 5: Global Deviance = -2371.563
## GAMLSS-RS iteration 6: Global Deviance = -2371.562
## GAMLSS-RS iteration 7: Global Deviance = -2371.561
```

```
# BEOI
g1p1 <- gamlss(y1~pb(x), sigma.fo=~pb(x), nu.fo=~pb(x), family=BEOI)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -818.9885
## GAMLSS-RS iteration 2: Global Deviance = -1709.734
## GAMLSS-RS iteration 3: Global Deviance = -2162.788
## GAMLSS-RS iteration 4: Global Deviance = -2340.568
## GAMLSS-RS iteration 5: Global Deviance = -2370.671
## GAMLSS-RS iteration 6: Global Deviance = -2371.544
## GAMLSS-RS iteration 7: Global Deviance = -2371.543
## GAMLSS-RS iteration 8: Global Deviance = -2371.543
```

```
# BE inflated at 1
t1p <- gamlssInf0to1(y=y1, mu.fo=~pb(x), sigma.fo=~pb(x),
                     xi1.fo=~pb(x), trace=TRUE, family="BE")
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 420.6815
## GAMLSS-RS iteration 2: Global Deviance = 419.762
## GAMLSS-RS iteration 3: Global Deviance = 419.7616
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -1953.708
## GAMLSS-RS iteration 2: Global Deviance = -2571.681
## GAMLSS-RS iteration 3: Global Deviance = -2766.59
## GAMLSS-RS iteration 4: Global Deviance = -2790.954
## GAMLSS-RS iteration 5: Global Deviance = -2791.324
## GAMLSS-RS iteration 6: Global Deviance = -2791.323
## GAMLSS-RS iteration 7: Global Deviance = -2791.322
##             The Final Global Deviance  = -2371.561
```

```
# logitSST inflated at 1
gen.Family("SST", "logit")
```

```
## A  logit  family of distributions from SST has been generated
##  and saved under the names:
##  dlogitSST plogitSST qlogitSST rlogitSST logitSST
```

```
t1sst <- gamlssInf0to1(y=y1, mu.fo=~pb(x), sigma.fo=~pb(x),
                  xi1.fo=~pb(x), family="logitSST", trace=T)
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 420.6815
## GAMLSS-RS iteration 2: Global Deviance = 419.762
## GAMLSS-RS iteration 3: Global Deviance = 419.7616
```

```
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -2753.826
## GAMLSS-RS iteration 2: Global Deviance = -2776.583
## GAMLSS-RS iteration 3: Global Deviance = -2782.89
## GAMLSS-RS iteration 4: Global Deviance = -2783.245
## GAMLSS-RS iteration 5: Global Deviance = -2783.26
## GAMLSS-RS iteration 6: Global Deviance = -2783.272
## GAMLSS-RS iteration 7: Global Deviance = -2783.27
## GAMLSS-RS iteration 8: Global Deviance = -2783.27
##             The Final Global Deviance  = -2363.508
```

```
# truncated SST inflated at 1
# generate a truncated distribution from 0 to 1
library(gamlss.tr)
gen.trun(c(0,1),"SST",type="both")
```

```
## A truncated family of distributions from SST has been generated
##  and saved under the names:
##  dSSTtr pSSTtr qSSTtr rSSTtr SSTtr
## The type of truncation is both
##  and the truncation parameter is 0 1
```

```
# fit for starting values
m1 <-gamlss(y1~pb(x), sigma.fo=~pb(x), family=SST)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -3040.191
## GAMLSS-RS iteration 2: Global Deviance = -3085.284
## GAMLSS-RS iteration 3: Global Deviance = -3091.881
## GAMLSS-RS iteration 4: Global Deviance = -3092.604
## GAMLSS-RS iteration 5: Global Deviance = -3092.64
## GAMLSS-RS iteration 6: Global Deviance = -3092.646
## GAMLSS-RS iteration 7: Global Deviance = -3092.652
## GAMLSS-RS iteration 8: Global Deviance = -3092.652
```

```
# fit model
t1ssttr <- gamlssInf0to1(y=y1, mu.fo=~pb(x), sigma.fo=~pb(x),
      xi1.fo=~pb(x), family="SSTtr",
      sigma.start=fitted(m1,"sigma"), trace=T)
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 420.6815
## GAMLSS-RS iteration 2: Global Deviance = 419.762
## GAMLSS-RS iteration 3: Global Deviance = 419.7616
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -2480.116
## GAMLSS-RS iteration 2: Global Deviance = -2035.495
## GAMLSS-RS iteration 3: Global Deviance = -2012.888
## GAMLSS-RS iteration 4: Global Deviance = -2054.668
## GAMLSS-RS iteration 5: Global Deviance = -2093.23
## GAMLSS-RS iteration 6: Global Deviance = -2122.096
## GAMLSS-RS iteration 7: Global Deviance = -2149.248
```

```
## GAMLSS-RS iteration 8: Global Deviance = -2168.106
## GAMLSS-RS iteration 9: Global Deviance = -2180.094
## GAMLSS-RS iteration 10: Global Deviance = -2188.406
## GAMLSS-RS iteration 11: Global Deviance = -2194.776
## GAMLSS-RS iteration 12: Global Deviance = -2199.887
## GAMLSS-RS iteration 13: Global Deviance = -2204.118
## GAMLSS-RS iteration 14: Global Deviance = -2207.63
## GAMLSS-RS iteration 15: Global Deviance = -2210.681
## GAMLSS-RS iteration 16: Global Deviance = -2213.322
## GAMLSS-RS iteration 17: Global Deviance = -2215.657
## GAMLSS-RS iteration 18: Global Deviance = -2217.733
## GAMLSS-RS iteration 19: Global Deviance = -2219.6
## GAMLSS-RS iteration 20: Global Deviance = -2221.288

## Warning in RS(): Algorithm RS has not yet converged

##             The Final Global Deviance  = -1801.526
```

Below we fit a Tobit model and a Tobit SSD model, both right censored at 1 to provide a point probability at 1. Since $0 < Y \leq 1$ it may be preferable to fit a Tobit model from a distribution on $(0, \infty)$ censored at 1, for example example a BCCG model, see Hossain et al. [2016a].

```
# Tobit models
library(survival)
# creating the y variable as survival response
y1surv<- Surv(y1, y1!=1, type="right")
# creating the distributions
library(gamlss.cens)
gen.cens("SST", type="right")

## A censored family of distributions from SST has been generated
##  and saved under the names:
##  dSSTrc pSSTrc qSSTrc SSTrc
## The type of censoring is right

gen.cens("NO", type="right")

## A censored family of distributions from NO has been generated
##  and saved under the names:
##  dNOrc pNOrc qNOrc NOrc
## The type of censoring is right

# fitting the  models
# tobit model
s1no <- gamlss( y1surv ~ pb(x), sigma.formula=~pb(x),
                family=NOrc)

## GAMLSS-RS iteration 1: Global Deviance = -2150.399
## GAMLSS-RS iteration 2: Global Deviance = -2228.728
## GAMLSS-RS iteration 3: Global Deviance = -2240.273
## GAMLSS-RS iteration 4: Global Deviance = -2242.955
## GAMLSS-RS iteration 5: Global Deviance = -2243.266
```

```
## GAMLSS-RS iteration 6: Global Deviance = -2243.183
## GAMLSS-RS iteration 7: Global Deviance = -2243.076
## GAMLSS-RS iteration 8: Global Deviance = -2242.996
## GAMLSS-RS iteration 9: Global Deviance = -2242.931
## GAMLSS-RS iteration 10: Global Deviance = -2242.88
## GAMLSS-RS iteration 11: Global Deviance = -2242.842
## GAMLSS-RS iteration 12: Global Deviance = -2242.815
## GAMLSS-RS iteration 13: Global Deviance = -2242.795
## GAMLSS-RS iteration 14: Global Deviance = -2242.781
## GAMLSS-RS iteration 15: Global Deviance = -2242.771
## GAMLSS-RS iteration 16: Global Deviance = -2242.764
## GAMLSS-RS iteration 17: Global Deviance = -2242.759
## GAMLSS-RS iteration 18: Global Deviance = -2242.756
## GAMLSS-RS iteration 19: Global Deviance = -2242.753
## GAMLSS-RS iteration 20: Global Deviance = -2242.752
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
# generalised Tobit
s1sst <- gamlss( y1surv ~ pb(x), sigma.formula=~pb(x),
              family=SSTrc)
```

```
## GAMLSS-RS iteration 1: Global Deviance = -1442.216
## GAMLSS-RS iteration 2: Global Deviance = -1457.684
## GAMLSS-RS iteration 3: Global Deviance = -1521.539
## GAMLSS-RS iteration 4: Global Deviance = -1540.201
## GAMLSS-RS iteration 5: Global Deviance = -1545.946
## GAMLSS-RS iteration 6: Global Deviance = -1547.59
## GAMLSS-RS iteration 7: Global Deviance = -1548.87
## GAMLSS-RS iteration 8: Global Deviance = -1548.966
## GAMLSS-RS iteration 9: Global Deviance = -1549.189
## GAMLSS-RS iteration 10: Global Deviance = -1549.204
## GAMLSS-RS iteration 11: Global Deviance = -1549.301
## GAMLSS-RS iteration 12: Global Deviance = -1549.28
## GAMLSS-RS iteration 13: Global Deviance = -1549.313
## GAMLSS-RS iteration 14: Global Deviance = -1549.306
## GAMLSS-RS iteration 15: Global Deviance = -1549.317
## GAMLSS-RS iteration 16: Global Deviance = -1549.316
```

```
GAIC(g1p, g1p1, t1p, t1sst, t1ssttr, s1no, s1sst)
```

```
##                df       AIC
## t1p     12.386411 -2346.788
## g1p     12.386737 -2346.788
## g1p1    12.384038 -2346.774
## t1sst   17.705054 -2328.098
## s1no    14.563539 -2213.625
## t1ssttr  9.314857 -1782.897
## s1sst    6.225857 -1536.865
```

Note that two of the model have not converged but those two will be no contestant for the best

model.

6.5 Fitting a regression model on $[0, 1]$

Here we fit four different models to the data shown on the bottom of Figure 12. The models are:

- the BEINF distribution using gamlss()
- the beta inflated at zero and one using gamlssInf0to1()
- the inflated logitSST using gamlssInf0to1()
- the inflated truncated (at zero and one) SST using gamlssInf0to1()

```
# BEINF using gamlss
g01p <- gamlss(y01~pb(x), sigma.fo=~pb(x), nu.fo=~pb(x), tau.fo=~pb(x),
               family=BEINF)

## GAMLSS-RS iteration 1: Global Deviance = -993.1583
## GAMLSS-RS iteration 2: Global Deviance = -1572.375
## GAMLSS-RS iteration 3: Global Deviance = -1764.435
## GAMLSS-RS iteration 4: Global Deviance = -1793.266
## GAMLSS-RS iteration 5: Global Deviance = -1794.008
## GAMLSS-RS iteration 6: Global Deviance = -1794.064
## GAMLSS-RS iteration 7: Global Deviance = -1794.071
## GAMLSS-RS iteration 8: Global Deviance = -1794.072

# Beta inflated using gamlssInf0to1
t01p <- gamlssInf0to1(y=y01, mu.fo=~pb(x), sigma.fo=~pb(x),
               xi0.fo=~pb(x),   xi1.fo=~pb(x),  trace=TRUE, family="BE")

## ******      The multinomial model        ******
## GAMLSS-RS iteration 1: Global Deviance = 729.0011
## GAMLSS-RS iteration 2: Global Deviance = 726.3389
## GAMLSS-RS iteration 3: Global Deviance = 726.3382
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -1722.205
## GAMLSS-RS iteration 2: Global Deviance = -2298.72
## GAMLSS-RS iteration 3: Global Deviance = -2490.773
## GAMLSS-RS iteration 4: Global Deviance = -2519.604
## GAMLSS-RS iteration 5: Global Deviance = -2520.346
## GAMLSS-RS iteration 6: Global Deviance = -2520.402
## GAMLSS-RS iteration 7: Global Deviance = -2520.41
## GAMLSS-RS iteration 8: Global Deviance = -2520.41
##             The Final Global Deviance  = -1794.072

# logistic SST using gamlssInf0to1
gen.Family("SST", "logit")

## A  logit  family of distributions from SST has been generated
##  and saved under the names:
```

```
##  dlogitSST plogitSST qlogitSST rlogitSST logitSST

t01sst <- gamlssInf0to1(y=y01, mu.fo=~pb(x), sigma.fo=~pb(x),
           xi0.fo=~pb(x), xi1.fo=~pb(x), family="logitSST", trace=TRUE)

## ******      The multinomial model        ******
## GAMLSS-RS iteration 1: Global Deviance = 729.0011
## GAMLSS-RS iteration 2: Global Deviance = 726.3389
## GAMLSS-RS iteration 3: Global Deviance = 726.3382
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -2506.09
## GAMLSS-RS iteration 2: Global Deviance = -2518.508
## GAMLSS-RS iteration 3: Global Deviance = -2521.592
## GAMLSS-RS iteration 4: Global Deviance = -2521.735
## GAMLSS-RS iteration 5: Global Deviance = -2521.75
## GAMLSS-RS iteration 6: Global Deviance = -2521.75
##             The Final Global Deviance  = -1795.411

# generate a truncated distribution from 0 to 1
m1 <-gamlss(y1~pb(x), sigma.fo=~pb(x), family=SST, trace=FALSE)
gen.trun(c(0,1),"SST",type="both")

## A truncated family of distributions from SST has been generated
##  and saved under the names:
##  dSSTtr pSSTtr qSSTtr rSSTtr SSTtr
## The type of truncation is both
##  and the truncation parameter is 0 1

t01ssttr <- gamlssInf0to1(y=y1, mu.fo=~pb(x), sigma.fo=~pb(x),
       xi0.fo=~pb(x),xi1.fo=~pb(x), family="SSTtr",  trace=TRUE,
       sigma.start=fitted(m1,"sigma"))

## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 420.6815
## GAMLSS-RS iteration 2: Global Deviance = 419.762
## GAMLSS-RS iteration 3: Global Deviance = 419.7616
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -2480.116
## GAMLSS-RS iteration 2: Global Deviance = -2035.495
## GAMLSS-RS iteration 3: Global Deviance = -2012.888
## GAMLSS-RS iteration 4: Global Deviance = -2054.668
## GAMLSS-RS iteration 5: Global Deviance = -2093.23
## GAMLSS-RS iteration 6: Global Deviance = -2122.096
## GAMLSS-RS iteration 7: Global Deviance = -2149.248
## GAMLSS-RS iteration 8: Global Deviance = -2168.106
## GAMLSS-RS iteration 9: Global Deviance = -2180.094
## GAMLSS-RS iteration 10: Global Deviance = -2188.406
## GAMLSS-RS iteration 11: Global Deviance = -2194.776
## GAMLSS-RS iteration 12: Global Deviance = -2199.887
## GAMLSS-RS iteration 13: Global Deviance = -2204.118
## GAMLSS-RS iteration 14: Global Deviance = -2207.63
```

```
## GAMLSS-RS iteration 15: Global Deviance = -2210.681
## GAMLSS-RS iteration 16: Global Deviance = -2213.322
## GAMLSS-RS iteration 17: Global Deviance = -2215.657
## GAMLSS-RS iteration 18: Global Deviance = -2217.733
## GAMLSS-RS iteration 19: Global Deviance = -2219.6
## GAMLSS-RS iteration 20: Global Deviance = -2221.288
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
##             The Final Global Deviance  = -1801.526
```

```
AIC(g01p, t01p, t01sst, t01ssttr)
```

```
##                 df       AIC
## t01ssttr  9.314857 -1782.897
## g01p     14.897741 -1764.276
## t01p     14.898541 -1764.275
## t01sst   16.697599 -1762.016
```

Surprisingly the truncated `SST` distribution is doing well for this data set.

7 Conclusions

GAMLSS is a framework where different models can be fitted and compared. In this vignette, we have shown how models with response variable restricted to values from zero to one (including 0 and/or 1) can be fitted within the GALMSS framework.

The `gamlssInf0to1()` function can be used to fit a variety of different models in which the response variable lies between zero and one including zero and/or one. The function allows any continuous distribution on interval $(0, 1)$, available in the **gamlss** packages, to be inflated with point probabilities at 0 and/or 1. The continuous distribution on $(0, 1)$ can be a distribution that exists already within `gamlss.dist`, or it can be created using the `gen.Family()` or `gen.trun()` functions for transformed or truncated distribution, respectively. In addition function `gen.Inf0to1()` generates `d`, `p`, `q`, `r` and plot functions for a distribution defined between 0 and 1, inflated with point probability at 0 and/or 1.

More information about GAMLSS can be found in Stasinopoulos et al. [2017] or the GAMLSS website `www.gamlss.org`. We're hoping that the **gamlss.inf** package will be useful.

References

A. Hossain, R.A. Rigby, D.M. Stasinopoulos, and M. Enea. Centile estimation for a proportion response variable. Statistics in Medicine, 35(6):895–904, 2016a. URL `http://dx.doi.org/10.1002/sim.6748`.

A. Hossain, R.A. Rigby, D.M. Stasinopoulos, and M Enea. A flexible approach for modelling a proportion response variable: Loss given default. In Proceedings of the 31th International Workshop on Statistical Modelling,, pages 127–132, 2016b.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape, (with discussion). Applied Statistics, 54:507–554, 2005.

D. M. Stasinopoulos and R. A. Rigby. Generalized additive models for location scale and shape (GAMLSS) in R. Journal of Statistical Software, 23(7):1–46, 2007.

D. M. Stasinopoulos, R. A. Rigby, G. Z. Heller, V. Voudouris, and F. De Bastiani. Flexible Regression and Smoothing: Using GAMLSS in R. Chapman and Hall, Boca Raton, 2017.