# Zero adjusted distributions on the positive real line

**Mikis Stasinopoulos, Marco Enea and Robert A. Rigby**

April 29, 2017

## Contents



## 1 Introduction

The new R package `gamlss.inf` is designed to fit both inflated distributions on the interval $[0, 1]$, including 0 and 1, and also zero adjusted distributions defined on $[0, \infty)$ including 0. This vignette focuses on the zero adjusted distributions defined on $[0, \infty)$ including 0 (i.e. the non-negative real line). This is a mixed discrete-continuous distribution, comprising a value $Y = 0$ with probability $p_0$ and $Y = Y_1$ with probability $(1 - p_0)$ where $Y_1$ has a continuous distribution on the interval $(0, \infty)$, i.e. the positive real line.

The **gamlss** package, Rigby and Stasinopoulos [2005], Stasinopoulos and Rigby [2007], Stasinopoulos et al. [2017], already provides two zero adjusted distributions, the zero adjusted gamma, `ZAGA` and the zero adjusted inverse Gaussian, `ZAIG`. The probability at the value $Y = 0$ may depend on explanatory variables. Note the gamma distribution has 2 parameters, so the zero adjusted gamma with probability at 0 has a total of 3 parameters. In practice, and for complicated data sets, the part of the response which lies on positive real line may need more than 2 distribution parameters to be captured correctly.

There are three methods within gamlss packages to obtain a more flexible continuous distribution with up to 4 parameters with range $(0, \infty)$, i.e. the positive real line, not including zero:

1. use a flexible continuous distribution with range $(0, \infty)$ within the **gamlss.dist** package, e.g. `BCCG`$(\mu, \sigma, \nu)$, `BCT`$(\mu, \sigma, \nu, \tau)$ or `BCPE`$(\mu, \sigma, \nu, \tau)$, see Section 2.1
2. Generate (i.e. create) a flexible continuous distribution on $(0, \infty)$ by transforming any continuous distribution with range $(-\infty, \infty)$ available in the **gamlss.dist** package to range $(0, \infty)$ using the inverse log (i.e. exponential) transformation through the function `gen.Family()` with argument `type="log"`, see Section 2.2
3. Generate (i.e. create) a flexible continuous distribution with range $(0, \infty)$ by truncating any continuous distribution on $(-\infty, \infty)$ available in **gamlss.dist** package to range $(0, \infty)$ using the function `gen.trun()` of the package **gamlss.tr**, see Section 2.3

The new **R** package **gamlss.inf** allows any continuous distribution on $(0, \infty)$, obtained by any of the above three methods, to be zero adjusted to range $[0, \infty)$, by including a point probability at zero.

So the package **gamlss.inf** enhances the capability of the standard `gamlss()` function by allowing an extra parameter for modelling the probability at zero. The overall distribution can then have up to five parameters. Let $\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}$ represent the four parameters of the distribution defined on $(0, \infty)$ and parameter $\boldsymbol{\xi}_0 = p_0$, the probability at 0. Then the general zero adjusted model that the new package `gamlss.inf` can fit can be written as:

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

where $\mathcal{D}(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\nu}, \boldsymbol{\tau}, \boldsymbol{\xi}_0)$ is a zero adjusted distribution for the response variable $Y$ defined on $[0, \infty)$ including 0, 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 $k = 1, 2, 3, 4, 5$ 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 $\boldsymbol{\eta}_1$, $\boldsymbol{\eta}_2$, $\boldsymbol{\eta}_3$, $\boldsymbol{\eta}_4$ and $\boldsymbol{\eta}_5$ are called the *predictors* of the distribution parameters $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, $\boldsymbol{\nu}$, $\boldsymbol{\tau}$ and $\boldsymbol{\xi}_0$ respectively. Note that $\xi_0 = p_0 = P(Y = 0)$.

# 2 Distributions on the positive real line

## 2.1 Explicit continuous distributions on $(0, \infty)$

Within the `gamlss.dist` package there are currently several continuous distributions defined on $(0, \infty)$, including:

1. the exponential distribution, EXP($\mu$), with one parameter,
2. the gamma distribution, GA($\mu, \sigma$), with two parameters,
3. the inverse gamma distribution, IGAMMA($\mu, \sigma$), with two parameters,
4. the inverse Gaussian distribution, IG($\mu, \sigma$), with two parameters,
5. the log normal distribution, LOGNO($\mu, \sigma$), with two parameters
6. the Pareto type 2 distribution, PARETO2($\mu, \sigma$), with two parameters
7. the Weibull distribution, WEI($\mu, \sigma$), WEI2($\mu, \sigma$), WEI3($\mu, \sigma$), with two parameters
8. the Box-Cox Cole and Green distribution, BCCG($\mu, \sigma, \nu$) or BCCGo($\mu, \sigma, \nu$), with three parameters.
9. the generalized gamma distribution, GG($\mu, \sigma, \nu$), with three parameters.
10. the generalized inverse Gaussian distribution, GIG($\mu, \sigma, \nu$), with three parameters.
11. the Box-Cox $t$ distribution, BCT($\mu, \sigma, \nu, \tau$) or BCTo($\mu, \sigma, \nu, \tau$), with four parameters.
12. the Box-Cox power exponential distribution, BCPE($\mu, \sigma, \nu, \tau$) or BCPEo($\mu, \sigma, \nu, \tau$) ,with four parameters.
13. the generalized beta type 2 distribution, GB2($\mu, \sigma, \nu, \tau$), with four parameters

## 2.2 Log transform distributions on $(0, \infty)$.

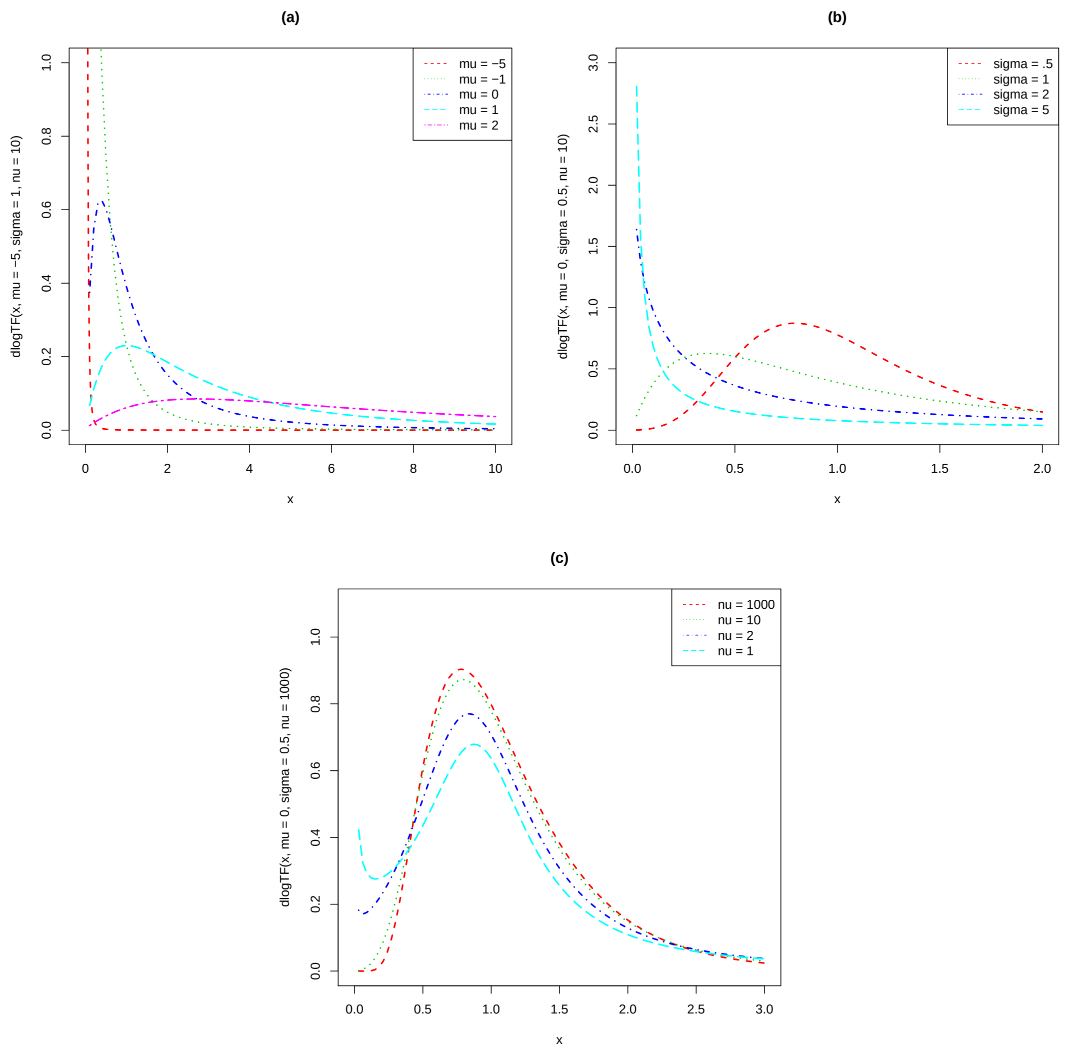
Any continuous random variable say $Z$ defined on $(-\infty, \infty)$ can be transformed by the inverse log (i.e. exponential) transformation $Y = \exp(Z)$ to a random variable Y defined on $(0, \infty)$. The resulting distribution is called a log-transform distribution. For example, if $Z$ is a $t$-family distributed variable i.e. $Z \sim$ TF($\mu, \sigma, \nu$), and the inverse log transformation is applied, then $Y \sim$ logTF($\mu, \sigma, \nu$), i.e. a log-$t$ family distribution on $(0, \infty)$.

The following is an example on how to take a `gamlss.family` distribution on $(-\infty, \infty)$ and create a corresponding log-transform distribution defined on $(0, \infty)$. 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 log-transform distribution, together with the function which can be used for fitting within **gamlss**. Here first generate a log-$t$ family distribution, i.e. logTF($\mu, \sigma, \nu$), and plot the distribution for different values of $\mu$, $\sigma$ and $\nu$. Note that $\mu$, $\sigma$ and $\nu$ are defined on the original $t$-distribution ranges, i.e. $(-\infty, \infty)$ for $\mu$ and $(0, \infty)$ for $\sigma$ and $\nu$. This implies that $\exp(\mu)$ is not the mean of the log-transform, logTF($\mu, \sigma, \nu$), distribution but its median. Also $\sigma$ and $\nu$ are related to the scale and shape of the distribution. We use `gen.Family("TF", type="log")` to generate a log-$t$ family distribution and in Figure 1 we plot the distribution for different values of $\mu$, $\sigma$ and $\nu$ using the function `curve()`.

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

## A  log  family of distributions from TF has been generated
##  and saved under the names:
##  dlogTF plogTF qlogTF rlogTF logTF

# different mu
curve(dlogTF(x, mu=-5, sigma=1, nu=10), 0,10, n = 1001, ylim=c(0,1),
      lwd=2, lty=2, col=2)
title("(a)")
curve(dlogTF(x, mu=-1, sigma=1, nu=10), 0,10, add=TRUE, lwd=2, lty=3, col=3)
curve(dlogTF(x, mu=0,  sigma=1, nu=10), 0,10, add=TRUE, lwd=2, lty=4, col=4)
curve(dlogTF(x, mu=1,  sigma=1, nu=10), 0,10, add=TRUE, lwd=2, lty=5, col=5)
curve(dlogTF(x, mu=2,  sigma=1, nu=10), 0,10, add=TRUE, lwd=2, lty=6, col=6)
legend("topright",
       legend=c("mu = -5","mu = -1","mu = 0","mu = 1","mu = 2"),
       lty=2:6, col=2:6, cex=1)
# different sigma
curve(dlogTF(x, mu=0, sigma=.5, nu=10), 0,2, ylim=c(0,3),
      lwd=2, lty=2, col=2)
title(("(b)"))
curve(dlogTF(x, mu=0, sigma=1, nu=10),  0,2, add=TRUE, lwd=2, lty=3, col=3)
curve(dlogTF(x, mu=0, sigma=2, nu=10),  0,2, add=TRUE, lwd=2, lty=4, col=4)
curve(dlogTF(x, mu=0, sigma=5, nu=10 ), 0,2, add=TRUE, lwd=2, lty=5, col=5)
legend("topright",
       legend=c("sigma = .5","sigma = 1","sigma = 2","sigma = 5"),
       lty=2:5, col=2:5, cex=1)
# different nu
curve(dlogTF(x, mu=0, sigma=.5, nu=1000), 0,3, ylim=c(0,1.1),
       lwd=2, lty=2, col=2)
title("(c)")
curve(dlogTF(x, mu=0, sigma=.5, nu=10), 0,3, add=TRUE, lwd=2, lty=3, col=3)
curve(dlogTF(x, mu=0, sigma=.5, nu=2 ), 0,3, add=TRUE, lwd=2, lty=4, col=4)
curve(dlogTF(x, mu=0, sigma=.5, nu=1 ), 0,3, add=TRUE, lwd=2, lty=5, col=5)
legend("topright",
       legend=c("nu = 1000","nu = 10","nu = 2","nu = 1"),
       lty=2:5, col=2:5, cex=1)
```

Figure 1

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




**R** code on page 4

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

## 2.3 Truncated distributions on $(0, \infty)$.

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

Below we transform a skew student $t$, `SST`$(\mu, \sigma, \nu, \tau)$, distribution by left truncating it at zero to give a truncated `SST` distribution, `SSTtr`$(\mu, \sigma, \nu, \tau)$ defined on $(0, \infty)$. The range of each parameter of `SSTtr`$(\mu, \sigma, \nu, \tau)$ is the same as for `SST`$(\mu, \sigma, \nu, \tau)$, i.e. $-\infty < \mu < \infty$, $\sigma > 0$, $\nu > 0$ and $\tau > 2$.

Figure 2

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

## 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 left
##  and the truncation parameter is 0

# different mu
curve(dSSTtr(x, mu=.1, sigma=.5, nu=1, tau=10), 0,4, lwd=2, lty=2, col=2)
curve(dSSTtr(x, mu= 1, sigma=.5, nu=1, tau=10), 0,4, lwd=2, lty=3, col=3,
      add=TRUE)
curve(dSSTtr(x, mu= 2, sigma=.5, nu=1, tau=10), 0,4, lwd=2, lty=4, col=4,
       add=TRUE)
title("(a)")
legend("topright",
       legend=c("mu = .1","mu = 1","mu = 2"),
       lty=2:4, col=2:4, cex=1)
# different sigma
curve(dSSTtr(x, mu=2, sigma=0.5, nu=1, tau=10), 0,4, lwd=2, lty=2, col=2 )
curve(dSSTtr(x, mu=2, sigma=  1, nu=1, tau=10), 0,4, lwd=2, lty=3, col=3,
      add=TRUE)
curve(dSSTtr(x, mu=2, sigma=  2, nu=1, tau=10), 0,4, lwd=2, lty=4, col=4,
      add=TRUE)
title("(b)")
legend("topright",
       legend=c("sigma = .5","sigma = 1","sigma = 2"),
       lty=2:4, col=2:4, cex=1)
# different nu
curve(dSSTtr(x, mu=2, sigma=.5, nu=0.1, tau=10), 0,4, lwd=2, lty=2, col=2)
curve(dSSTtr(x, mu=2, sigma=.5, nu=  1, tau=10), 0,4, lwd=2, lty=3, col=3,
      add=TRUE)
curve(dSSTtr(x, mu=2, sigma=.5, nu=  2, tau=10), 0,4, lwd=2, lty=4, col=4,
      add=TRUE)
title("(c)")
legend("topright",
       legend=c("nu = 0.1","nu = 1","nu = 2"),
       lty=2:4, col=2:4, cex=1)
```

```r
# different tau
curve(dSSTtr(x, mu=2, sigma=.5, nu=1, tau=  3),  0,4, lwd=2, lty=2, col=2)
curve(dSSTtr(x, mu=2, sigma=.5, nu=1, tau=  5),  0,4, lwd=2, lty=3, col=3,
      add=TRUE)
curve(dSSTtr(x, mu=2, sigma=.5, nu=1, tau=100),  0,4, lwd=2, lty=4, col=4,
       add=TRUE)
title("(d)")
legend("topright",
       legend=c("tau = 3","tau = 5","tau = 100"),
       lty=2:4, col=2:4, cex=1)
```

Figure 2 shows the different shapes of the $SSTtr(\mu, \sigma, \nu, \tau)$ distribution can take. Panel (a) shows, for fixed $\sigma = 0.5$ $\nu = 1$ and $\tau = 10$, how the distribution changes for different values of $\mu = (0.1, 1, 2)$. Panel (b) fixes $\mu = 2$, $\nu = 1$ and $\tau = 10$ and varies $\sigma = (0.5, 1, 2)$. Panel (c) fixes $\mu = 2$, $\sigma = 0.5$ and $\tau = 10$ and varies $\nu = (0.1, 1, 2)$. Panel (d) fixes $\mu = 2$, $\sigma = 0.5$ and $\nu = 1$ and varies $\tau = (3, 5, 100)$.

# 3 Generating zero adjusted distributions

Next it is shown how any `gamlss.family` distribution defined on $(0, \infty)$ can be zero adjusted to $[0, \infty)$, by including a point probability at 0.

The function `gen.Zadj()` takes as an argument a `gamlss.family` distribution on $(0, \infty)$, obtained by any of the three methods of Section 2, and generates a zero adjusted version of the distribution having a point probability at 0. The function has only one argument, `family`, which specifies a distribution family on $(0, \infty)$.

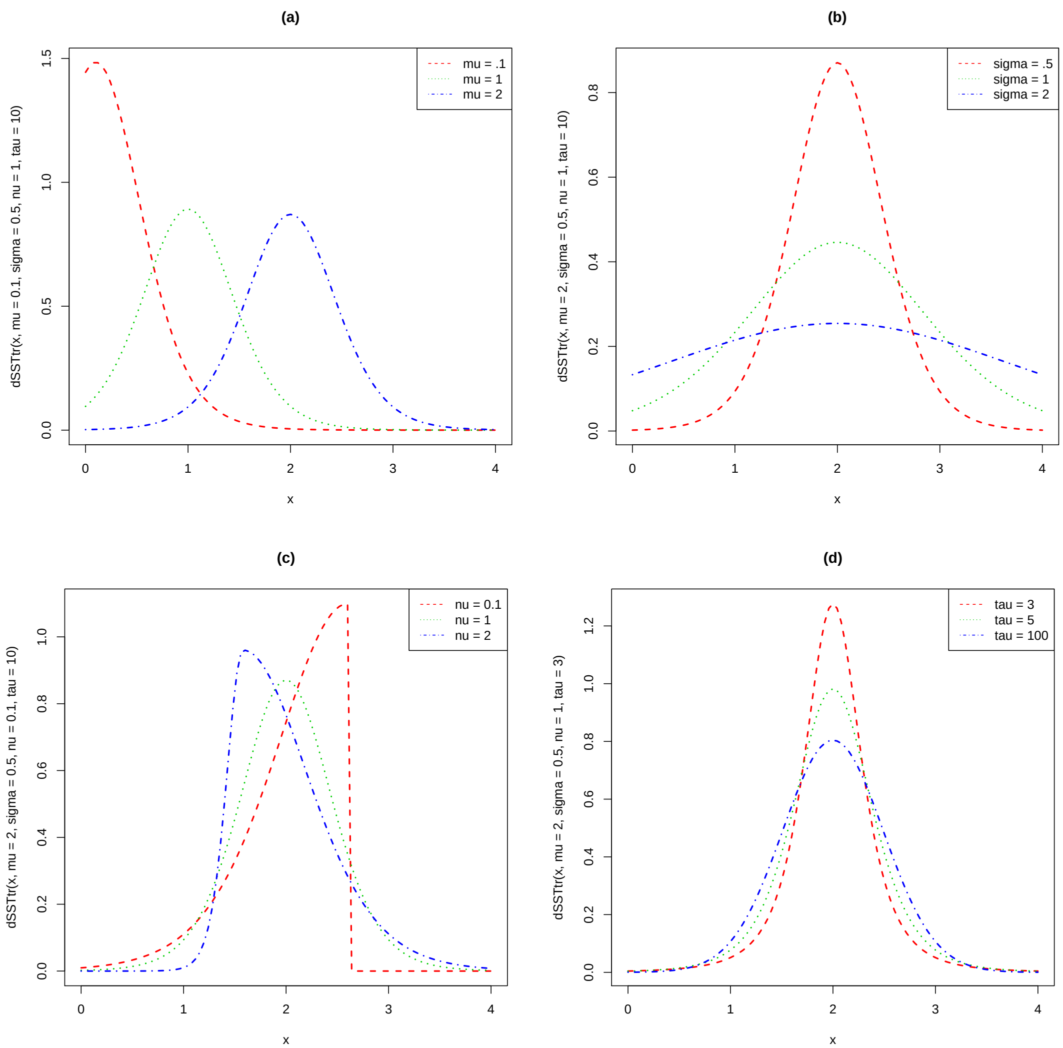
The resulting mixed (continuous-discrete) probability (density) function (pdf) is given by:

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

for $0 \leq y \leq \infty$, where $f_W(y|\boldsymbol{\theta})$ is any probability density function defined on $(0, \infty)$, i.e. for $0 < y < \infty$, with parameters $\boldsymbol{\theta}^\top = (\theta_1, \theta_2, \ldots, \theta_p)$ and $0 < \xi_0 < 1$, where $\xi_0 = P(Y = 0)$. Note that, in the **gamlss.inf** implementation, $\boldsymbol{\theta}$ has a maximum of four parameters denoted by $\boldsymbol{\theta}^T = (\mu, \sigma, \nu, \tau)$.

In the example below first take the skew student $t$-family distribution, `SST`$(\mu, \sigma, \nu, \tau)$, defined on $(-\infty, \infty)$, and use the `gen.Family()` function in the **gamlss.dist** package to generate the distribution `logSST`$(\mu, \sigma, \nu, \tau)$, defined on $(0, \infty)$. By using the function `gen.Zadj()` on the new generated `logSST`$(\mu, \sigma, \nu, \tau)$ distribution, a zero adjusted `logSST` distribution, `logSSTZadj`$(\mu, \sigma, \nu, \tau)$ is created, defined on $[0, \infty)$.

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

## A  log  family of distributions from SST has been generated
##  and saved under the names:
##  dlogSST plogSST qlogSST rlogSST logSST
```




**R** code on page 6

Figure 2: A truncated from zero SST distribution: (a) with values $\mu = (0.1, 1, 2)$, $\sigma = 0.5$, $\nu = 1$ and $\tau = 10$, (b) with values $\mu = 2$, $\sigma = (0.5, 1, 2)$, $\nu = 1$ and $\tau = 10$ (c) with values $\mu = 2$, $\sigma = 0.5$, $\nu = (0.1, 1, 2)$ and $\tau = 10$. (d) with values $\mu = 2$, $\sigma = 0.5$, $\nu = 1$ and $\tau = (3, 5, 100)$

```r
gen.Zadj(family="logSST")
```

```
## A zero adjusted logSST distribution has been generated
##  and saved under the names:
##  dlogSSTZadj plogSSTZadj qlogSSTZadj rlogSSTZadj
##  plotlogSSTZadj
```

There are five function generated here:

`dlogSSTZadj()` The pdf of the distribution, `d` function.

`plogSSTZadj()` The cdf of the distribution, `p` function.

`qlogSSTZadj()` The inverse cdf of the distribution, `q` function.

`rlogSSTZadj()` The random generating function of the distribution, `r` function.

`plotlogSSTZadj()` The function for plotting the pdf of the distribution.

To fit the distribution the function `gamlssZadj()` can be used, see Section 5.

# 4 Plotting zero adjusted distributions

The newly created `plotlogSSTZadj()` function can be used to plot the pdf of the adjusted distribution (which is a mixed continuous-discrete distribution). Figure 3 shows the use of the `plotlogSSTZadj()` function. The function plots the zero adjusted probability (density) function including the point probability at zero. Figure 3 shows eight different realisations of the distribution for different values of the parameters.

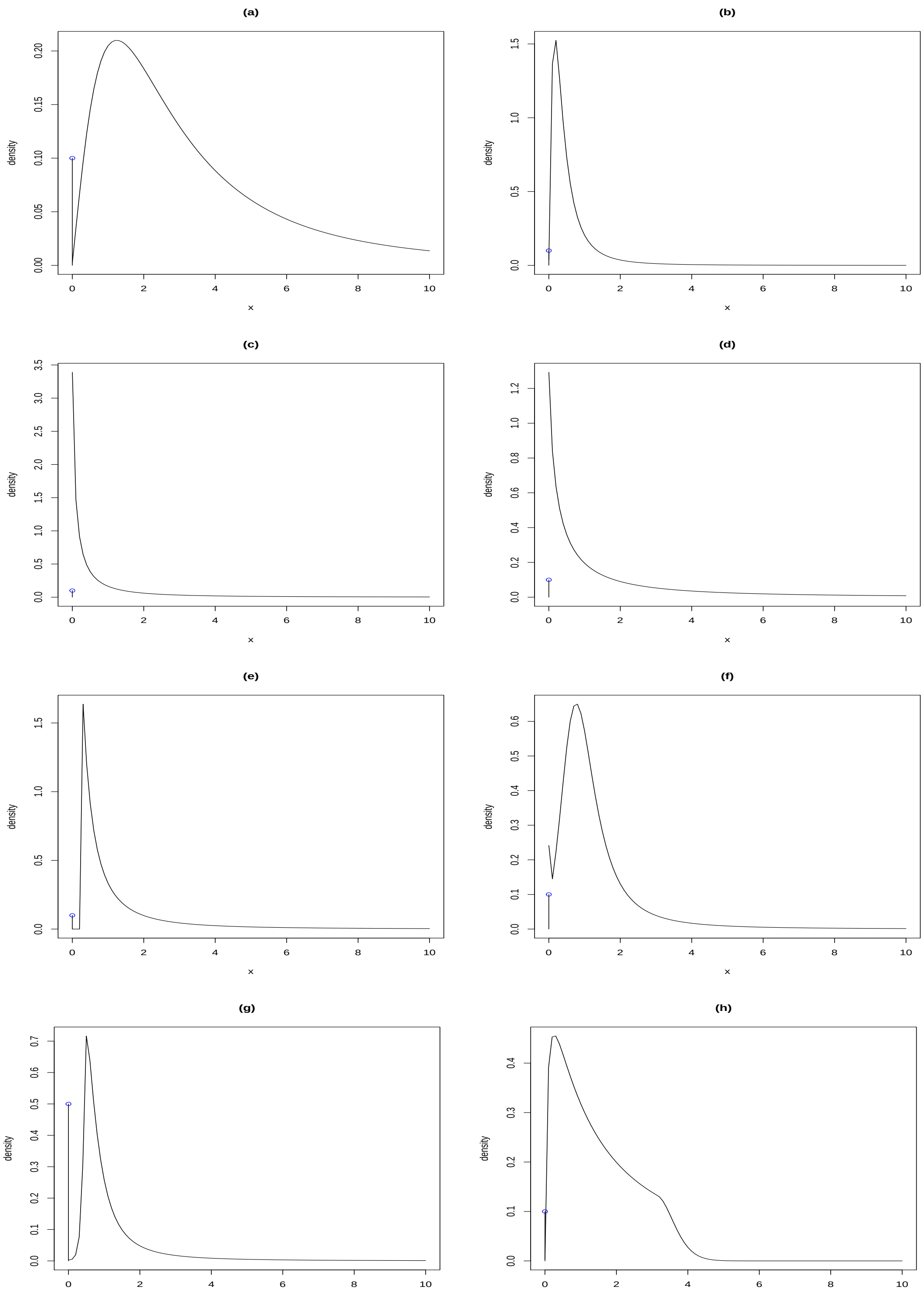
```r
plotlogSSTZadj(mu= 1, sigma=1, nu=1, tau=10, xi0=.1); title("(a)")
plotlogSSTZadj(mu=-1, sigma=1, nu=1, tau=10, xi0=.1); title("(b)")
plotlogSSTZadj(mu=-1, sigma=2, nu=1, tau=10, xi0=.1); title("(c)")
plotlogSSTZadj(mu=0,  sigma=2, nu=1, tau=10, xi0=.1); title("(d)")
plotlogSSTZadj(mu=0,  sigma=1,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

```r
# plotting the pdf ------------------------------
curve(dlogSSTZadj(x, mu=1, sigma=1, nu=.8, tau=10, xi0=.1),
      0.001,10, ylab="pdf",  main="(a)")
# getting the probabilities
p0 <- dlogSSTZadj(x=0, mu=1, sigma=1, nu=.8, tau=10, xi0=.1)
points(0, p0, col="blue")
lines(0, p0, col="blue", type="h")
# plotting the cdf ------------------------------
curve(plogSSTZadj(x, mu=1, sigma=1, nu=.8, tau=10, xi0=.1),
```

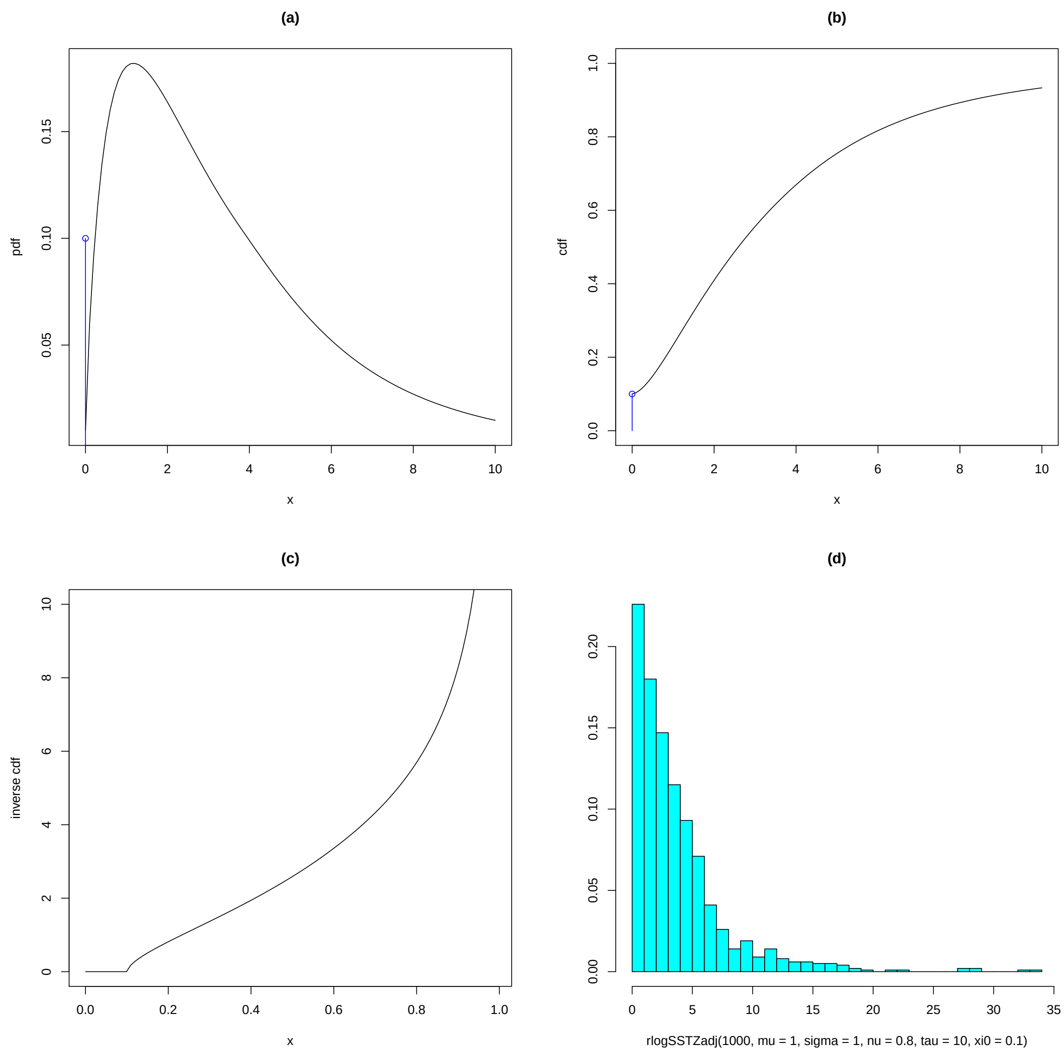
Figure 4




**R** code on page 9

Figure 3: A `logSST` distribution: (a) with values $\mu = 1$, $\sigma = 1$, $\nu = 1$, $\tau = 10$ and $\xi_0 = .1$ (b) with values $\mu = -1$, $\sigma = 1$, $\nu = 1$, $\tau = 10$ and $\xi_0 = .1$ (c) with values $\mu = -1$, $\sigma = 2$, $\nu = 1$, $\tau = 10$ and $\xi_0 = .1$ (d) with values $\mu = 0$, $\sigma = 2$, $\nu = 1$, $\tau = 10$ and $\xi_0 = .1$ (e) with values $\mu = 0$, $\sigma = 1$, $\nu = 10$, $\tau = 10$ and $\xi_0 = .1$ (f) with values $\mu = 0$, $\sigma = 1$, $\nu = 1$, $\tau = 3$ and $\xi_0 = .1$ (g) with values $\mu = 0$, $\sigma = 1$, $\nu = 2$, $\tau = 3$ and $\xi_0 = .5$ (h) with values $\mu = 0$, $\sigma = 1$, $\nu = .3$, $\tau = 100$ and $\xi_0 = .1$.

```
      0.0001,10, ylim=c(0,1), ylab="cdf", main="(b)")
points(c(0), c(p0), col="blue")
lines(c(0), c(p0), col="blue", type="h")
# plotting the inverse cdf -----------------------
curve(qlogSSTZadj(x, mu=1, sigma=1, nu=.8, tau=10, xi0=.1),
      0.0001,0.99, ylim=c(0,10), ylab="inverse cdf", main="(c)")
# plottind simulated data
set.seed(1000)
truehist(rlogSSTZadj(1000,mu=1, sigma=1, nu=.8, tau=10, xi0=.1),
         main="(d)")
```




**R** code on page 9

Figure 4: The (a) pdf (b) cdf (c) inverse pdf and (d) simulated data from a zero adjusted log-SST distribution with $\mu = 1$, $\sigma = 1$, $\nu = .8$, $\tau = 10$, $\xi_0 = .1$.

The next section demonstrates how to use the function `gamlssZadj()` to fit a model which has a response variable on the interval $[0, \infty)$, i.e. including 0.

# 5 Fitting a zero adjusted distribution

## 5.1 The `gamlssZadj()` function

The main function for fitting a zero adjusted distribution model to a response variable $Y$ on $[0, \infty)$ i.e. including 0, is `gamlssZadj()`. In a zero adjusted distribution the parameters $\mu$, $\sigma$, $\nu$ and $\tau$ (of the unadjusted distribution) are orthogonal to the parameter $\xi_0 = P(Y = 0)$ in the sense that the log-likelihood function can be factorised in two components, one containing the $\mu$, $\sigma$, $\nu$ and $\tau$ and another containing $\xi_0$. This means that the two sets of parameters can be estimated separately. The function `gamlssZadj()` takes advantage of this separation and works as follows:

- It picks the argument `family` which defines a `gamlss.family` distribution defined on $(0, \infty)$
- It creates a binary response variable $Y_1 = 1(\text{if } Y = 0) + 0(\text{if } Y > 0)$ and it fits a binary logistic model (using the `gamlss.family` `BI`)
- Fits a GAMLSS model to the data cases with Y in interval $(0, \infty)$ using the distribution defined by `family`, by weighting out the observations with $Y = 0$.
- Creates the (normalised randomized) quantile residuals for the all cases (including cases with $Y = 0$) using the fitted zero adjusted distribution model.
- Saves the output as an `gamlssZadj` object which is a subclass of a `gamlss` object.

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

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

The methods are demonstrated in the next sections.

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

**y** the response variable on $[0, \infty)$ (including values at zero)

**mu.formula** a model formula for the $\mu$ parameter

**sigma.formula** a model formula for the $\sigma$ parameter

**nu.formula** a model formula for the $\nu$ parameter

**tau.formula** a model formula for the $\tau$ parameter

**xi0.formula** a model formula for the $\xi_0$ parameter which is equals to the probability at zero

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

**family** any `gamlss()` distribution family defined on $(0, \infty)$

**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 `gamlssZadj()` are GAMLSS models, the parameter formulae above can take any linear or additive GAMLSS terms including smoothers and random effects.

To demonstrate the use of the `gamlssZadj()` function a simulated example is used below.

## 5.2 Simulating data

To compare the results obtained by the function `gamlssZadj()` to the ones obtained from standard `gamlss()`, simulate data from the zero adjusted gamma distribution, $\texttt{ZAGA}(\mu, \sigma, \nu)$.

```r
library(gamlss)     # loading gamlss package
library(gamlss.inf)
# creating  data
set.seed(3223)
 y0 <- rZAGA(1000, mu=3, sigma=.5, nu=.15)
truehist(y0)
```

Figure 5

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

$$f_Y(y) = \begin{cases} p_0 & \text{if } y = 0 \\ (1 - p_0) f_W(y) & \text{if } 0 < y < \infty \end{cases} \tag{3}$$

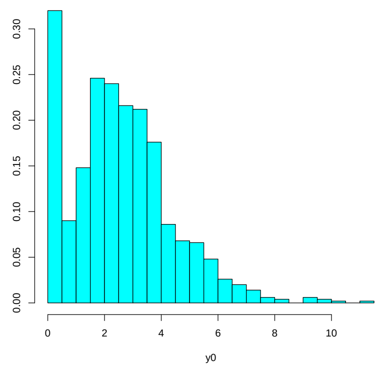
for $0 \leq y < \infty$, where $W \sim \texttt{GA}(\mu, \sigma)$ has a gamma distribution with $0 < \mu < \infty$ and $0 < \sigma < \infty$ and $0 < \nu < 1$, where $\nu = p_0$. For $\texttt{ZAGA}(\mu, \sigma, \nu)$ the default link functions for $\mu$ and $\sigma$ are the log link function and for $\nu$ the logit link function, giving predictors $\eta_1 = \log \mu$, $\eta_2 = \log \sigma$ and $\eta_3 = \log [\nu/(1 - \nu)]$.

Here $f_W(y)$ is given by

$$f_W(y|\mu, \sigma) = \frac{1}{(\sigma^2 \mu)^{1/\sigma^2}} \frac{y^{\frac{1}{\sigma^2} - 1} e^{-y/(\sigma^2 \mu)}}{\Gamma(1/\sigma^2)} \tag{4}$$

for $y > 0$, where $\mu > 0$ and $\sigma > 0$ and $E(W) = \mu$ and $Var(W) = \sigma^2 \mu^2$.

The simulated example comes from a zero adjusted gamma distribution, $\texttt{ZAGA}(\mu, \sigma, \nu)$, with $\mu = 3$ and $\sigma = 0.5$ and $\nu = 0.15$, so $p_0 = 0.15$.




**R** code on page 13

Figure 5: Generated data using the zero adjusted gamma distribution: with values $\mu = 3$, $\sigma = 0.5$, and $\nu = 0.15$.

## 5.3 Fitting a zero adjusted distribution

The zero adjusted distribution is fitted using both `gamlss()` and `gamlssZadj()` functions. Note that the `family` argument in `gamlssZadj()` takes a `gamlss.family` distribution defined on $(0, \infty)$. The `trace=TRUE` argument is used in `gamlssZadj()` to check the convergence of the two different models fitted, one using the `BI`$(\mu)$ family and the other using the `GA`$(\mu, \sigma)$.

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

## GAMLSS-RS iteration 1: Global Deviance = 3860.799
## GAMLSS-RS iteration 2: Global Deviance = 3860.799

t0 <- gamlssZadj(y=y0, mu.formula=~1, family=GA, trace=TRUE)

## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 865.9541
## GAMLSS-RS iteration 2: Global Deviance = 865.9541
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = 2994.845
## GAMLSS-RS iteration 2: Global Deviance = 2994.845
##             The Final Global Deviance  = 3860.799

AIC(g0,t0, k=0)

##    df      AIC
## g0  3 3860.799
## t0  3 3860.799
```

Note that the global deviance of the fitted `t0` model, using `gamlssZadj()`, is obtained by adding the individual deviances from the binomial and the gamma model. The third fitted parameter in both models, is equal to the the probability at zero. The third parameter is called `nu` (i.e. $\nu$) in `gamlss` but `xi0` (i.e. $\xi_0$) in `gamlssZadj()`, where $\nu = p_0$ and $\xi_0 = p_0$. The default link functions are $\text{logit}(\nu)$ and $\text{logit}(\xi_0)$ giving the same predictor $\eta_3 = \log[p_0/(1-p_0)]$. Hence the intercept coefficients ($\beta_{30}$) for the predictor $\eta_3$ for the third parameter are the same for both models as shown below.

```
coef(g0, "nu")

## (Intercept)
##   -1.688296

coef(t0, "xi0")

## (Intercept)
##   -1.688296
```

The fitted values for $\nu$ and $\xi_0$, which are the estimated probabilities at zero, are also identical.

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

## [1] 0.156

fitted(g0, "nu")[1]

## [1] 0.156
```

The `summary()` function makes it clear that the two models are identical.

```
summary(t0)

## *******************************************************************
## Family:  "ZadjGA"
##
## Call:  gamlssZadj(y = y0, mu.formula = ~1, family = GA, trace = TRUE)
##
## Fitting method: RS()
##
## -------------------------------------------------------------------
## Mu link function:  log
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.10322    0.01795   61.45   <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) -0.65095    0.02332  -27.91   <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) -1.68830    0.08715  -19.37   <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:     3860.799
##             AIC:     3866.799
##             SBC:     3881.522
## *******************************************************************
```

```r
summary(g0)
```

```
## ******************************************************************
## Family:  c("ZAGA", "Zero adjusted GA")
##
## Call:  gamlss(formula = y0 ~ 1, family = ZAGA)
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  log
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.10322    0.01795   61.45   <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) -0.65095    0.02332  -27.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  logit
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.68830    0.08715  -19.37   <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:  2
##
## Global Deviance:     3860.799
##             AIC:     3866.799
##             SBC:     3881.522
## ******************************************************************
```

The estimated variance covariance matrix of the estimates of the intercept coefficients ($\beta_{0k}$ for $k = 1, 2, 3$) in the predictors of $(\mu, \sigma, \nu)$ and $(\mu, \sigma, \xi_0)$ for the fitted `g0` and `t0` models, respectively, can be obtained as follows:

```
vcov(t0)

##               (Intercept)   (Intercept) (Intercept)
## (Intercept)  3.222917e-04 -1.812589e-10  0.00000000
## (Intercept) -1.812589e-10  5.438034e-04  0.00000000
## (Intercept)  0.000000e+00  0.000000e+00  0.00759509

vcov(g0)

##               (Intercept)   (Intercept) (Intercept)
## (Intercept)  3.222917e-04 -1.812639e-10  0.00000000
## (Intercept) -1.812639e-10  5.438034e-04  0.00000000
## (Intercept)  0.000000e+00  0.000000e+00  0.00759509
```

Note that, because of the partition of the likelihood function, parameters $\mu$ and $\sigma$ are orthogonal to $\nu$ or $\xi_0$.

The residuals for the two models should be identical for the not zero response variable values, i.e. $Y > 0$. Due to the randomization of the residuals at discrete values of $Y$ (i.e. $Y = 0$ here) we expect differences in the randomized residuals when the response $Y$ is zero. This is demonstrated in the lower part of Figure 6 where the residuals are plotted against the observation index.

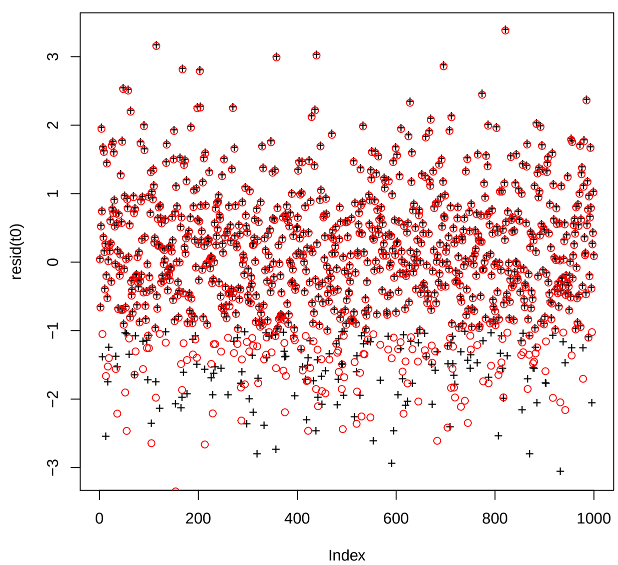
```
plot(resid(t0), pch="+")
points(resid(g0), col="red")
```

Figure 6

Next we will plot the fitted distribution in Figure 7. The standard `ZAGA`$(\mu, \sigma)$ distribution in **gamlss.dist** has its own plotting function called `plotZAGA()` which can be used here. For the model fitted with `gamlssZadj()` such a function `plotGAZadj()` is created using the `gen.Zadj()` function.

```
# generate the
gen.Zadj("GA")

## A zero adjusted GA distribution has been generated
##  and saved under the names:
##  dGAZadj pGAZadj qGAZadj rGAZadj
##  plotGAZadj

plotZAGA(mu=fitted(g0, "mu")[1], sigma=fitted(g0, "sigma")[1],
```

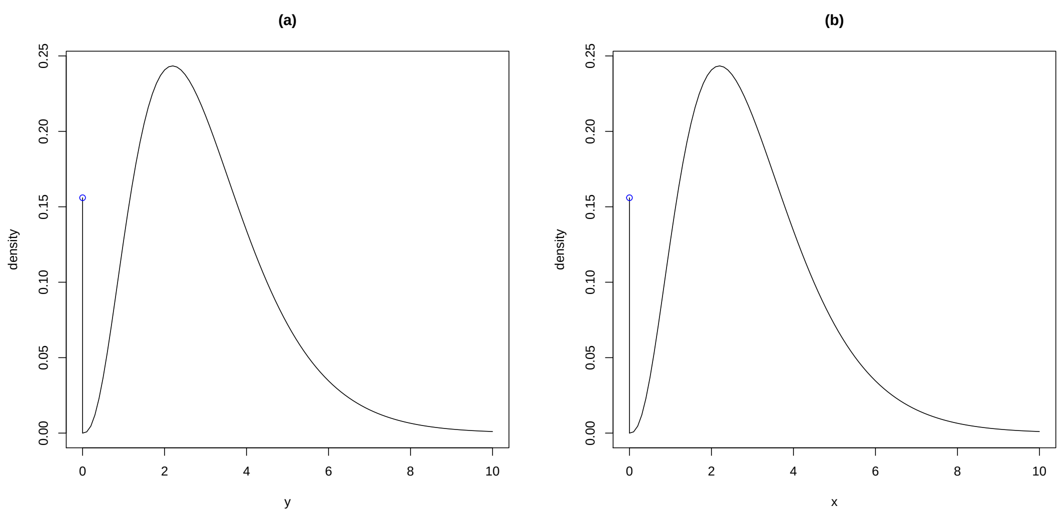
Figure 7




**R** code on page 17

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

```
           nu=fitted(g0, "nu")[1], main="(a)", ylab="density")
plotGAZadj(mu=fitted(g0, "mu")[1], sigma=fitted(g0, "sigma")[1],
           xi0=fitted(g0, "nu")[1]); title("(b)")
```




R code on page 17

Figure 7: The fitted distribution using (a) `plotZAGA()` and (b) `plotGAZadj()`.

The fitted distributions are identical.

# 6 Fitting a regression model

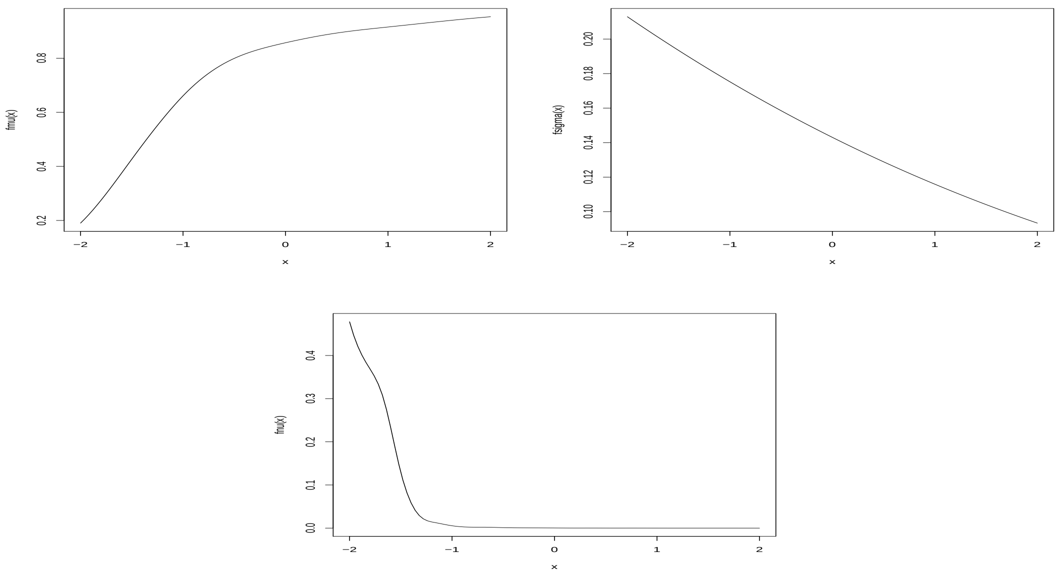
## 6.1 Simulation example.

We first generate the values for the explanatory variable $x$ and then create three different functions of $x$, $f_\mu(x)$, $f_\sigma(x)$ and $f_\nu(x)$, for the parameters $\mu$, $\sigma$, $\nu$; respectively. We will use for the simulations the functions `fmu()`, `fsigma()` and `fnu()` produced by the following code and shown in Figure 8. Note that the data frame `sda` contains values for a previous fitted model for $\mu$, $\sigma$ and $\nu$ given an explanatory variable $x$.

Figure 8

```
# 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)
```

```r
curve(fsigma, -2,2)
fnu <- function(x)
  {f <- splinefun(sda$x, sda$nu)
  f(x)/6
  }
curve(fnu, -2,2)
```



**R** code on page 19

Figure 8: Showing the three functions $\mu = f_\mu(x)$, $\sigma = f_\sigma(x)$ and $\nu = f_\nu(x)$ used for the simulation of the data in this section. From top left to the bottom we have the functions for $\mu$, $\sigma$ and $\nu$.

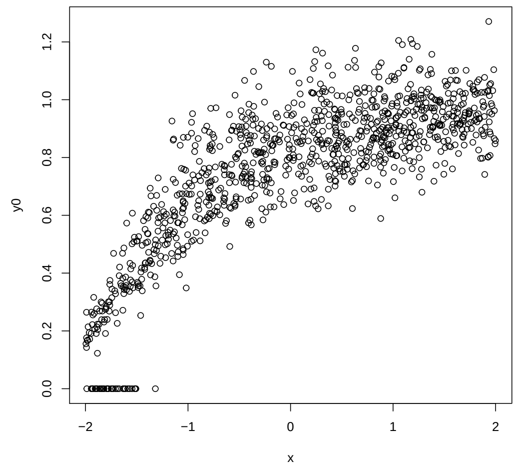
Next we randomly generate the response variable values, `y0`, from a zero adjusted gamma distribution model, ZAGA$(\mu, \sigma, \nu)$ where $\mu = f_\mu(x)$, $\sigma = f_\sigma(x)$ and $\nu = f_\nu(x)$. Figure 9 shows the response variable $y0$ against $x$.

Figure 9

```r
# generating y0 ----------
set.seed(123)
 y0 <- rZAGA(1000, mu=fmu(x), sigma=fsigma(x), nu=fnu(x))
plot(x,y0)
```

We will fit a zero adjusted distribution using the function `gamlss()`. The same model will be also fitted using the `gamlssZadj()` function. Since, in general, the type of relationship existing between the parameters and the explanatory variable $x$ 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()`.

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

## GAMLSS-RS iteration 1: Global Deviance = -1503.31
```




**R** code on page 19

Figure 9: Showing the response variable $y_0$ against the single explanatory variable $x$ for the simulated from a zero adjusted gamma, ZAGA$(\mu, \sigma, \nu)$ distributed response variable $y_0$ with values defined on $[0, \infty)$.

```
## GAMLSS-RS iteration 2: Global Deviance = -1502.221
## GAMLSS-RS iteration 3: Global Deviance = -1502.186
## GAMLSS-RS iteration 4: Global Deviance = -1502.184
## GAMLSS-RS iteration 5: Global Deviance = -1502.184

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

## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 155.0575
## GAMLSS-RS iteration 2: Global Deviance = 155.0575
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = -1658.367
## GAMLSS-RS iteration 2: Global Deviance = -1657.278
## GAMLSS-RS iteration 3: Global Deviance = -1657.243
## GAMLSS-RS iteration 4: Global Deviance = -1657.241
## GAMLSS-RS iteration 5: Global Deviance = -1657.241
##             The Final Global Deviance  = -1502.184

AIC(g0p, t0p)

##           df       AIC
## g0p 16.72940 -1468.725
## t0p 16.72942 -1468.725

summary(t0p)

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

```
## Family:  "ZadjGA"
##
## Call:  gamlssZadj(y = y0, mu.formula = ~pb(x), sigma.formula = ~pb(x),
##     xi0.formula = ~pb(x), family = "GA", trace = TRUE)
##
## Fitting method: RS()
##
## -------------------------------------------------------------------
## Mu link function:  log
## Mu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.317598   0.004905  -64.75   <2e-16 ***
## pb(x)        0.247489   0.004178   59.23   <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) -1.96008    0.02289  -85.64   <2e-16 ***
## pb(x)       -0.21021    0.02101  -10.01   <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)   -7.126      2.054  -3.470 0.000544 ***
## pb(x)         -2.781      1.176  -2.366 0.018174 *
## ---
## 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:  16.72942
##       Residual Deg. of Freedom:  983.2706
##                       at cycle:
##
## Global Deviance:     -1502.184
##             AIC:     -1468.725
##             SBC:     -1386.621
## *******************************************************************
```

The two models have the same deviance as we would expect. Make sure not to try to interpret the coefficients and the standard errors of the smoothing functions in the summary table. They are here as a consequence of how the model is fitted within the `gamlss()` algorithm, (which uses backfitting to alternate between fitting the linear and smoothing component) and they are for the linear term rather than indicating the coefficients and the standard errors of the smoothing functions. To roughly 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 the true functions $\mu = f_\mu(x)$, $\sigma = f_\sigma(x)$ and $\nu = f_\nu(x)$ with their fitted functions. Figure 10 shows the results. Note that the fitted values for the parameters $\mu$, $\sigma$ and $\nu$ (or $\xi_0$) are identical in both fitted models.

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

Figure 10

Note that the `term.plot()` functions is not working for models fitted through the `gamlssZadj()` function so use the function `term.plotZadj()`. Figure 11 shows the fitted additive terms using the `gamlss` `ZAGA` fit on the left and the `gamlssZadj` fit on the right.

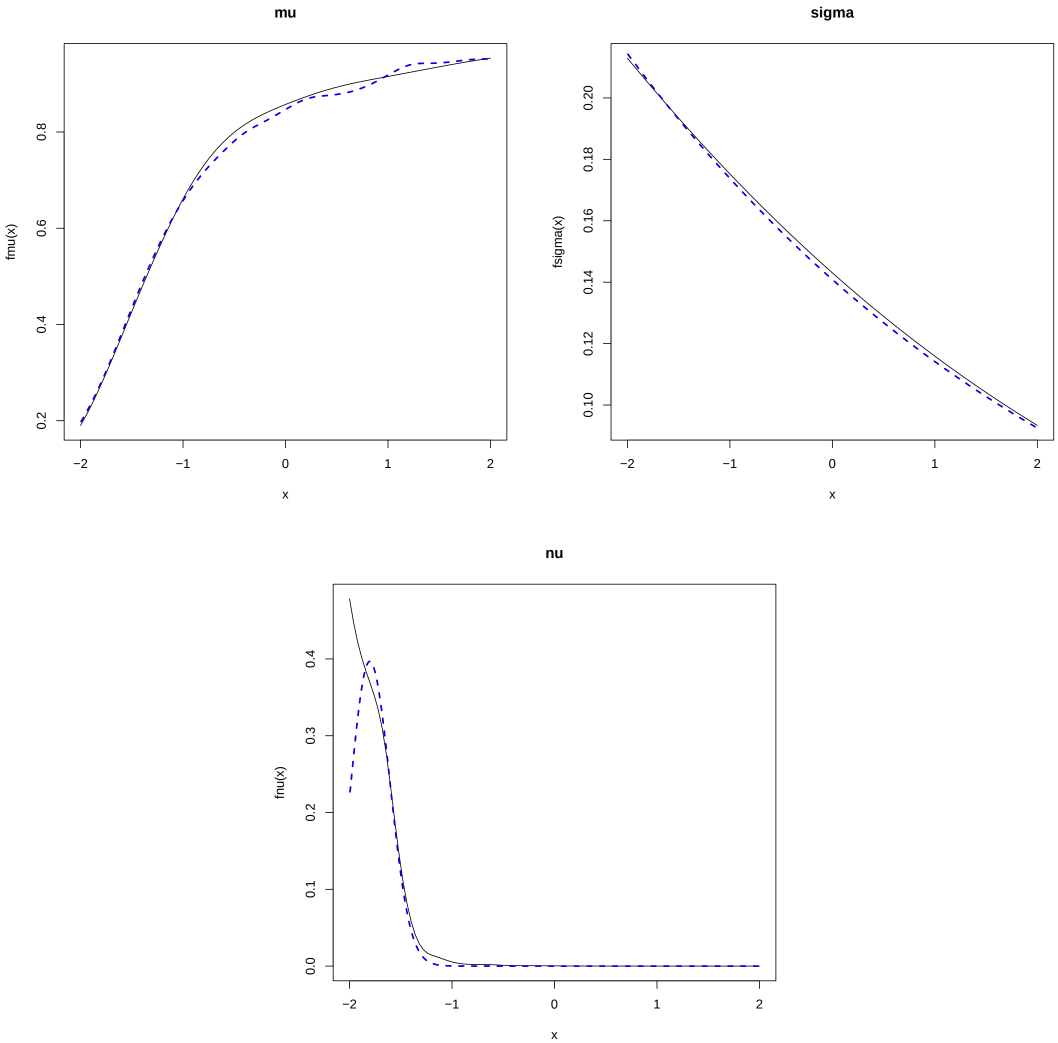
```
term.plot(g0p);title("gamlss")
term.plotZadj(t0p);title("gamlssZadj")
term.plot(g0p, "sigma")
term.plotZadj(t0p, "sigma")
term.plot(g0p, "nu")
term.plotZadj(t0p, "xi0")
```

Figure 11

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 `gamlssZadj` object. Also in the case in which only one explanatory variable exists, the function `centile.Zadj()` can be used to plot centile curves, see Figure 12.

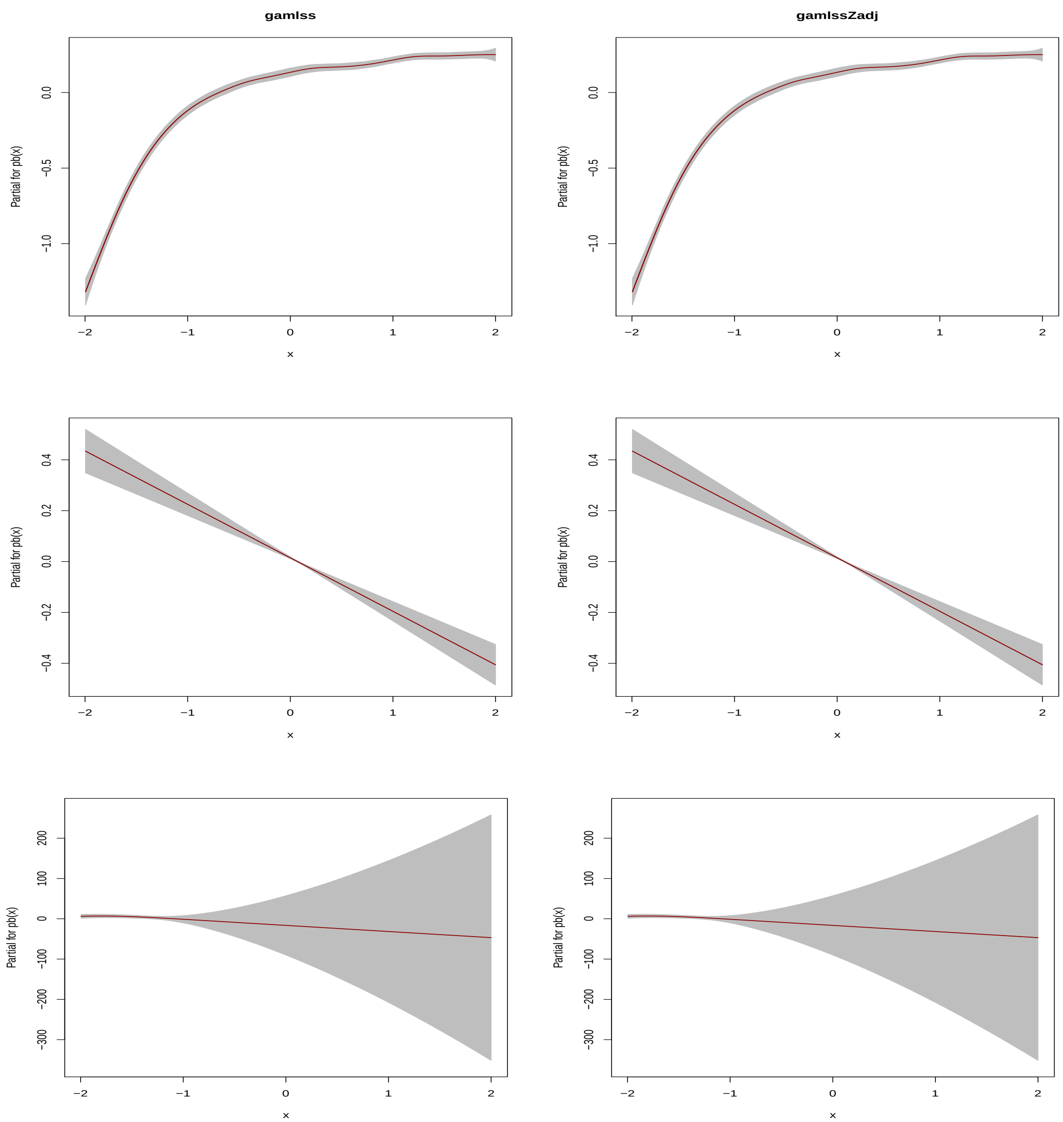
```
plot(t0p)

## ******************************************************************
##   Summary of the Randomised Quantile Residuals
##                            mean   =  -0.00233167
##                        variance   =  1.007856
##                coef. of skewness  =  -0.02696412
##                coef. of kurtosis  =  2.950955
## Filliben correlation coefficient  =  0.9996238
## ******************************************************************
```

Figure 12




**R** code on page 23

Figure 10: Showing the functions from which the data were simulated, together with the fitted smooth functions, for $\mu$, $\sigma$ and $\nu$ (or $\xi_0$). The solid black line is the true function. The dashed line is the fitted function from both models `g0p` and `t0p` since they are identical.




**R** code on page 23

Figure 11: Showing the fitted additive predictors for $\mu$, $\sigma$ and $\nu$ and their approximate 95% confidence bands for models fitted using the function `gamlss()` on the left and using the function `gamlssZadj()` on the right.

```r
wp(t0p, ylim.all=1)
Q.stats(t0p, xvar=x)
```

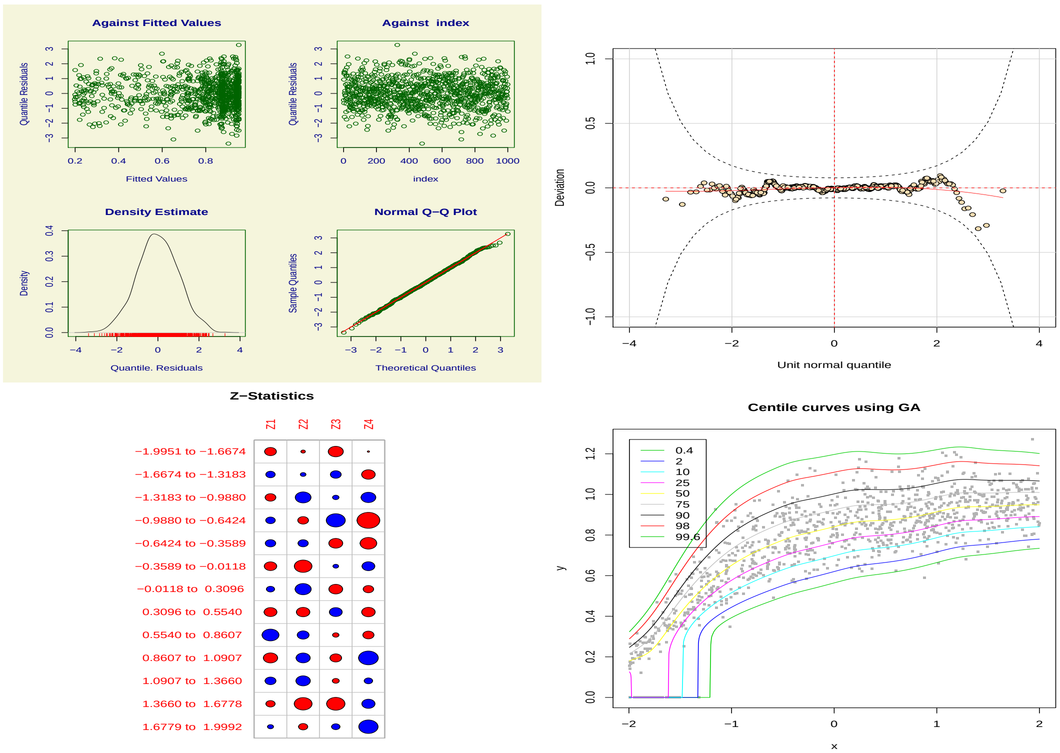
```
## Warning in Q.stats(t0p, xvar = x): the number of intervals have change to 13
##
```

```
##                              Z1          Z2         Z3          Z4
## -1.9951 to -1.6674 -0.5160602 -0.07459136 -0.8466418 -0.01777688
## -1.6674 to -1.3183  0.3294259  0.11634319  0.4385309 -0.68914681
## -1.3183 to -0.9880 -0.4294565  0.92735955  0.1518008  0.81464548
## -0.9880 to -0.6424  0.3398515 -0.43128824  1.3475734 -1.94348926
## -0.6424 to -0.3589  0.4064417  0.39077080 -0.7241606 -1.03863541
## -0.3589 to -0.0118 -0.5828591 -1.16560779  0.1170826  0.61381914
## -0.0118 to  0.3096  0.2503127  0.96743562 -0.7127370 -0.40662572
##  0.3096 to  0.5540 -0.6179411 -0.67474528  0.6117738 -0.68051577
##  0.5540 to  0.8607  1.0656147  0.52222204 -0.1508687 -0.46407877
##  0.8607 to  1.0907 -0.7681959  0.74471171 -0.5059944  1.45314330
##  1.0907 to  1.3660  0.4438157  0.77534908 -0.1793687  0.26486940
##  1.3660 to  1.6778 -0.3202311 -1.18703600 -1.2576778  0.64733508
##  1.6779 to  1.9992  0.1351939 -0.31670204  0.2662090  1.38642236
## TOTAL Q stats       3.6676560  6.90549334  6.1320730 11.73838077
## df for Q stats      2.5274836 11.49838086 13.0000000 13.00000000
## p-val for Q stats   0.2304757  0.83699878  0.9412137  0.54920454
##                     AgostinoK2    N
## -1.9951 to -1.6674  0.7171184   77
## -1.6674 to -1.3183  0.6672327   77
## -1.3183 to -0.9880  0.6866907   77
## -0.9880 to -0.6424  5.5931046   77
## -0.6424 to -0.3589  1.6031721   77
## -0.3589 to -0.0118  0.3904823   77
## -0.0118 to  0.3096  0.6733385   76
##  0.3096 to  0.5540  0.8373689   77
##  0.5540 to  0.8607  0.2381305   77
##  0.8607 to  1.0907  2.3676558   77
##  1.0907 to  1.3660  0.1023289   77
##  1.3660 to  1.6778  2.0007961   77
##  1.6779 to  1.9992  1.9930342   77
## TOTAL Q stats      17.8704537 1000
## df for Q stats     26.0000000    0
## p-val for Q stats   0.8804356    0
```

```r
centiles.Zadj(t0p, xvar=x)
```

```
## % of cases below  0.4 centile is  4.1
## % of cases below  2 centile is  5.5
## % of cases below  10 centile is  11.8
## % of cases below  25 centile is  25.7
## % of cases below  50 centile is  50.3
## % of cases below  75 centile is  75.2
## % of cases below  90 centile is  90.5
```

```
## % of cases below  98 centile is  97.8
## % of cases below  99.6 centile is  99.8
```




**R** code on page **??**

Figure 12: Showing the diagnostic plots created by `plot()`, `wp()`, and `Q.stats()`, and the centile curves produced by `centile.Zadj()`.

Note that the % of cases below the 0.4, 2, 20 and 25 centile curves should be ignored because their centiles curves include value $Y = 0$, see the centiles curves in Figure 12.

## 6.2 Fitting alternative zero adjusted distributions on $[0, \infty)$.

### 6.2.1 Zero adjusted log-transform distributions on $[0, \infty)$.

The function `gamlssZadj()` can be used with a log-transform distribution on $(0, \infty)$ or with a truncated distribution on $(0, \infty)$. For zero adjusted log-transform distribution use the following code:

```
# generate a log-tranform distribution from 0 to infinity
gen.Family("SST", "log")

## A  log  family of distributions from SST has been generated
##  and saved under the names:
##  dlogSST plogSST qlogSST rlogSST logSST
```

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

### 6.2.2 Zero adjusted truncated distributions on $[0, \infty)$.

For a zero adjusted truncated distribution use:

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

## 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 left
##  and the truncation parameter is 0

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

##                df       AIC
## g0p     16.72940 -1468.725
## t0p     16.72942 -1468.725
## t0sst   18.67549 -1463.862
## t0ssttr 10.37618 -1060.759
```

Note that both `logSST`$(\mu, \sigma, \nu, \tau)$ and `SSTtr`$(\mu, \sigma, \nu, \tau)$ distributions have four parameters, so their corresponding zero adjusted distributions have five parameters [including the extra parametrers $\xi_0 = P(Y = 0)$]. Hence models `t0sst` and `t0ssttr` can include models for parameters $\nu$ and $\tau$, e.g. `formula.nu=~pb(x)`, as well as models for $\mu$, $\sigma$ and $\xi_0$.

### 6.2.3 Generalised Tobit model distributions on $[0, \infty)$.

In general for a restricted values response variable, that is, having a distribution on 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 any Tobit model the probability at zero can **not** be modelled independently as a function of explanatory variables, but is equal to the probability of being left censored at zero.

Below we fit a Tobit normal model and a Tobit `SST` model, both left censored at 0 to provide point probabilities at 0.

```r
library(survival)
y0surv<- Surv(y0, y0!=0, type="left")
# creating the distribution
library(gamlss.cens)
# Gaussian
```

```r
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 = -1315.557
## GAMLSS-RS iteration 2: Global Deviance = -1314.02
## GAMLSS-RS iteration 3: Global Deviance = -1313.902
## GAMLSS-RS iteration 4: Global Deviance = -1313.863
## GAMLSS-RS iteration 5: Global Deviance = -1313.848
## GAMLSS-RS iteration 6: Global Deviance = -1313.841
## GAMLSS-RS iteration 7: Global Deviance = -1313.839
## GAMLSS-RS iteration 8: Global Deviance = -1313.838
## GAMLSS-RS iteration 9: Global Deviance = -1313.837

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

## GAMLSS-RS iteration 1: Global Deviance = -1290.533
## GAMLSS-RS iteration 2: Global Deviance = -1298.536
## GAMLSS-RS iteration 3: Global Deviance = -1302.704
## GAMLSS-RS iteration 4: Global Deviance = -1304.586
## GAMLSS-RS iteration 5: Global Deviance = -1305.147
## GAMLSS-RS iteration 6: Global Deviance = -1305.298
## GAMLSS-RS iteration 7: Global Deviance = -1305.348
## GAMLSS-RS iteration 8: Global Deviance = -1305.365
## GAMLSS-RS iteration 9: Global Deviance = -1305.373
## GAMLSS-RS iteration 10: Global Deviance = -1305.374
## GAMLSS-RS iteration 11: Global Deviance = -1305.375

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

##                df       AIC
## g0p     16.72940 -1468.725
## t0p     16.72942 -1468.725
## t0sst   18.67549 -1463.862
```

```
## s0no    13.03170 -1287.774
## s0sst   11.98696 -1281.401
## t0ssttr 10.37618 -1060.759
```

The Tobit models do not perform as well as the zero adjusted models in this case. This is not surprising since the response variable `y0` was simulated from a $ZAGA$ distribution (see earlier Section 5.2) which is a zero adjusted distribution.

## 6.3 Real data example.

The data we will analyse here are motor vehicle insurance data, i.e. motor vehicle insurance policies. The data.frame `mvi` is a sample of 2000 observations from the data frame `mviBig` which has 67143 observations.

**R data file:** `mvi` in package **gamlss.data** of dimensions $2000 \times 11$

**var** **`retval`** : a numeric vector showing the value of the vehicle

- **`whetherclm`** : a numeric vector showing whether a claim is made, 0 no claim, 1 at least one claim
- **`numclaims`** : a numeric vector showing the number of claims
- **`claimcst0`** : a numeric vector showing the total amount of claim, so if for numclaims=0 it is zero.
- **`vehmake`** : a factor showing the make of the car with levels BMW DAEWOO FORD MITSUBISHI.
- **`vehbody`** : a factor showing the type of the car, with levels BUS CONT COUPE HACK HDTOP HRSE MCARA MIBUS PANVN RDSTR SEDAN STNWG TRUCK UTE.
- **`vehage`** : a numeric vector showing the age of the car
- **`gender`** : a factor showing the gender of the policy holder with levels F M
- **`area`** : factor showing the area of residence of the policy holder with levels A B C D E F
- **`agecat`** : a factor showing the age band of the policy holder with levels 1 2 3 4 5 6 one is youngest
- **`exposure`** : a numeric vector showing the time of exposure with values from zero to one

In the following analysis we ignore the exposure time and we fit zero adjusted models, with models for $\mu$ and for the probability at zero (i.e. $\nu$ or $\xi_0$).

```
data(mvi)
# zero adjusted GA
m1 <- gamlss(claimcst0~vehmake+vehbody+vehage+gender+area,
```

```r
             nu.fo=~vehmake+vehbody+vehage+gender+area,
             family=ZAGA, data=mvi )
```

```
## GAMLSS-RS iteration 1: Global Deviance = 3229.012
## GAMLSS-RS iteration 2: Global Deviance = 3229.011
```

```r
# zero adjusted IG
m2 <- gamlss(claimcst0~vehmake+vehbody+vehage+gender+area,
             nu.fo=~vehmake+vehbody+vehage+gender+area,
             family=ZAIG, data=mvi )
```

```
## GAMLSS-RS iteration 1: Global Deviance = 3227.486
## GAMLSS-RS iteration 2: Global Deviance = 3225.318
## GAMLSS-RS iteration 3: Global Deviance = 3225.317
```

```r
# zero adjusted GG
m3 <- gamlssZadj(claimcst0, ~vehmake+vehbody+vehage+gender+area,
             xi0.fo=~vehmake+vehbody+vehage+gender+area,
             family=GG, data=mvi, trace=T, n.cyc=30 )
```

```
## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 942.9831
## GAMLSS-RS iteration 2: Global Deviance = 942.9828
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = 2284.271
## GAMLSS-RS iteration 2: Global Deviance = 2280.202
## GAMLSS-RS iteration 3: Global Deviance = 2275.738
## GAMLSS-RS iteration 4: Global Deviance = 2271.351
## GAMLSS-RS iteration 5: Global Deviance = 2267.604
## GAMLSS-RS iteration 6: Global Deviance = 2264.868
## GAMLSS-RS iteration 7: Global Deviance = 2263.079
## GAMLSS-RS iteration 8: Global Deviance = 2261.965
## GAMLSS-RS iteration 9: Global Deviance = 2261.276
## GAMLSS-RS iteration 10: Global Deviance = 2260.844
## GAMLSS-RS iteration 11: Global Deviance = 2260.568
## GAMLSS-RS iteration 12: Global Deviance = 2260.389
## GAMLSS-RS iteration 13: Global Deviance = 2260.271
## GAMLSS-RS iteration 14: Global Deviance = 2260.192
## GAMLSS-RS iteration 15: Global Deviance = 2260.138
## GAMLSS-RS iteration 16: Global Deviance = 2260.101
## GAMLSS-RS iteration 17: Global Deviance = 2260.076
## GAMLSS-RS iteration 18: Global Deviance = 2260.059
## GAMLSS-RS iteration 19: Global Deviance = 2260.047
## GAMLSS-RS iteration 20: Global Deviance = 2260.039
## GAMLSS-RS iteration 21: Global Deviance = 2260.033
## GAMLSS-RS iteration 22: Global Deviance = 2260.029
## GAMLSS-RS iteration 23: Global Deviance = 2260.026
## GAMLSS-RS iteration 24: Global Deviance = 2260.024
## GAMLSS-RS iteration 25: Global Deviance = 2260.022
## GAMLSS-RS iteration 26: Global Deviance = 2260.021
```

```
##             The Final Global Deviance  = 3203.004

# zero adjusted BCTo
m4 <- gamlssZadj(claimcst0, ~vehmake+vehbody+vehage+gender+area,
             xi0.fo=~vehmake+vehbody+vehage+gender+area,
             family=BCTo, data=mvi, trace=T )

## *****       The binomial model        *****
## GAMLSS-RS iteration 1: Global Deviance = 942.9831
## GAMLSS-RS iteration 2: Global Deviance = 942.9828
## ***** The continuous distribution model *****
## GAMLSS-RS iteration 1: Global Deviance = 2297.874
## GAMLSS-RS iteration 2: Global Deviance = 2262.812
## GAMLSS-RS iteration 3: Global Deviance = 2261.301
## GAMLSS-RS iteration 4: Global Deviance = 2260.482
## GAMLSS-RS iteration 5: Global Deviance = 2260.253
## GAMLSS-RS iteration 6: Global Deviance = 2260.177
## GAMLSS-RS iteration 7: Global Deviance = 2260.155
## GAMLSS-RS iteration 8: Global Deviance = 2260.149
## GAMLSS-RS iteration 9: Global Deviance = 2260.147
## GAMLSS-RS iteration 10: Global Deviance = 2260.147
##             The Final Global Deviance  = 3203.129

AIC(m1,m2,m3,m4)

##    df      AIC
## m3 48 3299.004
## m4 49 3301.129
## m2 47 3319.317
## m1 47 3323.011
```

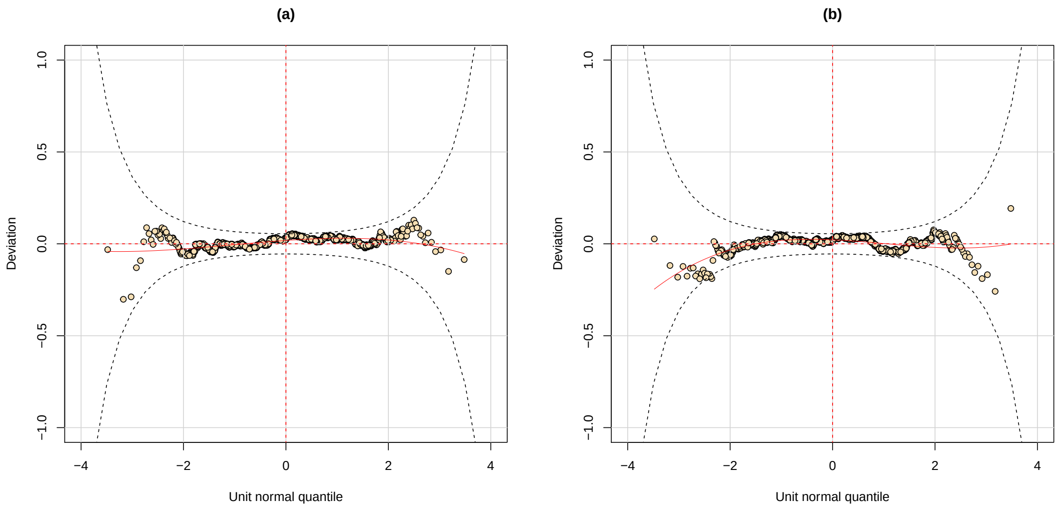
The best model according to AIC seems to be the model `m3` with a zero adjusted generalised gamma distribution, but it is not very far from model `m4`, the zero adjusted $BCTo$ distribution. Figure 13 shows the worm plot for both distributions with the zero adjusted $BCTo$ looking a bit better.

```
wp(m3, ylim.all=1); title('(a)')
wp(m4, ylim.all=1); title('(b)')
```

Figure 13

# 7 Conclusions

GAMLSS is a framework where different models can be fitted and compared. In this vignette, we have shown how models with a response variable defined on $[0, \infty)$, including 0, (i.e. the non-negative real line), can be fitted within the GALMSS framework. The `gamlssZadj()` function can be used to fit a variety of different zero adjusted models, in which the response variable lies on $[0, \infty)$. A zero adjusted distribution for $Y$ comprises a value $Y = 0$ with probability $p_0$ and $Y = Y_1$ with probability $(1 - p_0)$ where $Y_1$ has a continuous distribution on the interval $(0, \infty)$, i.e. the positive real line. In `gamlssZadj()` the parameter `xi0`, $(\xi_0)$ equals $p_0 = P(Y = 0)$. The continuous distribution component must either exist already within `gamlss.dist` or be created




R code on page 32

Figure 13: The worm plot of the residuals from model (a) `m3`, the zero adjusted generalised gamma distribution and (b) `m4` the zero adjusted $BCTo$ distribution.

using the `gen.Family()` or `gen.trun()` function for transformed or truncated distributions, respectively. In addition the function `gen.Zadj()` generates `d`, `p`, `q`, `r` and plot functions for a zero adjusted distribution.

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


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.