Truncated Normal Distribution Implementation in R

Section I

I.I Probability density function (PDF) - Truncated normal

Assume Y follows a normal distribution with mean μ and variance σ^2, with support (a, b). i.e., $-\infty \leq a < b \leq \infty$. Then Y follows a truncated normal distribution, with PDF:

$$f_Y(y; \mu, \sigma) = \frac{\frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right]}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)}. \tag{0.1}$$

Here, Φ is the CDF of the standard normal distribution.

$*$ For observations $y_1, \ldots, y_n \sim Y$ independent, the log–likelihood is given by:

$$\ell(\mu, \sigma; y) = -\frac{1}{2}\log(2\pi) - \log(\sigma) - \frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2 - \log(\Delta\Phi), \tag{0.2}$$

where $\Delta\Phi = \Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)$ for simplicity. Here, $\boldsymbol{\theta} = \left(\mu, \sigma\right)^T$.

I.II Score vector - Truncated normal (Y).

The score vector of $\boldsymbol{\theta}$ is given by:

Details:

Let $A = \dfrac{a-\mu}{\sigma}$ and $B = \dfrac{b-\mu}{\sigma}$.

Then:

$$\frac{\partial \ell}{\partial \mu} = \frac{y-\mu}{\sigma^2} + \frac{1}{\sigma} \cdot \left[\frac{\varphi(B) - \varphi(A)}{\Phi(B) - \Phi(A)}\right]. \tag{0.3}$$

Also,

$$\frac{\partial \ell}{\partial \sigma} = -\frac{1}{\sigma} + \frac{(y-\mu)^2}{\sigma^3} + \frac{1}{\sigma} \cdot \left[\frac{\varphi(B) \cdot B - \varphi(A) \cdot A}{\Phi(B) - \Phi(A)} \right]. \tag{0.4}$$

Here, φ is the standard normal distribution's PDF.

I.III Fisher information.

Denoted as $I(\mu,\sigma)$, the EIMs from observations distributed truncated-normal are given by:

$$I(\mu,\sigma) = \begin{pmatrix} I_{11} & I_{12} \\ I_{21} & I_{22} \end{pmatrix}, \tag{0.5}$$

where $\boldsymbol{\theta} = \left(\mu,\sigma\right)^T$.

Details:

(a) **Diagonal element (1, 1).**

$$I_{11} = -E\left(\frac{\partial^2 \ell}{\partial \mu^2}\right) = \frac{1}{\sigma^2} + \frac{1}{\sigma^2}\left(\left[\frac{\varphi(B) \cdot B - \varphi(A) \cdot A}{\Phi(B) - \Phi(A)}\right] - \left[\frac{\varphi(B) - \varphi(A)}{\Phi(B) - \Phi(A)}\right]^2\right). \tag{0.6}$$

(b) **Diagonal element (2, 2).**

$$\begin{aligned} I_{22} \quad = \quad & -E\left(\frac{\partial^2 \ell}{\partial \sigma^2}\right) = -\frac{1}{\sigma^2} - \frac{3}{\sigma^4}\left\{\sigma^2\left[1 + \frac{A \cdot \varphi(A) - B \cdot \varphi(B)}{\Phi(B) - \Phi(A)} - \left(\frac{\varphi(A) - \varphi(B)}{\Phi(B) - \Phi(A)}\right)^2\right]\right\} + \\ & \frac{1}{\sigma^2}\left\{\left[\frac{\varphi(B)B(1-B^2) - \varphi(A)A(1-A^2)}{\Phi(B) - \Phi(A)}\right] + \left[\frac{\varphi(B)B - \varphi(A)A}{\Phi(B) - \Phi(A)}\right]^2\right\} + \\ & \frac{1}{\sigma^2}\frac{\varphi(B)B - \varphi(A)A}{\Phi(B) - \Phi(A)}. \end{aligned} \tag{0.7}$$

(c) **Off-diagonal element (2, 1).**

$$\begin{aligned} I_{21} \quad = \quad & -E\left(\frac{\partial^2 \ell}{\partial \sigma \partial \mu}\right) = \frac{2}{\sigma^2}\left[\frac{\varphi(A) - \varphi(B)}{\Phi(B) - \Phi(A)}\right] + \frac{1}{\sigma^2}\left[\frac{\varphi(B)(1-B^2) - \varphi(A)(1-A^2)}{\Phi(B) - \Phi(A)}\right] - \\ & \frac{1}{\sigma^2}\left[\frac{(\varphi(B) - \varphi(A))(\varphi(B)B - \varphi(A)A)}{(\Phi(B) - \Phi(A))^2}\right]. \end{aligned} \tag{0.8}$$

(c) **Off-diagonal element (1, 2).**

In this case , we observe cross diagonal elements are equal: $I_{21} = I_{12}$.

Section II - Implementation in R

The new VGAM family function is called `truncnormal()`, an implementation of the truncated normal distribution. Table 1 gives the arguments of `truncnormal()`.

Table 1: Arguments for `truncnormal()`

Argument	Comment
`lmean`	Link function applied to mean.
`lsd`	Link function applied to standard deviation.
`min.support`	Vector of lower truncation, min.support enables RHS truncation (default is `-Inf`).
`max.support`	Vector of upper truncation, max.support enables LHS truncation with default `Inf`.
zero	See CommonVGAMffArguments for more information.

From the R terminal:

```
args(truncnormal)

## function (lmean = "identitylink", lsd = "loglink", min.support = -Inf,
##     max.support = Inf, zero = "sd")
## NULL
```

Section III - Simulated Example

Using simulated data, Table 2 shows the number of iterations for symmetric truncation, that is, truncated at the same percentage of lower and upper tails. Asymmetric truncation is also considered, and it includes lower tail and upper tail truncation. The model is intercept-only. See APPENDIX A.

Table 2: Number of iteration for symmetric and asymmetric truncation models

	Number of iteration till convergence		
Truncation percentage	Symmetric truncation	Lower tail truncation	Upper tail truncation
10%	6 iter (5% e/tail)	4 iter (10% LHS tail)	> 30 iter (10% RHS tail)
20%	> 30 iter (10% e/tail)	5 iter (20% LHS tail)	> 30 iter (20% RHS tail)
30%	> 30 iter (15% e/tail)	6 iter (30% LHS tail)	> 30 iter (30% RHS tail)
40%	> 30 iter (20% e/tail)	6 iter (40% LHS tail)	> 30 iter (40% RHS tail)
50%	> 30 iter (25% e/tail)	6 iter (50% LHS tail)	> 30 iter (50% RHS tail)
60%	> 30 iter (30% e/tail)	6 iter (60% LHS tail)	25 iter (60% RHS tail)
70%	> 30 iter (35% e/tail)	6 iter (70% LHS tail)	14 iter (70% RHS tail)

Note: LHS: left-hand side, RHS: right-hand side.

APPENDIX A

A.1 Symmetric Truncation Example

```
# library("VGAM")
# library("truncnorm")
# set.seed(150223)

# Symmetric truncation where mu is the midpoint
tpvec <- (1:7) / 10
nn <- 1000
mymu = 100
mysd = 15

fits_list <- list()

for (tp in 1:length(tpvec)) {

  y1 <- rtruncnorm(nn, mean = mymu, sd = mysd,
                   a = qnorm(tpvec[tp]/2, mean=mymu, sd=mysd),
                   b = qnorm(1 - tpvec[tp]/2, mean=mymu, sd=mysd))


  y1_df <- data.frame(y1 = y1)

  # Fit a truncated normal distribution to the simulated data
  fits_list[[tp]] <- vglm(y1 ~ 1,
                          truncnormal(min.supp = qnorm(tpvec[tp]/2,
                                                      mean=mymu, sd=mysd),
                                      max.supp = qnorm(1 - tpvec[tp]/2,
                                                      mean=mymu, sd=mysd)),
                          data = y1_df, trace = F)

  cat("Iterations for tp=", tpvec[tp], ": ", fits_list[[tp]]@iter, "\n")

}
```

A.2 Asymmetric Truncation Example

```
# Asymmetric truncation
tpvec <- (1:7) / 10
nn <- 1000
mymu = 100
mysd = 15

fit2 <- list()

for (tp in 1:length(tpvec)) {

  y1 <- rtruncnorm(nn, mean = mymu, sd = mysd,
                   a = qnorm(tpvec[tp]/2, mean=mymu, sd=mysd),
                   b = Inf)


  y1_df <- data.frame(y1 = y1)

  # Fit a truncated normal distribution to the simulated data
  fit2[[tp]] <- vglm(y1 ~ 1,
                     truncnormal(min.supp = qnorm(tpvec[tp]/2,
                                                  mean=mymu, sd=mysd),
                                 max.support = Inf),
                     data = y1_df, trace = F)

  cat("Iterations for tp=", tpvec[tp], ": ", fit2[[tp]]@iter, "\n")

}
```