An R package for Non-Normal Multivariate Distributions: Simulation and Probability Calculations from Multivariate Lomax (Pareto Type II) and Other Related Distributions

by Zhixin Lun and Ravindra Khattree

Abstract Convenient and easy-to-use programs are readily available in R to simulate data from, and probability calculations for several common multivariate distributions such as normal and t. However, functions for doing so from other less common multivariate distributions, especially those which are asymmetric, are not as readily available, either in R or otherwise. We introduce the R package **NonNorMvtDist** to generate random numbers from multivariate Lomax distribution which constitutes a very flexible family of skewed multivariate distributions. Further, by applying certain useful properties of multivariate Lomax distribution, multivariate cases of generalized Lomax, Mardia's Pareto of Type I, Logistic, Burr, Cook-Johnson's uniform, F and inverted beta can be also considered and random numbers from these distributions can be generated. Methods for the probability and the equicoordinate quantile calculations for all these distributions are then provided. This work substantially enriches the existing R toolbox for nonnormal or nonsymmetric multivariate probability distributions.

Introduction

A k-dimensional multivariate Lomax (Pareto Type II) probability distribution was first introduced by (Nayak, 1987) as a joint distribution of k skewed nonnegative random variables $X_1, \cdots, X_k$ with joint probability density function given by

$$f(x_1, \ldots, x_k) = \frac{\left[\prod_{i=1}^{k} \theta_i\right] a(a+1)\cdots(a+k-1)}{\left(1+\sum_{i=1}^{k} \theta_i x_i\right)^{a+k}}, \quad x_i > 0, a, \theta_i > 0, i = 1, \ldots, k. \tag{1}$$

We will denote above density function by $\mathrm{ML}_k(a; \theta_1, \ldots, \theta_k)$. Prior to Nayak (1987), the bivariate case of multivariate Lomax distribution was studied by Lindley and Singpurwalla (1986). Nayak (1987) indicated that the k-dimensional multivariate Lomax distribution can be obtained by mixing k independent univariate exponential distributions with different failure rates with the mixing parameter η that has a gamma distribution with certain shape parameter a and the scale parameter 1. This fact readily provides an approach to simulate the multidimensional random vectors from the multivariate Lomax distribution. The multivariate Lomax distribution is also transformable to many other useful multivariate distributions and therefore simulations from these distributions are also easily accomplished. Similarly, with appropriate transformations or reparameterizations (or otherwise directly from the probability density function (pdf)), we can also accomplish the cumulative probability calculations as well as the calculation of equicoordinate quantiles. The objective of this work is to formalize all of the above and to provide a ready-to-use R package titled **NonNorMvtDist** for practitioners to efficiently execute the same. See Lun and Khattree (2019).

The objective of our work is to enrich the existing R packages for supporting simulation and computations for non-normal continuous multivariate distributions. To the best of our knowledge and also from the list of packages for multivariate distributions in CRAN (`https://cran.r-project.org/web/views/Distributions.html`), there is no package that provides both simulation and probability computations for multivariate Pareto distribution. In our package, we provide functions for doing so to both Lomax (Pareto type II) and Mardia's Pareto type I distributions. For multivariate logistic distribution, package **VGAM** (Yee, 2019) implements the bivariate logistic distribution while we support p-variate logistic distribution for $p > 2$. Moreover, multivariate Burr, F and inverted beta distributions had not been implemented in R until we included them in the package **NonNorMvtDist**.

This paper is organized as follows. In Section Multivariate Lomax and related distributions, we provide simulation algorithms for generating data from k-dimensional multivariate Lomax and generalized Lomax (to be defined later) distributions. Through transformations, the tasks of random numbers generation from Mardia's multivariate Pareto Type I, Logistic, Burr, Cook-Johnson's uniform,

and F are achieved. Based on the remarks from Balakrishnan and Lai (2009), we then extend Nayak's work to simulate data from the multivariate inverted beta distribution. In Section Probability computations, we discuss numerical computations of cumulative distribution functions of equicoordinate quantiles and of survival functions for above distributions. In Section Illustrations of simulations and probability calculations, we illustrate the use of respective functions as implemented in R for each of above distributions for bivariate ($k = 2$) case. In Section Computation times, we provide run-time study to assess the computation times for functions as the dimension of data increases. In Section Maximum likelihood estimation of parameters, we implement maximum likelihood estimation of parameters for these distributions. In Section Two applications, we give two applications of package **NonNorMvtDist** namely, (i) data generation from certain non-elliptical symmetric multivariate distributions with univariate normal marginals and (ii) computation of critical values of multivariate F distribution. Section Concluding remarks includes some concluding remarks, pointing out other applications.

Multivariate Lomax and related distributions

Multivariate Lomax distribution can be derived as the probability distribution of a k-component system where k independent exponential random variables have a common environment or mixing parameter following a gamma distribution with shape parameter a and scale parameter b. Let the corresponding random vector be $\mathbf{X} = (X_1, \cdots, X_k)'$. The probability density function of $\mathbf{X}$ is given by (1) and the joint survival function of $\mathbf{X}$ is

$$S(x_1, \ldots, x_k) = \left(1 + \sum_{i=1}^{k} \theta_i x_i\right)^{-a}, \quad x_i > 0, a > 0, \theta_i > 0, i = 1, \ldots, k. \tag{2}$$

Specifically, the pivotal result that we use is given by the following theorem (see Nayak (1987)),

Theorem 2.1: Conditioned on fixed mixing parameter η, representing the environment effect, let $X_1, \cdots, X_k$ be independent exponentially distributed random variables with failure rates $\eta\lambda_1, \cdots, \eta\lambda_k$, respectively. Let the environment effect η be distributed as a Gamma random variate with probability density

$$g(\eta) = b^a \exp(-\eta b)\eta^{a-1}/\Gamma(a), \eta > 0, a, b > 0.$$

Then the unconditional joint density of $X_1, \cdots, X_k$ is given by (1) where $\theta_i = \lambda_i/b, i = 1, \cdots, k$. Clearly, without loss of generality, b can be taken as 1, in which case $\theta_i = \lambda_i, i = 1, \cdots, k$.

In view of above result, we implement the simulation from k-dimensional multivariate Lomax distribution by adopting the following algorithm.

Algorithm-ML$_k(a; \theta_1, \ldots, \theta_k)$**:**

1. Generate a random number η from Gamma$(a, 1)$ distribution.
2. With η as generated in Step 1, generate k-independent random variables X_i, $i = 1, \ldots, k$, each from exponential distribution with parameter $\eta\theta_i$, $i = 1, \ldots, k$, respectively. Let $\mathbf{X} = (X_1, X_2, \cdots, X_k)'$.
3. To obtain a random sample of size n, repeat the Steps 1 and 2 n times.

Nayak (1987) also generalized this distribution by mixing conditionally independent X_i having the Gamma$(l_i, \eta\theta_i)$ distribution, with mixing variable $\eta \sim$ Gamma$(a, 1)$, $i = 1, \ldots, k$. This is termed as generalized multivariate Lomax distribution denoted by GML$(a; \theta_1, \ldots, \theta_k, l_1, \ldots, l_k)$ and has the probability density function

$$f(x_1, \cdots, x_k) = \frac{\left[\prod_{i=1}^{k} \theta_i^{l_i}\right] \Gamma\left(\sum_{i=1}^{k} l_i + a\right) \prod_{i=1}^{k} x_i^{l_i - 1}}{\Gamma(a) \left[\prod_{i=1}^{k} \Gamma(l_i)\right] \left(1 + \sum_{i=1}^{k} \theta_i x_i\right)^{\sum_{i=1}^{k} l_i + a}}, \quad x_i > 0, a, \theta_i, l_i > 0, i = 1, \cdots, k \tag{3}$$

Accordingly, we perform the corresponding simulation by implementing the suitable changes in the above algorithm. The algorithm is given below.

Algorithm-GML$(a; \theta_1, \ldots, \theta_k, l_1, \ldots, l_k)$**:**

1. Generate a random number η from Gamma$(a, 1)$ distribution.

2. With η as generated in Step 1, generate k-independent X_i, each following Gamma$(l_i, \eta\theta_i)$, $i = 1, \ldots, k$, respectively.
3. To obtain a random sample of size n, repeat the Steps 1 and 2 n times.

Both algorithms are easily implemented using R **stats** (R Core Team, 2019) functions `rexp()` and `rgamma()`, respectively for generating univariate exponential and gamma random variates. In the following, we describe approaches to generate other distributions related to multivariate Lomax and generalized multivariate Lomax distributions.

Nayak (1987) has also discussed the inter-relationships between many other multivariate distributions and generalized multivariate Lomax distribution. In view of these inter-relationships, the above algorithm can accordingly be amended to simulate data from these distributions - a task which can be quite difficult to accomplish directly. These inter-relationships are described in Table 1. For convenience, we assume $\mathbf{X} = (X_1, \cdots, X_k)' \sim \text{ML}_k(a; \theta_1, \cdots, \theta_k)$ and $\mathbf{T} = (T_1, \cdots, T_k)' \sim \text{GML}_k(a; \theta_1, \cdots, \theta_k; l_1, \cdots, l_k)$.

Multivariate Distribution	**Transformation/ Parameter Substitutions**	**Probability Density Function**
Lomax	$l_i = 1,$ $i = 1, \cdots, k$	$f(x_1, \ldots, x_k) = \frac{\left[\prod_{i=1}^k \theta_i\right] a(a+1)\cdots(a+k-1)}{\left(1+\sum_{i=1}^k \theta_i x_i\right)^{a+k}}$ $x_i > 0, \quad a > 0, \quad \theta_i > 0$
Mardia's Pareto Type I	$l_i = 1,$ $Y_i = X_i + 1/\theta_i,$ $i = 1, \cdots, k$	$f(y_1, \cdots, y_k) = \frac{\left[\prod_{i=1}^k \theta_i\right] a(a+1)\cdots(a+k-1)}{\left(\sum_{i=1}^k \theta_i y_i - k + 1\right)^{a+k}},$ $y_i > 1/\theta_i > 0, \quad a > 0, \quad \theta_i > 0$
Logistic	$l_i = 1$ and $a = 1,$ $W_i = \mu_i - \sigma_i \ln(\theta_i X_i),$ $i = 1, \cdots, k$	$f(w_1, \cdots, w_k) = \frac{k! \exp\left(-\sum_{i=1}^k \frac{w_i - \mu_i}{\sigma_i}\right)}{\prod_{i=1}^k \sigma_i \left(1+\sum_{i=1}^k \exp\left(-\frac{w_i-\mu_i}{\sigma_i}\right)\right)^{1+k}},$ $-\infty < w_i, \mu_i < \infty, \quad \sigma_i > 0$
Burr	$l_i = 1,$ $B_i = (\theta_i X_i / d_i)^{1/c_i},$ $i = 1, \cdots, k$	$f(b_1, \cdots, b_k) = \frac{\left[\prod_{i=1}^k c_i d_i\right] a(a+1)\cdots(a+k-1) \left[\prod_{i=1}^k b_i^{c_i - 1}\right]}{\left(1+\sum_{i=1}^k d_i b_i^{c_i}\right)^{a+k}},$ $b_i > 0, \quad a > 0, \quad c_i > 0, \quad d_i > 0$
Cook-Johnson's uniform	$l_i = 1,$ $V_i = (1 + \theta_i X_i)^{-a},$ $i = 1, \cdots, k$	$f(v_1, \cdots, v_k) = \frac{\Gamma(a+k)}{\Gamma(a) a^k} \prod_{i=1}^k v_i^{(-1/a)-1} \left[\sum_{i=1}^k v_i^{-1/a} - k + 1\right]^{-(a+k)},$ $0 < v_i \leq 1, \quad a > 0$
F with degrees of freedom $(2a, 2l_1, \ldots, 2l_k)$	$l_i = 1,$ $\theta_i = l_i / a,$ $i = 1, \cdots, k$	$f(t_1, \cdots, t_k) = \frac{\left[\prod_{i=1}^k (l_i/a)^{l_i}\right] \Gamma\left(\sum_{i=1}^k l_i + a\right) \prod_{i=1}^k t_i^{l_i - 1}}{\Gamma(a)\left[\prod_{i=1}^k \Gamma(l_i)\right]\left(1+\sum_{i=1}^k \frac{l_i}{a} t_i\right)^{\sum_{i=1}^k l_i + a}},$ $t_i > 0, \quad a > 0, \quad l_i > 0$
Inverted Beta	$\theta_i = 1,$ $i = 1, \cdots, k$	$f(t_1, \cdots, t_k) = \frac{\Gamma\left(\sum_{i=1}^k l_i + a\right) \prod_{i=1}^k t_i^{l_i - 1}}{\Gamma(a)\left[\prod_{i=1}^k \Gamma(l_i)\right]\left(1+\sum_{i=1}^k t_i\right)^{\sum_{i=1}^k l_i + a}},$ $t_i > 0, \quad a > 0, \quad l_i > 0$

Table 1: Multivariate distributions related to $\text{GML}_k(a; \theta_1, \cdots, \theta_k; l_1, \cdots, l_k)$

The multivariate F distribution can also be obtained by considering

$$T_i = \frac{S_i/(2l_i)}{S_0/(2a)}, \quad i = 1, \ldots, k, \tag{4}$$

where $S_0, S_1, \ldots, S_k$ are independent Chi-square variables with $2a, 2l_1, \ldots, 2l_k$ degrees of freedom respectively; see Johnson and Kotz (1972). It is the joint distribution of the ratios of mean squares under certain linear hypotheses on treatments as discussed in Krishnaiah (1965) in the context of simultaneous ANOVA and MANOVA tests where S_0 is a residual sum of squares and S_i's are various effect sums of squares. The density given in Table 1 is a special case of generalized multivariate F distribution defined by Krishnaiah (1965) when S_i's, $i = 1, \cdots, k$, are all independent. This fact is useful in that Tables given by Armitage and Krishnaiah (1964) make use of this in constructing the statistical tables for certain linear hypotheses. We use these tables to confirm our calculation as done by our R programs. The multivariate Inverted Beta distribution, also called the multivariate inverted Dirichlet distribution, is essentially a special case of multivariate F distribution when $l_1 = l_2 = \cdots = l_k = a$.

Figure 1 pictorially summarizes the relationships between (generalized) Lomax and other distributions.

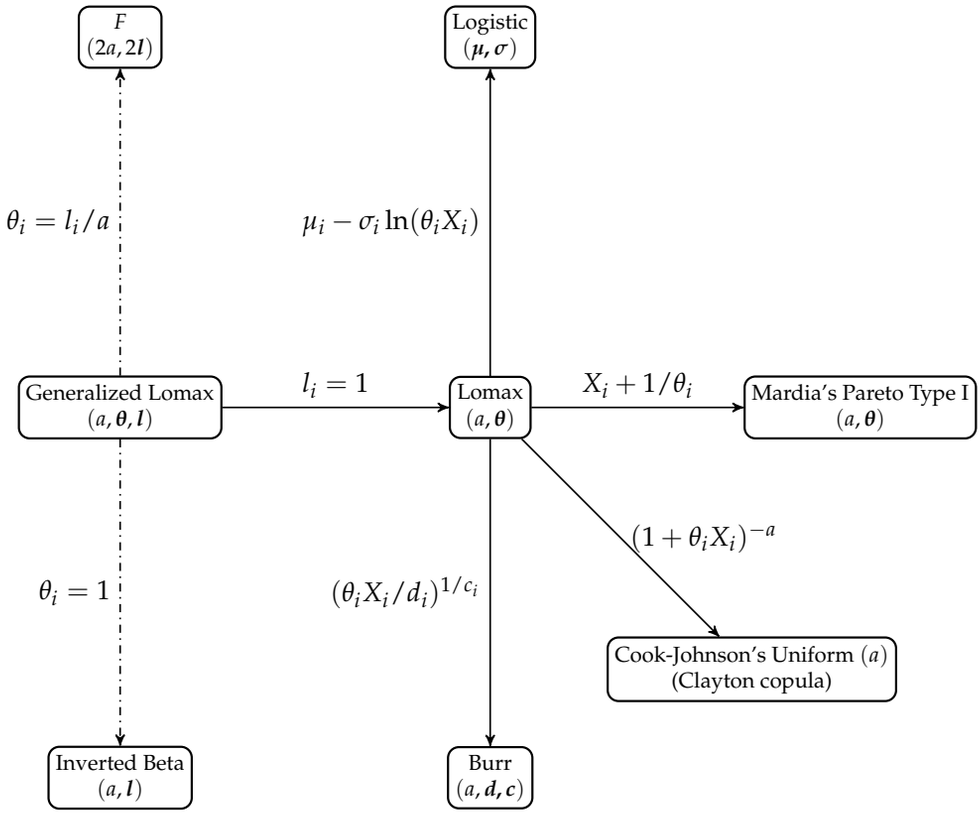


Figure 1: Relationships among (generalized) multivariate Lomax and other distributions. Solid lines represent transformations and dashed dot lines represent reparameterization (parameter substitutions).

Probability computations

Here, we give details of computations of cumulative probability distribution function (cdf), survival function, equicoordinate quantile function for each distribution introduced in Section Multivariate Lomax and related distributions. Depending on the situation, calculation may be sometimes simpler for ~~either~~ joint cdf or ~~for~~ joint survival function.

Distributions transformable from Lomax distribution

The multivariate Lomax distribution has an explicit closed-form expression for the joint survival function given by (2). The survival or cumulative distribution function of other related distributions can be obtained either directly or through appropriate transformations. We summarize these explicit expressions of cumulative distribution function $F(\cdot)$ and survival function $S(\cdot)$ in Table 2.

For the cumulative distribution functions or survival functions with no closed-form expressions, we rely on the following useful formulas (Joe, 1997):

$$S(\mathbf{x}) = 1 + \sum_{C \in \mathcal{C}} (-1)^{|C|} F_C(x_j, j \in C), \tag{5}$$

$$F(\mathbf{x}) = 1 + \sum_{C \in \mathcal{C}} (-1)^{|C|} S_C(x_j, j \in C) \tag{6}$$

where $F_C(x_j, j \in C)$ ($S_C(x_j, j \in C)$) is the joint cdf (joint survival function) of x_j where the subscripts belong to the set C which is a subset of $\{1, 2, \cdots, k\}$. Clearly, $C \in \mathcal{C}$ where $\mathcal{C}$ is the powerset of $\{1, 2, \cdots, k\}$. Also $|C|$ represents the cardinality of C.

The equation

$$P[x_i \leq q_i, i = 1, \cdots, k] = \int_0^{q_1} \cdots \int_0^{q_k} f(x_1, \cdots, x_k) dx_k \cdots dx_1 = p$$

Multivariate Distribution	Cumulative Distribution Function	Survival Function
Lomax	No closed form	$S(x_1,\cdots,x_k)=\left(1+\sum_{i=1}^{k}\theta_i x_i\right)^{-a}$ $x_i>0,\quad a,\theta_i>0$
Mardia's Pareto Type I	No closed form	$S(y_1,\cdots,y_k)=\left(\sum_{i=1}^{k}\theta_i y_i-k+1\right)^{-a}$ $y_i>0,\quad a,\theta_i>0$
Logistic	$F(w_1,\cdots,w_k)=\left[1+\sum_{i=1}^{k}\exp(-\frac{w_i-\mu_i}{\sigma_i})\right]^{-1}$ $-\infty<w_i,\mu_i<\infty,\quad \sigma_i>0$	No closed form
Burr	No closed form	$S(b_1,\cdots,b_k)=\left(1+\sum_{i=1}^{k}d_i b_i^{c_i}\right)^{-a}$ $b_i>0,\quad a,d_i,c_i>0$
Cook-Johnson's Uniform	$F(v_1,\cdots,v_k)=\left[\sum_{i=1}^{k}v_i^{-1/a}-k+1\right]^{-a}$ $0<v_i\leq 1,\quad a>0$	No closed form

Table 2: Cumulative distribution functions and survival functions of multivariate Lomax and related distributions

for a given p does not have a unique solution $(q_1,\cdots,q_k)$. We thus provide the quantile computations only for the equicoordinate quantile, obtained by solving the following equation for q,

$$P[X_i\leq q, i=1,\cdots,k]=\int_0^q\cdots\int_0^q f(x_1,\cdots,x_k)dx_k\cdots dx_1=p, \tag{7}$$

where $0<p<1$ is a (given) cumulative probability. We make use of the R **stats** function `uniroot()` which is used for finding one dimensional root.

Distributions related to generalized multivariate Lomax distribution

For generalized multivariate Lomax distribution and its related distributions, explicit expressions of cumulative distribution function and survival function are not available. Thus, we obtain the cumulative probabilities through multiple integral in (8) below over the unit cube $[0,1]^k$ by using the adaptive multivariate integration function `hcubature()` in package **cubature** (Narasimhan et al., 2018).

$$\begin{aligned}F(x_1,\ldots,x_k)&=\int_0^{x_1}\cdots\int_0^{x_k}f(t_1,\cdots,t_k)dt_k\cdots dt_1,\\&=\prod_{i=1}^{k}x_i\int_0^1\cdots\int_0^1 f(u_1x_1,\cdots,u_kx_k)du_k\cdots du_1,\quad (u_i=t_i/x_i, i=1,\cdots,k).\end{aligned} \tag{8}$$

The following result is used for the computation of cumulative distribution function for the generalized Lomax distribution.

Property 3.1: Let $T_1,\cdots,T_k$ be k continuous random variables that jointly follow the $\mathrm{GML}_k(a;\theta_1,\cdots,\theta_k,l_1,\cdots,l_k)$ distribution as given in (3). Then, the cumulative distribution function of $T_1,\cdots,T_k$ can be computed as

$$F(x_1,\cdots,x_k)=P(T_1\leq x_1,\cdots,T_k\leq x_k)=P(U_1\leq 1,\cdots,U_k\leq 1)$$

where $U_1,\cdots,U_k$ jointly follow $\mathrm{GML}_k(a;\theta_1x_1,\cdots,\theta_kx_k,l_1,\cdots,l_k)$ distribution.

Proof of the above is straightforward by making the substitutions $u_i=t_i/x_i$, $i=1,\cdots,k$.

Through parameter substitutions, the cumulative distribution functions of multivariate F and the inverted beta distribution can be found. These are summarized in Table 3.

For the above method, the run-time consumption rapidly increases as k becomes large. Thus as an alternative, we also provide the option of computation of cumulative distribution function via Monte

Distribution	Parameter Substitutions	Cumulative Distribution Function
Multivariate F with degrees of freedom $(2a, 2l_1, \ldots, 2l_k)$	$\theta_i = l_i/a$, $i = 1, \cdots, k$	$F(x_1, \cdots, x_k) = P(U_1 \leq 1, \cdots, U_k \leq 1)$, $(U_1, \cdots, U_k)$ follows $\text{GML}_k(a; x_1 l_1/a, \cdots, x_k l_k/a, l_1, \cdots, l_k)$
Multivariate Inverted Beta	$\theta_i = 1$, $i = 1, \cdots, k$	$F(x_1, \cdots, x_k) = P(U_1 \leq 1, \cdots, U_k \leq 1)$, $(U_1, \cdots, U_k)$ follows $\text{GML}_k(a; x_1, \cdots, x_k, l_1, \cdots, l_k)$

Table 3: Cumulative distribution functions of related multivariate distributions to generalized multivariate Lomax

Carlo method. The corresponding algorithm is

Algorithm - CDF Computation using Monte Carlo Method:

1. Generate N random vectors $\boldsymbol{t}^{(i)} = (t_1^{(i)}, \cdots, t_k^{(i)})', i = 1, \cdots, N$ from the desired distribution;
2. Compute

$$\hat{P}(T_1 \leq x_1, \cdots, T_k \leq x_k) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{I}(\boldsymbol{t}^{(i)} \leq \boldsymbol{x}),$$

where $\boldsymbol{x} = (x_1, \cdots, x_k)'$ and $\mathcal{I}(\cdot)$ is the zero-one indicator function corresponding to the conditions specified.

Step 1 above is readily carried out by the random numbers generation as described in Section Multivariate Lomax and related distributions using the package **NonNorMvtDist**. Since cdf is computable using adaptive multivariate integration over unit cube $[0, 1]^k$ or via Monte Carlo method, it follows that the survival function can also be calculated (by using (6)). The equicoordinate quantile is computed by using (7).

We also add in our package the calculations of joint probability density function - Being self explanatory with all $pdfs$ available in closed form, it needs no further elaboration. The corresponding function is `dmv*()`.

Illustrations of simulations and probability calculations

We will illustrate here the functions and corresponding arguments for **NonNorMvtDist**. Calling sequences include probability density calculation (`dmv*`), cumulative distribution calculation (`pmv*`), equicoordinate quantile calculation (`qmv*`), random numbers generation (`rmv*`) and survival function calculation (`smv*`) for each of the multivariate distributions introduced in Section Multivariate Lomax and related distributions. For each distribution, we consider the bivariate case ($k = 2$). This choice enables us to also succinctly and graphically present the probability density plots. The detailed description of calling sequence for each of the several cases have been moved into digital compliment of this work as Table 1.

For example, for the bivariate Lomax distribution ($k = 2$) with parameters $a = 5$, $\boldsymbol{\theta} = (0.5, 1)$, the calling sequences for various functions are

```
dmvlomax(x, parm1 = 5, parm2 = (0.5,1))
pmvlomax(q, parm1 = 5, parm2 = (0.5,1))
qmvlomax(p, parm1 = 5, parm2 = (0.5,1))
rmvlomax(n, parm1 = 5, parm2 = (0.5,1))
smvlomax(q, parm1 = 5, parm2 = (0.5,1))
```

It may be mentioned that our approach may be more efficient than NORTA method (Ghosh and Henderson, 2002) for simulation in that NORTA always first requires simulation from multivariate normal which are then transformed to multivariate uniform. Only often this step, one could subsequently transform the simulated data to the desired distribution. Consequently, for large dimensions the approach requires more computing power and time (in fact, to simulated data from normal distribution, many programs themselves first require random number generations from uniform distributions, from which normal random numbers are obtained).

3D bivariate density plot

For each bivariate distribution, we provide the plots of the density surface along with contours using the function `persp3D()` from the package **plot3D** (Soetaert, 2017). To illustrate, we define function `dplot2()` and we pass appropriate density functions to the density function argument `dfun` to create the density surfaces. These are summarized in Table 2 of the digital complement along with corresponding resulting plots in Figure 1 there. For the Lomax distribution with parameters indicated previously, the statements will be

```
library(plot3D)

dplot2 <- function(dfun, x1, x2, zlim) {
    zmat <- matrix(0, nrow = length(x1), ncol = length(x2))
    for (i in 1:length(x1)) {
        for (j in 1:length(x2)) {
            zmat[i, j] = dfun(x = c(x1[i], x2[j]))
        }
    }
    persp3D(z = zmat, x = x1, y = x2, theta = -60, phi = 10, ticktype = "detailed",
            zlim = zlim, contour = list(nlevels = 30, col = "red"),
            facets = FALSE, image = list(col = "white", side = "zmin"),
            xlab = "X1", ylab="X2", zlab = "Density", expand = 0.5, d = 2)
}

dplot2(dfun = function(x) dmvlomax(x, parm1 = 5, parm2 = c(0.5, 1)), x1 = seq(0, 4, 0.1),
       x2 = seq(0, 4, 0.1), zlim = c(-5, 13))
```

The plot that results is shown in Figure 2.

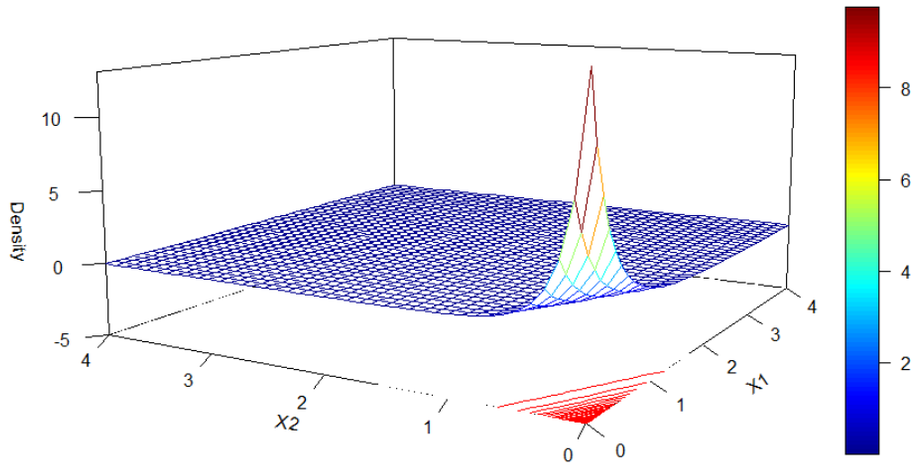


Figure 2: Density surface of bivariate Lomax distribution with parameters $a = 5$, $\boldsymbol{\theta} = (0.5, 1)$

Random number generation

The following code illustrates the use of the function `rmvlomax*()` with a bivariate sample of size $n = 2$. Sampling is done by setting `set.seed(2019)` in advance. The digital complement explicitly provides the code as well as output for all of the probability distributions discussed here. In the output each row represents a bivariate observation.

- **Bivariate Lomax:** $a = 5, \theta_1 = 0.5, \theta_2 = 1$

```
> set.seed(2019)
> rmvlomax(n = 2, parm1 = 5, parm2 = c(0.5, 1))
          [,1]      [,2]
[1,] 1.0174406 0.7076480
[2,] 0.3686253 0.7826978
```

CDF, survival function and equicoordinate quantile

The applications of *cdf* `pmv*()`, survival function `smv*()` and equicoordinate quantile function `qmv*()` are straightforward and follow the same pattern earlier. See Table 4 of digital complement for computation details. In the following, we give code as well as output only for Lomax distribution ($a = 5, \theta_1 = 0.5, \theta_2 = 1$) for specified coordinates (x_1, x_2) and for the cumulative probability $p = 0.5$.

- **Bivariate Lomax:** $a = 5, \theta_1 = 0.5, \theta_2 = 1$; quantiles: $(x_1, x_2) = (1, 0.5)$.

```
> pmvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))
[1] 0.7678755
> smvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))
[1] 0.03125
> qmvlomax(p = 0.5, parm1 = 5, parm2 = c(0.5, 1))
[1] 0.3928917
```

Computation times

We assess the run times for the computation of probability (`pmv*`) and equicoordinate quantile function (`qmv*`) for multivariate Lomax and generalized multivariate Lomax distributions for reference. The survival function (`smv*`) of multivariate Lomax distribution has a closed-form expression and hence the assessment of computation time is omitted. We have used the computer with Intel Core i5-8250U CPU and 8.00 GB RAM. The results for p-variate Lomax distribution are summarized in Table 4. As we can see, the run times for `pmvlomax()` are quite small, even for the dimension $p = 20$. However, `qmvlomax()` requires considerable longer time when $p \geq 17$, which seems to double for every extra dimension added to the size of the random vector. The computation times in Table 4 can be also used as a reference for the distributions related to multivariate Lomax which are in Table 2 since we apply the same approach for probability computations there as well.

p	`pmvlomax()`	`qmvlomax()`	p	`pmvlomax()`	`qmvlomax()`
1	0.01695895	0.03702188	11	0.04889703	0.7992568
2	0.01795197	0.02293706	12	0.09025002	1.879766
3	0.01794696	0.02789807	13	0.1545889	3.087925
4	0.01795197	0.03290296	14	0.245369	5.950396
5	0.01976085	0.04189205	15	0.4657819	11.37162
6	0.01995182	0.06582808	16	0.853502	22.04591
7	0.02094102	0.098768	17	1.708418	45.51909
8	0.02197218	0.146611	18	2.803703	90.70764
9	0.02396512	0.2393882	19	5.453189	181.55772
10	0.04587412	0.4296861	20	10.40903	351.58896

Table 4: Runtimes (in seconds) for functions `pmvlomax()` and `qmvlomax()` as functions of p

The results for generalized p-variate Lomax distribution are summarized in Tables 5 and 6. Both functions `pmvglomax()` and `smvglomax()` require relatively much longer time when $p > 4$ and `qmvglomax()` takes longer when $p > 2$. Based on the run time consumption, we recommend algorithm `MC` for larger dimensions (e.g., when $p > 5$). Similarly, this run-time study can be also used as a reference for the related distributions (to generalized Lomax distribution) as listed in Table 3 since we apply the same method for computations.

	`pmvglomax()`		`smvglomax()`	
p	`numerical`	`MC`	`numerical`	`MC`
1	0.03395414	4.600386	0.03084397	3.977374
2	0.02397585	6.536522	0.05378199	5.510087
3	0.101758	8.396577	0.208086	7.229721
4	0.9758801	10.64278	2.13447	8.741386
5	16.43411	19.30024	40.11997	9.762063
6	305.50092	19.57221	1344.1908	11.63218

Table 5: Runtimes (in seconds) for functions `pmvglomax()` and `smvglomax()` by using algorithms `numerical` and `MC` as functions of p

	qmvglomax()	
p	numerical	MC
1	0.4144461	12.5594
2	6.383504	18.97528
3	96.86238	27.23917
4	2929.5018	30.62036

Table 6: Runtimes (in seconds) for function `qmvglomax()` by using algorithms `numerical` and `MC` as functions of p

Maximum likelihood estimation of parameters

We also include the maximum likelihood estimation to estimate the parameters for various Lomax related distributions (except for the bivariate F distribution). Although many of the density functions in Section Multivariate Lomax and related distributions have complicated forms, maximum likelihood estimation can be easily accomplished by using the built-in optimization functions in R **stats**. The log-likelihood function for a given sample $\mathbf{x_1}, \ldots, \mathbf{x_n}$ is given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\mathbf{x_1}, \ldots, \mathbf{x_n}) &= \log\left[\prod_{j=1}^{n} f(x_{j1}, \ldots, x_{jk}|\boldsymbol{\theta})\right] \\ &= \sum_{j=1}^{n} \log\left[f(x_{j1}, \ldots, x_{jk}|\boldsymbol{\theta})\right], \qquad \boldsymbol{\theta} \in \Theta, \end{aligned} \tag{9}$$

where n is the sample size, $\boldsymbol{\theta}$ is a vector of parameters to be estimated and $\mathbf{x}_j = (x_{j1}, \ldots, x_{jk})'$ is the jth observation for the random vector $\mathbf{X} = (X_1, \ldots, X_k)'$, respectively. The maximizer $\hat{\boldsymbol{\theta}}$ of the log-likelihood function given in (9), namely,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta}|\mathbf{x_1}, \ldots, \mathbf{x_n}),$$

is obtained by using an appropriate optimization method. The parameter space Θ in each case must be appropriately constrained and these constraints must be taken into account during the optimization process. We have thus made use of three R **stats** functions namely, `optim()`, `constrOptim()` and `optimize()` in this work. The functionality of these optimization functions is described in Table 7.

Function	Number of parameters	Usage
`optim()`	Multiple	General-purpose Optimization
`constrOptim()`	Multiple	Optimization with linear constraints
`optimize()`	Single	One Dimensional Optimization

Table 7: Use of optimization functions in R stats

By default all these functions perform the task of minimization of a function. To maximize (9), we only need to add argument `control = list(fnscale = -1)` in functions `optim()` and `constrOptim()`, and set `maximum = TRUE` in function `optimize()`.

For example, for the multivariate Lomax distribution, we define the log-likelihood function `loglik.lomax()` by using the following code:

```
loglik.lomax <- function(data, par) {
    ll <- sum(dmvlomax(data, parm1 = par[1], parm2 = par[-1], log = TRUE))
}
```

The R **stats** function `constrOptim()` is chosen to obtain the maximizer of the log-likelihood function (or equivalently, `loglik.lomax()`). The linear constraints imposed on the parameters a, θ_1 and θ_2 are $a > 0, \theta_1 > 0$ and $\theta_2 > 0$. In matrix notation, it is,

$$\boldsymbol{U\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a \\ \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} a \\ \theta_1 \\ \theta_2 \end{pmatrix} > \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \boldsymbol{C}.$$

Thus, in the code that follows, the constraint matrix `ui` is set as an identity matrix $\mathbf{I}_3$ by using the function `diag(3)` and constraint vector `ci` is set as a zero vector `rep(0,3)`. For our illustration, let the

data be the one simulated by the methods described earlier, each with $n = 300$. It is done by

```
set.seed(1)
bvtlomax <- rmvlomax(n = 300, parm1 = 5, parm2 = c(0.5, 1))
```

The starting initial values for a, θ_1 and θ_2 are all set as 10 by assigning `rep(10,3)` to the argument `theta` in function `constrOptim()`. The gradient argument `grad` is optional and we have chosen this to be `NULL`.

```
> est = constrOptim(theta = rep(10, 3), f = loglik.lomax, grad = NULL, data = bvtlomax,
+                   ui = diag(3), ci = rep(0, 3), control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 5.0555691 0.4468724 0.9036692
```

The output consists of two important pieces of information, (i) whether convergence is successfully achieved (`est$convergence = 0`) or not (`est$convergence = 1`) and (ii) the values of the final maximum likelihood estimates. In our illustration, the convergence is successfully achieved and we have $\hat{a} = 5.0555691$ and $(\hat{\theta}_1, \hat{\theta}_2) = (0.4468724, 0.9036692)$.

We summarize the optimization methods, constraints and the resulting outputs for all the bivariate distributions (except for bivariate F distribution) in Table 8. The detailed illustrations and codes for the remaining distributions are included in the digital complement. Observe that these estimates are reasonably close to the true parameter values, thereby confirming that the program is functioning as it is expected to.

Multivariate Distribution	**Parameters**	**Optimization Method**	**Constraints**	**Estimated Parameters**
Lomax	$a = 5$ $\boldsymbol{\theta} = (0.5, 1)'$	`constrOptim()`	$U = I_3$ $C = (0, 0, 0)'$	$\hat{a} = 5.05556$ $\hat{\boldsymbol{\theta}} = (0.44687, 0.90366)'$
Mardia's Pareto Type I	$a = 5$ $\boldsymbol{\theta} = (0.5, 2)'$	`constrOptim()`	$U = I_3$ $C = (0, [\min(X_1)]^{-1}, [\min(X_2)]^{-1})'$	$\hat{a} = 4.63862$ $\hat{\boldsymbol{\theta}} = (0.49971, 1.99785)'$
Logistic	$\boldsymbol{\mu} = (0.5, 1)'$ $\boldsymbol{\sigma} = (1, 1.5)'$	`optim()`	N/A	$\hat{\boldsymbol{\mu}} = (0.39199, 0.89731)'$ $\hat{\boldsymbol{\sigma}} = (0.97573, 1.55976)'$
Burr	$a = 3$ $\boldsymbol{d} = (1, 3)'$ $\boldsymbol{c} = (2, 5)'$	`constrOptim()`	$U = I_5$ $C = (0, 0, 0, 0, 0)'$	$\hat{a} = 3.91498$ $\hat{\boldsymbol{d}} = (0.64889, 2.06858)'$ $\hat{\boldsymbol{c}} = (1.92719, 5.07697)'$
Cook-Johnson's Uniform	$a = 0.3$	`optimize()`	$a > 0$	$\hat{a} = 0.31064$
Generalized Lomax	$a = 5$ $\boldsymbol{\theta} = (0.5, 1)'$ $\boldsymbol{l} = (2, 4)'$	`constrOptim()`	$U = I_5$ $C = (0, 0, 0, 0, 0)'$	$\hat{a} = 3.91869$ $\hat{\boldsymbol{\theta}} = (0.54674, 1.79126)'$ $\hat{\boldsymbol{l}} = (1.70310, 5.21736)'$
Inverted Beta	$a = 4$ $\boldsymbol{l} = (2, 6)'$	`constrOptim()`	$U = I_3$ $C = (0, 0, 0)'$	$\hat{a} = 3.67153$ $\hat{\boldsymbol{l}} = (1.82450, 5.46465)'$

Table 8: Comparison between True and Estimated Values of Parameters for each of the Distributions (except for bivariate F distribution).

Two applications

In this last section, we give two brief applications, which not only demonstrate the use but also confirm the accuracy and verify the correctness of our work.

Generating data from the non-elliptical symmetric distributions with univariate normal marginals

Cook-Johnson's multivariate uniform distribution is a family of distributions which can be used for modeling non-elliptical symmetric data. Further, in view of uniform distribution for marginal it has been as one of the useful choices for modeling through copula (in fact, Cook-Johnson's uniform distribution is indeed a Clayton copula (Nelsen, 2006)). The value of parameter a impacts the common correlation coefficient ρ among variates in that $\rho \to 0$ as $a \to \infty$ and $\rho \to 1$ as $a \to 0$ (Cook and

Johnson, 1981). An interesting application of Cook-Johnson's multivariate uniform distribution is to obtain new joint distributions by marginal transformation. Specifically, we consider the problem of generating random numbers from a multivariate distribution ~~which~~ that is not elliptically symmetric but has univariate normal marginals. Let U_i's, $i = 1, 2$, be two random variables corresponding to the Cook-Johnson's bivariate uniform distribution. The following code yields the pairs of random numbers each having the normal marginals by the transformation $X_i = \Phi^{-1}(Ui)$ where $\Phi^{-1}(\cdot)$ is the quantile function of a standard normal distribution. Clearly, the joint distribution of X_1 and X_2 is not bivariate normal. To begin with, the parameter a is taken to be $a = 2$.

```
set.seed(1)
biv.unif <- rmvunif(8000, parm = 2, dim = 2)
biv.norm <- as.data.frame(apply(biv.unif, 2, qnorm))
```

New paragraph should be indented.

The sample correlation coefficient ρ of data set `biv.norm` is computed by the following code.

```
> cor(biv.norm$V1, biv.norm$V2)
[1] 0.3180119
```

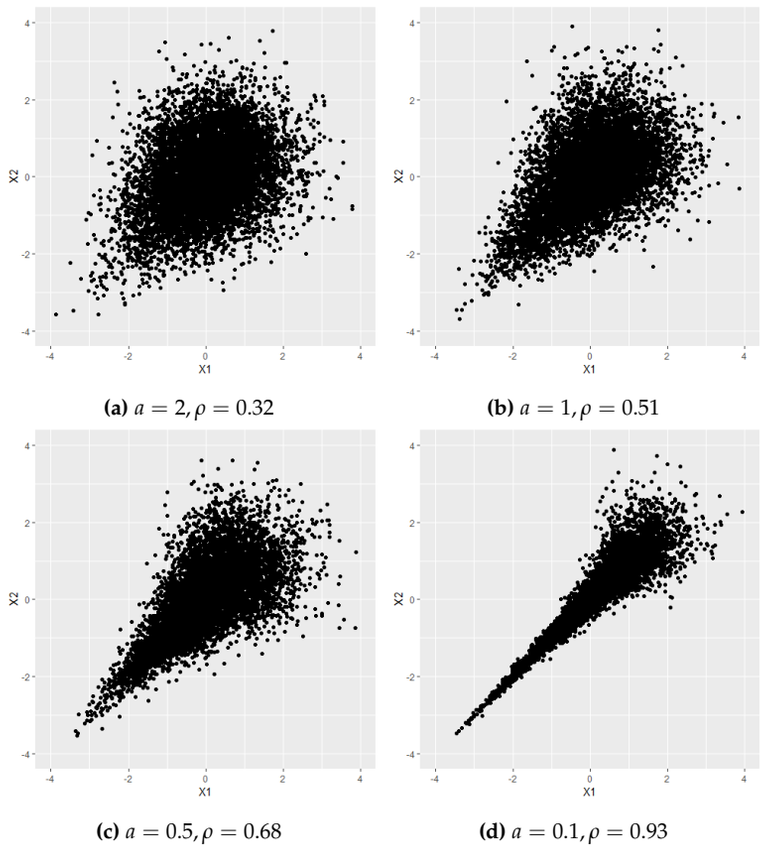
We create a bivariate scatter plot using the function `ggplot()` in package **ggplot2** (Wickham, 2016) for data set `biv.norm`. This is shown in Figure 3 (a).

```
library(ggplot2)

ggplot(biv.norm, aes(x = V1, y = V2)) + xlim(c(-4, 4)) +
ylim(c(-4, 4)) + xlab("X1") + ylab("X2") + geom_point()
```

Should be indented

To assess the behavior as a function of a, we now decrease the parameter a to $1.0, 0.5, 0.1$ resulting in higher correlations ρ ($= 0.51, 0.68, 0.93$, respectively) between the two variates. The bivariate scatter plots for the four cases that is, when $a = 2.0, 1.0, 0.5, 0.1$ are shown in Figure 3 (a)-(d). It is easy to observe that the generated bivariate data have non-elliptical yet, symmetric contours.



(a) $a = 2, \rho = 0.32$ **(b)** $a = 1, \rho = 0.51$

(c) $a = 0.5, \rho = 0.68$ **(d)** $a = 0.1, \rho = 0.93$

Figure 3: The scatterplots of non-elliptical symmetric normal data generated from transformed Cook-Johnson's uniform random numbers with $a = 2, 1, 0.5$ and 0.1.

Creating tables for simultaneous MANOVA hypothesis tests

Multivariate F distribution arises naturally as the distribution of test statistics in several testing problems in simultaneous MANOVA. Let $s_1^2, \cdots, s_k^2$ be k independent sums of squares and s_0^2 be the sum of squares due to error in the classical ANOVA model. Also, let $H_1, \cdots, H_k$ be certain individual linear hypotheses with corresponding sum of squares $s_1^2, \cdots, s_k^2$. Assume that under $H_1, \cdots, H_k$ respectively the sums of squares $s_1^2, \cdots, s_k^2$ are all χ^2 random variables each with n degrees of freedom and s_0^2 is a χ^2 random variable with m degrees of freedom and is independent of $s_1^2, \cdots, s_k^2$. Armitage and Krishnaiah (1964) defined the critical values F_α at α level of significance for simultaneously testing hypotheses $H_1, \cdots, H_k$ by the probability statement,

$$P\left[\frac{m \max(s_1^2, \cdots, s_k^2)}{n s_0^2} \leq F_\alpha \middle| \bigcap_{i=1}^{k} H_i\right] = P\left[\frac{m s_i^2}{n s_0^2} \leq F_\alpha, i = 1, \cdots, k \middle| \bigcap_{i=1}^{k} H_i\right] = 1 - \alpha.$$

The quantities $(ms_i^2)/(ns_0^2), i = 1, \cdots, k$ jointly follow multivariate F distribution if the overall null hypothesis $H_0 = \bigcap_{i=1}^{k} H_i$ is true. In this case, the critical value F_α can be readily computed using the equicoordinate quantile function `qmvf()` by setting the argument corresponding to $k+1$ values of the degrees of freedom as `df = c(m,n,...,n)`. The following code gives $F_{0.05} = 9.551505$ for the bivariate F case when $m = 5$ and $n = 1$ with default algorithm by using adaptive multiple integration over unit cube (`algorithm = "numerical"`). With Monte Carlo algorithm (`algorithm = "MC"` with `nsim=1,000,000`), we obtain $F_{0.05} = 9.550944$. Note that the Monte Carlo method is seed dependent so the output from different runs may slightly differ from each other.

```
> qmvf(0.95, df = c(5, 1, 1))
[1] 9.551505
> qmvf(0.95, df = c(5, 1, 1), algorithm = "MC")
[1] 9.550944
```

For further demonstration and also to further affirm our trust in the calculations, we compare the output of quantile function `qmvf()` using both adaptive multivariate integration and Monte Carlo methods with the values given in Armitage and Krishnaiah (1964). These three calculations are reported in Table 9 for a few choices of m and n. The agreement among the three columns shows that the package **NonNorMvtDist** provides a convenient way to obtain percentage points for the hypothesis testing problems considered by Armitage and Krishnaiah (1964) and Krishnaiah (1965). Clearly unlike the tables in Armitage and Krishnaiah, the choices of α and degrees of freedom are not restricted and in that sense our package is very comprehensive and exhaustive in this respect.

α	**df** (m, n, n)	`qmvf()` **Output** (`algorithm = "numerical"`)	`qmvf()` **Output** (`algorithm = "MC"`)	**Tabulated Values in Armitage and Krishnaiah (1964, pp. 33-42)**
0.05	(5, 1, 1)	9.551505	9.550944	9.55
	(5, 2, 2)	7.879999	7.881698	7.88
	(5, 3, 3)	7.136473	7.165361	7.14
	(5, 4, 4)	6.702224	6.715759	6.70
	(5, 5, 5)	6.412372	6.399167	6.41
	(10, 6, 6)	3.899335	3.898442	3.90
	(10, 7, 7)	3.768494	3.767915	3.77
	(10, 8, 8)	3.665646	3.661366	3.67
	(10, 9, 9)	3.582271	3.583923	3.58
	(10, 10, 10)	3.513163	3.514059	3.51

Table 9: Comparison between the output of `qmvf()` with the values given in Armitage and Krishnaiah (1964).

Concluding remarks

We have developed a new R package **NonNorMvtDist** for generating multivariate random numbers from Lomax (Pareto type II), generalized Lomax, Mardia's Pareto type I, logistic, Burr, Cook-Johnson's uniform, F, and inverted beta distributions. Detailed examples of each distribution are given to illustrate data simulation, probability calculations and statistical modeling.

The fact that nonnormal and skewed multivariate distributions are common in real world but are rarely pursued for analysis due to lack of ready-to-use computational support, underscores the

importance of this package. Possibilities of the use of these distributions are practically limitless and yet unforeseen in a variety of areas, starting from the biomedical sciences, reliability and engineering as well as in statistical finance in the contexts of volatility estimation. Simulations, probability calculations as well as calculations of quantiles and the maximum likelihood estimation of parameters are the natural first set of computations in such studies. We have addressed all of these in this work.

The calculations of probabilities of hypercubes (for example, of $P[a_1 < X_1 < b_1, a_2 < X_2 < b_2, a_3 < X_3 < b_3]$) can be easily implemented by appropriately combining several cdf calculations. Alternatively, our codes for `pmv*()` can be suitably modified for this purpose. The probability density surface plots for *any* bivariate marginal can be easily constructed since for the multivariate Lomax distribution, the marginal distributions of any subset of random variables also follow the multivariate Lomax distribution in the appropriate dimension. Further, our work provides a way to generate data from, probability calculations for, as well as modeling for, the data which are marginally distributed as normal but jointly are not.

Acknowledgements

We would like to thank the anonymous referee for his/her helpful comments and remarks to improve both package and this paper.

Bibliography

J. V. Armitage and P. R. Krishnaiah. *Tables for Studentized Largest Chi-Square and Their Application*. Report ARL 64-188, Aerospace Research Laboratories Wright-Patterson Air Force Base, Ohio, 1964. [p3, 12]

N. Balakrishnan and C.-D. Lai. *Continuous Bivariate Distributions*. Springer-Verlag New York, 2nd edition, 2009. ISBN 978-0-387-09614-8. URL https://doi.org/10.1007/b101765. [p2]

R. E. Cook and M. E. Johnson. A family of distributions for modeling non-elliptically symmetric multivariate data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 43(2):210–218, 1981. URL www.jstor.org/stable/2984851. [p10]

S. Ghosh and S. Henderson. Properties of the norta method in higher dimensions. *Proceedings of the 2002 Winter Simulation Conference*, 1:263–269, 2002. URL https://doi.org/10.1109/WSC.2002.1172894. [p6]

H. Joe. *Multivariate Models and Dependence Concepts*. Chapman and Hall, London, 1997. URL https://doi.org/10.1201/b13150. [p4]

N. L. Johnson and S. Kotz. *Distribution in Statistics: Continuous Multivariate Distributions*. John Wiley & Sons, New York, 1972. [p3]

P. R. Krishnaiah. On the simultaneous anova and manova tests. *Annals of the Institute of Statistical Mathematics*, 17(1):35–53, 1965. URL https://doi.org/10.1007/BF02868151. [p3, 12]

D. V. Lindley and N. D. Singpurwalla. Multivariate distributions for the life lengths of a system sharing a common environment. *Journal of Applied Probability*, 23:418–431, 1986. URL https://doi.org/10.2307/3214184. [p1]

Z. Lun and R. Khattree. *NonNorMvtDist: Multivariate Lomax (Pareto Type II) and Its Related Distributions*, 2019. URL https://CRAN.R-project.org/package=NonNorMvtDist. R package version 1.0.1. [p1]

B. Narasimhan, S. G. Johnson, T. Hahn, A. Bouvier, and K. Kiêu. *cubature: Adaptive Multivariate Integration over Hypercubes*, 2018. URL https://CRAN.R-project.org/package=cubature. R package version 2.0.3. [p5]

T. K. Nayak. Multivariate lomax distribution: Properties and usefulness in reliability theory. *Journal of Applied Probability*, 24(1):170–177, 1987. URL https://doi.org/10.2307/3214068. [p1, 2, 3]

R. B. Nelsen. *An Introduction to Copulas*. Springer-Verlag New York, 2nd edition, 2006. ISBN 978-0-387-28659-4. URL https://doi.org/10.1007/0-387-28678-0. [p10]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. URL https://www.R-project.org/. [p3]

K. Soetaert. *plot3D: Plotting Multi-Dimensional Data*, 2017. URL https://CRAN.R-project.org/package=plot3D. R package version 1.1.1. [p7]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p11]

T. Yee. *VGAM: Vector Generalized Linear and Additive Models*, 2019. URL https://CRAN.R-project.org/package=VGAM. R package version 1.1-2. [p1]

Zhixin Lun
Department of Mathematics and Statistics
Oakland University
Rochester, MI, USA
ORCiD: 0000-0002-8980-1554
zlun@oakland.edu

Ravindra Khattree
Department of Mathematics and Statistics
Oakland University
Rochester, MI, USA
ORCiD: 0000-0002-9305-2365
khattree@oakland.edu

Digital Complement: An R package for Non-Normal Multivariate Distributions: Simulation and Probability Calculations from Multivariate Lomax (Pareto Type II) and Other Related Distributions

by Zhixin Lun and Ravindra Khattree

Calling sequences and examples

Multivariate Distribution	Calling sequence

Lomax (a = parm1, $\boldsymbol{\theta}$ = parm2)	`dmvlomax(x,parm1 = 1,parm2 = rep(1,k),...)`
	`pmvlomax(q,parm1 = 1,parm2 = rep(1,k))`
	`qmvlomax(p,parm1 = 1,parm2 = rep(1,k),...)`
	`rmvlomax(n,parm1 = 1,parm2 = rep(1,k))`
	`smvlomax(q,parm1 = 1,parm2 = rep(1,k))`

Mardia's Pareto Type I (a = parm1, $\boldsymbol{\theta}$ = parm2)	`dmvmpareto1(x,parm1 = 1,parm2 = rep(1,k),...)`
	`pmvmpareto1(q,parm1 = 1,parm2 = rep(1,k))`
	`qmvmpareto1(p,parm1 = 1,parm2 = rep(1,k),...)`
	`rmvmpareto1(n,parm1 = 1,parm2 = rep(1,k))`
	`smvmpareto1(q,parm1 = 1,parm2 = rep(1,k))`

Logistic ($\boldsymbol{\mu}$ = parm1, $\boldsymbol{\sigma}$ = parm2)	`dmvlogis(x,parm1 = rep(1,k),parm2 = rep(1,k),...)`
	`pmvlogis(q,parm1 = rep(1,k),parm2 = rep(1,k))`
	`qmvlogis(p,parm1 = rep(1,k),parm2 = rep(1,k),...)`
	`rmvlogis(n,parm1 = rep(1,k),parm2 = rep(1,k))`
	`smvlogis(q,parm1 = rep(1,k),parm2 = rep(1,k))`

Burr (a = parm1, $\boldsymbol{d}$ = parm2, $\boldsymbol{c}$ = parm3)	`dmvburr(x,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`
	`pmvburr(q,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k))`
	`qmvburr(p,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`
	`rmvburr(n,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k))`
	`smvburr(q,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k))`

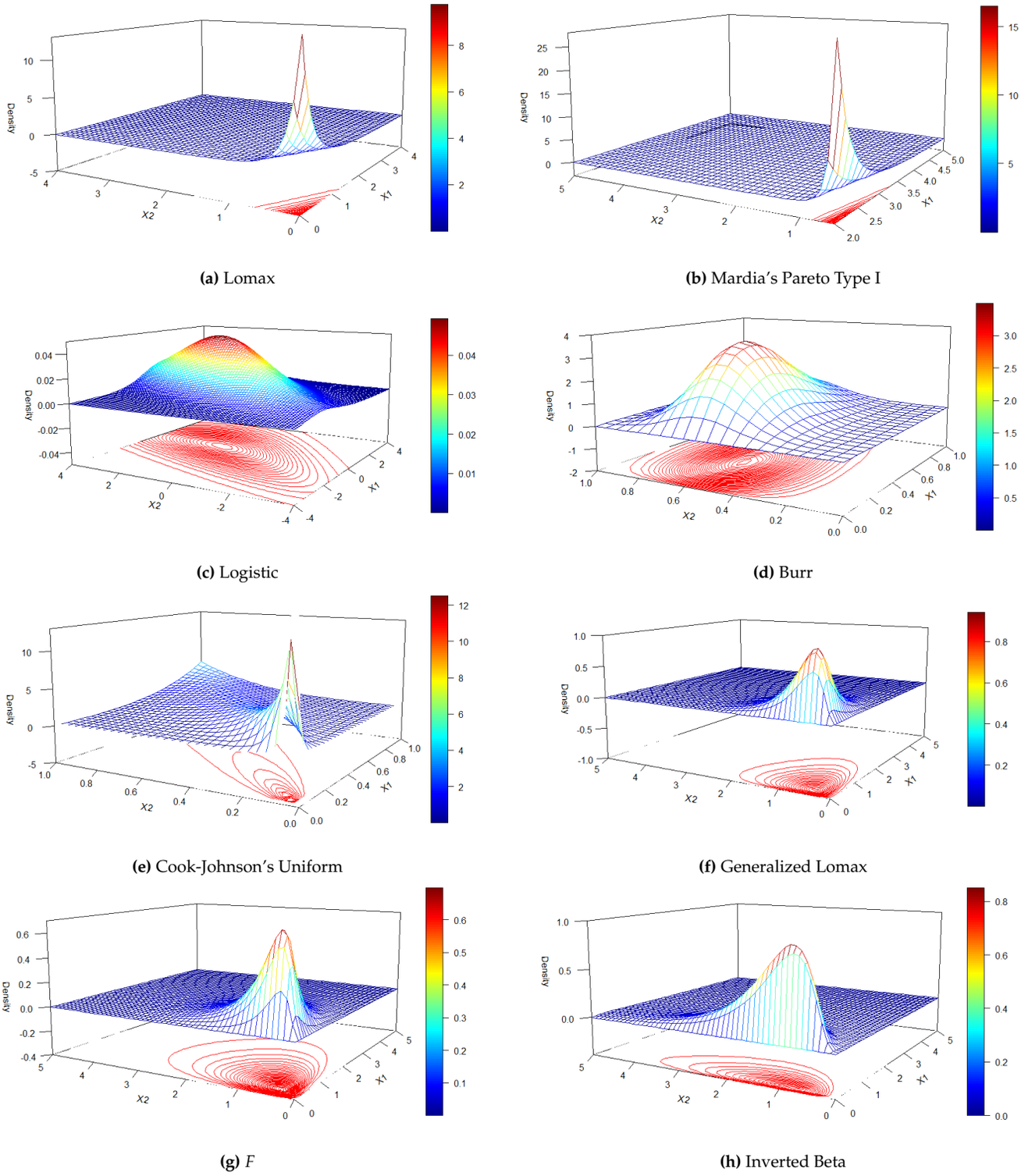
Cook-Johnson's Uniform (a = parm)	`dmvunif(x,parm = 1,...)`
	`pmvunif(q,parm = 1)`
	`qmvunif(p,parm = 1,dim = k,...)`
	`rmvunif(n,parm = 1,dim = 1)`
	`smvunif(q,parm = 1)`

Generalized Lomax (a = parm1, $\boldsymbol{\theta}$ = parm2, $\boldsymbol{l}$ = parm3)	`dmvglomax(x,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`
	`pmvglomax(q,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`
	`qmvglomax(p,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`
	`rmvglomax(n,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k))`
	`smvglomax(q,parm1 = 1,parm2 = rep(1,k),parm3 = rep(1,k),...)`

F (df = df)	`dmvf(x,df = rep(1,k + 1),...)`
	`pmvf(q,df = rep(1,k + 1),...)`
	`qmvf(p,df = rep(1,k + 1),...)`
	`rmvf(n,df = rep(1,k + 1))`
	`smvf(q,df = rep(1,k + 1),...)`

Inverted Beta ($\boldsymbol{a}$ = parm1, $\boldsymbol{l}$ = parm2)	`dmvinvbeta(x,parm1 = 1,parm2 = rep(1,k),...)`
	`pmvinvbeta(q,parm1 = 1,parm2 = rep(1,k),...)`
	`qmvinvbeta(p,parm1 = 1,parm2 = rep(1,k),...)`
	`rmvinvbeta(n,parm1 = 1,parm2 = rep(1,k))`
	`smvinvbeta(q,parm1 = 1,parm2 = rep(1,k),...)`

Table 1: Calling sequence for probability density calculation (`dmv*`), cumulative distribution calculation (`pmv*`), equicoordinate quantile calculation (`qmv*`), random numbers generation (`rmv*`) and survival function calculation (`smv*`).



(a) Lomax

(b) Mardia's Pareto Type I

(c) Logistic

(d) Burr

(e) Cook-Johnson's Uniform

(f) Generalized Lomax

(g) *F*

(h) Inverted Beta

Figure 1: Density surfaces created by the code in Table 2.

Multivariate Distribution	Command	Output
Lomax ($a = 5, \theta_1 = 0.5, \theta_2 = 1$)	`dplot2(dfun = function(x) dmvlomax(x, parm1 = 5, parm2 = c(0.5, 1)), x1 = seq(0, 4, 0.1), x2 = seq(0, 4, 0.1), zlim = c(-5, 13))`	Figure 1 (a)
Mardia's Pareto Type I ($a = 5, \theta_1 = 0.5, \theta_2 = 2$)	`dplot2(dfun = function(x) dmvmpareto1(x, parm1 = 5, parm2 = c(0.5, 2)), x1 = seq(2, 5, 0.1), x2 = seq(0.5, 5, 0.1), zlim = c(-3, 28))`	Figure 1 (b)
Logistic ($\mu_1 = 0.5, \mu_2 = 1, \sigma_1 = 1, \sigma_2 = 1.5$)	`dplot2(dfun = function(x) dmvlogis(x, parm1 = c(0.5, 1), parm2 = c(1, 1.5)), x1 = seq(-4, 4, 0.1), x2 = seq(-4, 4, 0.1), zlim = c(-0.05, 0.05))`	Figure 1 (c)
Burr ($a = 3, d_1 = 1, d_2 = 3, c_1 = 2, c_2 = 5$)	`dplot2(dfun = function(x) dmvburr(x, parm1 = 3, parm2 = c(1, 3), parm3 = c(2, 5)), x1 = seq(0, 1, 0.05), x2 = seq(0, 1, 0.05), zlim = c(-2, 4))`	Figure 1 (d)
Cook-Johnson's Uniform ($a = 0.3$)	`dplot2(dfun = function(x) dmvunif(x, parm = 0.3), x1 = seq(0, 1, 1 / 30), x2 = seq(0, 1, 1 / 30), zlim = c(-5, 13))`	Figure 1 (e)
Generalized Lomax ($a = 5, \theta_1 = 0.5, \theta_2 = 1, l_1 = 2, l_2 = 4$)	`dplot2(dfun = function(x) dmvglomax(x, parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4)), x1 = seq(0, 5, 0.1), x2 = seq(0, 5, 0.1), zlim = c(-1, 1))`	Figure 1 (f)
$F\ (df = (4, 6, 9))$	`dplot2(dfun = function(x) dmvf(x, df = c(4, 6, 9)), x1 = seq(0, 5, 0.1), x2 = seq(0, 5, 0.1), zlim = c(-0.4, 0.7))`	Figure 1 (g)
Inverted Beta ($a = 4, l_1 = 2, l_2 = 6$)	`dplot2(dfun = function(x) dmvinvbeta(x, parm1 = 4, parm2 = c(2, 6)), x1 = seq(0, 5, 0.1), x2 = seq(0, 5, 0.1), zlim = c(-0.4, 1))`	Figure 1 (h)

Table 2: Example code for creating bivariate density plot.

Bivariate Distribution	Command	Output (x_1, x_2)
Lomax ($a = 5$, $\theta_1 = 0.5, \theta_2 = 1$)	`rmvlomax(n = 2, parm1 = 5, parm2 = c(0.5, 1))`	`1.0174406 0.7076480` `0.3686253 0.7826978`
Mardia's Pareto Type I ($a = 5, \theta_1 = 0.5, \theta_2 = 2$)	`rmvmpareto1(n = 2, parm1 = 5, parm2 = c(0.5, 2))`	`3.017441 0.8538240` `2.368625 0.8913489`
Logistic ($\mu_1 = 0.5$, $\mu_2 = 1, \sigma_1 = 1, \sigma_2 = 1.5$)	`rmvlogis(n = 2, parm1 = c(0.5, 1), parm2 = c(1, 1.5))`	`-0.5019906 -0.9980586` `-0.6524945 -2.8979116`
Burr ($a = 3, d_1 = 1$, $d_2 = 3, c_1 = 2, c_2 = 5$)	`rmvburr(n = 2, parm1 = 3, parm2 = c(1, 3),` `parm3 = c(2, 5))`	`0.9107865 0.8260570` `0.6045062 0.8764837`
Cook-Johnson's Uniform ($a = 0.3$)	`rmvunif(n = 2, parm = 0.3, dim = 2`	`0.38036196 0.24878181` `0.03596575 0.08862409`
Generalized Lomax ($a = 5, \theta_1 = 0.5, \theta_2 = 1$, $l_1 = 2, l_2 = 4$)	`rmvglomax(n = 2, parm1 = 5, parm2 = c(0.5, 1),` `parm3 = c(2, 4))`	`0.6928037 1.281699` `0.7038999 1.184839`
$F\ (df = (4, 6, 9))$	`rmvf(n = 2, df = c(4, 6, 9))`	`1.145489 1.517436` `1.523112 2.211925`
Inverted Beta ($a = 4, l_1 = 2, l_2 = 6$)	`rmvinvbeta(n = 2, parm1 = 4, parm2 = c(2, 6))`	`0.4282041 1.878369` `0.4697152 2.459994`

Table 3: Example code for random numbers generation from bivariate distributions with `set.seed(2019)` and $n = 2$

Example code for maximum likelihood estimation of parameters

We define the following log-likelihood functions according to (9) for Mardia's Pareto Type I, Logistic, Burr, Cook-Johnson's uniform, generalized Lomax, and inverted beta distributions as follows.

```
loglik.mpareto1 <- function(data, par) {
        ll <- sum(dmvmpareto1(data, parm1 = par[1], parm2 = par[-1], log = TRUE))
}

loglik.logis <- function(data, par) {
        mu <- par[1:((length(par)) / 2)]
```

Bivariate Distribution and Quantiles	**Command**	**Output**
Lomax	`pmvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))`	`0.7678755`
$(a = 5, \theta_1 = 0.5, \theta_2 = 2)$	`smvlomax(q = c(1, 0.5), parm1 = 5, parm2 = c(0.5, 1))`	`0.03125`
$(x_1, x_2) = (1, 0.5)$	`qmvlomax(p = 0.5, parm1 = 5, parm2 = c(0.5, 1))`	`0.3928917`
Mardia's Pareto Type I	`pmvmpareto1(q = c(3, 1), parm1 = 5, parm2 = c(0.5, 2))`	`0.8473028`
$(a = 5, \theta_1 = 0.5, \theta_2 = 2)$	`smvmpareto1(q = c(3, 1), parm1 = 5, parm2 = c(0.5, 2))`	`0.01024`
$(x_1, x_2) = (3, 1)$	`qmvmpareto1(p = 0.5, parm1 = 5, parm2 = c(0.5, 2))`	`2.297463`
Logistic ($\mu_1 = 0.5$,	`pmvlogis(q = c(1, 2), parm1 = c(0.5, 1), parm2 = c(1, 1.5))`	`0.4717097`
$\mu_2 = 1, \sigma_1 = 1, \sigma_2 = 1.5$)	`smvlogis(q = c(1, 2), parm1 = c(0.5, 1), parm2 = c(1, 1.5))`	`0.188494`
$(x_1, x_2) = (1, 2)$	`qmvlogis(p = 0.5, parm1 = c(0.5, 1), parm2 = c(1, 1.5))`	`1.6041`
Burr ($a = 3, d_1 = 1$,	`pmvburr(q = c(0.5, 1), parm1 = 3, parm2 = c(1, 3), parm3 = c(2, 5))`	`0.4854017`
$d_2 = 3, c_1 = 2, c_2 = 5$)	`smvburr(q = c(0.5, 1), parm1 = 3, parm2 = c(1, 3), parm3 = c(2, 5))`	`0.01302666`
$(x_1, x_2) = (0.5, 1)$	`qmvburr(p = 0.5, parm1 = 3, parm2 = c(1, 3), parm3 = c(2, 5))`	`0.6853613`
Cook-Johnson's	`pmvunif(q = c(0.5, 0.75), parm = 0.3)`	`0.4782716`
Uniform ($a = 0.3$)	`smvunif(q = c(0.5, 0.75), parm = 0.3)`	`0.2282716`
$(x_1, x_2) = (0.5, 0.75)$	`qmvunif(p = 0.5, parm = 0.3, dim = 2)`	`0.598348`
Generalized Lomax ($a = 5$,	`pmvglomax(q = c(0.5, 1), parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`0.2645588`
$\theta_1 = 0.5$,	`smvglomax(q = c(0.5, 1), parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`0.2832001`
$\theta_2 = 1, l_1 = 2, l_2 = 4$)	`qmvglomax(p = 0.5, parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`1.050651`
$(x_1, x_2) = (0.5, 1)$		
F	`pmvf(q = c(1, 2), df = c(4, 6, 9))`	`0.4418794`
$(df = (4, 6, 9))$	`smvf(q = c(1, 2), df = c(4, 6, 9))`	`0.2296895`
$(x_1, x_2) = (1, 2)$	`qmvf(p = 0.5, df = c(4, 6, 9))`	`1.448382`
Inverted Beta	`pmvinvbeta(q = c(1, 2), parm1 = 4, parm2 = c(2, 6))`	`0.5873188`
$(a = 4, l_1 = 2, l_2 = 6)$	`smvinvbeta(q = c(1, 2), parm1 = 4, parm2 = c(2, 6))`	`0.1245114`
$(x_1, x_2) = (1, 2)$	`qmvinvbeta(p = 0.5, parm1 = 4, parm2 = c(2, 6))`	`1.568017`

Table 4: Example code for computation of CDF, Survival function and equcoordinate quantile from bivariate distributions

```
        sigma <- par[((length(par))/2 + 1):(length(par))]
        ll <- sum(dmvlogis(data, parm1 = mu, parm2 = sigma, log = TRUE))
}

loglik.burr <- function(data, par) {
        a <- par[1]
        d <- par[2:((length(par) + 1) / 2)]
        c <- par[((length(par) + 1) /2 + 1):(length(par))]
        ll = sum(dmvburr(data, parm1 = a, parm2 = d, parm3 = c, log = TRUE))
}

loglik.uniform <- function(data, par) {
        ll = sum(dmvunif(data, parm = par, log = TRUE))
}

loglik.glomax <- function(data, par) {
        a = par[1]
        theta = par[2:((length(par) + 1) / 2)]
        L = par[((length(par) + 1) / 2 + 1):(length(par))]
        ll = sum(dmvglomax(data, parm1 = a, parm2 = theta, parm3 = L, log = TRUE))
}
```

```
loglik.invbeta <- function(data, par) {
        a = par[1]
        l = par[-1]
        ll = sum(dmvinvbeta(data, parm1 = a, parm2 = l, log = TRUE))
}
```

Next, in order to create a data set in each case, we draw random sample of size $n = 300$. In each case, the random sampling is done by setting `set.seed(1)` in advance but not shown in the following code.

```
bvtMP1 <- rmvmpareto1(n = 300, parm1 = 5, parm2 = c(0.5, 2))

bvtlogis <- rmvlogis(n = 300, parm1 = c(0.5, 1), parm2 = c(1, 1.5))

bvtburr <- rmvburr(n = 300, parm1 = 3, parm2 = c(1, 3), parm3 = c(2, 5))

bvtuniform <- rmvunif(n = 300, parm = 0.3, dim = 2)

bvtglomax <- rmvglomax(n = 300, parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))

bvtinvbeta <- rmvinvbeta(n = 300, parm1 = 4, parm2 = c(2, 6))
```

The following code performs the maximum likelihood estimation for our sample data sets with appropriate optimization methods and constraints as described in Table 8. The output `est$convergence = 0` shows that in each case, the convergence is successfully achieved and `est$par` are the resulting estimated parameters.

- **Mardia's Pareto Type I** ($a = 5, \theta_1 = 0.5, \theta_2 = 2$):

```
> minParm2 = 1 / apply(bvtMP1, 2, min)
> est = constrOptim(theta = c(10, 10, 10), f = loglik.mpareto1, grad = NULL,
+                   data = bvtMP1, ui = diag(3), ci = c(0, minParm2),
+                   control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 4.6386220 0.4997126 1.9978535
```

- **Logistic** ($\mu_1 = 0.5, \mu_2 = 1, \sigma_1 = 1, \sigma_2 = 1.5$):

```
> est = optim(par = rep(10, 4), fn = loglik.logis, data = bvtlogis,
+             control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 0.3919998 0.8973157 0.9757377 1.5597609
```

- **Burr** ($a = 3, d_1 = 1, d_2 = 3, c_1 = 2, c_2 = 5$):

```
> est = constrOptim(theta = rep(10, 5), f = loglik.burr, grad = NULL,
+                   data = bvtburr, ui = diag(5), ci = rep(0, 5),
+                   control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 3.9149862 0.6488968 2.0685892 1.9271949 5.0769719
```

- **Cook-Johnson's uniform** ($a = 0.3$):

```
> est = optimize(f = loglik.uniform, data = bvtuniform,
+                interval = c(0, 100000), maximum = TRUE)
> est$maximum
[1] 0.3106469
```

- **Generalized Lomax** ($a = 5, \theta_1 = 0.5, \theta_2 = 1, l_1 = 2, l_2 = 4$):

```
> est = constrOptim(theta = rep(10, 5), f = loglik.glomax,
+                   grad = NULL, data = bvtglomax, ui = diag(5),
+                   ci = rep(0, 5), control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 3.918698 0.546744 1.791263 1.703105 5.217363
```

- **Inverted Beta** ($a = 4, l_1 = 2, l_2 = 6$):

```
> est = constrOptim(theta = rep(10, 3), f = loglik.invbeta, grad = NULL,
+                   data = bvtinvbeta, ui = diag(3), ci = rep(0, 3),
+                   control = list(fnscale = -1))
> est$convergence
[1] 0
> est$par
[1] 3.671536 1.824509 5.464654
```