Digital Complement

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),...)`

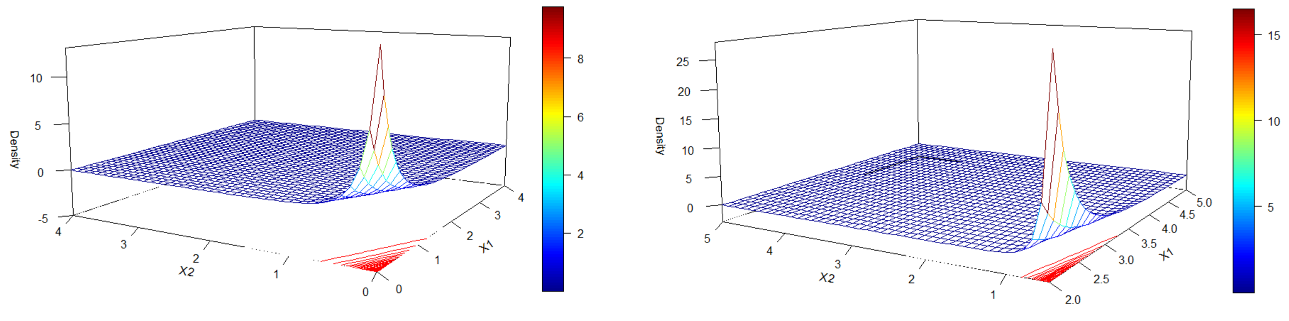
Inverted Beta ($\boldsymbol{a}$ = `parm1`, $\boldsymbol{l}$ = `parm2`)	`dmvinvbeta(x,parm1 = 1,parm2 = rep(1,k),...)`
	`pmvinvbeta(q,parm1 = 1,parm2 = rep(1,k),...)`
	`qmvinvbeta(p,parm1 = 1,parm2 = rep(1,k),...)`
	`rmvinvbeta(n,parm1 = 1,parm2 = rep(1,k))`
	`smvinvbeta(q,parm1 = 1,parm2 = rep(1,k),...)`

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*`).

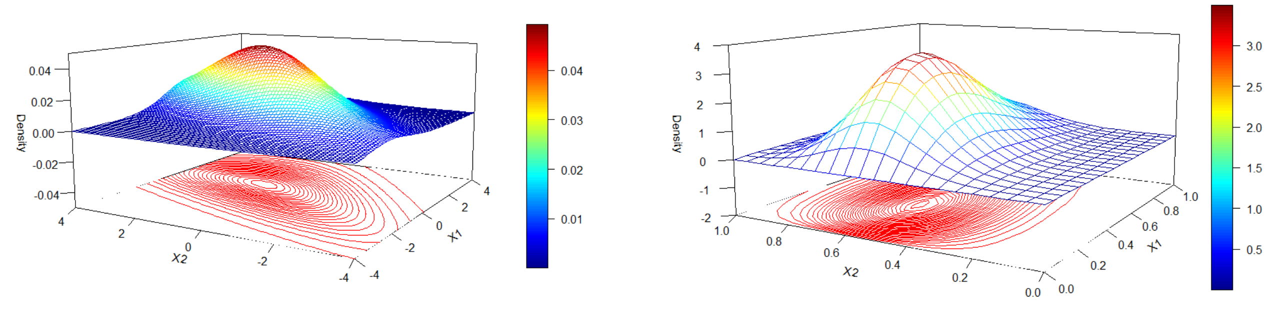
Example code for maximum likelihood estimation of parameters

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

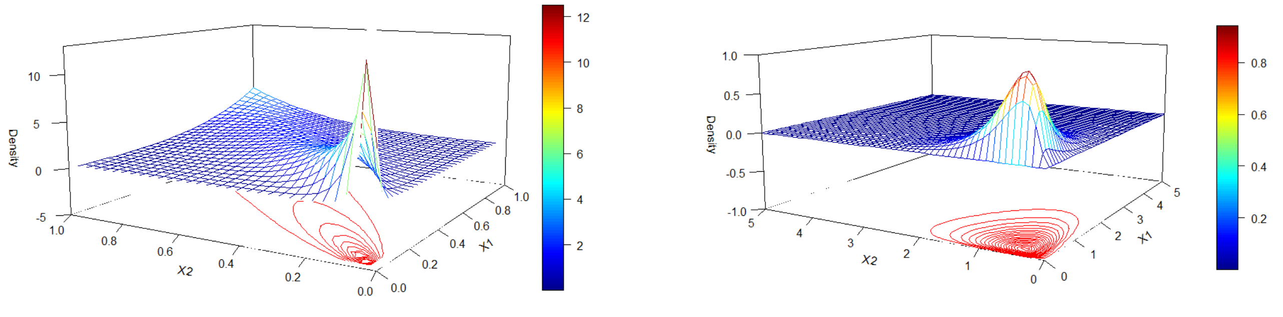
```
loglik.mpareto1 <- function(data, par) {
       ll <- sum(dmvmpareto1(data, parm1 = par[1], parm2 = par[-1], log = TRUE))
}
```



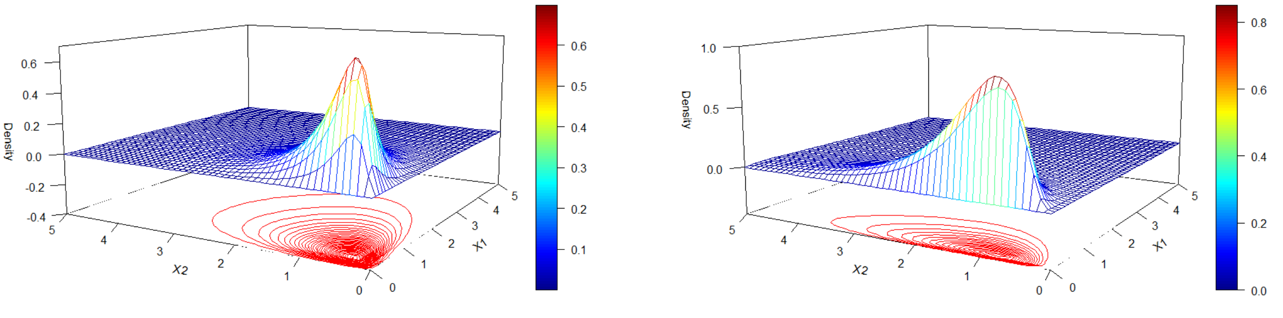
(a) Lomax **(b)** Mardia's Pareto Type I



(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 $n = 2$.

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

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 ($a = 5$, $\theta_1 = 0.5$,	`pmvglomax(q = c(0.5, 1), parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`0.2645588`
$\theta_2 = 1, l_1 = 2, l_2 = 4$)	`smvglomax(q = c(0.5, 1), parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`0.2832001`
$(x_1, x_2) = (0.5, 1)$	`qmvglomax(p = 0.5, parm1 = 5, parm2 = c(0.5, 1), parm3 = c(2, 4))`	`1.050651`
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.

```
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 **??**. 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
```