Supplementary material to:
C.C. Kokonendji and P. Puig, "Fisher dispersion index for multivariate count distributions: a review and a new proposal", *Journal of Multivariate Analysis.*

Appendix C: The GDI of the bivariate COM-Poisson distribution

Consider the more recent bivariate COM-Poisson distribution of Sellers et al. (2016)[1]. It has been proposed to include the bivariate Bernoulli of Section 4.1, the bivariate Poisson of Section 4.2, and the bivariate geometric distributions all as special cases. Following Sellers et al. (2016) who consider the index FI_2, we also show that numerical computations are required for the complete investigation of our proposed indices GDI and MDI.

Indeed, let us denote it by $\boldsymbol{Y} = (Y_1, Y_2)^\top \sim \mathcal{COMP}_2(\boldsymbol{p}, \mu, \gamma)$ where $\boldsymbol{p} = \{p_{00}, p_{01}, p_{10}, p_{11}\}$ is related to the bivariate Bernoulli and geometric components, $\mu > 0$ for generating Poisson components with σ_0^2, μ_1, μ_2, and $\gamma \geq 0$ is a kind of dispersion parameter. For instance, for $\gamma = 1$ both marginal distributions are univariate Poisson with mean parameters $\mu p_{1+} = \mu_1 + \sigma_0^2$ and $\mu p_{+1} = \mu_2 + \sigma_0^2$ such that $\mu p_{11} = \sigma_0^2$; for $\gamma \to \infty$ we have a bivariate Bernoulli distribution with their four individual probabilities

$$p_{00}^* = \frac{1 + \mu p_{00}}{1 + \mu}\,,\ p_{01}^* = \frac{\mu p_{01}}{1 + \mu}\,,\ p_{10}^* = \frac{\mu p_{10}}{1 + \mu}\,,\ p_{11}^* = \frac{\mu p_{11}}{1 + \mu}\,;$$

and, for $\gamma = 0$ this is a bivariate geometric distribution. Hence, Sellers et al. (2016) show the following ingredients needed here:

$$\mathrm{E}Y_1 = \mu p_{1+}\left(\frac{\partial \log Z(\mu,\gamma)}{\partial \mu}\right),\ \ \mathrm{E}Y_2 = \mu p_{+1}\left(\frac{\partial \log Z(\mu,\gamma)}{\partial \mu}\right),$$

$$\mathrm{var}Y_1 = \mu^2 p_{1+}^2\left(\frac{\partial^2 \log Z(\mu,\gamma)}{\partial \mu^2}\right) + \mu p_{1+}\left(\frac{\partial \log Z(\mu,\gamma)}{\partial \mu}\right),$$

$$\mathrm{var}Y_2 = \mu^2 p_{+1}^2\left(\frac{\partial^2 \log Z(\mu,\gamma)}{\partial \mu^2}\right) + \mu p_{+1}\left(\frac{\partial \log Z(\mu,\gamma)}{\partial \mu}\right),$$

$$\mathrm{cov}(Y_1, Y_2) = \mu^2 p_{1+}p_{+1}\left(\frac{\partial^2 \log Z(\mu,\gamma)}{\partial \mu^2}\right) + \mu p_{11}\left(\frac{\partial \log Z(\mu,\gamma)}{\partial \mu}\right),$$

with $Z(\mu,\gamma) = \sum_{s=0}^{\infty} \mu^s/(s!)^\gamma$. Denoting $Z_1 = \partial \log Z(\mu,\gamma)/\partial\mu$ and $Z_2 = \partial^2 \log Z(\mu,\gamma)/\partial\mu^2$, therefore one has

$$\begin{aligned}\mathrm{GDI}(\boldsymbol{Y}) &= \frac{\mu p_{1+}^3 Z_2 + p_{1+}^2 Z_1 + \mu p_{+1}^3 Z_2 + p_{+1}^2 Z_1 + 2(\mu p_{1+}p_{+1}Z_2 + p_{11}Z_1)\sqrt{p_{1+}p_{+1}}}{(p_{1+}^2 + p_{+1}^2)Z_1} \\ &= 1 + \frac{\mu p_{1+}^3 Z_2 + \mu p_{+1}^3 Z_2 + 2(\mu p_{1+}p_{+1}Z_2 + p_{11}Z_1)\sqrt{p_{1+}p_{+1}}}{(p_{1+}^2 + p_{+1}^2)Z_1} \geq 0\end{aligned}$$

and $\mathrm{MDI}(\boldsymbol{Y}) = \mu^2\{(p_{1+}^2 + p_{+1}^2)Z_1 + \mu(p_{1+}^3 + p_{+1}^3)Z_3\}/(p_{1+}^2 + p_{+1}^2) \geq 0$.

It is important to remark that, depending on the values of the parameters $(\boldsymbol{p}, \mu, \gamma)$, the four indices FI_2, GDI, MDI and also sGV can provide different conclusions with respect to the bivariate over-, equi- and under-dispersion as in the previous particular cases. One can refer to Kokonendji et al. (2008)[2] who clearly established the result for the univariate COM-Poisson with overdispersion for $\gamma \in [0, 1)$, equidispersion for $\gamma = 1$, and underdispersion for $\gamma \in (1, \infty]$.

[1] Sellers, K.F., Morris, D.S. and Balakrishnan, N. (2016). Bivariate Conway-Maxwell-Poisson distribution: formulation, properties, and inference. *Journal of Multivariate Analysis* **150**, 152-168.

[2] Kokonendji, C.C., Mizère, D. and Balakrishnan, N. (2008). Connections of the Poisson weight function to overdispersion and underdispersion. *Journal of Statistical Planning and Inference* **138**, 1287-1296.

Appendix D: Script in R for making bootstrap with bivariate GDI and MDI

```
# Data no.1 (Holgate, 1966)
y=c(rep(0,34),rep(0,12),rep(0,4),rep(0,5),rep(0,2),rep(1,8),rep(1,13),
rep(1,3),rep(1,3),rep(2,3),rep(2,6),rep(2,1),rep(2,2),rep(3,1),rep(3,1),
rep(3,1),0)

x=c(rep(0,34),rep(1,12),rep(2,4),rep(3,5),rep(4,2),rep(0,8),rep(1,13),
rep(2,3),rep(3,3),rep(0,3),rep(1,6),rep(2,1),rep(3,2),rep(0,1),rep(1,1),
rep(3,1),6)

data=cbind(y,x)
m=c(mean(data[,1]),mean(data[,2]))

# GDI
(t(m^.5)%*%var(data)%*%(m^.5))/(t(m)%*%m)
# MDI
(t(m^.5)%*%diag(diag(var(data)))%*%(m^.5))/(t(m)%*%m)

# bootstrap gdi and mdi (\textbf{10000 replicates})
n=length(y)
nb=10000; z=seq(1,n);gdi=numeric(nb);mdi=numeric(nb)
for(i in 1:nb){
zb=sample(z,n,replace=T)
datab=cbind(data[zb,1],data[zb,2])
m=c(mean(datab[,1]),mean(datab[,2]))
gdi[i]=(t(m^.5)%*%var(data)%*%(m^.5))/(t(m)%*%m)}
mdi[i]=(t(m^.5)%*%diag(diag(var(datab)))%*%(m^.5))/(t(m)%*%m)}
hist(gdi)
hist(mdi)

# bootstrap mean and sd
c(mean(gdi),sd(gdi))
c(mean(mdi),sd(mdi))
# 95% CI approx.
quantile(gdi,c(0.025,0.975))
quantile(mdi,c(0.025,0.975))
```

Appendix E: Simulated count data of Table 4 from the NORTA algorithm

```
# Simulation of multivariate count data using NORTARA package
# Loading extra packages
require(NORTARA)
# Marginal distributions:
# X1 ~ NB(\mu = 1.15, phi = 2.35) E(X1) = 1.15 and var(X1) = 1.91
# X2 ~ P(\mu = 5) E(X2) = 5 and var(X2) = 5
# X3 ~ NB(\mu = 4.91, phi = 16) E(X3) = 4.91 and var(X3) = 6.42
# X4 ~ B(\mu = 0.5) E(X4) = 0.5 and var(X4) = 0.25
# X5 ~ NB(\mu = 2.66, phi = 2.15) E(X5) = 2.66 and var(X5) = 6.67
# Correlation matrix
COR <- matrix(c(1,-0.33, 0.57, 0.12, 0.42, -0.33, 1, -0.15, 0.09, 0.36,
                0.57, -0.15, 1, 0.31, 0.27, 0.12, 0.09, 0.31, 1, 0.18,
```

```
                0.42, 0.36, 0.27, 0.18, 1), ncol = 5, nrow = 5)

invcdfnames <- c("qnbinom","qpois","qnbinom","qbinom","qnbinom")
paramslists <- list(
    m1 = list(mu = 1.15, size = 2.35),
    m2 = list(lambda = 5),
    m3 = list(mu = 4.91, size = 16),
    m4 = list(prob = 0.5, size = 1),
    m5 = list(mu = 2.66, size = 2.15)
)

data <- NORTARA::genNORTARA(n = 300, cor_matrix = COR,
                            invcdfnames = invcdfnames,
                            paramslists = paramslists)
apply(data, 2, mean)
apply(data, 2, var)
```

Data of Table 4 "*i*" $x1_i$ $x2_i$ $x3_i$ $x4_i$ $x5_i$**:** "1" 2 4 7 1 4 "2" 5 2 10 1 1 "3" 1 9 3 1 2 "4" 0 3 3 0 1 "5" 0 5 4 1 1 "6" 3 4 6 0 3 "7" 6 1 9 1 1 "8" 0 8 4 0 1 "9" 3 2 5 0 3 "10" 1 4 6 1 4 "11" 1 5 2 1 3 "12" 2 10 6 1 6 "13" 0 6 0 0 1 "14" 4 3 13 1 2 "15" 0 8 2 1 1 "16" 0 8 2 0 5 "17" 0 4 1 0 1 "18" 3 8 4 1 8 "19" 2 4 7 0 4 "20" 3 3 7 1 5 "21" 2 4 6 0 2 "22" 0 2 0 0 1 "23" 4 9 7 1 10 "24" 1 5 6 0 0 "25" 1 6 6 0 3 "26" 1 9 4 1 9 "27" 0 5 1 0 2 "28" 1 8 8 1 2 "29" 1 7 5 0 4 "30" 1 6 6 0 4 "31" 3 5 8 1 2 "32" 1 3 3 1 2 "33" 0 13 6 1 5 "34" 1 7 7 1 3 "35" 2 6 5 1 7 "36" 1 4 3 1 2 "37" 3 4 8 1 2 "38" 2 3 7 0 3 "39" 0 5 2 1 0 "40" 1 5 3 0 7 "41" 0 4 5 0 2 "42" 1 7 7 0 5 "43" 0 7 2 1 3 "44" 0 3 3 0 0 "45" 1 3 3 1 0 "46" 0 2 6 1 0 "47" 1 5 5 1 1 "48" 2 5 5 1 2 "49" 4 4 7 1 7 "50" 4 8 10 1 14 "51" 2 5 2 0 2 "52" 2 2 4 0 1 "53" 0 4 2 0 1 "54" 1 4 12 1 3 "55" 0 7 6 1 1 "56" 1 4 3 0 3 "57" 1 1 4 0 1 "58" 2 8 6 1 1 "59" 2 4 7 1 4 "60" 4 3 9 0 5 "61" 1 5 8 1 2 "62" 0 3 2 0 1 "63" 0 1 2 0 0 "64" 3 3 4 0 6 "65" 1 6 3 0 4 "66" 3 0 12 0 0 "67" 0 3 3 0 0 "68" 1 6 3 0 2 "69" 0 9 2 0 1 "70" 1 5 3 1 4 "71" 1 4 4 1 1 "72" 0 9 2 0 8 "73" 3 5 9 1 3 "74" 2 7 6 1 4 "75" 0 5 5 0 0 "76" 3 5 5 0 5 "77" 1 4 4 0 2 "78" 0 4 5 0 2 "79" 2 2 5 0 1 "80" 1 9 8 1 3 "81" 2 3 4 1 4 "82" 3 4 9 1 5 "83" 0 5 1 0 0 "84" 2 4 3 0 3 "85" 4 3 3 0 3 "86" 1 6 4 1 2 "87" 0 8 3 0 2 "88" 1 2 5 0 1 "89" 0 6 1 0 0 "90" 0 6 1 0 0 "91" 0 4 6 0 1 "92" 0 8 4 1 4 "93" 0 7 3 0 2 "94" 1 5 5 0 1 "95" 0 8 2 0 0 "96" 1 4 3 1 2 "97" 2 8 6 1 3 "98" 2 4 5 1 3 "99" 1 5 3 0 4 "100" 2 3 5 1 2 "101" 0 6 4 1 0 "102" 2 5 7 1 3 "103" 2 4 3 1 5 "104" 1 2 7 0 3 "105" 0 4 3 0 1 "106" 0 7 5 1 0 "107" 1 8 7 1 6 "108" 1 3 5 0 1 "109" 1 8 7 0 3 "110" 4 5 6 0 8 "111" 0 5 5 1 0 "112" 1 9 6 1 4 "113" 2 5 5 0 5 "114" 1 2 2 1 3 "115" 1 7 9 1 5 "116" 3 3 7 0 3 "117" 2 4 4 0 0 "118" 0 12 4 0 6 "119" 2 2 6 1 1 "120" 4 2 8 0 1 "121" 2 5 4 1 4 "122" 1 4 5 1 3 "123" 0 4 7 1 1 "124" 0 9 3 0 4 "125" 1 4 7 0 0 "126" 0 10 3 1 4 "127" 1 6 4 0 2 "128" 0 8 7 0 2 "129" 1 6 6 0 3 "130" 0 10 7 1 1 "131" 5 3 8 1 6 "132" 3 8 5 0 16 "133" 2 3 6 0 1 "134" 0 8 3 0 6 "135" 4 4 9 1 8 "136" 3 3 3 0 4 "137" 1 4 8 1 3 "138" 1 7 5 1 3 "139" 0 8 5 1 3 "140" 1 7 3 0 5 "141" 1 6 7 1 1 "142" 0 6 5 1 3 "143" 1 9 6 0 6 "144" 1 7 4 1 5 "145" 1 5 7 1 6 "146" 1 7 4 0 3 "147" 1 3 4 1 0 "148" 1 3 5 0 1 "149" 0 6 1 0 3 "150" 3 1 7 0 3 "151" 1 9 7 1 6 "152" 0 9 3 1 1 "153" 2 2 5 0 0 "154" 0 6 8 1 2 "155" 1 3 4 0 1 "156" 0 4 3 1 0 "157" 1 4 2 1 2 "158" 2 3 8 1 6 "159" 0 5 3 0 0 "160" 0 6 7 1 1 "161" 0 8 7 0 2 "162" 4 6 6 1 6 "163" 0 11 2 1 4 "164" 0 6 5 0 0 "165" 1 1 6 1 0 "166" 1 3 7 1 2 "167" 0 7 3 1 0 "168" 1 4 5 0 1 "169" 3 5 11 0 10 "170" 2 5 5 0 1 "171" 1 3 10 1 2 "172" 2 4 6 0 2 "173" 1 5 7 0 4 "174" 2 3 3 0 3 "175" 2 4 6 1 2 "176" 0 5 2 0 1 "177" 0 6 6 0 2 "178" 2 4 8 0 6 "179" 0 8 3 0 3 "180" 1 3 5 1 0 "181" 0 4 3 0 0 "182" 0 8 7 1 1 "183" 1 7 4 0 3 "184" 0 7 1 0 0 "185" 1 3 6 1 1 "186" 0 6 0 0 1 "187" 2 4 8 1 1 "188" 1 6 4 0 2 "189" 2 3 3 0 3 "190" 0 3 2 0 2 "191" 1 3 5 0 3 "192" 2 6 5 1 6 "193" 3 3 7 0 2 "194" 0 4 2 1 0 "195" 4 0 6 1 3 "196" 0 8 4 0 3 "197" 2 4 6 1 3 "198" 1 3 1 0 2 "199" 0 5 5 0 4 "200" 1 8 5 1 4 "201" 2 3 6 1 2 "202" 0 11 4 1 5 "203" 1 6 5 1 4 "204" 1 6 6 0 2 "205" 1 5 3 0 0 "206" 1 5 6 1 2 "207" 3 3 8 1 1 "208" 3 5 7 1 9 "209" 3 3 6 0 2 "210" 1 7 4 0 3 "211" 3 6 6 1 5 "212" 0 5 1 1 1 "213" 0 5 0 1 1 "214" 2 6 4 1 11 "215" 0 9 3 0 6 "216" 3 1 8 1 1 "217" 3 2 5 1 1 "218" 1 1 7 1 1 "219" 4 4 12 1 4 "220" 4 7 5 0 14 "221" 0 7 3 0 2 "222" 0 6 6 1 3 "223" 1 5 6 0 5 "224" 2 6 7 1 4 "225" 4 1 11 1 1 "226" 1 6 6 0 5 "227" 0 9 1 1 7 "228" 1 3 5 0 2 "229" 1 4 7 0 2 "230" 0 8 4 0 3 "231" 6 2 11 1 3 "232" 3 3 8 0 3 "233" 0 1 3 0 0 "234" 0 4 4 0 0 "235" 5 3 8 0 3 "236" 2 4 8 1 3 "237" 0 5 4 1 1 "238" 4 3 8 0 2 "239" 2 5 4 0 3 "240" 2 3 5 0 3 "241" 2 5 8 1 3 "242" 0 9 0 0 3 "243" 1 4 4 0 2 "244" 0 4 6 0 1 "245" 1 4 5 1 4 "246" 3 6 8 1 7 "247" 0 4 3 0 0 "248" 1 7 5 0 3 "249" 0 5 6 0 1 "250" 3 2 5 0 0 "251" 1 0 4 1 0 "252" 2 2 2 0 2 "253" 0 7 4 0 0 "254" 1 3 7 1 0 "255" 1 6 5 1 4 "256" 0 4 2 1 0 "257" 1 4 10 1 2 "258" 1 3 4 0 1 "259" 1 4 7 0 3 "260" 0 5 3 0

1 ”261” 2 4 10 0 4 ”262” 0 7 2 1 3 ”263” 0 3 2 0 0 ”264” 1 6 5 0 5 ”265” 1 5 10 1 3 ”266” 2 3 5 1 2 ”267” 0 4 2 0 4 ”268” 1 4 6 1 2 ”269” 0 4 4 0 1 ”270” 0 4 2 0 2 ”271” 0 4 1 1 1 ”272” 0 1 2 0 0 ”273” 1 4 8 0 5 ”274” 1 5 6 1 1 ”275” 3 3 7 1 4 ”276” 1 12 4 1 17 ”277” 4 6 9 1 3 ”278” 1 7 4 1 3 ”279” 0 7 5 1 0 ”280” 1 3 4 1 1 ”281” 4 2 5 0 2 ”282” 0 8 1 0 0 ”283” 0 6 3 0 2 ”284” 0 4 2 0 0 ”285” 0 2 1 0 0 ”286” 0 4 4 1 0 ”287” 0 9 1 1 2 ”288” 1 5 2 0 1 ”289” 1 3 5 1 2 ”290” 1 5 9 0 2 ”291” 2 0 7 1 0 ”292” 0 5 6 0 2 ”293” 0 4 2 0 1 ”294” 2 3 4 0 0 ”295” 0 3 4 0 0 ”296” 1 4 6 1 2 ”297” 1 2 4 0 0 ”298” 4 6 16 1 9 ”299” 4 6 7 0 10 ”300” 0 2 3 0 0

Appendix F: Scripts in R for asymptotic variances and confidence intervals of GDI and MDI

```
# Reading the table of data
data=read.table("Holgate_data.txt", sep = " ", header = TRUE)
#Function that returns the vector composed by the averages of each variable
aux_esp<-function(data){
  res<-c()
  n=dim(data)
  n=n[2]
  for (i in 1:n){
    res[i]=mean(data[,i])
  }
  return(res)
}
#Function that returns the vector composed by the variances of each variable
aux_var<-function(data){
  res<-c()
  n=dim(data)
  n=n[2]
  for (i in 1:n){
    res[i]=var(data[,i])
  }
  return(res)
}
# Function that returns the matrix composed by the variances on its diagonal and 0 else
aux_diag_var<-function(data){
  res<-c()
  n=dim(data)
  n=n[2]
  for (i in 1:n){
    res[i]=var(data[,i])
  }
  return(diag(res))
}
# Function that calculates the MDI
MDI<-function(data){
  tmp_E=aux_esp(data)
  tmp_D=aux_diag_var(data)
  res=sqrt(tmp_E)%*%tmp_D%*%sqrt((tmp_E))/(tmp_E%*%(tmp_E))
  return(res)
}
# Function that calculates the GDI
GDI<-function(data){
  tmp_E=aux_esp(data)
  tmp_C=cov(data)
```

```
  res=sqrt(tmp_E)%*%tmp_C%*%sqrt((tmp_E))/(tmp_E%*%(tmp_E))
  return(res)
}
# Fonction that returns the delta vector
vec_delta<-function(data){
  k=dim(data)
  k=k[2]
  res<-c()
  tmp_E=aux_esp(data)
  for (i in 1:k){
    res[i]=0
    for (j in 1:k){
      res[i]=res[i]+tmp_E[j]/tmp_E[i]*cov(data[,i],data[,j])
    }
    res[i]=(res[i]-2*tmp_E[i]*GDI(data))/(tmp_E%*%(tmp_E))
  }
  for (i in 1:k){
    for(j in i:k){
      if (i==j){
        tmp=tmp_E[i]/(tmp_E%*%(tmp_E))
      }
      else{
        tmp=2*sqrt(tmp_E[i]*tmp_E[j])/(tmp_E%*%(tmp_E))
      }
      res<-c(res,tmp)
    }
  }
  return(res)
}
# Function that returns the gamma3 matrix
mat_gamma3<-function(data){
  k=dim(data)
  k=k[2]
  tmp_E=aux_esp(data)
  res<-c()
  for (j in 1:k){
    tmp_vec<-c()
    for (jj in 1:k){
      for (ll in jj:k){
        tmp=cov(data[,j],data[,jj]*data[,ll])
        tmp_vec=c(tmp_vec,tmp)
      }
    }
    res=rbind(res,tmp_vec)
  }
  rownames(res)=c()
  return(res)
}
# Function that returns the gamma4 matrix
mat_gamma4<-function(data){
  k=dim(data)
  k=k[2]
```

```
  tmp_E=aux_esp(data)
  res<-c()
  for (j in 1:k){
    for (l in j:k){
      tmp_vec<-c()
      for (jj in 1:k){
        for (ll in jj:k){
          tmp=cov(data[,j]*data[,l],data[,jj]*data[,ll])
          tmp_vec=c(tmp_vec,tmp)
        }
      }
      res=rbind(res,tmp_vec)
    }
  }
  rownames(res)=c()
  return(res)
}
# Function that returns the gamma matrix
mat_gamma<-function(data){
  tmp=cbind(cov(data),mat_gamma3(data))
  tmp2=cbind(t(mat_gamma3(data)),mat_gamma4(data))
  res=rbind(tmp,tmp2)
  return(res)
}
# Function that calculates the variance of GDI
sigma_GDI<-function(data){
  delta=vec_delta(data)
  gamma=mat_gamma(data)
  res=delta%*%gamma%*%delta
  return(res)
}

vec_lambda<-function(data){ # Function that returns the  lambda  vector
  k=dim(data)
  k=k[2]
  res<-c()
  tmp_E=aux_esp(data)
  tmp_V=aux_var(data)
  for (i in 1:k){
    res[i]=(tmp_V[i]-2*tmp_E[i]*MDI(data))/(tmp_E%*%tmp_E)
  }
  for (i in 1:k){
    res[i+k]=tmp_E[i]/(tmp_E%*%tmp_E)
  }
  return(res)
}

mat_pi3<-function(data){ # Function that returns the pi3 matrix
  k=dim(data)
  k=k[2]
  tmp_E=aux_esp(data)
  res<-c()
```

```
  for (j in 1:k){
    tmp_vec<-c()
    for (jj in 1:k){
      tmp=cov(data[,j],data[,jj]*data[,jj])
      tmp_vec=c(tmp_vec,tmp)
    }
    res=rbind(res,tmp_vec)
  }
  rownames(res)=c()
  return(res)
}

mat_pi4<-function(data){ # Function that returns the  pi4 matrix
  k=dim(data)
  k=k[2]
  tmp_E=aux_esp(data)
  res<-c()
  for (j in 1:k){
    tmp_vec<-c()
    for (jj in 1:k){
      tmp=cov(data[,j]*data[,j],data[,jj]*data[,jj])
      tmp_vec=c(tmp_vec,tmp)
    }
    res=rbind(res,tmp_vec)
  }
  rownames(res)=c()
  return(res)
}

mat_pi<-function(data){ # Function that returns the  pi matrix
  tmp=cbind(cov(data),mat_pi3(data))
  tmp2=cbind(t(mat_pi3(data)),mat_pi4(data))
  res=rbind(tmp,tmp2)
  return(res)
}

sigma_MDI<-function(data){ # Function that calculates the variance of MDI
  lambda=vec_lambda(data)
  pi=mat_pi(data)
  res=lambda%*%pi%*%lambda
  return(res)
}

ind_tab<-function(data,n){ # Function that returns a table with all the information
  k=length(n)
  res<-c()
  for (i in 1:k){
    tmp_vec<-c()
    tdata=data[1:n[i],]
    tmp_vec=c(det(cor(tdata)),sigma_GDI(tdata),sigma_MDI(tdata),GDI(tdata)
    ,qnorm(0.975)*sqrt(sigma_GDI(tdata)/n[i]),MDI(tdata),
    qnorm(0.975)*sqrt(sigma_MDI(tdata)/n[i]))
```

```
    res=rbind(res,tmp_vec)
  }
  rownames(res)=n
  colnames(res)=c("det(rho)","sigma2GDI","sigma2MDI","GDI +-","E.t(int)",
  "MDI+-","E.t(int)")
  return(res)
}
# For this function ind_tab, data is the data set and n a vector
# with values (sample size) to calculate the different indices
n=c(50,100,300,500,1000,3000,5000,10000)
ind_tab(data,n)

invcdfnames <- c("qnbinom","qpois","qnbinom","qbinom","qnbinom")
paramslists <- list(
  m1 = list(mu = 1.15, size = 2.35),
  m2 = list(lambda = 5),
  m3 = list(mu = 4.91, size = 16),
  m4 = list(prob = 0.5, size = 1),
  m5 = list(mu = 2.66, size = 2.15)
)
# Matrix of specified input correlation
COR <- matrix(c(1,-0.33, 0.57, 0.12, 0.42, -0.33, 1, -0.15, 0.09, 0.36,
                0.57, -0.15, 1, 0.31, 0.27, 0.12, 0.09, 0.31, 1, 0.18,
                0.42, 0.36, 0.27, 0.18, 1), ncol = 5, nrow = 5)
# Function which calculates the correlation matrix
sigma1<- BoundingRA(cor_matrix = COR, invcdfnames, paramslists)
data<-NORTARA::genNORTARA(n=10000, cor_matrix=COR, invcdfnames=invcdfnames,
 paramslists = paramslists)
apply(data, 2, mean) # Function that applies the average of each column
apply(data, 2, var) # Function that applies the variance of each column
# Function that returns two matrix containing the GDI and MDI in n
#simulated replicates
# n is a vector with the same purpose as previously
n=c(50,100,300,500,1000,3000,5000,10000)
ind_tab(data,n)
#Function that returns two matrices gdi and mdi for n replicates
aux_box<-function(n,nrep){
  mat_gdi=c()
  mat_mdi=c()
  nsimu=n[length(n)]
  for (i in 1:nrep){
    data <- NORTARA::genNORTARA(nsimu, cor_matrix = COR,
                                invcdfnames = invcdfnames,
                                paramslists = paramslists)
    tmp=ind_tab(data,n)
    mat_gdi=rbind(mat_gdi,tmp[,4])
    mat_mdi=rbind(mat_mdi,tmp[,6])
  }
  return(list(mat_gdi,mat_mdi))
}

n=c(50,100,300,500,1000,3000,5000,10000)
```

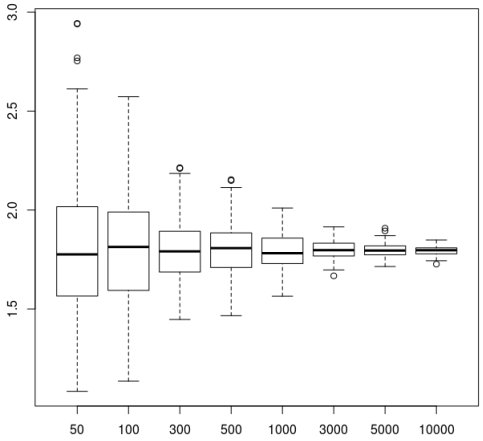
```
data_box=aux_box(n,100) #Function that returns boxplots for 100 replicates
gdi=data_box[[1]]
mdi=data_box[[2]]

boxplot(gdi)
boxplot(mdi)
```

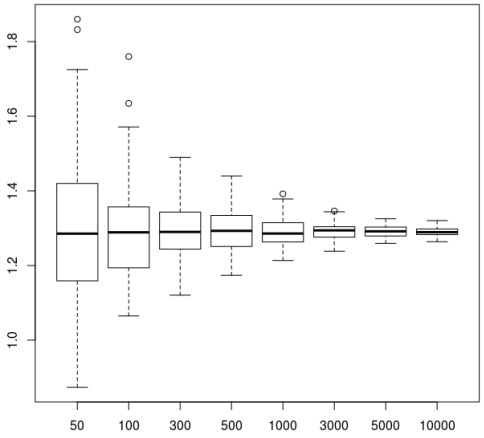
Figure 1: Boxplots for the targets GDI and MDI with 100 replicates according to sample sizes and simulation parameters of Tables 5, 6 and 7

(a) GDI = 1.8717

(b) MDI = 1.3254

k = 5: **over-**

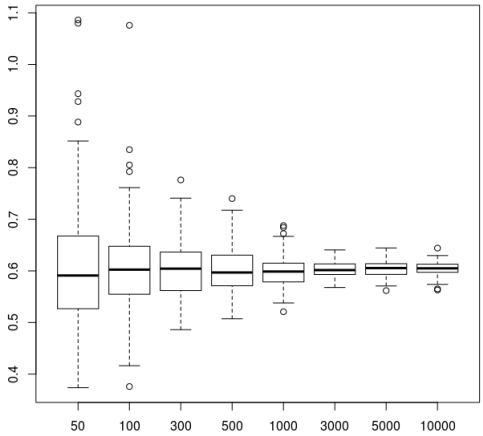


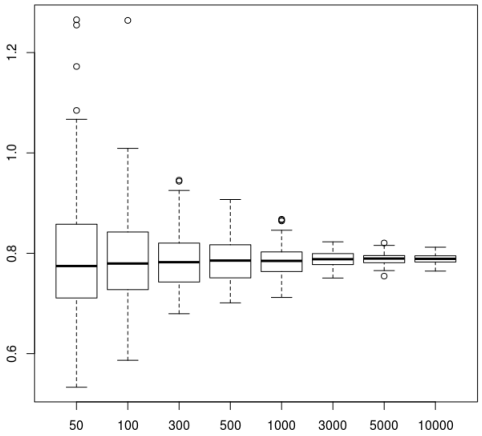


(c) GDI = 0.6049

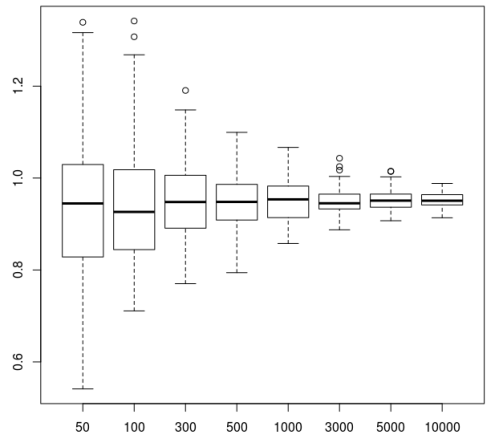
(d) MDI = 0.7882

k = 3: **under-**

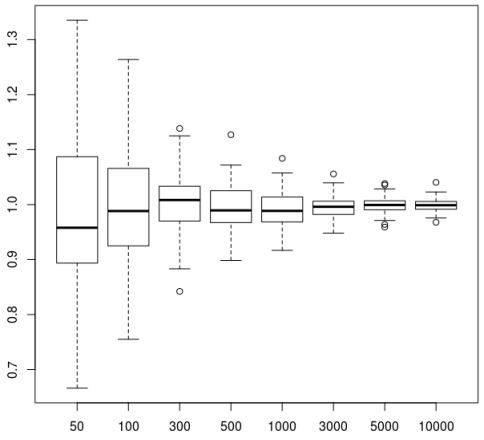




(e) GDI = 0.9520



(f) MDI = 1



$k = 2$: **equi-**