

Supplementary Material :
A variable clustering approach for overdispersed high – dimensional
count data using a copula – based mixture model

Keywords: High-dimensional data, Count Data, Overdispersion, Mixture modelling, Copula modelling, Clustering variables.

Contents

A	Supplementary material	3
A.1	R code for the data generation	3
A.2	Simulation settings	4
A.3	R code for ML estimation	5
A.4	R code for MM estimation	6
A.5	R code for the clustering algorithm	6
A.6	R code for the model selection	8
A.7	ANOVA table	11

A Supplementary material

A.1 R code for the data generation

```
GenerateCorCount <- function (nsub, ngenes, nclus, alpha, lambda, PI, copula, seed){
  library(copula)
  nobj = nsub
  K = nclus
  parmarg = list()
  for (k in 1:K){
    parmarg[[k]] = list(shape=alpha[k], rate=alpha[k])
  }
  set.seed(seed[1])
  switch(copula,
    clayton = {
      myCop = claytonCopula(PI, dim=K)
      myMvd = mvdc(copula=myCop, margins=rep("gamma", K), paramMargins=parmarg)
      Z = rMvdc(nobj, myMvd)
    },
    gumbel = {
      myCop = gumbelCopula(PI, dim=K)
      myMvd = mvdc(copula=myCop, margins=rep("gamma", K), paramMargins=parmarg)
      Z = rMvdc(nobj, myMvd)
    },
    frank = {
      myCop = frankCopula(PI, dim=K)
      myMvd = mvdc(copula=myCop, margins=rep("gamma", K), paramMargins=parmarg)
      Z = rMvdc(nobj, myMvd)
    }
  )
  Y = matrix(rep(0, nobj), ncol=1)
  seed.latent = seed[2:length(seed)]
  for (J in 1:K){
    set.seed(seed.latent[J])
    Y.J = rep(0, nobj)
    for (I in 1:nobj){
      Y.J[I] = rpois(1, lambda=ngenes[J]*lambda[J]*(Z[I, J]))
    }
    Y = cbind(Y, Y.J)
  }
  Y = data.frame(Y[, -c(1)])
  colnames(Y) = c(paste0("cluster", 1:K), "label")[1:K]
  return(list(Y, seed))
}
```

A.2 Simulation settings

Settings nr.	θ	N	P	λ	α	Partitioning
1	2	50	640	500, 1000	4, 6	HOM
2				500, 1000	4, 6	HET
3				200, 1200, 5000, 7500	3, 4, 5, 6	HOM
4				200, 1200, 5000, 7500	3, 4, 5, 6	HET
5				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HOM
6				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HET
7		8400	640	500, 1000	4, 6	HOM
8				500, 1000	4, 6	HET
9				200, 1200, 5000, 7500	3, 4, 5, 6	HOM
10				200, 1200, 5000, 7500	3, 4, 5, 6	HET
11				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HOM
12				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HET
13	500	500	8400	500, 1000	4, 6	HOM
14				500, 1000	4, 6	HET
15				200, 1200, 5000, 7500	3, 4, 5, 6	HOM
16				200, 1200, 5000, 7500	3, 4, 5, 6	HET
17				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HOM
18				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HET
19		500	8400	500, 1000	4, 6	HOM
20				500, 1000	4, 6	HET
21				200, 1200, 5000, 7500	3, 4, 5, 6	HOM
22				200, 1200, 5000, 7500	3, 4, 5, 6	HET
23				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HOM
24				20, 100, 500, 1000, 2500, 5000, 10000, 15000	1.5, 2, 3, 4, 4.5, 5, 5.5, 6	HET

Table 1: Simulation settings

A.3 R code for ML estimation

```

library(numDeriv)
library(MASS)
library("optimParallel")
parms = c(eta,lambda,pi)
ll.fn = function(parms,nclus,Y,m,copula){
alpha = parms[1:nclus]
LBDA = parms[(nclus+1):(2*nclus)]
PI = tail(parms,n=1)
ll = rep(0,dim(Y)[1])
switch(copula,
      clayton = {
        f=parse(text=c("(" ,paste("(u",1:nclus,"^(-PI)-1)",sep="",collapse="+"),"+1)^(-1/PI)"))},
      gumbel = {
        f=parse(text=c("exp(-(",paste("(-log(u",1:nclus,")) ^ PI",
        sep="",collapse="+"),")^(1/PI)")))},
      frank = {
        f=parse(text=c("-(1/PI)*log(1-",paste("(1-exp(-u",1:nclus,"*PI)"))",
        sep="",collapse="*"),"/((1-exp(-PI))^(",nclus-1,")")))}))}
for (I in 1:nclus){
  f=D(f,paste0("u",I))
  f=f}
for (J in 1:dim(Y)[1]){
  Y_dot = as.numeric(Y[J,])
  f1 = function(z_k){
    P = rep(0,nclus)
    u = rep(0,nclus)
    denGamma=rep(0,nclus)
    for (I in 1:nclus){
      P[I] = dpois((Y_dot[I]),(m[I])*(z_k[I])*(LBDA[I]),log=FALSE)
      u[I] = pgamma(z_k[I],shape=ETA[I],rate=ETA[I],lower.tail=TRUE,log.p=FALSE)
      denGamma[I] = dgamma(z_k[I],shape=ETA[I],rate=ETA[I],log=FALSE)
      assign(paste0("u",I),u[I])
    }
    P1=prod(P)
    P2=eval(f)
    P3=prod(denGamma)
    P1*P2*P3}
  res=adaptIntegrate(f1,lowerLimit=rep(0,nclus),upperLimit=rep(Inf,nclus))
  ll[J]=log(res$integral)}
sum(ll)}
nclus=2
m=rep(20,2)

####version A####
mle.optim = optim(parms,ll.fn,Y=Yfin,m=m,nclus=nclus,copula="clayton",
  lower=c(0.00001,rep(0.00001,nclus),rep(0.00001,nclus)),
  upper=c(Inf,rep(Inf,nclus),rep(Inf,nclus)),
  method="L-BFGS-B",control=list(fnscale = -1))

estimates = mle.optim$par
hes = hessian(ll.fn,x=estimates,method="Richardson")

####version B####
cl = makeCluster(3)
setDefaultCluster(cl=cl)
clusterEvalQ(cl,library("cubature"))
clusterEvalQ(cl,library("copula"))
optimParallel(par=parms,fn=ll.fn,Y=Y,m=m,K=K,
  method="L-BFGS-B",lower=c(0.00001,rep(0.00001,K),rep(0.00001,K)),
  upper=c(Inf,rep(Inf,K),rep(Inf,K)),control=list(fnscale=-1),maxit=100,
  parallel=list(loginfo=TRUE))

stopCluster(cl)

####version C####
lb = c(rep(0.00001,2),rep(0.00001,2),0.00001)
ub = c(rep(Inf,2),rep(Inf,2),Inf)
opts = list("algorithm"="NLOPT_LN_BOBYQA","xtol_rel"=1.0e-3,"maxeval"=250,print_level=2)
res = nloptr(x0=parms,eval_f=ll.fn,lb=lb,ub=ub,opts=opts,nclus=2,Y=Yfin,m=m,copula='clayton')
estimates = c(p,res$solution, res$status)

```

A.4 R code for MM estimation

```
MMestim = function (Y,nclus,copula,ngenes){
  library(nleqslv)
  LABDA = rep(0,nclus)
  ETA = rep(0,nclus)
  for (J in 1:nclus){
    m1 = sum(Y[,J])/length(Y[,J])
    m2 = sum(Y[,J]^2)/length(Y[,J])
    LABDA[J] = m1/ngenes[J]
    ETA[J] = (m1^2)/(m2-m1-(m1^2))
    tau = cor(Y,method="kendall",use="pairwise")
    meantau = mean(tau[row(tau)==(col(tau)-1)])
    switch(copula,
      clayton = {
        PI = (2*meantau)/(1-meantau),
      },
      gumbel = {
        PI = 1/(1-meantau),
      },
      frank = {
        fn = function(PI){
          InnerFunc = function(t) {t/(exp(t)-1)}
          RES = function(meantau) {
            sapply(PI, function(z) {integrate(InnerFunc,0,z)$value})
          }
          res = numeric(1)
          res[1] = (4*(RES(PI)))+PI*(1-meantau)-4
          res
        }
        PI=nleqslv(c(1),fn)$x)
      }
    )
  }
  ETA[ETA<=0] = 0.001
  ETA[ETA==Inf] = 100
  return(c(ETA,LABDA,PI))
}
```

A.5 R code for the clustering algorithm

```
kmin = kmin
kmax = kmax
maxit = maxit
QIC = rep(0,(kmax-kmin+1))
CH = rep(0,(kmax-kmin+1))
CCC = rep(0,(kmax-kmin+1))
Duda.Stat = rep(0,(kmax-kmin))
Duda.Crit = rep(0,(kmax-kmin))
CL.index = list()
gene.parms = MMestim(Y=Y,nclus=sum(ngenes),copula=copula,ngenes=rep(1,length(ngenes)))
for (T in kmin:kmax){
  nclus = T
  idx.cent = sample(1:nvar,size=1)
  centr = matrix(gene.parms[idx.cent,],ncol=2)
  for (O in 1:nclus) {
    min.dist = rep(0,nvar)
    Euclid.dist = matrix(rep(0,nvar*(length(idx.cent)+1)),
                        ncol=(length(idx.cent)+1),nrow=nvar)
    for (k in 1:nvar){
      ec.dist = rep(0,length(idx.cent))
      a = gene.parms[k,]
      for (L in 1:length(idx.cent)){
        X = rbind(a,centr[L,])
        rownames(X) = c("a","a.x")
        ec.dist[L] = dist(X,method="euclidean")
      }
      min.dist[k] = min(ec.dist^2)
      Euclid.dist[k,] = c(ec.dist,which(ec.dist==min(ec.dist))[1])
    }
    new.idx.cent = which(min.dist==max(min.dist))[1]
    idx.cent = c(idx.cent,new.idx.cent)
    centr = rbind(centr,gene.parms[new.idx.cent,])
  }
  new.idx = Euclid.dist[,nclus+1]
  NEW.IDX = matrix(rep(0,(length(new.idx))*maxit),ncol=maxit,byrow=F)
  PARM.CLU = matrix(rep(0,nclus*2),ncol=2,byrow=T)
  for (iter in 1:maxit){
    NEW.IDX[,iter] = new.idx
    n.clu = length(as.numeric(names(table(new.idx))))
    if(n.clu!=nclus) {break}
    for (D in 1:nclus){

```

```

      assign(paste0("idx.",D),which(new.idx==D))}
Ycl = matrix(c(rep(0,nclus*nsub)),ncol=nclus,byrow=T)
for (j in 1:nsub){
  for (P in 1:nclus){
    Ycl[j,P] = eval(parse(text=(paste0("sum(Y[j,idx.",P,"]))")))}
numg = as.vector(table(new.idx))
parms.clu = matrix(MMestim(Y=Ycl,nclus=nclus,copula=copula,ngenes=numg)[1:(nclus*2)],
  nrow=nclus,ncol=2,byrow=F)
colnames(parms.clu) = c("Alpha","Lambda")
Euclid.dist = matrix(rep(0,nvar*(nclus+1)),ncol=(nclus+1),nrow=nvar)
for (k in 1:nvar){
  ec.dist = rep(0,nclus)
  a = gene.parms[k,]
  for (L in 1:nclus){
    X = rbind(a,parms.clu[L,])
    rownames(X) = c("a","a.x")
    ec.dist[L] = dist(X,method="euclidean")}
  Euclid.dist[k,] = c(ec.dist, which(ec.dist==min(ec.dist))[1])
  new.idx = Euclid.dist[,ncol(Euclid.dist)]
  C.DIST = rep(0,nclus)
  for (G in 1:nclus){
    CenD = rbind(PARM.CLU[G,],parms.clu[G,])
    rownames(CenD) = c("C1","C2")
    C.DIST[G] = dist(CenD,method="euclidean")}
  if((sum(C.DIST)<rel.c){break}
  PARM.CLU = parms.clu}
cols = unique(which(NEW.IDX == 0,arr.ind=TRUE)[,2])
if(length(cols)>0){NEW.IDX1 = as.matrix(NEW.IDX[,-cols])
if(length(as.numeric(names(table(NEW.IDX1[,ncol(NEW.IDX1)])))!=nclus){
new.idx = NEW.IDX1[,ncol(NEW.IDX1)-1]} else{
new.idx = NEW.IDX1[,ncol(NEW.IDX1)]}
CL.index = append(CL.index,list(new.idx))
for (D in 1:nclus){
  assign(paste0("idx.",D),which(new.idx==D))}
Yfcl = matrix(c(rep(0,nclus*nsub)),ncol=nclus,byrow=T)
for (j in 1:nsub){
  for (P in 1:nclus){
    Yfcl[j,P] = eval(parse(text=(paste0("sum(Y[j,idx.",P,"]))")))}
numgenes = as.vector(table(new.idx))
estimates = MMestim(Y=Yfcl,nclus=nclus,copula=copula,ngenes=numgenes)[1:(nclus*2)]
parms.clu = matrix(estimates,nrow=nclus,ncol=2,byrow=FALSE)
nCL=nclus
alpha=estimates[1:nCL]
lambda = estimates[(nCL+1):(2*nCL)]
QIC[T-kmin+1] <- Index.QIC(Y=Yfcl,lambda=lambda,alpha=alpha,numgenes=numgenes,nCL=nCL)
TT = t(t(Y))%*%t(Y)
nn = dim(t(Y))[1]
sizeEigenTT = length(eigen(TT)$value)
eigenValues = eigen(TT/(nn-1))$value
s1 = sqrt(eigenValues)
ss = rep(1,sizeEigenTT)
for (i in 1:sizeEigenTT){
  if (s1[i]!=0) ss[i]=s1[i]}
vvfin = prod(ss/10)
CCC[T-kmin+1] = Index.CCC(x=t(Y),cl=new.idx,P=TT,s=s1,vv=vvfin)
CH[T-kmin+1] = index.G1(x=t(Y),cl=new.idx,d=NULL,centrotypes="centroids")
if (length(CL.index)>1){
  res.Duda = Index.Duda(x=t(Y),cl1=CL.index[[T-kmin]],cl2=CL.index[[T-kmin+1]])
  Duda.Stat[T-kmin] = res.Duda$duda
  Duda.Crit[T-kmin] = res.Duda$crit.val}
}

GAP = Index.GAP(t(Yfin),FUN=CL.index,K.max=(kmax-kmin+1),B=50,d.power=2,spaceH0="scaledPCA")

```

A.6 R code for the model selection

```

Index.QIC <- function (Y, lambda, alpha, numgenes, nCL){
  q1 = rep(0, dim(Y)[1])
  for (J in 1:dim(Y)[1]){
    Y_dot = as.numeric(Y[J,])
    res.int = rep(0, nCL)
    for (I in 1:nCL){
      res.int[I] = Y_dot[I]*
        log((lambda[I]*numgenes[I])/
          ((alpha[I]*lambda[I]*numgenes[I]+1)))+
        (1/alpha[I])*log((1/alpha[I])/
          ((1/alpha[I])+(lambda[I]*numgenes[I])))}
    q1[J]=sum(res.int)
  }
  return (-2*sum(q1)+2*nCL)
}

Index.CCC <- function(x, cl, P, s, vv){
  n = dim(x)[1]
  pp = dim(x)[2]
  qq = max(cl)
  z = matrix(0, ncol=qq, nrow=n)
  clX = as.matrix(cl)
  for (i in 1:n){
    for (j in 1:qq){
      z[i, j]==0
      if (clX[i, 1]==j)
        {z[i, j]=1}
    }
    xbar = solve(t(z)%*%z)%*%t(z)%*%x
    B = t(xbar)%*%t(z)%*%z)%*%xbar
    W = P-B
    R2 = 1-sum(diag(W))/sum(diag(P))
    v1 = 1
    u = rep(0, pp)
    c = (vv/(qq))^(1/pp)
    u = s/c
    k1 = sum((u>=1)==TRUE)
    p1 = min(k1, qq-1)
    if (all(p1>0, p1<pp)){
      for (i in 1:p1)
        v1 <- v1*s[i]
    }
    c = (v1/(qq))^(1/p1)
    u = s/c
    b1 = sum(1/(n+u[1:p1]))
    b2 = sum(u[p1+1:pp]^2/(n+u[p1+1:pp]), na.rm=TRUE)
    E_R2 = 1-((b1+b2)/sum(u^2))*((n-qq)^2/n)*(1+4/n)
    ccc = log((1-E_R2)/(1-R2))*(sqrt(n*p1/2)/((0.001+E_R2)^1.2)) else {
      b1 = sum(1/(n+u))
      E_R2 = 1-(b1/sum(u^2))*((n-qq)^2/n)*(1+4/n)
      ccc = log((1-E_R2)/(1-R2))*(sqrt(n*pp/2)/((0.001+E_R2)^1.2))
    }
  }
  return(ccc)
}

Index.Duda = function (x, cl1=cl1, cl2=cl2) {
  dim2 = dim(x)[2]
  wss = function(x){
    x = as.matrix(x)
    n = length(x)
    centers = matrix(nrow=1, ncol=ncol(x))
    if (ncol(x)==1) centers[1,] = mean(x)
    if (is.null(dim(x))){
      bb = matrix(x, byrow=FALSE, nrow=1, ncol=ncol(x))
      centers[1,] = apply(bb, 2, mean)} else {
      centers[1,] = apply(x, 2, mean)}
    x.2 = sweep(x, 2, centers[1,], "-")
    within = sum(x.2^2)
    wss = sum(within)
    return(wss)
  }
  ncg1 = 1
  ncg1max = max(cl1)
  while((sum(cl1==ncg1)==sum(cl2==ncg1)) && ncg1<=ncg1max) {
    ncg1 = ncg1+1}
}

```



```

g1 = ncg1
ncg2 = max(c12)
nc2g2 = ncg2-1
while ((sum(c11==nc2g2)==sum(c12==ncg2)) && nc2g2>=1) {
  ncg2 = ncg2-1
  nc2g2 = nc2g2-1}
g2 = ncg2
NK = sum(c12==g1)
WK.x = x[c12==g1,]
WK = wss(x=WK.x)
NL = sum(c12==g2)
WL.x = x[c12==g2,]
WL = wss(x=WL.x)
NM = sum(c11==g1)
WM.x = x[c11==g1,]
WM = wss(x=WM.x)
duda = (WK+WL)/WM
zz = 3.20
zzz = zz*sqrt(2*(1-8/((pi^2)*pp))/(NM*pp))
resCritical = 1-(2/(pi*pp))-zzz
return (list(duda=duda,crit.val=resCritical))
}

Index.GAP = function(x,FUNcluster,K.max,B,d.power,spaceH0=c("scaledPCA","original")){
  stopifnot(length(dim(x))==2,K.max>=2,(n<-nrow(x))>=1,ncol(x)>= 1)
  if(B!=(B.=as.integer(B)) || (B=B.)<=0)
    stop("'B' has to be a positive integer")
  cl.=match.call()
  if(is.data.frame(x))
    x=as.matrix(x)
  ii=seq_len(n)
  W.k=function(X,kk){
    clus=if(kk>1)
      FUNcluster[[kk]] else rep.int(1L,nrow(X))
    0.5*sum(vapply(split(ii,clus),function(l){xs=X[l,,drop=FALSE]
      sum(dist(xs)^d.power/nrow(xs))},0.))}
  logW=E.logW=SE.sim=numeric(K.max)
  if(verbose) cat("Clustering k = 1,2,..., K.max (= ",K.max,"): .. ",sep='')
  for(k in 1:K.max) logW[k] = log(W.k(x,k))
  if(verbose) cat("done\n")
  spaceH0 = match.arg(spaceH0)
  xs = scale(x,center=TRUE,scale=FALSE)
  m.x = rep(attr(xs,"scaled:center"),each=n)
  switch(spaceH0,
    "scaledPCA" = {
      V.sx = svd(xs,nu=0)$v
      xs = xs%%V.sx
    },
    "original" = {},
    stop("invalid 'spaceH0':",spaceH0))
  rng.x1 = apply(xs,2L,range)
  logWks = matrix(0,B,K.max)
  if(verbose) cat("Bootstrapping, b = 1,2,...,B(=",B,") [one \" per sample]:\n",sep="")
  for(b in 1:B) {
    z1 = apply(rng.x1,2,function(M,nn) runif(nn,min=M[1],max=M[2]),nn=n)
    z = switch(spaceH0,"scaledPCA"=tcrossprod(z1,V.sx),"original"=z1)+m.x
    for(k in 1:K.max) {
      logWks[b,k] = log(W.k(z,k))}
    if(verbose) cat(".",if(b%%50==0) paste(b,"\n"))}
  if(verbose && (B%%50!=0)) cat("'",B,"\n")
  E.logW = colMeans(logWks)
  SE.sim = sqrt((1+1/B)*apply(logWks,2,var))
  structure(class="clusGap",list(Tab=cbind(logW,E.logW,gap=E.logW-logW,SE.sim),
    call=cl.,spaceH0=spaceH0,n=n,B=B,FUNcluster=FUNcluster))}

maxSE = function(f,SE.f,method=c("firstSEmax","Tibs2001SEmax","globalSEmax",
  "firstmax","globalmax"),SE.factor=1){
  method = match.arg(method)
  stopifnot((K=length(f))>=1,K=length(SE.f),SE.f>=0,SE.factor>=0)
  fSE = SE.factor*SE.f
  switch(method,
    "firstmax" = {
      decr = diff(f) = 0
      if(any(decr)) which.max(decr) else K

```

```

    },
    "globalmax" = {
        which.max(f)
    },
    "Tibs2001SEmax" = {
        g.s = f-fSE
        if (any(mp = f[-K]>=g.s[-1])) which.max(mp) else K
    },
    "firstSEmax" = {
        decr = diff(f) = 0
        nc = if (any(decr)) which.max(decr) else K
        if (any(mp = f[seq_len(nc-1)]>=f[nc]-fSE[nc]))
            which(mp)[1] else nc
    },
    "globalSEmax" = {
        nc = which.max(f)
        if (any(mp = f[seq_len(nc - 1)]>=f[nc]-fSE[nc]))
            which(mp)[1]
        else nc
    }) }

```

A.7 ANOVA table

Factor	F value		
	Rel. Bias	CoV	Coverage
<i>H</i>	0.152	0.082	0.82
<i>Parm</i>	337.87**	342.66**	5.60**
<i>N</i>	418.99**	428.40**	4.36**
<i>P</i>	3.33	3.12	0.002
<i>Split</i>	1.43	0.56	0.024
<i>H</i> \times <i>Parm</i>	4.59**	4.47**	0.085
<i>H</i> \times <i>N</i>	2.66	2.79	0.835
<i>H</i> \times <i>P</i>	1.24	1.22	4.85**
<i>H</i> \times <i>Split</i>	3.80**	3.83**	2.37
<i>Parm</i> \times <i>N</i>	169.87**	174.89**	9.67**
<i>Parm</i> \times <i>P</i>	1.02	0.93	0.073
<i>Parm</i> \times <i>Split</i>	0.50	0.55	2.22
<i>N</i> \times <i>P</i>	0.12	0.25	0.77
<i>N</i> \times <i>Split</i>	7.18**	8.81**	0.11
<i>P</i> \times <i>Split</i>	0.54	0.96	2.43

Table 2: ANOVA fixed effects analysis for the 3 performance measures relative bias ("Rel Bias"), coefficient of variation ("CoV") and coverage. Factor coding: *H*=number of clusters, *Parm*=type of parameter, *N*=sample size, *P*=number of variables, *Split*=variable splitting (homogeneous/heterogeneous). The product (\times) indicates a first order interaction. ** = significant effect (<0.05).