|  |
| --- |
| rm(list=ls()) |
|  | ############################### |
|  | ##### Data Simulation ##### |
|  | ############################### |
|  | DINA.SIM = function (Q, N, J, K, R){ |
|  | m = matrix(R, K, K) |
|  | diag(m) = 1 |
|  | ch = chol(m) |
|  | u = matrix(rnorm(N\*K), ncol=K) |
|  | uc = u %\*% ch |
|  | cr = pnorm(uc) |
|  | alpha = matrix(0, N, K) |
|  | for (i in 1:K) alpha[,i] = ifelse(cr[,i]>i/(K+1), 1, 0) |
|  | tm = alpha%\*%t(Q) |
|  | at = matrix(rep(apply(Q, 1, sum), N), nrow=N, byrow=T) |
|  | eta = ifelse(tm==at, 1, 0) |
|  | y = ifelse(eta==1, 0.8, 0.2) |
|  | comp = c(runif(N\*J, 0, 1)) |
|  | y = ifelse(y>comp, 1, 0) |
|  | out = list(cr=cr, alpha=alpha, y=y) |
|  | out |
|  | } |
|  |  |
|  | ############################ |
|  | ##### Simulation I ##### |
|  | ############################ |
|  | q = matrix(c(1,0,0,0, |
|  | 0,1,0,0, |
|  | 0,0,1,0, |
|  | 0,0,0,1, |
|  | 1,1,0,0, |
|  | 1,0,1,0, |
|  | 1,0,0,1, |
|  | 0,1,1,0, |
|  | 0,1,0,1, |
|  | 0,0,1,1, |
|  | 1,1,1,0, |
|  | 1,1,0,1, |
|  | 1,0,1,1, |
|  | 0,1,1,1, |
|  | 1,1,1,1), byrow=TRUE, nrow=15, ncol=4) |
|  | out = DINA.SIM(Q=q, N=1000, J=nrow(q), K=ncol(q), R=0.3) |
|  | yy = out$y |
|  |  |
|  | ########################## |
|  | ##### Estimation ##### |
|  | ########################## |
|  | as.binary = function(x){ |
|  | ans = NULL |
|  | while(any(x!=0)){ |
|  | ans = cbind(x%%2,ans) |
|  | x = floor(x/2) |
|  | } |
|  | ans |
|  | } |
|  |  |
|  | EstQMCMC = function(Y, K=NULL,q.start = NULL, g.start=NULL, s.start=NULL,pi.start=NULL, a.start, niter){ |
|  |  |
|  | N = nrow(Y) |
|  | J = ncol(Y) |
|  | Y = as.matrix(Y) |
|  |  |
|  | if(is.null(K) & is.null(q.start)) |
|  | stop('User must supply either the number of attributes or a starting Q matrix!\n\n') |
|  | if(is.null(q.start)){ |
|  | Q = matrix(rbinom(J\*K, 1, 0.5), J, K) |
|  | Q[which(apply(Q, 1, sum)==0), 1] = 1 |
|  | } |
|  | else{ |
|  | Q = q.start |
|  | } |
|  | if(is.null(g.start)) |
|  | g = runif(J, 0.1, 0.3) |
|  | else |
|  | g = g.start |
|  | if(is.null(s.start)) |
|  | s = runif(J, 0.1, 0.3) |
|  | else |
|  | s = s.start |
|  | if(is.null(pi.start)){ |
|  | pi = exp(rnorm(2^K)) |
|  | pi = pi/sum(pi) |
|  | } |
|  | else |
|  | pi = pi.start |
|  |  |
|  | all.a = as.binary(0:(2^K-1)) |
|  | natt = apply(Q,1,sum) |
|  | Yt = t(Y) |
|  | a = a.start |
|  | p = ifelse(Q==1, 0.6, 0.4) |
|  | pi.out = pi; Q.out = Q; p.out = p; g.out = g; s.out = s |
|  |  |
|  | for(ii in 1:niter){ |
|  | etajm = tcrossprod(Q,all.a) |
|  | natt = apply(Q,1,sum) |
|  | etajm = (etajm == natt) |
|  | pp = g\*(1-etajm) + (1-s)\*etajm |
|  | ll = Y %\*% log(pp) + (1-Y)%\*%log(1-pp) |
|  | ll = sweep(ll,2,log(pi),'+') |
|  | pp = exp(ll) |
|  | pp = apply(pp,1,cumsum) |
|  | pp = sweep(pp,2,pp[2^K,],'/') |
|  | u = runif(N) |
|  | alpha = apply(sweep(pp,2,u,'<'),2,sum) |
|  | alpha = as.binary(c(2^K-1,alpha))[-1,] |
|  |  |
|  | cc = as.vector(alpha%\*%(2^((K-1):0))) |
|  | cc = apply(outer(cc,0:(2^K-1),'=='),2,sum) |
|  | pi = rgamma(2^K, 1+cc) |
|  | pi = pi/sum(pi) |
|  | pi.out = rbind(pi.out,pi) |
|  |  |
|  | etaim = tcrossprod(Q,alpha) |
|  | etaim = (etaim == natt) |
|  | ga = apply((1-etaim)\*Yt,1,sum) |
|  | gb = apply((1-etaim)\*(1-Yt),1,sum) |
|  | sa = apply(etaim\*(1-Yt),1,sum) |
|  | sb = apply(etaim\*Yt,1,sum) |
|  | g = qbeta(runif(J, 0,pbeta(1-s,1+ga,1+gb)),1+ga,1+gb) |
|  | s = qbeta(runif(J,0,pbeta(1-g,1+sa,1+sb)),1+sa,1+sb) |
|  | g.out = rbind(g.out,g) |
|  | s.out = rbind(s.out,s) |
|  |  |
|  | Q = t(sapply(1:J,function(j){sample.Q(all.a[-1,],g[j],s[j],Y[,j],alpha,p[j,])})) |
|  | Q.out = rbind(Q.out,Q) |
|  |  |
|  | pa = a + Q; pb = a +1-Q |
|  | ppp = rbeta(J\*K, pa, pb) |
|  | p = matrix(ppp, J, K) |
|  | p.out = cbind(p.out, p) |
|  | } |
|  | p.out = array(p.out, c(J,K,niter)) |
|  | Q.out = array(Q.out, c(J,niter,K)) |
|  | out = list(pi=pi.out, Q=Q.out, g=g.out, s=s.out, p=p.out) |
|  | class(out) = 'cdmcmc' |
|  | out |
|  | } |
|  |  |
|  | sample.Q = function(all.a, g, s, Y, alpha,pp){ |
|  | natt = apply(all.a,1,sum) |
|  | cc = tcrossprod(all.a,alpha) |
|  | etaim = (cc==natt) |
|  | pp[pp<1e-8] = 1e-8 |
|  | pp[pp>1-1e-8] = 1-1e-8 |
|  | pp = all.a%\*%log(pp) + (1-all.a)%\*%log(1-pp) |
|  | ga = (1-etaim)%\*%Y |
|  | gb = (1-etaim)%\*%(1-Y) |
|  | sa = etaim%\*%(1-Y) |
|  | sb = etaim%\*%Y |
|  | pm = ga\*log(g) + gb\*log(1-g) + sa\*log(s) + sb\*log(1-s) |
|  | pm = pm + pp |
|  | pm = pm - max(pm) |
|  | pm = as.vector(exp(pm)) |
|  | pm = pm/sum(pm) |
|  | kk = nrow(all.a) |
|  | q = sample(1:kk,size=1,prob=pm) |
|  | q = (as.binary(c(q,kk))[1,]) |
|  | q |
|  | } |
|  |  |
|  | system.time(out <- EstQMCMC(yy, K=4, a.start=1, niter=100000)) |
|  |  |
|  | niter = 100000 |
|  | bn = 50000 |
|  | nmb = niter - bn |
|  | qest = apply(out$Q[,-(1:bn),], c(1,3), mean) |
|  | qqq = ifelse(q==1, 0.66, 0.33) |
|  | permutations = function(n){ |
|  | if(n==1){ |
|  | return(matrix(1)) |
|  | } else { |
|  | sp = permutations(n-1) |
|  | p = nrow(sp) |
|  | A = matrix(nrow=n\*p,ncol=n) |
|  | for(i in 1:n){ |
|  | A[(i-1)\*p+1:p,] = cbind(i,sp+(sp>=i)) |
|  | } |
|  | return(A) |
|  | } |
|  | } |
|  |  |
|  | reorder <- function(J, K, a, b){ |
|  | vec.a = matrix(a, ncol=1) |
|  | vec.b = matrix(b, ncol=1) |
|  | pm = permutations(K) |
|  | tpm = t(pm) |
|  | vec.b.matrix = matrix(as.vector(b[,c(tpm[,1:factorial(K)])]), J\*K, factorial(K)) |
|  | vec.bind.ab = as.matrix(cbind(vec.a, vec.b.matrix)) |
|  | dist.matrix = as.matrix(dist(t(vec.bind.ab), method="euclidean")) |
|  | ds = dist.matrix[,1] |
|  | min.value = (min(ds[ds>0])) |
|  | matrix.number = as.numeric(which(ds == min.value)) |
|  | reorder.b = matrix(vec.b.matrix[, matrix.number-1], J, K) |
|  | output = list(reorder.b=reorder.b) |
|  | output$reorder.b |
|  | } |
|  |  |
|  | rqest = reorder(J=nrow(q), K=ncol(q), a=qqq, b=qest) |
|  | J = nrow(q) |
|  | K = ncol(q) |
|  | diff1 = rqest-q |
|  | delta1 = 1-(sum(abs(diff1))/(J\*K)) |
|  |  |
|  | QA = array(dim=c(J, K, nmb)) |
|  | QB = out$Q[,-(1:bn),] |
|  |  |
|  | for (rr in 1:nmb){ |
|  | QA[,,rr]=reorder(J, K, qest, QB[,rr,]) |
|  | } |
|  | Qest = apply(QA[,,(1:nmb)], c(1,2), mean) |
|  |  |
|  | Rqest = reorder(J, K, a=qqq, b=Qest) |
|  | diff2 = Rqest - q |
|  | delta2 = 1-(sum(abs(diff2))/(J\*K)) |
|  |  |
|  | for(mmm in 1:1000){ |
|  | for (rr in 1:nmb){ |
|  | QA[,,rr]=reorder(J, K, Qest, QA[,,rr]) |
|  | } |
|  | RQest = apply(QA[,,(1:nmb)], c(1,2), mean) |
|  | dif = (sum(abs(RQest-Qest)))/(J\*K) |
|  | if (dif < 0.0001){ |
|  | RQest = reorder(J, K, a=qqq, b=RQest) |
|  | diff3 = RQest - q |
|  | delta3 = 1-(sum(abs(diff3))/(J\*K)) |
|  | cat("delta (original) = ", delta1, "\n") |
|  | cat("delta (final) =", delta3, "\n") |
|  | break |
|  | } |
|  | Qest = RQest |
|  | } |
|  |  |
|  |  |