```
# Bayesiam MDS (Oh & Raftery, 2001, JASA) code
# with slight modification from original paper to solve identification problems.
# Written by Kensuke Okada, 2008.

# CPU time check
now <- proc.time()[1:2]

# This function is from S-plus manual
dist2full <- function(dis)
{
    n <- attr(dis, "Size")
    full <- matrix(0, n, n)
    full[lower.tri(full)] <- dis
    full + t(full)
}

# Function for computing raw stress
compssr <- function(x,delta){
    dd <- dist2full(dist(x))
    ssr1 <- (dd-delta)^2
    ssr1[t(lower.tri(ssr1, diag=F))]<-0
    sum(ssr1)
}

# Main program
set.seed(5353)

# Read distance data from a file
# Here, we analyze Ekman(1954)'s color similarity data
deltasim <- read.table("c:/usr/ken/MDS/Ekman.dat", header=F)
deltasim <- as.matrix(deltasim)
n<-nrow(deltasim)

# Transform similarities into dissimilarities
ones <- matrix(1,n,n)
delta <- ones - deltasim
diag(delta) <- rep(0,n)

# Number of iterations
nwarm<-0
maxn<-1000
```

```
scale<-2.38^2
m<-n*(n-1)/2

deltasq <- delta^2
deltasq[t(lower.tri(deltasq, diag=F))]<-0
sdsq <- sum(deltasq)

# Number of dimensions
p <- 3
rIp <- diag(rep(1,p))

# We use the cmds solution as an initial value
tmpx <- cmdscale(delta,k=p)
x <- tmpx

# Rotation in order to solve indeterminancy
xbar <- t(apply(x,2,mean)) # Mean about objects
xbarmat <-  matrix(1,n,1) %*% xbar
x <- x - xbarmat
sigx <- var(x,unbiased=F) # covariance matrix devided by n

eigx <- eigen(sigx)
eval <- diag(eigx$values) # Eigenvalues
rotat <- t(eigx$vectors)

x <- x %*% rotat # Written in matrix form. Different from Oh & Raftery.
sigx <- rotat %*% sigx %*% t(rotat)

xint <- x # This is real "initial X"

# Output the initial settings
dd <- dist2full(dist(x))

sres1 <- (dd-delta)^2
sres1[t(lower.tri(sres1, diag=F))]<-0
sres <- sum(sres1)

sigma <- sres / m
stress <- sqrt(sres/sdsq)

txpre <- "initial sigma: "
```

```
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=F)
write(sigma, file="BMDSplus_Ekman3D_initstat.txt",append=T)

txpre <- "initial stress: "
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=T)
write(stress, file="BMDSplus_Ekman3D_initstat.txt",append=T)

txpre <- "sdsq: "
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=T)
write(sdsq, file="BMDSplus_Ekman3D_initstat.txt",append=T)

txpre <- "sres: "
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=T)
write(sres, file="BMDSplus_Ekman3D_initstat.txt",append=T)

lambda <- 1/diag(sigx)

# Set hyperpriors
pralphasig<- 5
prbetasig <- (pralphasig-1)*sigma
alphalam<-0.5
betalam<-0.5*diag(sigx)

txpre <- "nwarm: "
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=T)
write(nwarm, file="BMDSplus_Ekman3D_initstat.txt",append=T)

txpre <- "maxn: "
write(txpre, file="BMDSplus_Ekman3D_initstat.txt",append=T)
write(maxn, file="BMDSplus_Ekman3D_initstat.txt",append=T)

ssigma<-0
sqsigma<-0
rminssr<-stress^2*sdsq

# Start iteration
for (niter in 1:maxn){

# Generate X
    # loop about objects
    for (ii in 1:n){
```

```
    cdsig <- sqrt(scale*sigma/(n-1)) * rIp # it's OK with "*" here
    xold <- t(x[ii,])
    cdm <- xold
    z <- rnorm(p,mean=0,sd=1)

    xnew <- t(cdsig %*% t(t(z)) + t(cdm)) # confusing t() …

    quad <- sum(xold^2*lambda)
    quadst <- sum(xnew^2*lambda)

    t1 <- 0
    t1st <- 0
    stemp <- 0

    for (jj in 1:n){
        if (jj != ii){
            delij <- sqrt(sum((xold-t(x[jj,]))^2))
            delstij <- sqrt(sum((xnew-t(x[jj,]))^2))

            temp <- log( pnorm(delstij/sqrt(sigma)) / pnorm(delij/sqrt(sigma)) )
            t1 <- t1 - 0.5 / sigma * (delij - delta[ii,jj])^2
            t1st <- t1st - 0.5 / sigma * (delstij - delta[ii,jj])^2
            stemp <- stemp – temp
        }
    }

   rlgw <- 0
   flfw <- t1st - t1 - stemp - 0.5 *(quadst-quad)

   ranunif <- runif(1)
   if (log(ranunif) < (flfw-rlgw)){
       xold <- xnew
   }

   x[ii,] <- t(xold)

}

# Centralization and then rotate X using procrustean technique
# (see Oh & Raftery, 2007; Hoff, Raftery & Handcock, 2002)
```

```
xbar <- t(apply(x,2,mean))
xbarmat <- matrix(1,n,p)*diag(xbar)
x <- x - xbarmat

sigx <- var(x,unbiased=F) # Covariance matrix divided by n

eigx <- eigen(sigx)
eval <- diag(eigx$values)
rotat <- t(eigx$vectors)

x <- x %*% rotat

sigx <- rotat %*% sigx %*% t(rotat)

Jmat <- diag(1,n) - (1/n)  * ( matrix(1,n,1) %*% t(matrix(1,n,1)))
Cmat <- t(xint) %*% Jmat %*% x
svdres <- svd(Cmat)

Pmat <- svdres$u
Dmat <- diag(svdres$d)
Qmat <- t(svdres$v)

Tmat <- Qmat %*% t(Pmat)
tvec <- (1/n) * t(xint - x %*% Tmat) %*% matrix(1,n,1)
x <- x %*% Tmat + matrix(1,n,1)%*%t(tvec)

# Generate sigma (which is phi^2 in our notation)
sres <- compssr(x,delta)
sigold <- sigma
cdvarsig <- scale * 2 * sres^2 / ((m-2)^2*(m-4))

rannor <- rnorm(1)
signew <- -1
while (signew < 0){
    rannor <- rnorm(1)
    signew <- rannor * sqrt(cdvarsig) + sigold
}

dd <- dist2full(dist(x)) # dd is represented by delta in Oh & Raftery (2001)

ddvec1 <- as.vector(dd)
```

```
nonzerosvec <- ddvec1 != 0
ddvec <- ddvec1[nonzerosvec]

flf <- sum( -log ( pnorm( ddvec / sqrt(signew)) / pnorm ( ddvec / sqrt(sigold))))
flf <- flf-(0.5*sres+prbetasig)*(1/signew-1/sigold)
    -(0.5*m+pralphasig+1)*log(signew/sigold)
rlg <- log( pnorm( signew / sqrt(cdvarsig) ) / pnorm( sigold / sqrt(cdvarsig) ) )
rlw <- flf-rlg

ranunif <- runif(1)
if (log(ranunif) < rlw){
    sigold <- signew
}
sigma <- sigold

# Generate lambda
s1 <-  diag(var(x))

pstbetalam <- 0.5 * s1 +betalam
pstalphalam <- 0.5 * n + alphalam
rangam <- rgamma(p,alphalam)
lambda <- 1 / pstbetalam

ssigma <- ssigma+sigma
sqsigma<-sqsigma+sigma^2

# Compute stress and end iteration
ssr <- compssr(x,delta)

if (ssr < rminssr){
    rminssr <- ssr
    indssr <- niter
}

if (niter %% 10 == 0){
    print(niter)
}

# Write each MCMC sample into text file
if (niter == 1){
    write.table(t(as.vector(x)), "c:/usr/ken/MDS/BMDSplus_Ekman3D_X1.txt",append=F)
```

```
        write(sigma, "c:/usr/ken/MDS/BMDSplus_Ekman3D_sigma1.txt",append=F)
        write(lambda, "c:/usr/ken/MDS/BMDSplus_Ekman3D_lambda1.txt",append=F)
    } else {
        write.table(t(as.vector(x)), "c:/usr/ken/MDS/BMDSplus_Ekman3D_X1.txt",append=T)
        write(sigma, "c:/usr/ken/MDS/BMDSplus_Ekman3D_sigma1.txt",append=T)
        write(lambda, "c:/usr/ken/MDS/BMDSplus_Ekman3D_lambda1.txt",append=T)
    }

# Iteration ends
}

Stress <- sqrt(rminssr/sdsq)
Esigma <- ssigma/maxn
Varsigma <- sqsigma/maxn-esigma^2

# Check CPU time
speed <- proc.time()[1:2] - now
print(speed)

# Summary statistics
postx <- read.table('c:/usr/ken/MDS/BMDSplus_Ekman3D_X1.txt', sep=",");
postx <- as.matrix(postx)
meanpostxvec <- apply(postx[301:1000,],2,mean)
matpostx <- matrix(meanpostxvec,nrow=n,ncol=p)

# Simple 2D plot
plot(matpostx[,1],matpostx[,2])
```