**Assn 3 KEY – R history / trace**

> ##################################

> # HUDM 5124 - Homework #3 KEY #

> # R code to run #

> # Torgerson's Metric MDS #

> ##################################

>

> # SELECT SUBSET OF CONFUSION DATA MATRIX

> n <- 5

> S <- matrix(c(97.00, 4.00, 4.00, 7.00, 2.00,

+ 9.00, 87.00, 8.00, 37.00, 9.00,

+ 8.00, 16.00, 93.00, 12.00, 12.00,

+ 11.00, 59.00, 17.00, 96.00, 12.00,

+ 9.00, 15.00, 26.00, 12.00, 86.00),

+ nrow= n, byrow=TRUE, dimnames = list(c("E", "H", "N", "S", "W"), c("E", "H", "N", "S", "W")))

> S

E H N S W

E 97 4 4 7 2

H 9 87 8 37 9

N 8 16 93 12 12

S 11 59 17 96 12

W 9 15 26 12 86

>

> # symmetrize raw confusion data by averaging off-diagonal entries

> SS <- (1/2) \* (t(S) + S)

> SS

E H N S W

E 97.0 6.5 6.0 9.0 5.5

H 6.5 87.0 12.0 48.0 12.0

N 6.0 12.0 93.0 14.5 19.0

S 9.0 48.0 14.5 96.0 12.0

W 5.5 12.0 19.0 12.0 86.0

>

> # set diagonal entries of SS to 0

> diag(SS) <- rep(0, n)

> SS

E H N S W

E 0.0 6.5 6.0 9.0 5.5

H 6.5 0.0 12.0 48.0 12.0

N 6.0 12.0 0.0 14.5 19.0

S 9.0 48.0 14.5 0.0 12.0

W 5.5 12.0 19.0 12.0 0.0

>

> # transform symmetrized confusion data into dissimilarities (by subtracting each element from the largest)

> MaxC <- max(SS)

> MaxC

[1] 48

> D <- matrix(rep(MaxC, n^2), nrow= n) - SS

> diag(D) <- rep(0, n)

> D

E H N S W

E 0.0 41.5 42.0 39.0 42.5

H 41.5 0.0 36.0 0.0 36.0

N 42.0 36.0 0.0 33.5 29.0

S 39.0 0.0 33.5 0.0 36.0

W 42.5 36.0 29.0 36.0 0.0

>

> # check that the Triangle Inequality holds; fix data if it doesn't

> maxC <- 0; nTriples <- 0;

> for(x in 3:n) {

+ for(y in 2:(x-1)) {

+ for(z in 1:(y-1)) {

+ nTriples <- nTriples + 1

+ c <- D[x,z] - D[x,y] - D[y,z]

+ if(c >= maxC) maxC <- c

+ c <- D[x,y] - D[x,z] - D[z,y]

+ if(c >= maxC) maxC <- c

+ c <- D[y,z] - D[y,x] - D[x,z]

+ if(c >= maxC) maxC <- c

+ }

+ }

+ }

> maxC

[1] 2.5

> nTriples

[1] 10

>

> # add the additive constant to all entries

> D <- D + matrix(rep(maxC, n^2), nrow= n)

# comment -- or just use: D <- D + c

> diag(D) <- rep(0, n) # reset diagonal to 0’s

> D

E H N S W

E 0.0 44.0 44.5 41.5 45.0

H 44.0 0.0 38.5 2.5 38.5

N 44.5 38.5 0.0 36.0 31.5

S 41.5 2.5 36.0 0.0 38.5

W 45.0 38.5 31.5 38.5 0.0

>

>

> # square each element of D, put into a new matrix D.Sq

> D.Sq <- matrix(rep(NA, 25), nrow= n, byrow=TRUE,

+ dimnames = list(c("E", "H", "N", "S", "W"), c("E", "H", "N", "S", "W")))

> D.Sq <- D^2

> D.Sq

E H N S W

E 0.00 1936.00 1980.25 1722.25 2025.00

H 1936.00 0.00 1482.25 6.25 1482.25

N 1980.25 1482.25 0.00 1296.00 992.25

S 1722.25 6.25 1296.00 0.00 1482.25

W 2025.00 1482.25 992.25 1482.25 0.00

>

> # Create matrix B by double-centering D.Sq

> B <- matrix(rep(NA, 25), nrow= n, byrow=TRUE,

+ dimnames = list(c("E", "H", "N", "S", "W"), c("E", "H", "N", "S", "W")))

> for(i in 1:n) {

+ for(j in 1:n) {

+ B[i,j] <- (-.5)\*(D.Sq[i,j] - mean(D.Sq[i,]) - mean(D.Sq[,j]) + mean(D.Sq))

+ }

+ }

> B

E H N S W

E 956.510 -287.165 -224.890 -220.290 -224.165

H -287.165 405.160 -251.565 362.035 -228.465

N -224.890 -251.565 573.960 -198.440 100.935

S -220.290 362.035 -198.440 325.160 -268.465

W -224.165 -228.465 100.935 -268.465 620.160

>

> # perform a PCA – “eigen” by default treats data as covariances (i.e., uses diagonal entries)

> eigendecomp <- eigen(B)

> eigendecomp

$values

[1] 1.227466e+03 1.158874e+03 4.991536e+02 2.273737e-12 -4.544095e+00

$vectors

[,1] [,2] [,3] [,4] [,5]

[1,] 0.6741496 0.5865000 0.007564536 -0.4472136 -0.03850789

[2,] -0.5306785 0.2232794 0.087270933 -0.4472136 -0.67890385

[3,] 0.1398122 -0.4970385 -0.726828529 -0.4472136 -0.07159301

[4,] -0.4611964 0.2374654 -0.047554867 -0.4472136 0.72708086

[5,] 0.1779132 -0.5502063 0.679547928 -0.4472136 0.06192390

> sqrtLambda <- as.matrix(diag(sqrt(eigendecomp$values)))

Warning message:

In sqrt(eigendecomp$values) : NaNs produced

> sqrtLambda

[,1] [,2] [,3] [,4] [,5]

[1,] 35.03521 0.00000 0.00000 0.000000e+00 0

[2,] 0.00000 34.04224 0.00000 0.000000e+00 0

[3,] 0.00000 0.00000 22.34174 0.000000e+00 0

[4,] 0.00000 0.00000 0.00000 1.507891e-06 0

[5,] 0.00000 0.00000 0.00000 0.000000e+00 NaN

> prinC <- as.matrix(eigendecomp$vectors) %\*% sqrtLambda

> prinC

[,1] [,2] [,3] [,4] [,5]

[1,] 23.618974 19.965777 0.1690049 -6.743496e-07 NaN

[2,] -18.592434 7.600931 1.9497849 -6.743496e-07 NaN

[3,] 4.898350 -16.920306 -16.2386174 -6.743496e-07 NaN

[4,] -16.158115 8.083855 -1.0624587 -6.743496e-07 NaN

[5,] 6.233226 -18.730256 15.1822863 -6.743496e-07 NaN

>

> plot(eigendecomp$values, type="b", lty=2, main = "Scree Plot", ylab = "Eigenvalue", xlab = "Principal Component Number")

>



> plot(prinC[,1:2], main= "CONFIGURATION: Principal Component Loadings", pch= "",ylab = "PC #2 Loading", xlab = "PC #1 Loading")

> abline(h=0); abline(v=0)

> text(prinC[,1:2],c("E","H","N","S","W"))

>



> #########################################################################

> # Analyze Assn 3 data using R's "classical MDS" package, cmdscale

> D\_cmds<-cmdscale(D, eig=TRUE, add = FALSE, x.ret = TRUE)

> plot(D\_cmds$points)

> text(D\_cmds$points,rownames(D))

>

> # use regression to aid in interpreting dimensions

> # E .

> # H ....

> # N -.

> # S ...

> # W .--

>

> # plot results of regression for # signal components

> ncomponents=c(1,4,2,3,3)

> ncompsol<-lm(ncomponents ~ D\_cmds$points[,1:2])

> nxy<-ncompsol$coefficients[2:3]\*300

> arrows(0,0,nxy[1],nxy[2])

> text(nxy[1],nxy[2],"#comp")

>

> #plot results of regression for proportion of dots in signal

> propdot=c(100,100,50,100,33)

> propdotsol<-lm(propdot ~ D\_cmds$points[,1:2])

> pxy<-propdotsol$coefficients[2:3]\*10

> arrows(0,0,pxy[1],pxy[2])

> text(pxy[1],pxy[2],"propdot")

