**Instructions for using Terrence Tracey’s RAND R package on an 8 octant circumplex measure**

1) compute correlation matrix for intercorrelations among the 8 (ordered octants)

2) save correlation matrix as a “txt” file (e.g., see example immediately below). Name file something like “matrix.txt”.

1.000

.275 1.000

-.027 .405 1.000

-.242 .072 .322 1.000

-.592 -.443 -.237 .127 1.000

-.420 -.677 -.565 -.308 .412 1.000

-.047 -.494 -.604 -.530 -.079 .440 1.000

.267 -.056 -.351 -.488 -.368 -.020 .286 1.000

3) Open R and set working directory to folder containing your “matrix.txt” file.

4) Run the following in R.

randall<-function(n,nmat,ord,input,samp){

library("permute")

setMaxperm<-(50000)

library("utils")

np<- ((n\*n)-n)/2

#read input file

#read in diagonal matrix as vector and then fill in

za<-scan(input) #goodf read in vector

dmatm<-array(dim=c(n,n,nmat))

for (m in 1:nmat){

ii<-(m-1)\*(np+n)

for(j in 1:n){

for(i in 1:n){

if (i > j) next

ii<-ii+1

dmatm[i,j,m]<-za[ii]}} #set up matrix

ii<-(m-1)\*(np+n)

for(i in 1:n){

for(j in 1:n){

if (i < j) next

ii<-ii+1

dmatm[i,j,m]<-za[ii]}} #fill in

} #end nmat loop

#mathhyp generation

mathyp<-matrix(nrow=np,ncol=np)

for(i in 1:np){

for(j in 1:np){

if(ord[j]<ord[i]) mathyp[i,j]<-1 else mathyp[i,j]<-0}}

#count hyp

nhyp<-0

for(i in mathyp){

if(i==1)nhyp<-nhyp+1}

#print header and nhyp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#out1<-matrix(nrow=2,ncol=2)

zz<-matrix(nrow=1,ncol=n)

out<- matrix(nrow=nmat,ncol=7)

dmatp<-matrix(nrow=n,ncol=n)

dmat<-matrix(nrow=n,ncol=n)

pp<-vector(length=n)

#set up permutation file

nper<-1

for(i in 1:n){

nper<- i\*nper} ##good n permutation

if(nper> 50000)nper<-50000

permat<-matrix(nrow=nper,ncol=n)

zz<-matrix(nrow=1,ncol=n)

f<-function (zz){

zz<-sample.int(n,n,replace=FALSE,prob=NULL)

return(zz)}

if(nper<50000)permat<-allPerms(n, control = how(maxperm = 50000), check = TRUE)

if(nper>=50000) for(ll in 1:nper){permat[ll,]<-t(apply(zz,1,f))}

#do big loop over nmat kk

for(kk in 1:nmat){

#select matrix from array

dmat<-dmatm[,,kk]

#run on original data prior to permutations

#do data 1,0 matc gt=1 eq =2

nagr<-0

ntie<-0

ii=0

scal<-vector(length=np)

for(i in 1:n){

for(j in 1:n){

if (i >= j) next

ii<-ii+1

scal[ii]<-dmat[i,j]}}

#match data matc with mathyp 1=conf 2=tie, 3=less

matc <-matrix(nrow=np,ncol=np)

for(i in 1:np){

for(j in 1:np){

if(scal[j] > scal[i]) matc[i,j]<-1

if(scal[j] == scal[i]) matc[i,j]<-2

if(scal[j] < scal[i]) matc[i,j]<-0}}

nsup<-0

nntie<-0

for(i in 1:np){

for(j in 1:np){

if(matc[i,j]==1 & mathyp[i,j]==1) nagr<-nagr+1

if(matc[i,j]==2 & mathyp[i,j]==1) ntie<-ntie+1}}

ci<- (nagr-(nhyp-(nagr+ntie)))/nhyp

count<- 1 #counter for number exceed values in original permuation

nperx<-nper-1 ##check on this correction

#do small loop over nper

for(k in 1:nperx){ #minus 1

# if(k >50000) break #stop if greater than 50,000

#set up new matrix

pp<-permat[k,]

for(i in 1:n){

for(j in 1:n){

dmatp[i,j]<-dmat[pp[i],pp[j]]

}}

#do data 1,0 matc gt=1 eq =2

ii=0

scal2<-vector(length=np)

for(i in 1:n){

for(j in 1:n){

if (i >= j) next

ii<-ii+1

scal2[ii]<-dmatp[i,j]}}

#match data matc with mathyp 1=conf 2=tie, 3=less

matc2 <-matrix(nrow=np,ncol=np)

for(i in 1:np){

for(j in 1:np){

if(scal2[j] > scal2[i]) matc2[i,j] <- 1

if(scal2[j] == scal2[i]) matc2[i,j] <- 2

if(scal2[j] < scal2[i]) matc2[i,j] <- 0}}

nsup<-0

nntie<-0

for(i in 1:np){

for(j in 1:np){

if(matc2[i,j]==1 & mathyp[i,j]==1) nsup<-nsup+1

if(matc2[i,j]==2 & mathyp[i,j]==1) nntie<-nntie+1}}

if(nsup >= nagr) count <- count+1 #count number of cases where fit is equal or greater

} #end first loop kz

prob<-count/nper

out[kk,1]<-kk

out[kk,2]<-nhyp

out[kk,3]<-nagr

out[kk,4]<-ntie

out[kk,5]<-ci

out[kk,6]<-prob

out[kk,7]<-samp[kk]

}#end loop kk different matrices

colnames(out)<-c("samp","pred","met","tie","CI","p"," ")

rownames(out)<-c(1:nmat)

print(out,quote=FALSE)

}

#end randall

randall(n=8,nmat=1,ord=(c(1,2,3,4,3,2,1,1,2,3,4,3,2,1,2,3,4,3,1,2,3,4,1,2,3,1,2,1)),input="matrix.txt",samp=("NameOfMeasure"))