**Online Resource 2: *R* code**

**A permutation version of Kuiper’s two sample test for circular data**

**as an alternative to Watson’s U² test**

Graeme D. Ruxton1, E. Pascal Malkemper2, Lukas Landler3\*

1School of Biology, University of St Andrews, St Andrews KY16 9TH, UK

2Max Planck Research Group Neurobiology of Magnetoreception, Center of Advanced European Studies and Research (caesar), Ludwig-Erhard-Allee 2, Bonn 53175, Germany

3Institute of Zoology, University of Natural Resources and Life Sciences (BOKU),  
1180 Vienna, Gregor-Mendel-Straße 33/I

\*corresponding author: [lukas.landler@boku.ac.at](mailto:lukas.landler@boku.ac.at)

**R code to reproduce the simulations in the paper**

**Code for comparing 2 unimodal von Mises Distributions**

# Code to test which tests are more powerful in detecting differences between distributions

library(kuiper.2samp)

library(circular)

library(NPCirc)

library(permute)

library(foreach)

library(doParallel)

options("scipen" = 10)

enableJIT(3)

#setting sample size for dist 1 - in two steps to reduce typing effort

nvseq=c(10,20,50,20,10)

nv1=c(nvseq,nvseq,nvseq)

#setting sample size for dist 2

nvseq2=c(10,20,50,30,50)

nv2=c(nvseq2,nvseq2,nvseq2)

#combine sample sizes in matrix to make calling in the loop practical

n\_matrix = cbind(nv1,nv2)

#preparing certain vectors of concentration parameters and direction - all 7 elements long (they need to be same length)

kseq=c(0,0.25,0.5,1,2,4,8) #standard kappa sequence for plot

k0=rep(0,7)

k2=rep(2,7)

mseq= c(rad(0),rad(30),rad(60),rad(90),rad(120),rad(150),rad(180))

m0=rep(0,7)

#this would be needed of lengths differ, at the present it doesn't matter which l is used

lk=length(kseq)

lm=length(mseq)

l=lk # this one is in this script used for the inner loop

#preparing the sequences in a list so they can be called in inner loop

k\_list\_1=list(kseq,kseq,kseq,kseq,kseq,k0,k0,k0,k0,k0,k2,k2,k2,k2,k2)

k\_list\_2=list(kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,k2,k2,k2,k2,k2)

m\_list\_1=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0)

m\_list\_2=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,mseq,mseq,mseq,mseq,mseq)

#defining the distribution type, there are always 5 combinations of sample sizes

#therefore I prepared "chunks" of distribution and then combined them, decreasing typing effort

type\_vm=c("vm","vm","vm","vm","vm")

type\_matrix\_1 = c(type\_vm,type\_vm,type\_vm)

type\_matrix\_2 = c(type\_vm,type\_vm,type\_vm)

#again in chunks of 5: naming the file that is created (see text for write.csv)

Type\_1\_errorn=c("Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error")

Powern=c("Power","Power","Power","Power","Power")

#again combining the chunks

Name=c(Type\_1\_errorn,Powern,Powern)

#starting the outer loop, it loops through the sample size vector, taking the respective vectors (of length 7) from the respective list

#the outer loop is stored one csv file, this means one panel in the plot.

for (i in 1:length(nv1)) {

########### set number of iterations, Sample size mean and kappa (or a sequence of any of that)

rans = 10000 #iterations for distribution, could have placed outside the loops as well

#grabbing the correct vectors for the simulation from the list

nv1 = n\_matrix[,1][i]

nv2=n\_matrix[,2][i]

m1seq=m\_list\_1[[i]]

m2seq=m\_list\_2[[i]]

k1seq=k\_list\_1[[i]]

k2seq=k\_list\_2[[i]]

typeDist1 =type\_matrix\_1[i]

typeDist2 = type\_matrix\_2[i]

#setup parallel back-end to use many processors

cores=detectCores()

cl <- parallel::makeCluster(cores[1]-2, setup\_strategy = "sequential") #used "sequential" otherwise it crashed for me

registerDoParallel(cl)

#start the inner loop. Here I use a foreach instead of for. It splits the jobs on the number of cores set up.

#Parallelization is done with "%dopar%". The packages used in the foreach loop must be passed to the loop.

#they are not grabbed automatically from the environment.

#the results of each of the loops are combined (with rbind) in the file Power.sample

Power.sample<-foreach(e = 1:l,.combine=rbind,.packages=c("kuiper.2samp","NPCirc","permute","circular")) %dopar% {

###self written functions

kuiper.stat <- function(x,y){

n1 <- length(x)

n2 <- length(y)

n <- n1+n2

cf1 <- 0

cf2 <- 0

D1 <- 0

D2 <- 0

groups <- c(rep(1,n1),rep(2,n2))

both <- c(x,y)

ordered <- groups[order(both)]

for (i in 1:n){

if(ordered[i] == 1){cf1 <- cf1+(1/n1)} else {cf2 <- cf2+(1/n2)}

test1 <- cf1-cf2

if(test1 > D1){D1 <- test1}

test2 <- -test1

if(test2 > D2){D2 <- test2}}

return(D1+D2)}

kuiper.perm <- function(x,y){

n1 <- length(x)

n2 <- length(y)

reps <- 999

lower <- 0

both <- c(x,y)

n <- length(both)

calc <- kuiper.stat(x,y)

for (i in 1:reps){

perm <- shuffle(1:n)

x1 <- both[perm[1:n1]]

x2 <- both[perm[(n1+1):n]]

if(kuiper.stat(x1,x2) >= calc){lower <- lower+1}}

return((1+lower)/(reps+1))}

#take correct values for mean direction as well as the concentration parameter from the vector

m1 = m1seq[e]

m2 = m2seq[e]

k1 = k1seq[e]

k2 = k2seq[e]

#First distribution generated rans times, the "units=" could be deleted. Its just here to remind me what it is, but doesn't do anything.

Distribution1 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv1,model = NULL,

dist = typeDist1,

param=list(p=1,

mu=m1,con=k1))))

###### set up the 2nd distribution

Distribution2 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv2,model = NULL,

dist = typeDist2,

param=list(p=1,

mu=m2,con=k2))))

test<-watson.two.test(Distribution1[1],Distribution2[2],alpha=0.05)

###### underneath we apply each of the tests used on the generated distribution

#using mapply (if the groups are arranged in separate column) and apply (if they are arranged underneath each other)

##watson2SampleTest

watson1 = mapply(function(x,y) {watson.two.test(x,y)},Distribution1,Distribution2)

watson1 <- array(as.matrix(unlist(watson1)), dim=c(5, rans))

watson1<- watson1[1,] # critical valueCritical Value: 0.187

##kuiper

kuiper1 = mapply(function(x,y) {kuiper.2samp(x, y)},Distribution1,Distribution2)

kuiper1 <- array(as.matrix(unlist(kuiper1)), dim=c(2, rans))

kuiperp<- kuiper1[2,]

#testst<-lapply(1:rans,function(z) mapply(function(x,y) {kuiper.2samp(as.data.frame(x), as.data.frame(y))},Rand1[[z]],Rand2[[z]]))

kuiper\_rand = mapply(function(x,y) {kuiper.perm (x, y)},Distribution1,Distribution2)

#create a temp data.frame to store results

temp <- data.frame(matrix(nrow = 1, ncol = 3))

names(temp) <- c('Kuiper','Kuiper\_rand','WatsonU2')

#####Calculate type I error/power in temp file

temp$Kuiper<- (sum(1\*(kuiperp<0.05)))/rans

temp$Kuiper\_rand<- (sum(1\*(kuiper\_rand<0.05)))/rans

temp$WatsonU2<- (sum(1\*(watson1>0.187)))/rans

temp # this is stored in the power.sample

} #e

#the x-axis is calculated from the concentration of the second distribution, if the maximum is larger the minimum number

#other wise it takes the mean direction vector

Power.sample$x=if(max(k2seq)>min(k2seq)) k2seq else deg(m2seq)

dataaa=Power.sample

#prepare the text of the csv. file

text = paste(Name[i],"\_n1\_",nv1,"\_n2\_",nv2,"\_its\_",rans,"\_Dist1\_",typeDist1,"\_Dist2\_",typeDist2,

"\_m1=",deg(min(m1seq)),"-",deg(max(m1seq)),"\_m2\_",deg(min(m2seq)),"-",deg(max(m2seq)),

"\_k1=",min(k1seq),"-",max(k1seq),"\_k2\_",min(k2seq),"-",max(k2seq),".csv",sep="")

#write csv file in the working directory

write.csv(dataaa, file = text,row.names=FALSE )

}#i

**Code for comparing 2 axial von Mises Distributions**

# Code to test which tests are more powerful in detecting differences between distributions

library(kuiper.2samp)

library(circular)

library(NPCirc)

library(permute)

library(foreach)

library(doParallel)

options("scipen" = 10)

enableJIT(3)

nvseq=c(10,20,50,20,10)\*2

nvseq2=c(10,20,50,30,50)\*2

nv1=c(nvseq,nvseq,nvseq)

nv2=c(nvseq2,nvseq2,nvseq2)

n\_matrix = cbind(nv1,nv2)

kseq=c(0,0.25,0.5,1,2,4,8)

lk=length(kseq)

k0=rep(0,7)

k2=rep(2,7)

mseq= c(rad(0),rad(30),rad(60),rad(90),rad(120),rad(150),rad(180))/2

lm=length(mseq)

m0=rep(0,7)

k\_list\_1=list(kseq,kseq,kseq,kseq,kseq,k0,k0,k0,k0,k0,k2,k2,k2,k2,k2)

k\_list\_2=list(kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,k2,k2,k2,k2,k2)

m\_list\_1=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0)

m\_list\_2=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,mseq,mseq,mseq,mseq,mseq)

l=length(kseq)

type\_vm=c("vm","vm","vm","vm","vm")

type\_matrix\_1 = c(type\_vm,type\_vm,type\_vm)

type\_matrix\_2 = c(type\_vm,type\_vm,type\_vm)

Type\_1\_errorn=c("Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error")

Powern=c("Power","Power","Power","Power","Power")

Name=c(Type\_1\_errorn,Powern,Powern)

########### set number of iterations, Sample size mean and kappa (or a sequence of any of that)

rans = 10000 #iterations for distribution

for (i in 1:length(nv1)) {

nv1 = n\_matrix[,1][i]

nv2=n\_matrix[,2][i]

m1seq=m\_list\_1[[i]]

m2seq=m\_list\_2[[i]]

k1seq=k\_list\_1[[i]]

k2seq=k\_list\_2[[i]]

typeDist1 =type\_matrix\_1[i]

typeDist2 = type\_matrix\_2[i]

# make two groups for the test where the data needs to b sturctured in the same column

Group1 <-(as.matrix( rep((as.matrix((strrep(c("A"), 1)))),nv1),dim=c(nv1, 1)))

Group2 <- (as.matrix( rep((as.matrix((strrep(c("B"), 1)))),nv2),dim=c(nv2, 1)))

GroupsAB<-as.data.frame(c(Group1[,1], Group2[,1]))

#setup parallel backend to use many processors

cores=detectCores()

cl <- parallel::makeCluster(cores[1]-1, setup\_strategy = "sequential")

registerDoParallel(cl)

Power.sample<-foreach(e = 1:l,.combine=rbind,.packages=c("kuiper.2samp","NPCirc","permute","circular")) %dopar% {

###self written functions

kuiper.stat <- function(x,y){

n1 <- length(x)

n2 <- length(y)

n <- n1+n2

cf1 <- 0

cf2 <- 0

D1 <- 0

D2 <- 0

groups <- c(rep(1,n1),rep(2,n2))

both <- c(x,y)

ordered <- groups[order(both)]

for (i in 1:n){

if(ordered[i] == 1){cf1 <- cf1+(1/n1)} else {cf2 <- cf2+(1/n2)}

test1 <- cf1-cf2

if(test1 > D1){D1 <- test1}

test2 <- -test1

if(test2 > D2){D2 <- test2}}

return(D1+D2)}

kuiper.perm <- function(x,y){

n1 <- length(x)

n2 <- length(y)

reps <- 999

lower <- 0

both <- c(x,y)

n <- length(both)

calc <- kuiper.stat(x,y)

for (i in 1:reps){

perm <- shuffle(1:n)

x1 <- both[perm[1:n1]]

x2 <- both[perm[(n1+1):n]]

if(kuiper.stat(x1,x2) >= calc){lower <- lower+1}}

return((1+lower)/(reps+1))}

m1 = m1seq[e]

m2 = m2seq[e]

k1 = k1seq[e]

k2 = k2seq[e]

#First distribution

Distribution1 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv1,model = NULL,

dist = c(typeDist1,typeDist1),

param=list(p=c(0.5,0.5),

mu=c(m1,m1-pi),

con=c(k1,k1)))),units = 'radians')

###### set the 2nd distribution

Distribution2 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv2,model = NULL,

dist = c(typeDist2,typeDist2),

param=list(p=c(0.5,0.5),

mu=c(m2,m2-pi),

con=c(k2,k2)))),units = 'radians')

##watson2SampleTest

watson1 = mapply(function(x,y) {watson.two.test(x,y)},Distribution1,Distribution2)

watson1 <- array(as.matrix(unlist(watson1)), dim=c(5, rans))

watson1<- watson1[1,] # critical valueCritical Value: 0.187

#kuiper

kuiper1 = mapply(function(x,y) {kuiper.2samp(x, y)},Distribution1,Distribution2)

kuiper1 <- array(as.matrix(unlist(kuiper1)), dim=c(2, rans))

kuiperp<- kuiper1[2,]

#testst<-lapply(1:rans,function(z) mapply(function(x,y) {kuiper.2samp(as.data.frame(x), as.data.frame(y))},Rand1[[z]],Rand2[[z]]))

kuiper\_rand = mapply(function(x,y) {kuiper.perm (x, y)},Distribution1,Distribution2)

#create a temp data.frame to store results

temp <- data.frame(matrix(nrow = 1, ncol = 3))

names(temp) <- c('Kuiper','Kuiper\_rand','WatsonU2')

#####Calculate type I error/power in temp file

temp$Kuiper<- (sum(1\*(kuiperp<0.05)))/rans

temp$Kuiper\_rand<- (sum(1\*(kuiper\_rand<0.05)))/rans

temp$WatsonU2<- (sum(1\*(watson1>0.187)))/rans

temp # this is stored in the power.sample

} #e

stopCluster(cl)

Power.sample$x=if(max(k2seq)>min(k2seq)) k2seq else deg(m2seq)

Power.sample

text = paste(Name[i],"\_n1\_",nv1,"\_n2\_",nv2,"\_its\_",rans,"\_Dist1\_","Axial",typeDist1,"\_Dist2\_",typeDist2,

"\_m1=",deg(min(m1seq)),"-",deg(max(m1seq)),"\_m2\_",deg(min(m2seq)),"-",deg(max(m2seq)),

"\_k1=",min(k1seq),"-",max(k1seq),"\_k2\_",min(k2seq),"-",max(k2seq),".csv",sep="")

write.csv(Power.sample, file = text,row.names=FALSE )

}#i

**Code for comparing 2 asymmetrical bimodal von Mises Distributions**

# Code to test which tests are more powerful in detecting differences between distributions

library(kuiper.2samp)

library(circular)

library(NPCirc)

library(permute)

library(foreach)

library(doParallel)

options("scipen" = 10)

enableJIT(3)

nvseq=c(10,20,50,20,10)\*2

nvseq2=c(10,20,50,30,50)\*2

nv1=c(nvseq,nvseq,nvseq)

nv2=c(nvseq2,nvseq2,nvseq2)

n\_matrix = cbind(nv1,nv2)

kseq=c(0,0.25,0.5,1,2,4,8)

lk=length(kseq)

k0=rep(0,7)

k2=rep(2,7)

mseq= c(rad(0),rad(30),rad(60),rad(90),rad(120),rad(150),rad(180))/2

lm=length(mseq)

m0=rep(0,7)

k\_list\_1=list(kseq,kseq,kseq,kseq,kseq,k0,k0,k0,k0,k0,k2,k2,k2,k2,k2)

k\_list\_2=list(kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,k2,k2,k2,k2,k2)

m\_list\_1=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0)

m\_list\_2=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,mseq,mseq,mseq,mseq,mseq)

l=length(kseq)

type\_vm=c("vm","vm","vm","vm","vm")

type\_matrix\_1 = c(type\_vm,type\_vm,type\_vm)

type\_matrix\_2 = c(type\_vm,type\_vm,type\_vm)

Type\_1\_errorn=c("Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error")

Powern=c("Power","Power","Power","Power","Power")

Name=c(Type\_1\_errorn,Powern,Powern)

########### set number of iterations, Sample size mean and kappa (or a sequence of any of that)

rans = 10000 #iterations for distribution

for (i in 1:length(nv1)) {

nv1 = n\_matrix[,1][i]

nv2=n\_matrix[,2][i]

m1seq=m\_list\_1[[i]]

m2seq=m\_list\_2[[i]]

k1seq=k\_list\_1[[i]]

k2seq=k\_list\_2[[i]]

typeDist1 =type\_matrix\_1[i]

typeDist2 = type\_matrix\_2[i]

# make two groups for the test where the data needs to b sturctured in the same column

Group1 <-(as.matrix( rep((as.matrix((strrep(c("A"), 1)))),nv1),dim=c(nv1, 1)))

Group2 <- (as.matrix( rep((as.matrix((strrep(c("B"), 1)))),nv2),dim=c(nv2, 1)))

GroupsAB<-as.data.frame(c(Group1[,1], Group2[,1]))

#setup parallel backend to use many processors

cores=detectCores()

cl <- parallel::makeCluster(cores[1]-1, setup\_strategy = "sequential")

registerDoParallel(cl)

Power.sample<-foreach(e = 1:l,.combine=rbind,.packages=c("kuiper.2samp","NPCirc","permute","circular")) %dopar% {

###self written functions

kuiper.stat <- function(x,y){

n1 <- length(x)

n2 <- length(y)

n <- n1+n2

cf1 <- 0

cf2 <- 0

D1 <- 0

D2 <- 0

groups <- c(rep(1,n1),rep(2,n2))

both <- c(x,y)

ordered <- groups[order(both)]

for (i in 1:n){

if(ordered[i] == 1){cf1 <- cf1+(1/n1)} else {cf2 <- cf2+(1/n2)}

test1 <- cf1-cf2

if(test1 > D1){D1 <- test1}

test2 <- -test1

if(test2 > D2){D2 <- test2}}

return(D1+D2)}

kuiper.perm <- function(x,y){

n1 <- length(x)

n2 <- length(y)

reps <- 999

lower <- 0

both <- c(x,y)

n <- length(both)

calc <- kuiper.stat(x,y)

for (i in 1:reps){

perm <- shuffle(1:n)

x1 <- both[perm[1:n1]]

x2 <- both[perm[(n1+1):n]]

if(kuiper.stat(x1,x2) >= calc){lower <- lower+1}}

return((1+lower)/(reps+1))}

m1 = m1seq[e]

m2 = m2seq[e]

k1 = k1seq[e]

k2 = k2seq[e]

#First distribution

Distribution1 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv1,model = NULL,

dist = c(typeDist1,typeDist1),

param=list(p=c(0.5,0.5),

mu=c(m1,m1-rad(120)),

con=c(k1,k1)))))

###### set the 2nd distribution

Distribution2 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv2,model = NULL,

dist = c(typeDist2,typeDist2),

param=list(p=c(0.5,0.5),

mu=c(m2,m2-rad(120)),

con=c(k2,k2)))))

##watson2SampleTest

watson1 = mapply(function(x,y) {watson.two.test(x,y)},Distribution1,Distribution2)

watson1 <- array(as.matrix(unlist(watson1)), dim=c(5, rans))

watson1<- watson1[1,] # critical valueCritical Value: 0.187

#kuiper

kuiper1 = mapply(function(x,y) {kuiper.2samp(x, y)},Distribution1,Distribution2)

kuiper1 <- array(as.matrix(unlist(kuiper1)), dim=c(2, rans))

kuiperp<- kuiper1[2,]

#testst<-lapply(1:rans,function(z) mapply(function(x,y) {kuiper.2samp(as.data.frame(x), as.data.frame(y))},Rand1[[z]],Rand2[[z]]))

kuiper\_rand = mapply(function(x,y) {kuiper.perm (x, y)},Distribution1,Distribution2)

#create a temp data.frame to store results

temp <- data.frame(matrix(nrow = 1, ncol = 3))

names(temp) <- c('Kuiper','Kuiper\_rand','WatsonU2')

#####Calculate type I error/power in temp file

temp$Kuiper<- (sum(1\*(kuiperp<0.05)))/rans

temp$Kuiper\_rand<- (sum(1\*(kuiper\_rand<0.05)))/rans

temp$WatsonU2<- (sum(1\*(watson1>0.187)))/rans

temp # this is stored in the power.sample

} #e

stopCluster(cl)

Power.sample$x=if(max(k2seq)>min(k2seq)) k2seq else deg(m2seq)

text = paste(Name[i],"\_n1\_",nv1,"\_n2\_",nv2,"\_its\_",rans,"\_Dist1\_","Asymm\_bimodal",typeDist1,"\_Dist2\_",typeDist2,

"\_m1=",deg(min(m1seq)),"-",deg(max(m1seq)),"\_m2\_",deg(min(m2seq)),"-",deg(max(m2seq)),

"\_k1=",min(k1seq),"-",max(k1seq),"\_k2\_",min(k2seq),"-",max(k2seq),".csv",sep="")

write.csv(Power.sample, file = text,row.names=FALSE )

}#i

**Code for comparing 2 symmetrical trimodal von Mises Distributions**

# Code to test which tests are more powerful in detecting differences between distributions

library(kuiper.2samp)

library(circular)

library(NPCirc)

library(permute)

library(foreach)

library(doParallel)

options("scipen" = 10)

enableJIT(3)

nvseq=c(10,20,50,20,10)\*3

nvseq2=c(10,20,50,30,50)\*3

nv1=c(nvseq,nvseq,nvseq)

nv2=c(nvseq2,nvseq2,nvseq2)

n\_matrix = cbind(nv1,nv2)

kseq=c(0,0.25,0.5,1,2,4,8)

lk=length(kseq)

k0=rep(0,7)

k2=rep(2,7)

mseq= c(rad(0),rad(30),rad(60),rad(90),rad(120),rad(150),rad(180))/3

lm=length(mseq)

m0=rep(0,7)

k\_list\_1=list(kseq,kseq,kseq,kseq,kseq,k0,k0,k0,k0,k0,k2,k2,k2,k2,k2)

k\_list\_2=list(kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,k2,k2,k2,k2,k2)

m\_list\_1=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0)

m\_list\_2=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,mseq,mseq,mseq,mseq,mseq)

l=length(kseq)

type\_vm=c("vm","vm","vm","vm","vm")

type\_matrix\_1 = c(type\_vm,type\_vm,type\_vm)

type\_matrix\_2 = c(type\_vm,type\_vm,type\_vm)

Type\_1\_errorn=c("Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error")

Powern=c("Power","Power","Power","Power","Power")

Name=c(Type\_1\_errorn,Powern,Powern)

########### set number of iterations, Sample size mean and kappa (or a sequence of any of that)

rans = 10000 #iterations for distribution

for (i in 1:length(nv1)) {

nv1 = n\_matrix[,1][i]

nv2=n\_matrix[,2][i]

m1seq=m\_list\_1[[i]]

m2seq=m\_list\_2[[i]]

k1seq=k\_list\_1[[i]]

k2seq=k\_list\_2[[i]]

typeDist1 =type\_matrix\_1[i]

typeDist2 = type\_matrix\_2[i]

# make two groups for the test where the data needs to b sturctured in the same column

Group1 <-(as.matrix( rep((as.matrix((strrep(c("A"), 1)))),nv1),dim=c(nv1, 1)))

Group2 <- (as.matrix( rep((as.matrix((strrep(c("B"), 1)))),nv2),dim=c(nv2, 1)))

GroupsAB<-as.data.frame(c(Group1[,1], Group2[,1]))

#setup parallel backend to use many processors

cores=detectCores()

cl <- parallel::makeCluster(cores[1]-1, setup\_strategy = "sequential")

registerDoParallel(cl)

Power.sample<-foreach(e = 1:l,.combine=rbind,.packages=c("kuiper.2samp","NPCirc","permute","circular")) %dopar% {

###self written functions

kuiper.stat <- function(x,y){

n1 <- length(x)

n2 <- length(y)

n <- n1+n2

cf1 <- 0

cf2 <- 0

D1 <- 0

D2 <- 0

groups <- c(rep(1,n1),rep(2,n2))

both <- c(x,y)

ordered <- groups[order(both)]

for (i in 1:n){

if(ordered[i] == 1){cf1 <- cf1+(1/n1)} else {cf2 <- cf2+(1/n2)}

test1 <- cf1-cf2

if(test1 > D1){D1 <- test1}

test2 <- -test1

if(test2 > D2){D2 <- test2}}

return(D1+D2)}

kuiper.perm <- function(x,y){

n1 <- length(x)

n2 <- length(y)

reps <- 999

lower <- 0

both <- c(x,y)

n <- length(both)

calc <- kuiper.stat(x,y)

for (i in 1:reps){

perm <- shuffle(1:n)

x1 <- both[perm[1:n1]]

x2 <- both[perm[(n1+1):n]]

if(kuiper.stat(x1,x2) >= calc){lower <- lower+1}}

return((1+lower)/(reps+1))}

m1 = m1seq[e]

m2 = m2seq[e]

k1 = k1seq[e]

k2 = k2seq[e]

#First distribution

Distribution1 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv1,model = NULL,

dist = c(typeDist1,typeDist1,typeDist1),

param=list(p=c(0.333,0.333,0.333),

mu=c(m1,m1-rad(120),m1-rad(240)),

con=c(k1,k1,k1)))),units = 'radians')

###### set the 2nd distribution

Distribution2 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv2,model = NULL,

dist = c(typeDist2,typeDist2,typeDist2),

param=list(p=c(0.333,0.333,0.333),

mu=c(m2,m2-rad(120),m2-rad(240)),

con=c(k2,k2,k2)))),units = 'radians')

######

##watson2SampleTest

watson1 = mapply(function(x,y) {watson.two.test(x,y)},Distribution1,Distribution2)

watson1 <- array(as.matrix(unlist(watson1)), dim=c(5, rans))

watson1<- watson1[1,] # critical valueCritical Value: 0.187

#kuiper

kuiper1 = mapply(function(x,y) {kuiper.2samp(x, y)},Distribution1,Distribution2)

kuiper1 <- array(as.matrix(unlist(kuiper1)), dim=c(2, rans))

kuiperp<- kuiper1[2,]

#testst<-lapply(1:rans,function(z) mapply(function(x,y) {kuiper.2samp(as.data.frame(x), as.data.frame(y))},Rand1[[z]],Rand2[[z]]))

kuiper\_rand = mapply(function(x,y) {kuiper.perm (x, y)},Distribution1,Distribution2)

#create a temp data.frame to store results

temp <- data.frame(matrix(nrow = 1, ncol = 3))

names(temp) <- c('Kuiper','Kuiper\_rand','WatsonU2')

#####Calculate type I error/power in temp file

temp$Kuiper<- (sum(1\*(kuiperp<0.05)))/rans

temp$Kuiper\_rand<- (sum(1\*(kuiper\_rand<0.05)))/rans

temp$WatsonU2<- (sum(1\*(watson1>0.187)))/rans

temp # this is stored in the power.sample

} #e

stopCluster(cl)

Power.sample$x=if(max(k2seq)>min(k2seq)) k2seq else deg(m2seq)

text = paste(Name[i],"\_n1\_",nv1,"\_n2\_",nv2,"\_its\_",rans,"\_Dist1\_","Symm\_trimodal",typeDist1,"\_Dist2\_",typeDist2,

"\_m1=",deg(min(m1seq)),"-",deg(max(m1seq)),"\_m2\_",deg(min(m2seq)),"-",deg(max(m2seq)),

"\_k1=",min(k1seq),"-",max(k1seq),"\_k2\_",min(k2seq),"-",max(k2seq),".csv",sep="")

write.csv(Power.sample, file = text,row.names=FALSE )

}#i

**Code for comparing 2 asymmetrical trimodal von Mises Distributions**

# Code to test which tests are more powerful in detecting differences between distributions

library(kuiper.2samp)

library(circular)

library(NPCirc)

library(permute)

library(foreach)

library(doParallel)

options("scipen" = 10)

enableJIT(3)

nvseq=c(10,20,50,20,10)\*3

nvseq2=c(10,20,50,30,50)\*3

nv1=c(nvseq,nvseq,nvseq)

nv2=c(nvseq2,nvseq2,nvseq2)

n\_matrix = cbind(nv1,nv2)

kseq=c(0,0.25,0.5,1,2,4,8)

lk=length(kseq)

k0=rep(0,7)

k2=rep(2,7)

mseq= c(rad(0),rad(30),rad(60),rad(90),rad(120),rad(150),rad(180))/3

lm=length(mseq)

m0=rep(0,7)

k\_list\_1=list(kseq,kseq,kseq,kseq,kseq,k0,k0,k0,k0,k0,k2,k2,k2,k2,k2)

k\_list\_2=list(kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,kseq,k2,k2,k2,k2,k2)

m\_list\_1=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,m0)

m\_list\_2=list(m0,m0,m0,m0,m0,m0,m0,m0,m0,m0,mseq,mseq,mseq,mseq,mseq)

l=length(kseq)

type\_vm=c("vm","vm","vm","vm","vm")

type\_matrix\_1 = c(type\_vm,type\_vm,type\_vm)

type\_matrix\_2 = c(type\_vm,type\_vm,type\_vm)

Type\_1\_errorn=c("Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error","Type\_1\_error")

Powern=c("Power","Power","Power","Power","Power")

Name=c(Type\_1\_errorn,Powern,Powern)

for (i in 1:length(nv1)) {

########### set number of iterations, Sample size mean and kappa (or a sequence of any of that)

rans = 9999 #iterations for distribution

nv1 = n\_matrix[,1][i]

nv2=n\_matrix[,2][i]

m1seq=m\_list\_1[[i]]

m2seq=m\_list\_2[[i]]

k1seq=k\_list\_1[[i]]

k2seq=k\_list\_2[[i]]

typeDist1 =type\_matrix\_1[i]

typeDist2 = type\_matrix\_2[i]

# make two groups for the test where the data needs to b sturctured in the same column

Group1 <-(as.matrix( rep((as.matrix((strrep(c("A"), 1)))),nv1),dim=c(nv1, 1)))

Group2 <- (as.matrix( rep((as.matrix((strrep(c("B"), 1)))),nv2),dim=c(nv2, 1)))

GroupsAB<-as.data.frame(c(Group1[,1], Group2[,1]))

#setup parallel backend to use many processors

cores=detectCores()

cl <- parallel::makeCluster(cores[1]-1, setup\_strategy = "sequential")

registerDoParallel(cl)

Power.sample<-foreach(e = 1:l,.combine=rbind,.packages=c("kuiper.2samp","NPCirc","permute","circular")) %dopar% {

###self written fucntions (Graeme werote them)

kuiper.stat <- function(x,y){

n1 <- length(x)

n2 <- length(y)

n <- n1+n2

cf1 <- 0

cf2 <- 0

D1 <- 0

D2 <- 0

groups <- c(rep(1,n1),rep(2,n2))

both <- c(x,y)

ordered <- groups[order(both)]

for (i in 1:n){

if(ordered[i] == 1){cf1 <- cf1+(1/n1)} else {cf2 <- cf2+(1/n2)}

test1 <- cf1-cf2

if(test1 > D1){D1 <- test1}

test2 <- -test1

if(test2 > D2){D2 <- test2}}

return(D1+D2)}

kuiper.perm <- function(x,y){

n1 <- length(x)

n2 <- length(y)

reps <- 999

lower <- 0

both <- c(x,y)

n <- length(both)

calc <- kuiper.stat(x,y)

for (i in 1:reps){

perm <- shuffle(1:n)

x1 <- both[perm[1:n1]]

x2 <- both[perm[(n1+1):n]]

if(kuiper.stat(x1,x2) >= calc){lower <- lower+1}}

return((1+lower)/(reps+1))}

m1 = m1seq[e]

m2 = m2seq[e]

k1 = k1seq[e]

k2 = k2seq[e]

#First distribution

Distribution1 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv1,model = NULL,

dist = c(typeDist1,typeDist1,typeDist1),

param=list(p=c(1/3,1/3,1/3),

mu=c(m1,m1-rad(90),m1-rad(200)),

con=c(k1,k1,k1)))),units = 'radians')

###### set the 2nd distribution

Distribution2 <- as.data.frame(sapply(1:rans, function(x) rcircmix(nv2,model = NULL,

dist = c(typeDist2,typeDist2,typeDist2),

param=list(p=c(1/3,1/3,1/3),

mu=c(m2,m2-rad(90),m2-rad(200)),

con=c(k2,k2,k2)))),units = 'radians')

######

##watson2SampleTest

watson1 = mapply(function(x,y) {watson.two.test(x,y)},Distribution1,Distribution2)

watson1 <- array(as.matrix(unlist(watson1)), dim=c(5, rans))

watson1<- watson1[1,] # critical valueCritical Value: 0.187

#kuiper

kuiper1 = mapply(function(x,y) {kuiper.2samp(x, y)},Distribution1,Distribution2)

kuiper1 <- array(as.matrix(unlist(kuiper1)), dim=c(2, rans))

kuiperp<- kuiper1[2,]

#kuiper\_stat<- kuiper1[1,]

#testst<-lapply(1:rans,function(z) mapply(function(x,y) {kuiper.2samp(as.data.frame(x), as.data.frame(y))},Rand1[[z]],Rand2[[z]]))

kuiper\_rand = mapply(function(x,y) {kuiper.perm (x, y)},Distribution1,Distribution2)

#create a temp data.frame to store results

temp <- data.frame(matrix(nrow = 1, ncol = 3))

names(temp) <- c('Kuiper','Kuiper\_rand','WatsonU2')

#####Calculate type I error/power in temp file

temp$Kuiper<- (sum(1\*(kuiperp<0.05)))/rans

temp$Kuiper\_rand<- (sum(1\*(kuiper\_rand<0.05)))/rans

temp$WatsonU2<- (sum(1\*(watson1>0.187)))/rans

temp # this is stored in the power.sample

} #e

stopCluster(cl)

Power.sample$x=if(max(k2seq)>min(k2seq)) k2seq else deg(m2seq)

text = paste(Name[i],"\_n1\_",nv1,"\_n2\_",nv2,"\_its\_",rans,"\_Dist1\_","asymm\_trimodal",typeDist1,"\_Dist2\_",typeDist2,

"\_m1=",deg(min(m1seq)),"-",deg(max(m1seq)),"\_m2\_",deg(min(m2seq)),"-",deg(max(m2seq)),

"\_k1=",min(k1seq),"-",max(k1seq),"\_k2\_",min(k2seq),"-",max(k2seq),".csv",sep="")

write.csv(Power.sample, file = text,row.names=FALSE )

}#i