SUPPLEMENT

1. Pseudocode for calculating and plotting codispersion coefficients:

**Input**: two datasets (*X*, *Y*), each with spatial information and associated data (e.g. an R geodata object (Ribeiro and Diggle 2001)).

**begin**

*k* ← kernel bandwidth

**for** *h*1 in 1 to 20 columns **do** # 20 lags in *x* direction

**for** *h*2 in 1 to 10 rows **do** # 10 lags in *y* direction

. # Compute and store the codispersion coefficient

**end**

**end**

**plot** # Create graph

**end**

2. R code for all analyses and graphs in the paper. Note that to run the first two sections of code that analyze the (1) simulated datasets and (2) the Harvard Forest plot data, you need to save the code in the section at the bottom of this file as "CoDisp\_functions.R" and run this file as a source code before you begin. Note that CoDisp\_functions.R loads the required R libraries – spatstat, geoR, fields, SpatialPack, ggplot2, grid, raster, and gstat – which, along with their dependencies, should be installed on your local machine. This code was developed and run in RStudio version 0.98.1103 using R version 3.1.2 “Pumpkin Helmet” on platform: x86\_64-w64-mingw32/x64 (64-bit).

##################################################################

### Spatial point pattern simulations for demonstrating

### the codispersion function

##############################################################

source("CoDisp\_functions.R") # see code at bottom of this file for these functions

##################################################################

### Simulated co-occurrence grid patterns for Ecology paper

##################################################################

copp.ls <- vector("list",16)

copp.ls[[1]] <- copp.fn(grid.points = 20,sp1.pattern="CSR",sp2.pattern="CSR",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[2]] <- copp.fn(grid.points = 20,sp1.pattern="CSR",sp2.pattern="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[3]] <- copp.fn(grid.points = 20,sp1.pattern="CSR",sp2.pattern="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[4]] <- copp.fn(grid.points = 20,sp1.pattern="CSR",sp2.pattern="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[5]] <- copp.fn(grid.points = 20,sp1.pattern="CSR",sp2.pattern="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[6]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.x",sp2.pattern="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[7]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.x",sp2.pattern="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[8]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.x",sp2.pattern="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[9]] <- copp.fn(grid.points = 20,sp1.pattern="increasing.x",sp2.pattern="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[10]] <- copp.fn(grid.points = 20,sp1.pattern="increasing.x",sp2.pattern="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[11]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.xy",sp2.pattern="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[12]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.x",sp2.pattern="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[13]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.xy",sp2.pattern="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[14]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.x",sp2.pattern="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[15]] <- copp.fn(grid.points = 20,sp1.pattern="dec.x.inc.y",sp2.pattern="inc.x.dec.y",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

copp.ls[[16]] <- copp.fn(grid.points = 20,sp1.pattern="decreasing.y",sp2.pattern="increasing.y",xmin=0,xmax=300,ymin=0,ymax=300,Print="TRUE")

gtitles=c("CSR,CSR","CSR,decreasing.x","CSR,increasing.x","CSR,decreasing.xy","CSR,bivariate.normal","decreasing.x,increasing.x","decreasing.x,decreasing.xy","decreasing.x,bivariate.normal","increasing.x,decreasing.xy","increasing.x,bivariate.normal","decreasing.xy,bivariate.normal","decreasing.x,decreasing.x","decreasing.xy,increasing.xy","decreasing.x,increasing.xy","decreasing.x.increasing.y,increasing.x.decreasing.y","decreasing.y, increasing.y")

# Specify parameters and options for CoDisp analysis

k=c(20,20,20)

max.window.size = 300/4

binwidth=0.1

xmin=0

xmax=300

ymin=0

ymax=300

Codisp\_out <- vector(mode="list",length=length(copp.ls)) # list for output CoDisp objects

Graphs\_out <- vector(mode="list",length=length(copp.ls)) # list for graph outputs

Means.df <- data.frame(sim=gtitles,mean\_CoDisp=NA,sd\_CoDisp=NA) # create object to hold mean and sd codispersion values for each simulation

for(i in 1:length(copp.ls)){

print(date())

print(paste("i =",i))

Graphs\_ls <- vector(mode="list",length=4) # empty list for output graphs

## Extract the data

sp1.geo <- copp.ls[[i]][[1]]

sp2.geo <- copp.ls[[i]][[2]]

CoDisp\_sim <- codisp.fn(sp1.geo,sp2.geo,k=k,max.window.size=max.window.size)

sp1.dat <- data.frame(X=sp1.geo$coords[,1],Y=sp1.geo$coords[,2],AB=sp1.geo$data)

sp2.dat <- data.frame(X=sp2.geo$coords[,1],Y=sp1.geo$coords[,2],AB=sp2.geo$data)

## Graph the output

Graphs\_ls[[1]] <- ggplot(sp1.dat, aes(x=X, y=Y, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.lab.20pt

Graphs\_ls[[2]] <- ggplot(sp2.dat, aes(x=X, y=Y, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.lab.20pt

## Save the output object

nam=(paste("CoDisp\_copp\_sim",i,sep="\_"))

assign(nam,CoDisp\_sim)

CoDisp\_sim <- codisp.fn(sp1.geo,sp2.geo,k=k,max.window.size=max.window.size)

## Graph the output

Graphs\_ls[[3]] <- ggplot(CoDisp\_sim[[1]],aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1.no.lab.20pt

Graphs\_ls[[4]] <- ggplot(CoDisp\_sim[[1]],aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1, 1))+coord\_fixed(ratio=1)+t1.no.leg\_lab.20+xlab(NULL)+ylab(NULL)+stat\_contour(aes(x=xx,y=yy,z=Codispersion),binwidth=binwidth)

## Calculate the mean values

Means.df$mean\_CoDisp[i] <- round(mean(CoDisp\_sim[[1]]$Codispersion),2)

Means.df$sd\_CoDisp[i] <- round(sd(CoDisp\_sim[[1]]$Codispersion),2)

## Save the output object

Codisp\_out[[i]] <- CoDisp\_sim

nam=(paste("CoDisp\_copp\_sim",i,sep="\_"))

assign(nam,CoDisp\_sim)

## Save the output objects

Graphs\_out[[i]] <- Graphs\_ls

nam=(paste("Graphs\_ls",i,sep="\_"))

assign(nam,Graphs\_ls)

} # end i loop

save.image("copp\_sim.RData")

###################################

### Graph output

##################################

###############

# Graph species patterns for FIGURE 1

###############

#load("copp\_sim.RData")

png("CoDispEcolSims\_Figure.png",width=1400,height=1000)

grphs <- c(2,6,16,7,13,15) # the graphs we want in the figure

grid.newpage()

pushViewport(viewport(layout=grid.layout(6,6))) # 5 sims by 4 graphs in 6 columns

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

out <- Graphs\_out[[grphs[i]]] # select simulation output

g1 <- out[[1]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp1 raster graph

g2 <- out[[2]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp2 raster graph

g3 <- out[[3]]+xlab(NULL)+ylab(NULL) # Unscaled graph

g4 <- out[[4]]+xlab(NULL)+ylab(NULL) # Scaled graph

## Print the graphs to the layout

print(g1, vp=vplayout(i,1))

print(g2, vp=vplayout(i,2))

print(g3, vp=vplayout(i,3:4))

print(g4, vp=vplayout(i,5:6))

}

dev.off()

## Example figure for Box 1

ggplot(Codisp\_out[[9]][[1]],aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+t1

ggsave("example.png")

###############

# Graph species patterns for APPENDIX A

###############

png("Graphs\_CoDispSims\_AppFig1.png",width=1400,height=1327)

grid.newpage()

pushViewport(viewport(layout=grid.layout(8,6))) # 15 sims by 4 graphs in 6 columns

vplayout <- function(x,y)

viewport(layout.pos.row=x,layout.pos.col=y)

for(i in 1:8){

out <- Graphs\_out[[i]] # select simulation output

g1 <- out[[1]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp1 raster graph

g2 <- out[[2]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp2 raster graph

g3 <- out[[3]]+xlab(NULL)+ylab(NULL) # Unscaled graph

g4 <- out[[4]]+xlab(NULL)+ylab(NULL) # Scaled graph

## Print the graphs to the layout

print(g1, vp=vplayout(i,1))

print(g2, vp=vplayout(i,2))

print(g3, vp=vplayout(i,3:4))

print(g4, vp=vplayout(i,5:6))

}

dev.off()

png("Graphs\_CoDispSims\_AppFig2.png",width=1400,height=1327)

grid.newpage()

pushViewport(viewport(layout=grid.layout(8,6))) # 15 sims by 4 graphs in 6 columns

vplayout <- function(x,y)

viewport(layout.pos.row=x,layout.pos.col=y)

for(i in 9:16){

out <- Graphs\_out[[i]] # select simulation output

g1 <- out[[1]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp1 raster graph

g2 <- out[[2]]+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab.20 # Sp2 raster graph

g3 <- out[[3]]+xlab(NULL)+ylab(NULL) # Unscaled graph

g4 <- out[[4]]+xlab(NULL)+ylab(NULL) # Scaled graph

## Print the graphs to the layout

print(g1, vp=vplayout(i-8,1))

print(g2, vp=vplayout(i-8,2))

print(g3, vp=vplayout(i-8,3:4))

print(g4, vp=vplayout(i-8,5:6))

}

dev.off()

###############

### APPENDIX B: Null model analysis of CSR, CSR to obtain error rates

###############

## Plot species grid plots

# copp.ls[[1]]

plot(copp.ls[[1]][[1]])

CC1.dat <- data.frame(xx=copp.ls[[1]][[1]]$coords[,1],yy=copp.ls[[1]][[1]]$coords[,2],AB=copp.ls[[1]][[1]]$data)

CC2.dat <- data.frame(xx=copp.ls[[1]][[2]]$coords[,1],yy=copp.ls[[1]][[2]]$coords[,2],AB=copp.ls[[1]][[2]]$data)

ggplot(CC1.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg+xlab("X (m)")+ylab("Y (m)")

ggsave("CoDisp\_Sims\_CC1.png")

ggplot(CC2.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg+xlab("X (m)")+ylab("Y (m)")

ggsave("CoDisp\_Sims\_CC2.png")

# Codisp\_out[[1]]

load("copp\_sim.RData")

source("CoDisp\_functions.R")

# settings for codispersion analysis

k=c(20,20,20)

max.window.size = 300/4

binwidth=0.1

xmin=0

xmax=300

ymin=0

ymax=300

nsim = 199

# turn simulated geodata objects into ppp patterns for randomisations

CSR1.ppp <- ppp(x=copp.ls[[1]][[1]]$coords[,1]+0.001,y=copp.ls[[1]][[1]]$coords[,2]+0.001,window=owin(c(xmin,xmax),c(ymin,ymax)),marks=copp.ls[[1]][[1]]$data) # add a small amount to move points off the lattice

CSR2.ppp <- ppp(x=copp.ls[[1]][[2]]$coords[,1]+0.001,y=copp.ls[[1]][[2]]$coords[,2]+0.001,window=owin(c(xmin,xmax),c(ymin,ymax)),marks=copp.ls[[1]][[2]]$data)

# randomise 'species 2' patterns using Homogeneous Poisson and Toroidal shift null models

HomP\_CSR2.ls <- ppp.null.fn(CSR2.ppp,nsim=nsim,model="HomP",marks=TRUE)

Tor\_CSR2.ls <- ppp.null.fn(CSR2.ppp,nsim=nsim,model="Tor",marks=TRUE)

# turn ppp objects back into geodata objects for codispersion analysis

geo.CSR1 <- ppp.to.geoR.fn(CSR1.ppp,quad.size=20,xmin,xmax,ymin,ymax,method="sum")

geo.HomP.CSR2 <- lapply(HomP\_CSR2.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="sum")

geo.Tor.CSR2 <- lapply(Tor\_CSR2.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="sum")

# make empty lists to hold null model results

CoDisp\_CSR\_HomP <- vector("list",nsim)

CoDisp\_CSR\_Tor <- vector("list",nsim)

# Run codispersion analysis on null model data

for(j in 1:nsim){

print(paste("CoDisp\_HomP\_CSR, j",j)) # HomP

CoDisp\_CSR\_HomP[[j]] <- codisp.fn(geo.CSR1,geo.HomP.CSR2[[j]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_CSR, i j",i,j)) # Toroidal shift

CoDisp\_CSR\_Tor[[j]] <- codisp.fn(geo.CSR1,geo.Tor.CSR2[[j]], k=k,max.window.size=max.window.size)

} # end simulations j loop

# Convert output lists to array objects

CoDisp\_CSR\_HomP\_ary <- list2ary(CoDisp\_CSR\_HomP)

CoDisp\_CSR\_Tor\_ary <- list2ary(CoDisp\_CSR\_Tor)

save.image("Simulated\_CSR&CSR\_null\_199.RData")

load("copp\_sim.RData")

load("Simulated\_CSR&CSR\_null\_199.RData")

source("CoDisp\_functions.R")

CSR\_out.df <- codisp.compare(CoDisp\_CSR\_HomP\_ary,Codisp\_out[[1]][[1]],round=TRUE)

Tor\_out.df <- codisp.compare(CoDisp\_CSR\_Tor\_ary,Codisp\_out[[1]][[1]],round=TRUE)

write.table(CSR\_out.df,"CSR\_type I error rate.csv",sep=",")

write.table(Tor\_out.df,"Tor\_type I error rate.csv",sep=",")

### CSR

# Observed minus expected CoDispersion value graph

( g2 <- ggplot(CSR\_out.df,aes(xx,yy))+geom\_tile(aes(fill=diff))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1, 1))+coord\_fixed(ratio=1)+t1+xlab(NULL)+ylab(NULL)+stat\_contour(aes(x=xx,y=yy,z=diff),binwidth=binwidth)+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") )

ggsave("CSR\_type1\_OE.png")

# P-value category graph

my.cols <- c("steelblue3","firebrick3")

if(levels(CSR\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g3 <- ggplot(CSR\_out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value.cat))+scale\_fill\_manual(values=my.cols)+scale\_colour\_discrete(name="P.value.cat",limits=c(0,1))+coord\_fixed(ratio=1)+t1.no.leg+xlab(NULL)+ylab(NULL) )

### Toroidal shift

# Observed minus expected CoDispersion value graph

( g2 <- ggplot(Tor\_out.df,aes(xx,yy))+geom\_tile(aes(fill=diff))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1, 1))+coord\_fixed(ratio=1)+t1+xlab(NULL)+ylab(NULL)+stat\_contour(aes(x=xx,y=yy,z=diff),binwidth=binwidth)+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") )

ggsave("Tor\_type1\_OE.png")

# P-value category graph

my.cols <- c("steelblue3","firebrick3")

if(levels(Tor\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g3 <- ggplot(Tor\_out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value.cat))+scale\_fill\_manual(values=my.cols)+scale\_colour\_discrete(name="P.value.cat",limits=c(0,1))+coord\_fixed(ratio=1)+t1.no.leg+xlab(NULL)+ylab(NULL) )

######################################################

######################################################

##### Analysis of Harvard Forest data

######################################################

######################################################

###################################

#### Load required functions

###################################

source("CoDisp\_functions.R") # see code below for these functions

########################

#### Read in the data

########################

dat <- read.csv("http://harvardforest.fas.harvard.edu/data/p25/hf253/hf253-03-trees-2014.csv",header=TRUE)

head(dat)

dat$sp <- as.factor(dat$sp) # make species code a factor

dat$ba <- basal.area.fn(dat$dbh) # calculate basal area for each tree in the dataset

dat$xt <- cut(dat$gx,seq(0,round(max(dat$gx),0),20)) # generate vectors for 20x20 quadrat grid

dat$yt <- cut(dat$gy,seq(0,round(max(dat$gy),0),20))

dat <- dat[ order(dat[,"gx"]), ]

unique(dat$sp) # view species list

nspp <- length(unique(dat$sp)) # number of species

########################

#### Set the plot dimensions

########################

plot(dat$gx,dat$gy)

max(dat$gx)

max(dat$gy)

xmin=0; xmax=700; ymin=0; ymax=500

###################################

## Create ppp objects

###################################

spp.list <- sort(unique(dat$sp))

ppp.ls <- vector("list",nspp)

for (i in 1:nspp){

ppp.ls[[i]] <- ppp(dat$gx[dat$sp==spp.list[i]],dat$gy[dat$sp==spp.list[i]],xrange=c(xmin,xmax),yrange=c(ymin,ymax),marks=dat$dbh[dat$sp==spp.list[i]])

}

# Print maps for all species

#pdf("HF\_ppp\_spp200.pdf")

#for(i in 1:nspp){ if(ppp.ls[[i]]$n>200){plot(ppp.ls[[i]],main=spp.list[i])}}

#dev.off()

###################################

## Extract target species

###################################

spp.list <- c("tsugca","pinust","querru","acerru")

nspp <- length(spp.list)

ppp.ls <- vector("list",nspp)

for (i in 1:nspp){

ppp.ls[[i]] <- ppp(dat$gx[dat$sp==spp.list[i]],dat$gy[dat$sp==spp.list[i]],xrange=c(xmin,xmax),yrange=c(ymin,ymax),marks=dat$dbh[dat$sp==spp.list[i]])

}

ppp.ls

###################################

## Create geodata objects

###################################

ppp.sp1 <- ppp.ls[[1]]

ppp.sp2 <- ppp.ls[[2]]

ppp.sp3 <- ppp.ls[[3]]

ppp.sp4 <- ppp.ls[[4]]

plot(ppp.ls[[1]],main=spp.list[1])

plot(ppp.ls[[2]],main=spp.list[2])

plot(ppp.ls[[3]],main=spp.list[3])

plot(ppp.ls[[4]],main=spp.list[4])

#### Rotate the patterns

#ppp.dat <- rotate(ppp.ls[[i]],centre="midpoint")

#xmin=0; xmax=500; ymin=0; ymax=700

####

geo.obs.ab.sp1 <- ppp.to.geoR.fn(ppp.sp1,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.obs.ab.sp2 <- ppp.to.geoR.fn(ppp.sp2,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.obs.ab.sp3 <- ppp.to.geoR.fn(ppp.sp3,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.obs.ab.sp4 <- ppp.to.geoR.fn(ppp.sp4,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

###################################

## Calculate the codispersion coefficient

###################################

# observed data

CoDispCoef\_sp12 <- codisp.coef.fn(geo.obs.ab.sp1,geo.obs.ab.sp2)

CoDispCoef\_sp13 <- codisp.coef.fn(geo.obs.ab.sp1,geo.obs.ab.sp3)

CoDispCoef\_sp14 <- codisp.coef.fn(geo.obs.ab.sp1,geo.obs.ab.sp4)

CoDispCoef\_sp23 <- codisp.coef.fn(geo.obs.ab.sp2,geo.obs.ab.sp3)

CoDispCoef\_sp24 <- codisp.coef.fn(geo.obs.ab.sp2,geo.obs.ab.sp4)

CoDispCoef\_sp34 <- codisp.coef.fn(geo.obs.ab.sp3,geo.obs.ab.sp4)

plot(CoDispCoef\_sp12[[1]])

plot(CoDispCoef\_sp13[[1]])

plot(CoDispCoef\_sp14[[1]])

plot(CoDispCoef\_sp23[[1]])

plot(CoDispCoef\_sp24[[1]])

plot(CoDispCoef\_sp34[[1]])

###################################

## Run the codispersion analysis (this takes several hours)

###################################

# Settings for the codispersion analysis functions

k=c(20,20,20)

max.window.size=500/4

pdf("HF\_Bivariate\_obs.pdf")

# observed data

CoDisp\_sp12 <- codisp.fn(geo.obs.ab.sp1,geo.obs.ab.sp2, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp12[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[2]))

print.CoDisp(CoDisp\_sp12[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[2]))

CoDisp\_sp13 <- codisp.fn(geo.obs.ab.sp1,geo.obs.ab.sp3, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp13[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[3]))

print.CoDisp(CoDisp\_sp13[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[3]))

CoDisp\_sp14 <- codisp.fn(geo.obs.ab.sp1,geo.obs.ab.sp4, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp14[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[4]))

print.CoDisp(CoDisp\_sp14[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[1],"vs",spp.list[4]))

CoDisp\_sp23 <- codisp.fn(geo.obs.ab.sp2,geo.obs.ab.sp3, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp23[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[2],"vs",spp.list[3]))

print.CoDisp(CoDisp\_sp23[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[2],"vs",spp.list[3]))

CoDisp\_sp24 <- codisp.fn(geo.obs.ab.sp2,geo.obs.ab.sp4, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp24[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[2],"vs",spp.list[3]))

print.CoDisp(CoDisp\_sp24[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[2],"vs",spp.list[4]))

CoDisp\_sp34 <- codisp.fn(geo.obs.ab.sp3,geo.obs.ab.sp4, k=k,max.window.size=max.window.size)

print.CoDisp(CoDisp\_sp34[[1]],scaled=FALSE,input="Abundance\n",gtitle=paste(spp.list[3],"vs",spp.list[4]))

print.CoDisp(CoDisp\_sp34[[1]],scaled=TRUE,contours=TRUE,binwidth=0.1,input="Abundance\n",gtitle=paste(spp.list[3],"vs",spp.list[4]))

dev.off()

save.image("HF\_bivariate\_spp1\_4.RData")

#########################################################

## Run the null models and compare to the observed data

## This takes several days

#########################################################

nsim=199 # set the number of null model simulations

##########

## CSR null model (homogeneous Poisson)

##########

# Create randomised point patterns

HomP\_sp1.ls <- ppp.null.fn(ppp.sp1,nsim=nsim,model=c("HomP"))

HomP\_sp2.ls <- ppp.null.fn(ppp.sp2,nsim=nsim,model=c("HomP"))

HomP\_sp3.ls <- ppp.null.fn(ppp.sp3,nsim=nsim,model=c("HomP"))

HomP\_sp4.ls <- ppp.null.fn(ppp.sp4,nsim=nsim,model=c("HomP"))

# Generate HomP null model geodata objects

geo.HomP.sp1 <- lapply(HomP\_sp1.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.HomP.sp2 <- lapply(HomP\_sp2.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.HomP.sp3 <- lapply(HomP\_sp3.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.HomP.sp4 <- lapply(HomP\_sp4.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

CoDisp\_HomP\_sp12 <- vector("list",nsim)

CoDisp\_HomP\_sp13 <- vector("list",nsim)

CoDisp\_HomP\_sp14 <- vector("list",nsim)

CoDisp\_HomP\_sp23 <- vector("list",nsim)

CoDisp\_HomP\_sp24 <- vector("list",nsim)

CoDisp\_HomP\_sp34 <- vector("list",nsim)

for (i in 1:nsim) {

# Run codispersion analysis on HomP null model data

print(paste("CoDisp\_HomP\_sp12, i =",i))

CoDisp\_HomP\_sp12[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.HomP.sp2[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Hom2P\_sp13, i =",i))

CoDisp\_HomP\_sp13[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.HomP.sp3[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp14, i =",i))

CoDisp\_HomP\_sp14[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.HomP.sp4[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp23, i =",i))

CoDisp\_HomP\_sp23[[i]] <- codisp.fn(geo.obs.ab.sp2,geo.HomP.sp3[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp24, i =",i))

CoDisp\_HomP\_sp24[[i]] <- codisp.fn(geo.obs.ab.sp2,geo.HomP.sp4[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp34, i =",i))

CoDisp\_HomP\_sp34[[i]] <- codisp.fn(geo.obs.ab.sp3,geo.HomP.sp4[[i]], k=k,max.window.size=max.window.size)

} # end i loop

CoDisp\_HomP\_sp12.ary <- list2ary(CoDisp\_HomP\_sp12)

CoDisp\_HomP\_sp13.ary <- list2ary(CoDisp\_HomP\_sp13)

CoDisp\_HomP\_sp14.ary <- list2ary(CoDisp\_HomP\_sp14)

CoDisp\_HomP\_sp23.ary <- list2ary(CoDisp\_HomP\_sp23)

CoDisp\_HomP\_sp24.ary <- list2ary(CoDisp\_HomP\_sp24)

CoDisp\_HomP\_sp34.ary <- list2ary(CoDisp\_HomP\_sp34)

save.image("HF\_bivariate\_spp1\_4\_HomP\_199.RData")

##########

## Toroidal shift null model

##########

# Create randomised point patterns

Tor\_sp1.ls <- ppp.null.fn(ppp.sp1,nsim=nsim,model=c("Tor"),marks=FALSE)

Tor\_sp2.ls <- ppp.null.fn(ppp.sp2,nsim=nsim,model=c("Tor"),marks=FALSE)

Tor\_sp3.ls <- ppp.null.fn(ppp.sp3,nsim=nsim,model=c("Tor"),marks=FALSE)

Tor\_sp4.ls <- ppp.null.fn(ppp.sp4,nsim=nsim,model=c("Tor"),marks=FALSE)

# Generate Tor null model geodata objects

geo.Tor.sp1 <- lapply(Tor\_sp1.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.Tor.sp2 <- lapply(Tor\_sp2.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.Tor.sp3 <- lapply(Tor\_sp3.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

geo.Tor.sp4 <- lapply(Tor\_sp4.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="abundance")

CoDisp\_Tor\_sp12 <- vector("list",nsim)

CoDisp\_Tor\_sp13 <- vector("list",nsim)

CoDisp\_Tor\_sp14 <- vector("list",nsim)

CoDisp\_Tor\_sp23 <- vector("list",nsim)

CoDisp\_Tor\_sp24 <- vector("list",nsim)

CoDisp\_Tor\_sp34 <- vector("list",nsim)

for (i in 1:nsim) {

# Run codispersion analysis on Tor null model data

print(paste("CoDisp\_Tor\_sp12, i =",i))

CoDisp\_Tor\_sp12[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.Tor.sp2[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Hom2P\_sp13, i =",i))

CoDisp\_Tor\_sp13[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.Tor.sp3[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp14, i =",i))

CoDisp\_Tor\_sp14[[i]] <- codisp.fn(geo.obs.ab.sp1,geo.Tor.sp4[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp23, i =",i))

CoDisp\_Tor\_sp23[[i]] <- codisp.fn(geo.obs.ab.sp2,geo.Tor.sp3[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp24, i =",i))

CoDisp\_Tor\_sp24[[i]] <- codisp.fn(geo.obs.ab.sp2,geo.Tor.sp4[[i]], k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp34, i =",i))

CoDisp\_Tor\_sp34[[i]] <- codisp.fn(geo.obs.ab.sp3,geo.Tor.sp4[[i]], k=k,max.window.size=max.window.size)

} # end i loop

CoDisp\_Tor\_sp12.ary <- list2ary(CoDisp\_Tor\_sp12)

CoDisp\_Tor\_sp13.ary <- list2ary(CoDisp\_Tor\_sp13)

CoDisp\_Tor\_sp14.ary <- list2ary(CoDisp\_Tor\_sp14)

CoDisp\_Tor\_sp23.ary <- list2ary(CoDisp\_Tor\_sp23)

CoDisp\_Tor\_sp24.ary <- list2ary(CoDisp\_Tor\_sp24)

CoDisp\_Tor\_sp34.ary <- list2ary(CoDisp\_Tor\_sp34)

save.image("HF\_bivariate\_spp1\_4\_Tor\_199.RData")

##########

## Graph the observed and null model comparisons

##########

load("HF\_bivariate\_spp1\_4.RData")

load("HF\_bivariate\_spp1\_4\_HomP\_199.RData")

load("HF\_bivariate\_spp1\_4\_Tor\_199.RData")

source("CoDisp\_functions.R")

pdf("HF\_bivariate\_spp1\_4\_HomP\_199.pdf")

comparison.abab.fn(CoDisp\_HomP\_sp12.ary,CoDisp\_sp12,model="HomP",spe="tsugca and pinust",binwidth=0.1)

comparison.abab.fn(CoDisp\_HomP\_sp13.ary,CoDisp\_sp13,model="HomP",spe="tsugca and querru",binwidth=0.1)

comparison.abab.fn(CoDisp\_HomP\_sp14.ary,CoDisp\_sp14,model="HomP",spe="tsugca and acerru",binwidth=0.1)

comparison.abab.fn(CoDisp\_HomP\_sp23.ary,CoDisp\_sp23,model="HomP",spe="pinust and querru",binwidth=0.1)

comparison.abab.fn(CoDisp\_HomP\_sp24.ary,CoDisp\_sp24,model="HomP",spe="pinust and acerru",binwidth=0.1)

comparison.abab.fn(CoDisp\_HomP\_sp34.ary,CoDisp\_sp34,model="HomP",spe="querru and acerru",binwidth=0.1)

dev.off()

pdf("HF\_bivariate\_spp1\_4\_Tor\_199.pdf")

comparison.abab.fn(CoDisp\_Tor\_sp12.ary,CoDisp\_sp12,model="Tor",spe="tsugca and pinust",binwidth=0.1)

comparison.abab.fn(CoDisp\_Tor\_sp13.ary,CoDisp\_sp13,model="Tor",spe="tsugca and querru",binwidth=0.1)

comparison.abab.fn(CoDisp\_Tor\_sp14.ary,CoDisp\_sp14,model="Tor",spe="tsugca and acerru",binwidth=0.1)

comparison.abab.fn(CoDisp\_Tor\_sp23.ary,CoDisp\_sp23,model="Tor",spe="pinust and querru",binwidth=0.1)

comparison.abab.fn(CoDisp\_Tor\_sp24.ary,CoDisp\_sp24,model="Tor",spe="pinust and acerru",binwidth=0.1)

comparison.abab.fn(CoDisp\_Tor\_sp34.ary,CoDisp\_sp34,model="Tor",spe="querru and acerru",binwidth=0.1)

dev.off()

#######################

## Null model comparison figure

#######################

# Generate comparison output objects for each species and null model analysis combination

binwidth = 0.1

n.pairs = 6 # number of species pairs

n.mods = 2 # number of null models

png("HF\_Codisp\_Null\_Figure.png",width=1900,height=(n.pairs\*160))

grid.newpage()

pushViewport(viewport(layout=grid.layout(n.pairs,(4+4\*n.mods))))

# Loop through each species and plot the observed CoDisp graphs

obs.codisp.ls <- list(CoDisp\_sp12[[1]],CoDisp\_sp13[[1]],CoDisp\_sp14[[1]],CoDisp\_sp23[[1]],CoDisp\_sp24[[1]],CoDisp\_sp34[[1]])

for (i in 1:length(obs.codisp.ls)) {

# Observed graphs

g0 <- print.CoDisp.plain(obs.codisp.ls[[i]],labels="FALSE",legend="TRUE",scaled=FALSE,contours=FALSE,binwidth=binwidth)

#print(g0, vp=vplayout(i,1:2))

g1 <- print.CoDisp.plain(obs.codisp.ls[[i]],labels="FALSE",legend="FALSE",scaled=TRUE,contours=TRUE,binwidth=binwidth)

print(g1, vp=vplayout(i,3:4))

}

codisp.obj.ls <- list(CoDisp\_sp12[[1]],CoDisp\_sp12[[1]],CoDisp\_sp13[[1]],CoDisp\_sp13[[1]],CoDisp\_sp14[[1]],CoDisp\_sp14[[1]],CoDisp\_sp23[[1]],CoDisp\_sp23[[1]],CoDisp\_sp24[[1]],CoDisp\_sp24[[1]],CoDisp\_sp34[[1]],CoDisp\_sp34[[1]])

null.ary.ls <- list( CoDisp\_HomP\_sp12.ary,CoDisp\_Tor\_sp12.ary,CoDisp\_HomP\_sp13.ary,CoDisp\_Tor\_sp13.ary,CoDisp\_HomP\_sp14.ary,CoDisp\_Tor\_sp14.ary,CoDisp\_HomP\_sp23.ary,CoDisp\_Tor\_sp23.ary,CoDisp\_HomP\_sp24.ary,CoDisp\_Tor\_sp24.ary,CoDisp\_HomP\_sp34.ary,CoDisp\_Tor\_sp34.ary )

g2.ls <- vector("list",n.pairs\*n.mods)

g3.ls <- vector("list",n.pairs\*n.mods)

row.loop.no <- rep(1:n.pairs,each=n.mods)

g2.col.loop.no <- rep(c(5,9),n.pairs)

g3.col.loop.no <- rep(c(7,11),n.pairs)

for(i in 1:length(null.ary.ls)){

# Null model comparison graphs

out.df <- codisp.compare(null.ary.ls[[i]],codisp.obj.ls[[i]])

# Observed minus expected CoDispersion value graph

g2 <- ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=diff))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1, 1))+coord\_fixed(ratio=1)+t1.no.leg+xlab(NULL)+ylab(NULL)+stat\_contour(aes(x=xx,y=yy,z=diff),binwidth=binwidth)

# P-value category graph

my.cols <- c("steelblue3","firebrick3")

if(levels(out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

g3 <- ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value.cat))+scale\_fill\_manual(values=my.cols)+scale\_colour\_discrete(name="P.value.cat",limits=c(0,1))+coord\_fixed(ratio=1)+t1.no.leg+xlab(NULL)+ylab(NULL)

print(g2, vp=vplayout(row.loop.no[i],g2.col.loop.no[i]:(g2.col.loop.no[i]+1)))

print(g3, vp=vplayout(row.loop.no[i],g3.col.loop.no[i]:(g3.col.loop.no[i]+1)))

} # end create graph loop

dev.off()

############################

## Other figures for paper

## Note that all figures saved as .png files were labelled in

## the free software "Inkscape" and saved as high resolution

## eps files.

############################

# observed spatial patterns

sp1 <- data.frame(x=ppp.ls[[1]]$x,y=ppp.ls[[1]]$y,z=ppp.ls[[1]]$marks)

sp2 <- data.frame(x=ppp.ls[[2]]$x,y=ppp.ls[[2]]$y,z=ppp.ls[[2]]$marks)

sp3 <- data.frame(x=ppp.ls[[3]]$x,y=ppp.ls[[3]]$y,z=ppp.ls[[3]]$marks)

sp4 <- data.frame(x=ppp.ls[[4]]$x,y=ppp.ls[[4]]$y,z=ppp.ls[[4]]$marks)

sp1$sp <- "Tsuga canadensis"

sp2$sp <- "Pinus strobus"

sp3$sp <- "Quercus rubra "

sp4$sp <- "Acer rubrum"

ggdat <- rbind(sp1,sp2,sp3,sp4)

names(ggdat)[3] <- "DBH"

# plot all species together

#x11(6,5)

ggplot(ggdat,aes(x=x,y=y))+geom\_point(aes(size=DBH),shape=1)+scale\_size\_area(max\_size=3)+coord\_fixed(ratio=1)+ facet\_wrap(~sp)+theme(strip.text = element\_text(face = "italic",size=20),plot.background = element\_blank(),panel.grid.major = element\_blank(),panel.grid.minor = element\_blank(),panel.border = element\_blank(),panel.background = element\_blank(),axis.line = element\_line(size=.4),axis.text = element\_text(colour="black",size=20),axis.title = element\_text(colour="black",size=20),legend.text = element\_text(colour="black",size=18),legend.title = element\_text(colour="black",size=20))+xlab("X (m)")+ylab("Y (m)")

ggsave("HF\_spp\_ppp.png",height=10,width=12,units="in")

#######################

## Variograms for observed HF spatial patterns

#######################

sp1.var <- variog(geo.obs.ab.sp1,nugget.tolerance=39,max.dist=300 )

sp2.var <- variog(geo.obs.ab.sp2,nugget.tolerance=39,max.dist=300 )

sp3.var <- variog(geo.obs.ab.sp3,nugget.tolerance=39,max.dist=300 )

sp4.var <- variog(geo.obs.ab.sp4,nugget.tolerance=39,max.dist=300 )

gvdat.sp1 <- data.frame(semivariance=sp1.var$v,distance=sp1.var$u)

gvdat.sp2 <- data.frame(semivariance=sp2.var$v,distance=sp2.var$u)

gvdat.sp3 <- data.frame(semivariance=sp3.var$v,distance=sp3.var$u)

gvdat.sp4 <- data.frame(semivariance=sp4.var$v,distance=sp4.var$u)

gvdat.sp1$sp <- "Tsuga canadensis"

gvdat.sp2$sp <- "Pinus strobus"

gvdat.sp3$sp <- "Quercus rubra "

gvdat.sp4$sp <- "Acer rubrum"

gvdat <- rbind(gvdat.sp1,gvdat.sp2,gvdat.sp3,gvdat.sp4)

# plot all species together

#x11(6,5)

ggplot(gvdat,aes(x=distance,y=semivariance))+geom\_line(size=1.5)+facet\_wrap(~sp,scales = "free")+theme(strip.text = element\_text(face = "italic",size=20),plot.background = element\_blank(),panel.grid.major = element\_blank(),panel.grid.minor = element\_blank(),panel.border = element\_blank(),panel.background = element\_blank(),axis.line = element\_line(size=.4),axis.text = element\_text(colour="black",size=20),axis.title = element\_text(colour="black",size=20),legend.text = element\_text(colour="black",size=18),legend.title = element\_text(colour="black",size=20))+xlab("Distance (m)")+ylab("Semivariance")

ggsave("HF\_spp\_variograms.png",height=10,width=12,units="in")

######################################################

######################################################

##### Required functions for codispersion analysis

##### Save all the code below as "CoDisp\_functions.R"

##### Run as a source file: source("CoDisp\_functions.R")

######################################################

######################################################

##################################

### LOAD REQUIRED PACKAGES (these must be installed first)

##################################

library(spatstat)

library(geoR)

library(fields)

library(SpatialPack)

library(ggplot2)

library(grid)

library(raster)

library(gstat)

##################################

### SIMPLE FUNCTIONS

##################################

# basal area function: calculates basal area from DBH values (must be in cm)

basal.area.fn <- function(x){ (pi\*(x)^2)/40000 } # calculate basal area in m^2

### Function to draw random values from a truncated log normal distribution

rtlnorm <- function (n, meanlog = 0, sdlog = 1, lower = -Inf, upper = Inf)

{

ret <- numeric()

if (n > 1)

n <- n

while (length(ret) < n) {

y <- rlnorm(n - length(ret), meanlog, sdlog)

y <- y[y >= lower & y <= upper]

ret <- c(ret, y)

}

stopifnot(length(ret) == n)

ret

}

### Function for simulating a bivariate normal distribution

bivariate <- function(x,y){

mu1 <- 0 # expected value of x

mu2 <- 0 # expected value of y

sig1 <- 1 # variance of x

sig2 <- 1 # variance of y

rho <- 0.5 # corr(x, y)

term1 <- 1 / (2 \* pi \* sig1 \* sig2 \* sqrt(1 - rho^2))

term2 <- (x - mu1)^2 / sig1^2

term3 <- -(2 \* rho \* (x - mu1)\*(y - mu2))/(sig1 \* sig2)

term4 <- (y - mu2)^2 / sig2^2

z <- term2 + term3 + term4

term5 <- term1 \* exp((-z / (2 \*(1 - rho^2))))

return (term5)

}

##################################

### DATA MANIPULATION

##################################

# List to array function for Co\_disp null model output objects

list2ary = function(input.list){ #input a list of lists

temp.ls <- vector("list")

for(i in 1:length(input.list)) { temp.ls[i] <- input.list[[i]][1] } # take the dataframes out of the list and put them in a new list

rows.cols <- dim(temp.ls[[1]])

sheets <- length(temp.ls)

output.ary <- array(unlist(temp.ls), dim = c(rows.cols, sheets))

colnames(output.ary) <- colnames(temp.ls[[1]])

row.names(output.ary) <- row.names(temp.ls[[1]])

return(output.ary) # output as a 3-D array

}

#### Function to generate a geodata object (used by packages geoR and the codispersion function) from a ppp object.

# ppp.dat = input ppp object

# xmin, xmax, ymin, ymax = plot dimensions

# method = the measure that is used to generate the 'data' value for the geodata object

ppp.to.geoR.fn <- function(ppp.dat,xmin,xmax,ymin,ymax,quad.size,method=c("abundance","mean.mark","mean.ba","total.ba","sum")){ # function to generate geoR objects with abundance and basal area in 20x20m quadrats. Note that DBH must be measured in cm. Input data= ppp object.

x <- ppp.dat$x # extract x coordinate

y <- ppp.dat$y # extract y coordinate

z <- ppp.dat$marks # extract DBH values

ba <- (pi\*(z)^2)/40000 # calculate basal area in m^2

xt <- cut(x,seq(xmin,xmax,quad.size)) # cut x coordinates using 20m spacing

yt <- cut(y,seq(ymin,ymax,quad.size)) # cut y coordinates using 20m spacing

coords <- dimnames(table(yt,xt)) # extract quadrat coordinate lists

qx <- rep(seq(xmin,xmax-quad.size,length=length(coords$xt)),each=length(coords$yt)) # vector of x coordinates for the bottom left corner of the quadrat

qy <- rep(seq(ymin,ymax-quad.size,length=length(coords$yt)),length(coords$xt)) # vector of y coordinates for the bottom left corner of the quadrat

if(method=="abundance"){

out.grid <- table(yt,xt) # count the trees in each quadrat

out.grid[is.na(out.grid)==T] <- 0 # replace NAs in table with zeros for empty quadrats

}

if(method=="mean.mark"){

out.grid <- tapply(z,list(yt,xt),mean) # calculate mean DBH in each quadrat

out.grid[is.na(out.grid)==T] <- 0

}

if(method=="mean.ba"){

out.grid <- tapply(ba,list(yt,xt),mean) # calculate mean ba in each quadrat

out.grid[is.na(out.grid)==T] <- 0

}

if(method=="total.ba"){

out.grid <- tapply(ba,list(yt,xt),sum) # calculate total ba in each quadrat

out.grid[is.na(out.grid)==T] <- 0

}

if(method=="sum"){

out.grid <- tapply(z,list(yt,xt),sum) # calculate sum of the marks in each quadrat

out.grid[is.na(out.grid)==T] <- 0

}

out.df <- data.frame(qx,qy,as.vector(out.grid))

out.geo <- as.geodata(out.df,coords.col=1:2,data.col=3)

return(out.geo)

} # end function

##################################

### CODISPERSION ANALYSIS

##################################

#### Modified codispersion function (modified from Cuevas et al. 2013)

#### See 'Box 1' for a detailed explanation.

Codisp.Kern<-function(X,Y,h,k,gamma=1)

{

Kernel<-function(u,gamma)

{

v=0

v=ifelse(abs(u)<=1,(1/beta(0.5,gamma+1))\*(1-u^2)^gamma,0)

}

ifelse(X$coords==Y$coords,1,

{

break

print("The coordinates of X and Y are different")

})

n=length(X$data)

mX <- matrix(X$data,nrow=n,ncol=n,byrow=FALSE)

mY <- matrix(Y$data,nrow=n,ncol=n,byrow=FALSE)

MatriXX <- (mX - t(mX))^2

MatriYY <- (mY - t(mY))^2

MatriXY <- (mX - t(mX))\*(mY - t(mY))

mX <- matrix(X$coords[,1],nrow=n,ncol=n,byrow=FALSE)

DesignX <- mX - t(mX)

mY <- matrix(X$coords[,2],nrow=n,ncol=n,byrow=FALSE)

DesignY <- mY - t(mY)

KERNMATRIXX=Kernel((h[1]-DesignX)/k[1],gamma)\*Kernel((h[2]-DesignY)/k[1],gamma)

if(k[1]==k[2]&k[1]==k[3]){

KERNMATRIYY=KERNMATRIXX

KERNMATRIXY=KERNMATRIXX } else{

KERNMATRIYY=Kernel((h[1]-DesignX)/k[2],gamma)\*Kernel((h[2]-DesignY)/k[2],gamma)

KERNMATRIXY=Kernel((h[1]-DesignX)/k[3],gamma)\*Kernel((h[2]-DesignY)/k[3],gamma)

}

Numerador=sum(KERNMATRIXY\*MatriXY)/(2\*sum(KERNMATRIXY))

Denominador1=sum(KERNMATRIYY\*MatriYY)/(2\*sum(KERNMATRIYY))

Denominador2=sum(KERNMATRIXX\*MatriXX)/(2\*sum(KERNMATRIXX))

v1=Denominador1

v2=Denominador2

v3=Numerador

v4=Numerador/sqrt(Denominador1\*Denominador2)

print(c(v1,v2,v3,v4))

}

### Function to run codispersion window analysis (modified from Cuevas et al. 2013)

# geodata1 = first input data object (a geoR geodata object)

# geodata2 = second input object

# k = c(k1, k2, k3) = a vector of three bandwidth values for X, Y and XY

# max.window.size = the maximum lag distance

# lx = is the number of divisions in the lags in x (up to the max.window.size) that the kernal is applied over. Half of these divisions are in the 'left', or positive direction, and half are in the 'right', or negative x direction.

# ly = is the number of divisions in the lags in y (up to the max.window.size) that the kernal is applied over in the 'up' direction of the plot

codisp.fn <- function(geodata1, geodata2, k=k, max.window.size=max.window.size, lx=20, ly=10){

out <- vector("list",length=2)

X=geodata1 # input data process 1

Y=geodata2 # input data process 2

k=c(k[1],k[2],k[3]) # Set the bandwith for the kernel

h\_range <- max.window.size # set the spatial lags over which to calculate codisp

h1=seq(-h\_range,h\_range,l=lx) # x-axis values for codispersion graph (lags)

h2=seq(min(k),h\_range,l=ly) # y-axis values for codispersion graph (lags)

MCodisp=matrix(0,ncol=ly,nrow=lx) # loop through the lags

for(i in 1:lx) # 'left-right' lags

{

for(j in 1:ly) # 'up' lags

{

MCodisp[i,j]=Codisp.Kern(X,Y,c(h1[i],h2[j]),k)[4]; # calculate codisp

}

}

Codispersion <- as.numeric(MCodisp) # save codisp object as output

xx <- rep(h1,length(h2)) # write out values for x-axis

yy <- rep(h2,each=length(h1)) # write out values for y-axis

graphing.data <- data.frame(xx,yy,Codispersion) # graphing object

# put both the graphing object and the original object in an output list

out[[1]] <- graphing.data

out[[2]] <- MCodisp

return(out)

}

##################################

### NULL MODELS

##################################

#### Function to generate a list of 'nsim' ppp objects (marked point patterns) under four different null models

ppp.null.fn <- function(ppp.dat,nsim,model=c("RLM","HomP","HetP","Tor"),marks=TRUE) {

#ppp.dat <- ppp.dat[[1]]

ppp.out <- vector("list",nsim) # create output list object

if(model=="RLM"){ # Random labelling model

for(i in 1:nsim){ # start loop to generate simulations

ppp.out[[i]] <- rlabel(ppp.dat, labels=marks(ppp.dat), permute=TRUE) # randomise marks

} # end simulations loop

} # end RLM loop

if(model=="HomP"){ # Homogeneous Poisson model

for(i in 1:nsim){ # start loop to generate simulations

ppp.HomP <- rpoint(ppp.dat$n,win=ppp.dat$win) # randomise the observed ppp

ppp.HomP$marks <- sample(ppp.dat$marks, replace=F) # assign shuffled marks to new ppp

ppp.out[[i]] <- ppp.HomP # add new marked ppp to output list

} # end simulations loop

} # end HomP loop

if(marks==TRUE){

if(model=="HetP"){ # this null model generates random marks based on a lognormal fit to the DBH distribution

intensity\_function <- density.ppp(ppp.dat, bw.diggle) # generate the intensity function

LN\_params <- fitdistr(ppp.dat$marks,"log-normal") # fit lognormal to DBH distribution

for(i in 1:nsim){ # start loop to generate simulations

ppp.HetP <- rpoispp(intensity\_function) # generate randomised ppp using intensity function

ppp.HetP$marks <- rtlnorm(ppp.HetP$n,meanlog=LN\_params$estimate[[1]],sdlog=LN\_params$estimate[[2]],1,max(ppp.dat$marks)) # generate marks using parameters of DBH distribution

ppp.out[[i]] <- ppp.HetP # add new marked ppp to output list

} # end simulations loop

} # end HetP loop

} # end marks==TRUE

if(marks==FALSE){

if(model=="HetP"){ # this null model ignores the marks

intensity\_function <- density.ppp(ppp.dat, bw.diggle) # generate the intensity function

for(i in 1:nsim){ # start loop to generate simulations

ppp.HetP <- rpoispp(intensity\_function) # generate randomised ppp using intensity function

ppp.out[[i]] <- ppp.HetP # add new marked ppp to output list

} # end simulations loop

} # end HetP

} # end marks==FALSE

if(model=="Tor"){ # Toroidal shift null model

for(i in 1:nsim){ # start loop to generate simulations

ppp.out[[i]] <- rshift(ppp.dat, edge="torus") # toroidal shift randomisation

} # end simulations loop

} # end toroidal shift

return(ppp.out)

} # end function

##################################

### DEALING WITH CODISPERSION OUTPUTS

##################################

# Comparing observed values to null model results from output arrays and graphing average null model results

# inputs are the null model input array object, the observed CoDisp result list, and a choice of null model

# This function deals with species' co-occurrences

# 'spe' is text to identify the two species, e.g. spe="tsugca and querru"

comparison.abab.fn = function(null.input.ary,CoDisp\_obs,model=c("HomP","Tor","HetP"),spe=spe,binwidth=binwidth){

out.df <- CoDisp\_obs[[1]] # extract the observed Codispersion result as a dataframe

for(i in 1:length(null.input.ary[,1,1])){ # loop through each cell

nsims <- length(null.input.ary[1,1,])

obser <- out.df$Codispersion[i] # observed codispersion value

expec <- null.input.ary[i,3,]

prop.greater.than <- length(which(expec>obser))/nsims

prop.less.than <- length(which(expec<obser))/nsims

out.df$P.value[i]<-min(prop.greater.than,prop.less.than)

}

out.df$null\_mean <- apply(null.input.ary[,3,],MARGIN=1,mean) # calculate mean codispersion value for each cell from the array of null model results

out.df$diff <- out.df$Codispersion-out.df$null\_mean # observed minus expected

out.df$P.value.cat <- factor(ifelse(out.df$P.value<0.025,"Sig.","Non-sig.")) # significance at alpha=0.05

## Graph results

if(model=="HomP"){gtitle="Homogeneous Poisson model"}

if(model=="Tor"){gtitle="Toroidal shift model"}

if(model=="HetP"){gtitle="Heterogeneous Poisson model"}

# observed graphs

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Codispersion of abundance of",spe,"observed")))

print( ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") + stat\_contour(aes(x=xx,y=yy,z=Codispersion),binwidth=binwidth)+ggtitle(paste("Codispersion of abundance of",spe,"observed")) )

# null model graphs

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=null\_mean))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Mean codispersion of abundance of",spe,gtitle)))

print( ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=null\_mean))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") + stat\_contour(aes(x=xx,y=yy,z=null\_mean),binwidth=binwidth)+ggtitle(paste("Mean codispersion of abundance of",spe,gtitle)) )

# Significance and direction graph

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=diff))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Observed minus mean expected \n codispersion values for",spe,gtitle)))

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=diff))+scale\_fill\_gradientn(colours=rainbow(7),limits=c(-1, 1))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Observed minus mean expected \n codispersion values for",spe,gtitle)))

# P-value graph

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Significance of observed Codispersion values \n for",spe,gtitle)))

print( ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value))+scale\_fill\_gradientn(colours=c("#FF6666","#0000FF"),limits=c(0,0.5))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") + stat\_contour(aes(x=xx,y=yy,z=P.value),binwidth=binwidth)+ggtitle(paste("Significance of observed Codispersion values \n for",spe,gtitle)))

# P-value category graph

my.cols <- c("steelblue3","firebrick3")

if(levels(out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

print(ggplot(out.df,aes(xx,yy))+geom\_tile(aes(fill=P.value.cat))+scale\_fill\_manual(values=my.cols)+scale\_colour\_discrete(name="P.value.cat",limits=c(0,1))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Significance of observed Codispersion values \n for",spe,gtitle,"\n (alpha=5%, two sided test)")))

}

# Function to return a data frame with the null model comparison results

codisp.compare <- function(null.input.ary,CoDisp\_obs,round=FALSE){

out.df <- CoDisp\_obs # observed Codispersion result df

for(i in 1:length(null.input.ary[,1,1])) { # loop through each cell

nsims <- length(null.input.ary[1,1,])

obser <- out.df$Codispersion[i] # observed codispersion value

expec <- null.input.ary[i,3,]

prop.greater.than <- length(which(expec>obser))/nsims

prop.less.than <- length(which(expec<obser))/nsims

out.df$P.value[i]<-min(prop.greater.than,prop.less.than)

} # end cell loop

out.df$null\_mean <- apply(null.input.ary[,3,],MARGIN=1,mean) # calculate mean codispersion value for each cell from the array of null model results

out.df$diff <- out.df$Codispersion-out.df$null\_mean # observed minus expected

out.df$P.value.cat <- factor(ifelse(out.df$P.value<0.025,"Sig.","Non-sig.")) # significance at alpha=0.05

if(round==TRUE){ # for printing table of results

out.df$xx <- round(out.df$xx,1)

out.df$yy <- round(out.df$yy,1)

out.df$Codispersion <- round(out.df$Codispersion,3)

out.df$P.value <- round(out.df$P.value,3)

out.df$null\_mean <- round(out.df$null\_mean,3)

out.df$diff <- round(out.df$diff,3)

}

return(out.df)

}

##################################

### GRAPHING

##################################

#### Graphing function for ViewPort Grid graphics

vplayout <- function(x,y) { viewport(layout.pos.row=x,layout.pos.col=y) }

#### gglot theme options

t1<-theme(

plot.background = element\_blank(),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(size=.4),

axis.text = element\_text(colour="black",size=20,angle=0),

axis.title = element\_text(colour="black",size=20),

legend.key = element\_blank(),

legend.title = element\_text(colour="black",size=14),

legend.text = element\_text(colour="black",size=14),

plot.margin = unit(c(.2,.2,.2,.2),"lines"),

panel.margin = unit(.2,"lines"),

plot.background = element\_rect(fill=NA)

)

t1.no.leg <-theme(

plot.background = element\_blank(),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(size=.4),

axis.text = element\_text(colour="black",size=20,angle=0),

axis.title = element\_text(colour="black",size=20),

legend.text = element\_text(colour="black",size=18),

legend.position="none",

plot.margin = unit(c(.5,.2,.2,.2),"lines"),

panel.margin = unit(.2,"lines"),

plot.background = element\_rect(fill=NA)

#axis.title.x = element\_blank(),

#axis.title.y = element\_blank()

)

t1.unscaled.leg <- theme(

plot.background = element\_blank(),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(size=.4),

axis.text.y = element\_text(colour="black",size=20,angle=0),

axis.text.x = element\_text(colour="black",size=20,angle=0,hjust=1),

axis.title = element\_text(colour="black",size=20),

legend.key = element\_blank(),

legend.title = element\_blank(),

legend.text = element\_text(colour="black",size=20),

plot.margin = unit(c(.5,.2,.2,.2),"lines"),

panel.margin = unit(.2,"lines"),

plot.background = element\_rect(fill=NA)

)

t1.no.leg\_lab.20 <-theme(

plot.background = element\_blank(),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(size=.4),

axis.text = element\_text(colour="black",size=20,angle=0),

axis.title = element\_blank(),

legend.position="none",

plot.margin = unit(c(.5,.2,.2,.2),"lines"),

panel.margin = unit(.2,"lines"),

plot.background = element\_rect(fill=NA)

)

t1.no.lab.20pt <-theme(

plot.background = element\_blank(),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(size=.4),

axis.text = element\_text(colour="black",size=20,angle=0),

axis.title = element\_text(colour="black",size=20),

legend.key = element\_blank(),

#legend.title = element\_text(colour="black",size=20),

legend.title = element\_blank(),

legend.text = element\_text(colour="black",size=20),

plot.margin = unit(c(.2,.2,.2,.2),"lines"),

panel.margin = unit(.2,"lines"),

plot.background = element\_rect(fill=NA),

axis.title.x = element\_blank(),

axis.title.y = element\_blank()

)

## Function to generate variograms and cross variograms for the two geo.data objects used in codispersion analysis (observed patterns)

# labels is a two element vector used for labelling the graphs

# e.g. labels=c("species1","species2")

cross.variog.fn <- function(geodata1,geodata2,lab=missing(lab)){

Obs\_graphs <- vector(mode="list",length=3) # create empty object to store graphs

D1.dat <- data.frame(X=geodata1$coords[,1],Y=geodata1$coords[,2],D1=geodata1$data) # put geodata object into a dataframe

D2.dat <- data.frame(X=geodata2$coords[,1],Y=geodata2$coords[,2],D2=geodata2$data)

# Plot the observed raster patterns

g1 <- ggplot(D1.dat, aes(x=X, y=Y, size=D1))+geom\_point(colour="black", fill="steelblue2", shape=21)+coord\_fixed(ratio=1)

g2 <- ggplot(D2.dat, aes(x=X, y=Y, size=D2))+geom\_point(colour="black", fill="#4dac26", shape=21)+coord\_fixed(ratio=1)

## Plot the variograms and cross variogram

v.dat <- data.frame(x=geodata1$coords[,1],y=geodata1$coords[,2],dat1=scale(geodata1$data),dat2=scale(geodata2$data))

g <- gstat(id="D1", formula=dat1~1, locations=~x+y, data = v.dat)

g <- gstat(g, id="D2", formula=dat2~1, locations=~x+y, data = v.dat)

v <- variogram(g, cutoff=(min((max(v.dat$x)-min(v.dat$x)),(max(v.dat$y)-min(v.dat$y)))\*0.67), cross=TRUE) # 2/3 the min. of the two plot dimensions

g3 <- ggplot(v,aes(x=dist,y=gamma,group=id,colour=id))+geom\_line(lwd=2) + labs(x="Distance (m)",y = "Semivariance")

if(missing(lab)==FALSE){ # put labels on the graphs

Obs\_graphs[[1]] <- g1 + scale\_size\_continuous(name=lab[1])

Obs\_graphs[[2]] <- g2 + scale\_size\_continuous(name=lab[2])

Obs\_graphs[[3]] <- g3 + scale\_colour\_discrete(labels=c(paste(lab[1],"vs.",lab[2]),lab[2],lab[1])) + theme(legend.title=element\_blank())

}

if(missing(lab)==TRUE){ # don't put a label on the legend

Obs\_graphs[[1]] <- g1+t1.no.leg\_lab

Obs\_graphs[[2]] <- g2+t1.no.leg\_lab

Obs\_graphs[[3]] <- g3+t1.no.leg

}

return(Obs\_graphs)

} # end of function

# Function to print a codispersion graph using the CoDisp output object

print.CoDisp <- function(CoDisp.obj=CoDisp.obj,scaled=c("TRUE","FALSE"),contours=c("TRUE","FALSE"),binwidth=binwidth,input=input,gtitle=gtitle){

if(scaled=="FALSE"){

# print(ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab(expression(h[1]))+ylab(expression(h[2]))+ggtitle(paste("Codispersion of",input,gtitle)))

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")+ggtitle(paste("Codispersion of",input,gtitle))

}

if(scaled=="TRUE"){

if(contours=="TRUE"){

# print(ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+t1+xlab(expression(h[1]))+ylab(expression(h[2]))

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+t1+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") + stat\_contour(aes(x=xx,y=yy,z=Codispersion),binwidth=binwidth)+ggtitle(paste("Codispersion of",input,gtitle))

}

if(contours=="FALSE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+t1+xlab(expression(h[1]))+ylab(expression(h[2]))+ggtitle(paste("Codispersion of",input,gtitle))

}

} # end of scaled

return(g1)

} # end of function

# Function to print a codispersion graph using the CoDisp output object

# With plain output (no labels)

print.CoDisp.plain <- function(CoDisp.obj=CoDisp.obj,scaled=TRUE,contours=TRUE,labels=TRUE,legend=TRUE,binwidth=binwidth){

if(labels=="TRUE"){

if(scaled=="FALSE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1)+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)")

}

if(scaled=="TRUE"){

if(contours=="TRUE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+xlab("Spatial lag in X (m)")+ylab("Spatial lag in Y (m)") + stat\_contour(aes(x=xx,y=yy,z=Codispersion),binwidth=binwidth)

}

if(contours=="FALSE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+xlab(expression(h[1]))+ylab(expression(h[2]))

}

} # end of scaled

} # end of labels == TRUE

if(labels=="FALSE"){

if(scaled=="FALSE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=rainbow(7))+coord\_fixed(ratio=1) +xlab(NULL) +ylab(NULL)

}

if(scaled=="TRUE"){

if(contours=="TRUE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+ stat\_contour(aes(x=xx,y=yy,z=Codispersion),binwidth=binwidth)+xlab(NULL) +ylab(NULL)

}

if(contours=="FALSE"){

g1 <- ggplot(CoDisp.obj,aes(xx,yy))+geom\_tile(aes(fill=Codispersion))+scale\_fill\_gradientn(colours=c("#0000FF","#FFFFFF","#FF6666"),limits=c(-1,1))+coord\_fixed(ratio=1)+xlab(NULL) +ylab(NULL)

}

} # end of scaled

} # end of labels == FALSE

if(legend=="TRUE") { g1 <- g1 + t1.unscaled.leg }

if(legend=="FALSE") { g1 <- g1 + t1.no.leg }

return(g1)

} # end of function

##################################

### SIMULATING PATTERNS

##################################

##############

##### Function to simulate co-occurrence patterns on grids

# grid.points=20 # scale: grain of grid

# sp1.pattern="CSR" # The desired pattern for species 1: CSR,decreasing.x,increasing.x,decreasing.xy,bivariate.normal

# sp2.pattern="CSR" # The desired pattern for species 2: CSR,decreasing.x,increasing.x,decreasing.xy,bivariate.normal

# xmin=0 # Dimensions of the plot area, e.g. 300 x 300m

# xmax=300

# ymin=0

# ymax=300

# Print=TRUE # Whether you want plots of the spatial patterns or not

copp.fn <- function(grid.points = grid.points,sp1.pattern = c("CSR","decreasing.x","increasing.x","decreasing.y","decreasing.xy","increasing.xy","bivariate.normal"),sp2.pattern = c("CSR","decreasing.x","increasing.x","increasing.y","decreasing.xy","increasing.xy","bivariate.normal","inc.x.dec.y","dec.x.inc.y"),xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,Print=c("TRUE","FALSE")){

# 1. Create an empty list to add output geodata objects

copp.sim <- vector("list",2)

# 2. Set up underlying grid coordinates

X <- seq(from=xmin,to=xmax-grid.points,by=grid.points)

Y <- seq(from=ymin,to=ymax-grid.points,by=grid.points)

gridxy <- expand.grid(x=X,y=Y)

# 3a. Create a set of quadrat abundance values for sp1 based on the 'sp1.pattern' argument

if(sp1.pattern=="CSR"){Z <- rnorm(length(gridxy$x),mean=30,sd=10) }

if(sp1.pattern=="decreasing.x"){Z <- 1+(rev(2\*gridxy$x+5))/10 }

if(sp1.pattern=="increasing.x"){Z <- 1+(2\*gridxy$x+5)/10 }

if(sp1.pattern=="decreasing.y"){Z <- 1+(rev(2\*gridxy$y+5))/10 }

if(sp1.pattern=="decreasing.xy"){

Z <- 1+rev(((gridxy$x+1)^2+(gridxy$y+1)^2)/3000) # (x-u)^2+(y-v)^2

}

if(sp1.pattern=="increasing.xy"){

Z <- 1+((gridxy$x+2)^2+(gridxy$y+1)^2)/3000 # (x-u)^2+(y-v)^2

}

if(sp1.pattern=="bivariate.normal"){

Z <- 300\*bivariate(((gridxy$x-min(gridxy$x))/(max(gridxy$x)-min(gridxy$x))\*4)-2,((gridxy$y-min(gridxy$y))/(max(gridxy$y)-min(gridxy$y))\*4)-2)

} # bivariate normal

if(sp1.pattern=="inc.x.dec.y"){

Z <- 1+((gridxy$x+2)^2+(rev(gridxy$y+1))^2)/3000 # (x-u)^2+(y-v)^2

}

if(sp1.pattern=="dec.x.inc.y"){

Z <- 1+((rev(gridxy$x+2))^2+(gridxy$y+1)^2)/3000 # (x-u)^2+(y-v)^2

}

# 3b. Add data to the output list as a geodata object

copp.sp1.df <- data.frame(x=gridxy$x,y=gridxy$y,Z=jitter(Z,mean(Z)/5))

copp.sim[[1]] <- as.geodata(copp.sp1.df,coords.col=1:2,data.col=3)

# 4a. Create a set of quadrat abundance values for sp2 based on the 'sp1.pattern' argument

if(sp2.pattern=="CSR"){Z <- rnorm(length(gridxy$x),mean=30,sd=10) }

if(sp2.pattern=="decreasing.x"){Z <- 1+(rev(2\*gridxy$x+5))/10 }

if(sp2.pattern=="decreasing.x"){Z <- 1+(rev(2\*gridxy$x+5))/10 }

if(sp2.pattern=="increasing.x"){Z <- 1+(2\*gridxy$x+5)/10 }

if(sp2.pattern=="increasing.y"){Z <- 1+(2\*gridxy$y+5)/10 }

if(sp2.pattern=="decreasing.xy"){

Z <- 1+rev(((gridxy$x+1)^2+(gridxy$y+1)^2)/3000) # (x-u)^2+(y-v)^2

}

if(sp2.pattern=="increasing.xy"){

Z <- 1+((gridxy$x+2)^2+(gridxy$y+1)^2)/3000 # (x-u)^2+(y-v)^2

}

if(sp2.pattern=="bivariate.normal"){

Z <- 300\*bivariate(((gridxy$x-min(gridxy$x))/(max(gridxy$x)-min(gridxy$x))\*4)-2,((gridxy$y-min(gridxy$y))/(max(gridxy$y)-min(gridxy$y))\*4)-2)

} # bivariate normal

if(sp2.pattern=="inc.x.dec.y"){

Z <- 1+((gridxy$x+2)^2+(rev(gridxy$y+1))^2)/3000 # (x-u)^2+(y-v)^2

}

if(sp2.pattern=="dec.x.inc.y"){

Z <- 1+((rev(gridxy$x+2))^2+(gridxy$y+1)^2)/3000 # (x-u)^2+(y-v)^2

}

# 4b. Add data to the output list as a geodata object

copp.sp2.df <- data.frame(x=gridxy$x,y=gridxy$y,Z=jitter(Z,mean(Z)/10))

copp.sim[[2]] <- as.geodata(copp.sp2.df,coords.col=1:2,data.col=3)

# 5. Print map of points if desired

if(Print=="TRUE"){

print(qplot(x, y, data=copp.sp1.df, size=Z,main=paste("sp1.pattern = ",sp1.pattern))+ theme\_bw())

print(qplot(x, y, data=copp.sp2.df, size=Z,main=paste("sp2.pattern = ",sp2.pattern))+ theme\_bw())

} # end Print loop

# 6. Output the list of geodata objects

return(copp.sim)

} # end function

######################################################

######################################################

##### End of source file code

######################################################

######################################################