R code for the simulations and the analysis of the Luquillo and Tyson datasets. Note that to run the first three sections of code that analyze the (1) simulated datasets, (2) the LFDP data, (3) the TRCP 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 illustrating the use of

### the codispersion function

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

source("CoDisp\_functions.R")

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

## Simulated anisotropic spp-environment patterns for the MEE paper

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

for(k in 1:10){ # Ten types of spp-environment patterns

#### APP - 1500 trees with varying environmental gradients in 300 x 300m plots

gtitles <- vector("numeric")

app.ls <- vector("list")

###########----------------Basal Area

if(k==1){

title="CSR\_Thomas"

###### Quantitative marks = env gradient vs. DBH

#### Thomas = ppp of species

## CSR = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[1] = "CSR\_mpp\_Thomas\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[2] = "CSR\_mpp\_Thomas\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[3] = "CSR\_mpp\_Thomas\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[4] = "CSR\_mpp\_Thomas\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[5] = "CSR\_mpp\_Thomas\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[6] = "CSR\_mpp\_Thomas\_bivnorm"

}

if(k==2){

title="Uniform\_Thomas"

## uniform = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[1] = "uniform\_mpp\_Thomas\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[2] = "uniform\_mpp\_Thomas\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[3] = "uniform\_mpp\_Thomas\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[4] = "uniform\_mpp\_Thomas\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[5] = "uniform\_mpp\_Thomas\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="Thomas",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[6] = "uniform\_mpp\_Thomas\_bivnorm"

}

if(k==3){

title="decx\_Thomas"

## decreasing.x = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[1] = "decx\_mpp\_Thomas\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[2] = "decx\_mpp\_Thomas\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[3] = "decx\_mpp\_Thomas\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[4] = "decx\_mpp\_Thomas\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[5] = "decx\_mpp\_Thomas\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="Thomas",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[6] = "decx\_mpp\_Thomas\_bivnorm"

}

if(k==4){

title="decxy\_Thomas"

###### Quantitative marks = env gradient vs. DBH

#### Thomas = ppp of species

## decreasing.xy = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[1] = "decxy\_mpp\_Thomas\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[2] = "decxy\_mpp\_Thomas\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[3] = "decxy\_mpp\_Thomas\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[4] = "decxy\_mpp\_Thomas\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[5] = "decxy\_mpp\_Thomas\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="Thomas",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[6] = "decxy\_mpp\_Thomas\_bivnorm"

}

if(k==5){

title="bivnorm\_Thomas"

###### Quantitative marks = env gradient vs. DBH

#### Thomas = ppp of species

## bivariate.normal = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[1] = "bivnorm\_mpp\_Thomas\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[2] = "bivnorm\_mpp\_Thomas\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[3] = "bivnorm\_mpp\_Thomas\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[4] = "bivnorm\_mpp\_Thomas\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[5] = "bivnorm\_mpp\_Thomas\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="Thomas",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE); gtitles[6] = "env\_bivnorm\_mpp\_Thomas\_bivnorm"

}

if(k==6){

title="CSR\_CSR"

###### Quantitative marks = env gradient vs. DBH

#### CSR = ppp of species

## CSR = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=200,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[1] = "CSR\_mpp\_CSR\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[2] = "CSR\_mpp\_CSR\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[3] = "CSR\_mpp\_CSR\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[4] = "CSR\_mpp\_CSR\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[5] = "CSR\_mpp\_CSR\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[6] = "CSR\_mpp\_CSR\_bivnorm"

}

if(k==7){

title="Uniform\_CSR"

#### CSR = ppp of species

## uniform = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=200,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[1] = "uniform\_mpp\_CSR\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[2] = "uniform\_mpp\_CSR\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[3] = "uniform\_mpp\_CSR\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[4] = "uniform\_mpp\_CSR\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[5] = "uniform\_mpp\_CSR\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="uniform",pattern.method="quant.marks",ppp.model="CSR",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[6] = "uniform\_mpp\_CSR\_bivnorm"

}

if(k==8){

title="decx\_CSR"

#### CSR = ppp of species

## decreasing.x = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=200,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[1] = "decx\_mpp\_CSR\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[2] = "decx\_mpp\_CSR\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[3] = "decx\_mpp\_CSR\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[4] = "decx\_mpp\_CSR\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[5] = "decx\_mpp\_CSR\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="decreasing.x",pattern.method="quant.marks",ppp.model="CSR",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[6] = "decx\_mpp\_CSR\_bivnorm"

}

if(k==9){

title="decxy\_CSR"

###### Quantitative marks = env gradient vs. DBH

#### CSR = ppp of species

## decreasing.xy = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[1] = "decxy\_mpp\_CSR\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[2] = "decxy\_mpp\_CSR\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[3] = "decxy\_mpp\_CSR\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[4] = "decxy\_mpp\_CSR\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[5] = "decxy\_mpp\_CSR\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="decreasing.xy",pattern.method="quant.marks",ppp.model="CSR",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[6] = "decxy\_mpp\_CSR\_bivnorm"

}

if(k==10){

title="bivnorm\_CSR"

###### Quantitative marks = env gradient vs. DBH

#### CSR = ppp of species

## bivariate.normal = environmental gradient

app.ls[[1]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[1] = "bivnorm\_mpp\_CSR\_rand"

app.ls[[2]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[2] = "bivnorm\_mpp\_CSR\_decx"

app.ls[[3]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.x",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[3] = "bivnorm\_mpp\_CSR\_incx"

app.ls[[4]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="decreasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[4] = "bivnorm\_mpp\_CSR\_decxy"

app.ls[[5]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="increasing.xy",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[5] = "bivnorm\_mpp\_CSR\_incxy"

app.ls[[6]] <- app.sim.fn(grid.points=5,env.func="bivariate.normal",pattern.method="quant.marks",ppp.model="CSR",marks.method="bivariate.normal",xmin=0,xmax=300,ymin=0,ymax=300,lambda=300,minmark=1,maxmark=80,ntrees=1500,Print=FALSE); gtitles[6] = "bivnorm\_mpp\_CSR\_bivnorm"

}

# 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

Means.df <- data.frame(sim=1:6,mean\_CoDisp=NA,sd\_CoDisp=NA)

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

print(date())

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

Graphs\_ls <- vector(mode="list",length=5) # empty list for output graphs

## Extract the data

app.sims <- app.ls[[i]]

## Convert to point patterns to geodata objects

geo.env <- ppp.to.geoR.fn(app.sims[[1]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="mean.mark")

geo.sp <- ppp.to.geoR.fn(app.sims[[2]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="total.ba")

## Plot the observed patterns

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

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

Graphs\_ls[[1]] <- ggplot(env.dat, aes(x=X, y=Y, size=env))+geom\_point(colour="black", fill="steelblue4", shape=21)+t1.no.leg\_lab

Graphs\_ls[[2]] <- ggplot(sp.dat, aes(x=X, y=Y, size=BA))+geom\_point(colour="black", fill="#4dac26", shape=21)+t1.no.leg\_lab

## Plot the variograms and cross variogram

dat <- data.frame(geo.env$coords,env=scale(geo.env$data),sp=scale(geo.sp$data))

g <- gstat(id="env", formula=env~1, locations=~qx+qy, data = dat)

g <- gstat(g, id="sp", formula=sp~1, locations=~qx+qy, data = dat)

v <- variogram(g, cutoff=250, cross=TRUE) # half the min. of the two plot dimensions

#plot(v)

Graphs\_ls[[3]] <- ggplot(v,aes(x=dist,y=gamma,group=id,colour=id))+geom\_line(lwd=2)+t1.no.leg + labs(x="Distance (m)",y = "Semivariance")

## Run Codispersion Analysis

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

## Graph the output

Graphs\_ls[[4]] <- print.CoDisp.plain(CoDisp\_sim[[1]],scaled=FALSE)

Graphs\_ls[[5]] <- print.CoDisp.plain(CoDisp\_sim[[1]],scaled=TRUE,contours=TRUE,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 objects

nam=(paste("CoDisp\_app\_ba",k,i,sep="\_"))

assign(nam,CoDisp\_sim)

## Save the output objects

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

assign(nam,Graphs\_ls)

} # end i loop

nam=(paste("CoDisp\_app\_ba\_means",k,sep="\_"))

assign(nam,Means.df)

} # end k loop

save.image("mpp\_env\_ba.RData")

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

### Graph output

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

load("mpp\_env\_ba.RData")

source("CoDisp\_functions.R")

# Graph species patterns for FIGURE

png("Graphs\_SppEnvSims\_Figure.png",width=1400,height=1100)

grid.newpage()

pushViewport(viewport(layout=grid.layout(6,6)))

for(i in 1:6){

if(i==1){out <- Graphs\_ls\_CSR\_CSR\_1 }

if(i==2){out <- Graphs\_ls\_Uniform\_CSR\_4 }

if(i==3){out <- Graphs\_ls\_decx\_CSR\_2 }

if(i==4){out <- Graphs\_ls\_decx\_Thomas\_2 }

if(i==5){out <- Graphs\_ls\_decxy\_CSR\_3 }

if(i==6){out <- Graphs\_ls\_bivnorm\_CSR\_5 }

if(i==2){ out[[1]] <- out[[1]]+geom\_point(aes(size=0.1)) } # plot uniform point sizes

g1 <- out[[1]]+coord\_fixed(ratio=1)+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab # Env raster graph

g2 <- out[[2]]+coord\_fixed(ratio=1)+xlab(NULL)+ylab(NULL)+t1.no.leg\_lab # Spp raster graph

g3 <- out[[3]]+xlab(NULL)+ylab(NULL)+ylim(c(-1.2,2.1))+t1.fat.margins # Variogram graph

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

g5 <- out[[5]]+t1.fat.margins\_no.leg+xlab(NULL)+ylab(NULL) # Scaled CoDisp 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,4:5))

print(g5, vp=vplayout(i,5:6))

}

dev.off()

# Graph all spp-env patterns for APPENDIX

png("Graphs\_bivnorm\_CSR.png",width=1800,height=1200)

grid.newpage()

pushViewport(viewport(layout=grid.layout(6,7))) # 6 rows by 7 columns

vplayout <- function(x,y)

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

for(i in 1:6){

if(i==1){out <- Graphs.ls\_bivnorm\_CSR\_1}

if(i==2){out <- Graphs.ls\_bivnorm\_CSR\_2}

if(i==3){out <- Graphs.ls\_bivnorm\_CSR\_3}

if(i==4){out <- Graphs.ls\_bivnorm\_CSR\_4}

if(i==5){out <- Graphs.ls\_bivnorm\_CSR\_5}

if(i==6){out <- Graphs.ls\_bivnorm\_CSR\_6}

g1 <- out[[1]]

g2 <- out[[2]]

g3 <- out[[3]]

g4 <- out[[4]]

g5 <- out[[5]]

## Print the graphs to the layout

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

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

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

print(g4, vp=vplayout(i,4:5))

print(g5, vp=vplayout(i,6:7))

}

dev.off()

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

### Supplementary Online material: Null model analysis of CSR, CSR

### to obtain error rates

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

load("mpp\_env\_ba.RData")

source("CoDisp\_functions.R")

# Create data ppp objects and convert to geo.data objects for analysis

# Thomas

thomas.ppp <- app.sim.fn(grid.points=20,env.func="CSR",pattern.method="quant.marks",ppp.model="Thomas",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,minmark=1,maxmark=80,kappa=20,sigma=0.05,mu=10,ntrees=1500,Print=FALSE)

thomas.ppp[[1]]$x <- thomas.ppp[[1]]$x+0.001 # shift the values off the lattice

thomas.ppp[[1]]$y <- thomas.ppp[[1]]$y+0.001

thomas.env.geo <- ppp.to.geoR.fn(thomas.ppp[[1]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="mean.mark")

thomas.spp.geo <- ppp.to.geoR.fn(thomas.ppp[[2]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="total.ba")

# CSR

csr.ppp <- app.sim.fn(grid.points=20,env.func="CSR",pattern.method="quant.marks",ppp.model="CSR",marks.method="random",xmin=0,xmax=300,ymin=0,ymax=300,lambda=200,minmark=1,maxmark=80,ntrees=1500,Print=FALSE)

csr.ppp[[1]]$x <- csr.ppp[[1]]$x+0.001 # shift the values off the lattice

csr.ppp[[1]]$y <- csr.ppp[[1]]$y+0.001

CSR.env.geo <- ppp.to.geoR.fn(csr.ppp[[1]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="mean.mark")

CSR.spp.geo <- ppp.to.geoR.fn(csr.ppp[[2]],quad.size=20,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,method="total.ba")

## Plot species grid plots

# Thomas

t1.dat <- data.frame(xx=thomas.env.geo$coords[,1],yy=thomas.env.geo$coords[,2],AB=thomas.env.geo$data,Distribution="Thomas",Type="Environment")

t2.dat <- data.frame(xx=thomas.spp.geo$coords[,1],yy=thomas.spp.geo$coords[,2],AB=thomas.spp.geo$data,Distribution="Thomas",Type="Species")

# CSR

c1.dat <- data.frame(xx=CSR.env.geo$coords[,1],yy=CSR.env.geo$coords[,2],AB=CSR.env.geo$data,Distribution="CSR",Type="Environment")

c2.dat <- data.frame(xx=CSR.spp.geo$coords[,1],yy=CSR.spp.geo$coords[,2],AB=CSR.spp.geo$data,Distribution="CSR",Type="Species")

(p1 <- ggplot(t1.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg\_lab)

(p2 <- ggplot(t2.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="#4dac26", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg\_lab)

(p3 <- ggplot(c1.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="steelblue4", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg\_lab)

(p4 <- ggplot(c2.dat, aes(x=xx, y=yy, size=AB))+geom\_point(colour="black", fill="#4dac26", shape=21)+ coord\_fixed(ratio = 1)+t1.no.leg\_lab)

png("CoDisp\_Sims\_TypeIerr\_MEE.png",width=600,height=600)

grid.newpage()

pushViewport(viewport(layout=grid.layout(2,2))) # 6 rows by 7 columns

vplayout <- function(x,y)

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

## Print the graphs to the layout

print(p1, vp=vplayout(1,1))

print(p2, vp=vplayout(1,2))

print(p3, vp=vplayout(2,1))

print(p4, vp=vplayout(2,2))

dev.off()

# 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 = 49

# randomise species patterns using Homogeneous Poisson (CSR), RLM and Toroidal shift null models

# Thomas

HomP\_Thomas.ls <- ppp.null.fn(thomas.ppp[[2]],nsim=nsim,model="HomP",marks=TRUE)

RLM\_Thomas.ls <- ppp.null.fn(thomas.ppp[[2]],nsim=nsim,model="RLM",marks=TRUE)

Tor\_Thomas.ls <- ppp.null.fn(thomas.ppp[[2]],nsim=nsim,model="Tor",marks=TRUE)

# CSR

HomP\_CSR.ls <- ppp.null.fn(csr.ppp[[2]],nsim=nsim,model="HomP",marks=TRUE)

RLM\_CSR.ls <- ppp.null.fn(csr.ppp[[2]],nsim=nsim,model="RLM",marks=TRUE)

Tor\_CSR.ls <- ppp.null.fn(csr.ppp[[2]],nsim=nsim,model="Tor",marks=TRUE)

### make empty lists to hold null model results

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

CoDisp\_Thomas\_RLM <- vector("list",nsim)

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

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

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

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

### Convert null ppp objects into geo.data objects

# Thomas

geo.HomP.Thomas <- lapply(HomP\_Thomas.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.RLM.Thomas <- lapply(RLM\_Thomas.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.Tor.Thomas <- lapply(Tor\_Thomas.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

# CSR

geo.HomP.CSR <- lapply(HomP\_CSR.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.RLM.CSR <- lapply(RLM\_CSR.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.Tor.CSR <- lapply(Tor\_CSR.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

# Run codispersion analysis on null model data

for(j in 1:nsim){

# Thomas

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

CoDisp\_Thomas\_HomP[[j]] <- codisp.fn(thomas.env.geo,geo.HomP.Thomas[[j]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Thomas\_RLM, j",j)) # Random labelling

CoDisp\_Thomas\_RLM[[j]] <- codisp.fn(thomas.env.geo,geo.RLM.Thomas[[j]],k=k,max.window.size=max.window.size)

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

CoDisp\_Thomas\_Tor[[j]] <- codisp.fn(thomas.env.geo,geo.Tor.Thomas[[j]],k=k,max.window.size=max.window.size)

# CSR

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

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

print(paste("CoDisp\_CSR\_RLM, j",j)) # Random labelling

CoDisp\_CSR\_RLM[[j]] <- codisp.fn(CSR.env.geo,geo.RLM.CSR[[j]],k=k,max.window.size=max.window.size)

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

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

} # end simulations j loop

# Convert output lists to array objects

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

CoDisp\_Thomas\_RLM\_ary <- list2ary(CoDisp\_Thomas\_RLM)

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

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

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

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

### Run the codispersion analysis on the observed patterns

CoDisp\_Thomas <- codisp.fn(thomas.env.geo,thomas.spp.geo,k=k,max.window.size=max.window.size)

CoDisp\_CSR <- codisp.fn(CSR.env.geo,CSR.spp.geo,k=k,max.window.size=max.window.size)

save.image("Simulated\_MEE\_CSR&CSR\_null\_49.RData")

load("mpp\_env\_ba.RData")

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

source("CoDisp\_functions.R")

### Make comparisons between observed and null expectation

# Thomas

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

Thomas\_RLM\_out.df <- codisp.compare(CoDisp\_Thomas\_RLM\_ary,CoDisp\_Thomas[[1]],round=TRUE)

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

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

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

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

# CSR

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

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

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

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

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

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

### Draw graphs

# Thomas

### HomP

# Observed minus expected CoDispersion value graph

( g1 <- ggplot(Thomas\_HomP\_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(Thomas\_HomP\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g2 <- ggplot(Thomas\_HomP\_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) )

### RLM

# Observed minus expected CoDispersion value graph

( g3 <- ggplot(Thomas\_RLM\_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(Thomas\_RLM\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g4 <- ggplot(Thomas\_RLM\_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

( g5 <- ggplot(Thomas\_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.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(Thomas\_Tor\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g6 <- ggplot(Thomas\_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) )

# CSR

### HomP

# Observed minus expected CoDispersion value graph

( g7 <- ggplot(CSR\_HomP\_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(CSR\_HomP\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g8 <- ggplot(CSR\_HomP\_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) )

### RLM

# Observed minus expected CoDispersion value graph

( g9 <- ggplot(CSR\_RLM\_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(CSR\_RLM\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g10 <- ggplot(CSR\_RLM\_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

( g11 <- ggplot(CSR\_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.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(CSR\_Tor\_out.df$P.value.cat)[1]=="Sig."){my.cols <- c("firebrick3") }

( g12 <- ggplot(CSR\_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) )

png("CoDisp\_Sims\_TypeIerr\_output\_MEE\_0.025\_199.png",width=1400,height=250)

grid.newpage()

pushViewport(viewport(layout=grid.layout(2,12)))

vplayout <- function(x,y)

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

## Print the graphs to the layout

print(g1, vp=vplayout(1,1:2))

print(g2, vp=vplayout(1,3:4))

print(g3, vp=vplayout(1,5:6))

print(g4, vp=vplayout(1,7:8))

print(g5, vp=vplayout(1,9:10))

print(g6, vp=vplayout(1,11:12))

print(g7, vp=vplayout(2,1:2))

print(g8, vp=vplayout(2,3:4))

print(g9, vp=vplayout(2,5:6))

print(g10, vp=vplayout(2,7:8))

print(g11, vp=vplayout(2,9:10))

print(g12, vp=vplayout(2,11:12))

dev.off()

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

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

##### Analyse LFDP species on environmental gradients

##### using Codispersion analysis.

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

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

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

source("CoDisp\_functions.R")

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

#### Read in the datasets

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

# PuertoRico-LFDP data

LFDP\_full <- read.csv("http://luq.lternet.edu/sites/default/files/data/LFDP\_Census3.csv",header=TRUE)

dim(LFDP\_full[is.na(LFDP\_full$GX)==F,])

LFDP.new <- LFDP\_full[ order(LFDP\_full[,"GX"]), ]

head(LFDP.new)

names(LFDP.new) <- c("tag","stemtag","sp","quadrat","subquadrat","gx","gy","dbh","status","hom.m","date","census","status.and.codes")

temp <- strsplit(as.character(LFDP.new$status.and.codes),split=";")

levels(factor(LFDP.new$status.and.codes))

LFDP.new$codes1 <- sapply(temp,function(x)x[1])

LFDP.new$codes2 <- sapply(temp,function(x)x[2])

LFDP.new$codes3 <- sapply(temp,function(x)x[3])

LFDP.new$codes4 <- sapply(temp,function(x)x[4])

LFDP.new$codes5 <- sapply(temp,function(x)x[5])

LFDP\_sub2 <- LFDP.new[LFDP.new$codes1=="MAIN"&LFDP.new$codes2=="A",]

LFDP\_sub <- subset(LFDP\_sub2,select=c(sp,gx,gy,dbh))

LFDP <- LFDP\_sub[complete.cases(LFDP\_sub),]

LFDP$sp <- factor(LFDP$sp)

LFDP$dbh <- LFDP$dbh/10

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

unique(dat$sp) # species list

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

rm(LFDP\_full,LFDP.new,LFDP\_sub2,LFDP\_sub,LFDP)

# calculate basal area

dat$ba <- basal.area.fn(dat$dbh)

# set plot dimensions

plot(dat$gx,dat$gy)

max(dat$gx)

max(dat$gy)

xmin=0; xmax=320; ymin=0; ymax=500

# LFDP environmental data on 20 x 20m grid

env <- read.csv("http://luq.lternet.edu/sites/default/files/data/LFDPEnvironment20.csv", header=TRUE)

env$qx <- (env$Col-1)\*20

env$qy <- (env$Row-1)\*20

env <- env[ order(env[,"qx"]), ]

plot(geo.elev <- as.geodata(env,coords.col=19:20,data.col=12))

plot(geo.slope <- as.geodata(env,coords.col=19:20,data.col=15))

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

## Extract target species:

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

nspp <- 4

spp.list <- c("CASARB","PREMON","CECSCH","DACEXC")

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

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

## Analysis using rasters

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

for(i in 1:length(spp.list)){

ppp.dat <- ppp.ls[[i]]

Graphs\_ls <- vector(mode="list",length=9) # empty list for output graphs

Means.df <- data.frame(env=c("Elevation","Slope"),mean\_CoDisp=NA,sd\_CoDisp=NA)

spe <- spp.list[i]

geo.obs.ba <- ppp.to.geoR.fn(ppp.dat,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

# Graph the data

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

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

sp.dat <- data.frame(X=geo.obs.ba$coords[,1],Y=geo.obs.ba$coords[,2],BA=geo.obs.ba$data)

Graphs\_ls[[1]] <- ggplot(elev.dat, aes(x=X, y=Y, size=env))+geom\_point(colour="black", fill="steelblue2", shape=21)+t1.no.leg\_lab

Graphs\_ls[[2]] <- ggplot(slope.dat, aes(x=X, y=Y, size=env))+geom\_point(colour="black", fill="steelblue2", shape=21)+t1.no.leg\_lab

Graphs\_ls[[3]] <- ggplot(sp.dat, aes(x=X, y=Y, size=BA))+geom\_point(colour="black", fill="#4dac26", shape=21)+t1.no.leg\_lab

## Plot the variograms and cross variograms

# Elevation

ddat <- data.frame(geo.elev$coords,env=scale(geo.elev$data),sp=scale(geo.obs.ba$data))

g <- gstat(id="env", formula=env~1, locations=~qx+qy, data = ddat)

g <- gstat(g, id="sp", formula=sp~1, locations=~qx+qy, data = ddat)

v <- variogram(g, cutoff=(min(xmax,ymax)\*0.67), cross=TRUE)

Graphs\_ls[[4]] <- ggplot(v,aes(x=dist,y=gamma,group=id,colour=id))+geom\_line(lwd=2)+t1.no.leg + labs(x="Distance (m)",y = "Semivariance")

# Slope

ddat <- data.frame(geo.slope$coords,env=scale(geo.slope$data),sp=scale(geo.obs.ba$data))

g <- gstat(id="env", formula=env~1, locations=~qx+qy, data = ddat)

g <- gstat(g, id="sp", formula=sp~1, locations=~qx+qy, data = ddat)

v <- variogram(g, cutoff=(min(xmax,ymax)\*0.67), cross=TRUE)

Graphs\_ls[[5]] <- ggplot(v,aes(x=dist,y=gamma,group=id,colour=id))+geom\_line(lwd=2)+t1.no.leg + labs(x="Distance (m)",y = "Semivariance")

#### run the codispersion analysis

binwidth=0.1

k=c(20,20,20)

max.window.size=320/4

# observed data BA

print(paste("Elevation",spe))

CoDisp\_elev <- codisp.fn(geo.obs.ba,geo.elev,k=k,max.window.size=max.window.size)

print(paste("Slope",spe))

CoDisp\_slope <- codisp.fn(geo.obs.ba,geo.slope,k=k,max.window.size=max.window.size)

## Graph the output

Graphs\_ls[[6]] <- print.CoDisp.plain(CoDisp\_elev[[1]],scaled=FALSE,labels=FALSE)

Graphs\_ls[[7]] <- print.CoDisp.plain(CoDisp\_elev[[1]],scaled=TRUE,contours=TRUE,binwidth=binwidth,labels=FALSE)

Graphs\_ls[[8]] <- print.CoDisp.plain(CoDisp\_slope[[1]],scaled=FALSE,labels=FALSE)

Graphs\_ls[[9]] <- print.CoDisp.plain(CoDisp\_slope[[1]],scaled=TRUE,contours=TRUE,binwidth=binwidth,labels=FALSE)

## Calculate the mean values

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

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

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

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

## Save the output objects

nam=paste("CoDisp\_elev",spe,sep="\_")

assign(nam,CoDisp\_elev)

nam=paste("CoDisp\_slope",spe,sep="\_")

assign(nam,CoDisp\_slope)

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

assign(nam,Graphs\_ls)

nam=(paste("Means.df",spe,sep="\_"))

assign(nam,Means.df)

} # end i loop

save.image("LFDP\_spp\_env\_obs\_basalarea.RData")

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

## Observed graphs for paper

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

load("LFDP\_spp\_env\_obs\_basalarea.RData")

source("CoDisp\_functions.R")

png("LFDP\_Variograms\_Figure.png",width=860,height=1200)

grid.newpage()

pushViewport(viewport(layout=grid.layout(5,3)))

# Print observed elevation and slope raster plots

print(Graphs\_ls\_CASARB[[1]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(1,2))

print(Graphs\_ls\_CASARB[[2]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(1,3))

# g1 <- out[[1]] # env var 1: elevation

# g2 <- out[[2]] # env var 2: slope

# g3 <- out[[3]] # BA of species

# g4 <- out[[4]] # first variogram plot: elevation

# g5 <- out[[5]] # second variogram plot: slope

for(i in 1:4){

if(i==1){out <- Graphs\_ls\_CASARB

print(out[[3]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(2,1))

print(out[[4]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(2,2))

print(out[[5]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(2,3))

}

if(i==2){out <- Graphs\_ls\_CECSCH

print(out[[3]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(3,1))

print(out[[4]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(3,2))

print(out[[5]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(3,3))

}

if(i==3){out <- Graphs\_ls\_DACEXC

print(out[[3]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(4,1))

print(out[[4]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(4,2))

print(out[[5]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(4,3))

}

if(i==4){out <- Graphs\_ls\_PREMON

print(out[[3]]+coord\_fixed(ratio=1)+scale\_x\_continuous(breaks=c(0,100,300))+scale\_y\_continuous(breaks=c(0,100,300,500)), vp=vplayout(5,1))

print(out[[4]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(5,2))

print(out[[5]]+t1.no.lab+ylim(c(-0.2,1.25)), vp=vplayout(5,3))

}

}

dev.off()

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

## Null model comparison

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

load("LFDP\_spp\_env\_obs\_basalarea.RData")

source("CoDisp\_functions.R")

spp.list <- c("CASARB","PREMON","CECSCH","DACEXC")

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]])

}

k=c(20,20,20)

max.window.size=320/4

nsim=199

HomP\_sp1.ls <- ppp.null.fn(ppp.ls[[1]],nsim=nsim,model=c("HomP"))

HomP\_sp2.ls <- ppp.null.fn(ppp.ls[[2]],nsim=nsim,model=c("HomP"))

HomP\_sp3.ls <- ppp.null.fn(ppp.ls[[3]],nsim=nsim,model=c("HomP"))

HomP\_sp4.ls <- ppp.null.fn(ppp.ls[[4]],nsim=nsim,model=c("HomP"))

RLM\_sp1.ls <- ppp.null.fn(ppp.ls[[1]],nsim=nsim,model=c("RLM"))

RLM\_sp2.ls <- ppp.null.fn(ppp.ls[[2]],nsim=nsim,model=c("RLM"))

RLM\_sp3.ls <- ppp.null.fn(ppp.ls[[3]],nsim=nsim,model=c("RLM"))

RLM\_sp4.ls <- ppp.null.fn(ppp.ls[[4]],nsim=nsim,model=c("RLM"))

Tor\_sp1.ls <- ppp.null.fn(ppp.ls[[1]],nsim=nsim,model=c("Tor"))

Tor\_sp2.ls <- ppp.null.fn(ppp.ls[[2]],nsim=nsim,model=c("Tor"))

Tor\_sp3.ls <- ppp.null.fn(ppp.ls[[3]],nsim=nsim,model=c("Tor"))

Tor\_sp4.ls <- ppp.null.fn(ppp.ls[[4]],nsim=nsim,model=c("Tor"))

# Generate null model geodata objects

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

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

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

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

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

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

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

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

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

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

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

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

CoDisp\_HomP\_sp1elev <- vector("list",nsim)

CoDisp\_HomP\_sp2elev <- vector("list",nsim)

CoDisp\_HomP\_sp3elev <- vector("list",nsim)

CoDisp\_HomP\_sp4elev <- vector("list",nsim)

CoDisp\_HomP\_sp1slope <- vector("list",nsim)

CoDisp\_HomP\_sp2slope <- vector("list",nsim)

CoDisp\_HomP\_sp3slope <- vector("list",nsim)

CoDisp\_HomP\_sp4slope <- vector("list",nsim)

CoDisp\_RLM\_sp1elev <- vector("list",nsim)

CoDisp\_RLM\_sp2elev <- vector("list",nsim)

CoDisp\_RLM\_sp3elev <- vector("list",nsim)

CoDisp\_RLM\_sp4elev <- vector("list",nsim)

CoDisp\_RLM\_sp1slope <- vector("list",nsim)

CoDisp\_RLM\_sp2slope <- vector("list",nsim)

CoDisp\_RLM\_sp3slope <- vector("list",nsim)

CoDisp\_RLM\_sp4slope <- vector("list",nsim)

CoDisp\_Tor\_sp1elev <- vector("list",nsim)

CoDisp\_Tor\_sp2elev <- vector("list",nsim)

CoDisp\_Tor\_sp3elev <- vector("list",nsim)

CoDisp\_Tor\_sp4elev <- vector("list",nsim)

CoDisp\_Tor\_sp1slope <- vector("list",nsim)

CoDisp\_Tor\_sp2slope <- vector("list",nsim)

CoDisp\_Tor\_sp3slope <- vector("list",nsim)

CoDisp\_Tor\_sp4slope <- vector("list",nsim)

for (i in 1:nsim) {

# Run codispersion analysis on null model data

# HomP elevation

print(paste("CoDisp\_HomP\_sp1elev, i =",i))

CoDisp\_HomP\_sp1elev[[i]] <- codisp.fn(geo.elev,geo.HomP.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp2elev, i =",i))

CoDisp\_HomP\_sp2elev[[i]] <- codisp.fn(geo.elev,geo.HomP.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp3elev, i =",i))

CoDisp\_HomP\_sp3elev[[i]] <- codisp.fn(geo.elev,geo.HomP.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp4elev, i =",i))

CoDisp\_HomP\_sp4elev[[i]] <- codisp.fn(geo.elev,geo.HomP.sp4[[i]],k=k,max.window.size=max.window.size)

# HomP slope

print(paste("CoDisp\_HomP\_sp1slope, i =",i))

CoDisp\_HomP\_sp1slope[[i]] <- codisp.fn(geo.slope,geo.HomP.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp2slope, i =",i))

CoDisp\_HomP\_sp2slope[[i]] <- codisp.fn(geo.slope,geo.HomP.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp3slope, i =",i))

CoDisp\_HomP\_sp3slope[[i]] <- codisp.fn(geo.slope,geo.HomP.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_HomP\_sp4slope, i =",i))

CoDisp\_HomP\_sp4slope[[i]] <- codisp.fn(geo.slope,geo.HomP.sp4[[i]],k=k,max.window.size=max.window.size)

# RLM elevation

print(paste("CoDisp\_RLM\_sp1elev, i =",i))

CoDisp\_RLM\_sp1elev[[i]] <- codisp.fn(geo.elev,geo.RLM.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp2elev, i =",i))

CoDisp\_RLM\_sp2elev[[i]] <- codisp.fn(geo.elev,geo.RLM.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp3elev, i =",i))

CoDisp\_RLM\_sp3elev[[i]] <- codisp.fn(geo.elev,geo.RLM.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp4elev, i =",i))

CoDisp\_RLM\_sp4elev[[i]] <- codisp.fn(geo.elev,geo.RLM.sp4[[i]],k=k,max.window.size=max.window.size)

# RLM slope

print(paste("CoDisp\_RLM\_sp1slope, i =",i))

CoDisp\_RLM\_sp1slope[[i]] <- codisp.fn(geo.slope,geo.RLM.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp2slope, i =",i))

CoDisp\_RLM\_sp2slope[[i]] <- codisp.fn(geo.slope,geo.RLM.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp3slope, i =",i))

CoDisp\_RLM\_sp3slope[[i]] <- codisp.fn(geo.slope,geo.RLM.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp4slope, i =",i))

CoDisp\_RLM\_sp4slope[[i]] <- codisp.fn(geo.slope,geo.RLM.sp4[[i]],k=k,max.window.size=max.window.size)

# Tor elevation

print(paste("CoDisp\_Tor\_sp1elev, i =",i))

CoDisp\_Tor\_sp1elev[[i]] <- codisp.fn(geo.elev,geo.Tor.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp2elev, i =",i))

CoDisp\_Tor\_sp2elev[[i]] <- codisp.fn(geo.elev,geo.Tor.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp3elev, i =",i))

CoDisp\_Tor\_sp3elev[[i]] <- codisp.fn(geo.elev,geo.Tor.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp4elev, i =",i))

CoDisp\_Tor\_sp4elev[[i]] <- codisp.fn(geo.elev,geo.Tor.sp4[[i]],k=k,max.window.size=max.window.size)

# Tor slope

print(paste("CoDisp\_Tor\_sp1slope, i =",i))

CoDisp\_Tor\_sp1slope[[i]] <- codisp.fn(geo.slope,geo.Tor.sp1[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp2slope, i =",i))

CoDisp\_Tor\_sp2slope[[i]] <- codisp.fn(geo.slope,geo.Tor.sp2[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp3slope, i =",i))

CoDisp\_Tor\_sp3slope[[i]] <- codisp.fn(geo.slope,geo.Tor.sp3[[i]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp4slope, i =",i))

CoDisp\_Tor\_sp4slope[[i]] <- codisp.fn(geo.slope,geo.Tor.sp4[[i]],k=k,max.window.size=max.window.size)

} # end i loop

# Convert output lists to array objects

CoDisp\_HomP\_sp1elev.ary <- list2ary(CoDisp\_HomP\_sp1elev)

CoDisp\_HomP\_sp2elev.ary <- list2ary(CoDisp\_HomP\_sp2elev)

CoDisp\_HomP\_sp3elev.ary <- list2ary(CoDisp\_HomP\_sp3elev)

CoDisp\_HomP\_sp4elev.ary <- list2ary(CoDisp\_HomP\_sp4elev)

CoDisp\_HomP\_sp1slope.ary <- list2ary(CoDisp\_HomP\_sp1slope)

CoDisp\_HomP\_sp2slope.ary <- list2ary(CoDisp\_HomP\_sp2slope)

CoDisp\_HomP\_sp3slope.ary <- list2ary(CoDisp\_HomP\_sp3slope)

CoDisp\_HomP\_sp4slope.ary <- list2ary(CoDisp\_HomP\_sp4slope)

CoDisp\_RLM\_sp1elev.ary <- list2ary(CoDisp\_RLM\_sp1elev)

CoDisp\_RLM\_sp2elev.ary <- list2ary(CoDisp\_RLM\_sp2elev)

CoDisp\_RLM\_sp3elev.ary <- list2ary(CoDisp\_RLM\_sp3elev)

CoDisp\_RLM\_sp4elev.ary <- list2ary(CoDisp\_RLM\_sp4elev)

CoDisp\_RLM\_sp1slope.ary <- list2ary(CoDisp\_RLM\_sp1slope)

CoDisp\_RLM\_sp2slope.ary <- list2ary(CoDisp\_RLM\_sp2slope)

CoDisp\_RLM\_sp3slope.ary <- list2ary(CoDisp\_RLM\_sp3slope)

CoDisp\_RLM\_sp4slope.ary <- list2ary(CoDisp\_RLM\_sp4slope)

CoDisp\_Tor\_sp1elev.ary <- list2ary(CoDisp\_Tor\_sp1elev)

CoDisp\_Tor\_sp2elev.ary <- list2ary(CoDisp\_Tor\_sp2elev)

CoDisp\_Tor\_sp3elev.ary <- list2ary(CoDisp\_Tor\_sp3elev)

CoDisp\_Tor\_sp4elev.ary <- list2ary(CoDisp\_Tor\_sp4elev)

CoDisp\_Tor\_sp1slope.ary <- list2ary(CoDisp\_Tor\_sp1slope)

CoDisp\_Tor\_sp2slope.ary <- list2ary(CoDisp\_Tor\_sp2slope)

CoDisp\_Tor\_sp3slope.ary <- list2ary(CoDisp\_Tor\_sp3slope)

CoDisp\_Tor\_sp4slope.ary <- list2ary(CoDisp\_Tor\_sp4slope)

save.image("LFDP\_elev&slope\_spp1\_4\_Null\_199.RData")

load("LFDP\_spp\_env\_obs\_basalarea\_20x20.RData")

load("LFDP\_elev&slope\_spp1\_4\_Null\_199.RData")

source("CoDisp\_functions.R")

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

## Null model figure for paper

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

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

binwidth = 0.1

n.spp = 4 # number of species

n.vars = 2 # number of variables

n.mods = 3 # number of null models

png("LFDP\_CodispNull\_Figure.png",width=1600,height=(n.spp\*n.vars\*120))

grid.newpage()

pushViewport(viewport(layout=grid.layout((n.spp\*n.vars),14)))

# Loop through each species and plot the observed CoDisp graphs

obs.codisp.ls <- list(CoDisp\_elev\_CASARB[[1]],CoDisp\_slope\_CASARB[[1]],CoDisp\_elev\_PREMON[[1]],CoDisp\_slope\_PREMON[[1]],CoDisp\_elev\_CECSCH[[1]],CoDisp\_slope\_CECSCH[[1]],CoDisp\_elev\_DACEXC[[1]],CoDisp\_slope\_DACEXC[[1]])

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

# Observed graphs

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

print(g1, vp=vplayout(i,1:2))

}

codisp.obj.ls <- list( CoDisp\_elev\_CASARB[[1]],CoDisp\_elev\_CASARB[[1]],CoDisp\_elev\_CASARB[[1]],CoDisp\_slope\_CASARB[[1]],CoDisp\_slope\_CASARB[[1]],CoDisp\_slope\_CASARB[[1]],CoDisp\_elev\_PREMON[[1]],CoDisp\_elev\_PREMON[[1]],CoDisp\_elev\_PREMON[[1]],CoDisp\_slope\_PREMON[[1]],CoDisp\_slope\_PREMON[[1]],CoDisp\_slope\_PREMON[[1]],CoDisp\_elev\_CECSCH[[1]],CoDisp\_elev\_CECSCH[[1]],CoDisp\_elev\_CECSCH[[1]],CoDisp\_slope\_CECSCH[[1]],CoDisp\_slope\_CECSCH[[1]],CoDisp\_slope\_CECSCH[[1]],CoDisp\_elev\_DACEXC[[1]],CoDisp\_elev\_DACEXC[[1]],CoDisp\_elev\_DACEXC[[1]],CoDisp\_slope\_DACEXC[[1]],CoDisp\_slope\_DACEXC[[1]],CoDisp\_slope\_DACEXC[[1]] )

null.ary.ls <- list( CoDisp\_HomP\_sp1elev.ary,CoDisp\_RLM\_sp1elev.ary,CoDisp\_Tor\_sp1elev.ary,CoDisp\_HomP\_sp1slope.ary,CoDisp\_RLM\_sp1slope.ary,CoDisp\_Tor\_sp1slope.ary,CoDisp\_HomP\_sp2elev.ary,CoDisp\_RLM\_sp2elev.ary,CoDisp\_Tor\_sp2elev.ary,CoDisp\_HomP\_sp2slope.ary,CoDisp\_RLM\_sp2slope.ary,CoDisp\_Tor\_sp2slope.ary,CoDisp\_HomP\_sp3elev.ary,CoDisp\_RLM\_sp3elev.ary,CoDisp\_Tor\_sp3elev.ary,CoDisp\_HomP\_sp3slope.ary,CoDisp\_RLM\_sp3slope.ary,CoDisp\_Tor\_sp3slope.ary,CoDisp\_HomP\_sp4elev.ary,CoDisp\_RLM\_sp4elev.ary,CoDisp\_Tor\_sp4elev.ary,CoDisp\_HomP\_sp4slope.ary,CoDisp\_RLM\_sp4slope.ary,CoDisp\_Tor\_sp4slope.ary )

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

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

row.loop.no <- rep(1:(n.spp\*n.vars),each=n.mods)

g2.col.loop.no <- rep(c(3,7,11),(n.spp\*n.vars))

g3.col.loop.no <- rep(c(5,9,13),(n.spp\*n.vars))

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()

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

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

##### Analyse Tyson species on environmental gradients

##### using Codispersion analysis.

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

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

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

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

#### Load required functions

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

source("CoDisp\_functions.R")

library(labdsv)

library(vegan)

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

#### Read in data

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

dat <- read.csv("Tyson\_spp.csv")

env <- read.csv("Tyson\_env.csv")

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

#### Specify plot coordinates

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

xmin=0; xmax=500; ymin=0; ymax=440

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

#### Generate ppp objects from species data

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

spp.list <- c("fracar","linben","quealb","querub","quevel")

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]])

}

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

#### Create geodata objects from PC axes

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

plot(geo.pc1 <- as.geodata(env,coords.col=1:2,data.col=3))

plot(geo.pc2 <- as.geodata(env,coords.col=1:2,data.col=4))

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

## Analysis using rasters

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

# Select environmental dataset

geo.env <- geo.pc1; env.var = "PC1"

geo.env <- geo.pc2; env.var = "PC2"

for(i in 1:length(spp.list)){

ppp.dat <- ppp.ls[[i]]

Graphs\_ls <- vector(mode="list",length=5) # empty list for output graphs

Means.df <- data.frame(env=c(env.var),mean\_CoDisp=NA,sd\_CoDisp=NA)

spe <- spp.list[i]

#plot(ppp.dat,main=spe)

geo.obs.ba <- ppp.to.geoR.fn(ppp.dat,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

# Graph the data

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

sp.dat <- data.frame(X=geo.obs.ba$coords[,1],Y=geo.obs.ba$coords[,2],BA=geo.obs.ba$data)

Graphs\_ls[[1]] <- ggplot(env.dat, aes(x=X, y=Y, size=env))+geom\_point(colour="black", fill="steelblue2", shape=21)+coord\_fixed(ratio=1)+t1.no.leg\_lab # ENV

Graphs\_ls[[2]] <- ggplot(sp.dat, aes(x=X, y=Y, size=BA))+geom\_point(colour="black", fill="#4dac26", shape=21)+coord\_fixed(ratio=1)+t1.no.leg\_lab # SP

## Plot the variograms and cross variograms

# ENV

ddat <- data.frame(geo.env$coords,env=scale(geo.env$data),sp=scale(geo.obs.ba$data))

g <- gstat(id="env", formula=env~1, locations=~qx+qy, data = ddat)

g <- gstat(g, id="sp", formula=sp~1, locations=~qx+qy, data = ddat)

v <- variogram(g, cutoff=(min(xmax,ymax)\*0.67), cross=TRUE)

Graphs\_ls[[3]] <- ggplot(v,aes(x=dist,y=gamma,group=id,colour=id))+geom\_line(lwd=2)+t1.no.leg + labs(x="Distance (m)",y = "Semivariance")

#### run the codispersion analysis

binwidth=0.1

k=c(20,20,20)

max.window.size=min(xmax,ymax)/4

# observed data BA

print(paste("Env",spe))

CoDisp\_env <- codisp.fn(geo.obs.ba,geo.env,k=k,max.window.size=max.window.size)

## Graph the output

Graphs\_ls[[4]] <- print.CoDisp.plain(CoDisp\_env[[1]],scaled=FALSE)

Graphs\_ls[[5]] <- print.CoDisp.plain(CoDisp\_env[[1]],scaled=TRUE,binwidth=binwidth)

## Calculate the mean values

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

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

## Save the output objects

nam=paste("CoDisp\_env",env.var,spe,sep="\_")

assign(nam,CoDisp\_env)

nam=(paste("Graphs\_ls",env.var,spe,sep="\_"))

assign(nam,Graphs\_ls)

nam=(paste("Means.df",env.var,spe,sep="\_"))

assign(nam,Means.df)

} # end i loop

save.image("TY\_spp\_PC1&2\_obs\_basalarea\_20x20.RData")

## Plot results graphs

load("TY\_spp\_PC1&2\_obs\_basalarea\_20x20.RData")

source("CoDisp\_functions.R")

##########

## Figure for paper - Variograms

##########

load("TY\_spp\_PC1&2\_obs\_basalarea\_20x20.RData")

source("CoDisp\_functions.R")

png("TY\_Variograms\_Figure.png",width=1000,height=1350)

grid.newpage()

pushViewport(viewport(layout=grid.layout(6,3)))

# Print observed elevation and slope raster plots

print(Graphs\_ls\_PC1\_quealb[[1]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(1,2))

print(Graphs\_ls\_PC2\_quealb[[1]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(1,3))

for(i in 1:5){

if(i==1){out1 <- Graphs\_ls\_PC1\_fracar

out2 <- Graphs\_ls\_PC2\_fracar

print(out1[[2]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(2,1))

print(out1[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(2,2))

print(out2[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(2,3))

}

if(i==2){out1 <- Graphs\_ls\_PC1\_linben

out2 <- Graphs\_ls\_PC2\_linben

print(out1[[2]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(3,1))

print(out1[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(3,2))

print(out2[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(3,3))

}

if(i==3){out1 <- Graphs\_ls\_PC1\_quealb

out2 <- Graphs\_ls\_PC2\_quealb

print(out1[[2]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(4,1))

print(out1[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(4,2))

print(out2[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(4,3))

}

if(i==4){out1 <- Graphs\_ls\_PC1\_querub

out2 <- Graphs\_ls\_PC2\_querub

print(out1[[2]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(5,1))

print(out1[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(5,2))

print(out2[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(5,3))

}

if(i==5){out1 <- Graphs\_ls\_PC1\_quevel

out2 <- Graphs\_ls\_PC2\_quevel

print(out1[[2]]+scale\_x\_continuous(breaks=c(0,200,400))+scale\_y\_continuous(breaks=c(0,200,400)), vp=vplayout(6,1))

print(out1[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(6,2))

print(out2[[3]]+coord\_cartesian(ylim=c(-0.6,1.7))+scale\_y\_continuous(breaks=seq(-0.6, 1.7, 0.5))+t1.no.lab.20pt, vp=vplayout(6,3))

}

}

dev.off()

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

## Null model comparison: analysis using rasters

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

spp.list <- c("fracar","linben","quealb","querub","quevel")

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]])

}

k=c(20,20,20)

max.window.size=320/4

nsim = 199

for(i in 1:length(spp.list)) {

spe <- spp.list[i]

# Generate null point patterns

HomP\_sp.ls <- ppp.null.fn(ppp.ls[[i]],nsim=nsim,model="HomP")

RLM\_sp.ls <- ppp.null.fn(ppp.ls[[i]],nsim=nsim,model="RLM")

Tor\_sp.ls <- ppp.null.fn(ppp.ls[[i]],nsim=nsim,model="Tor")

# Generate null model geodata objects

geo.HomP.sp <- lapply(HomP\_sp.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.RLM.sp <- lapply(RLM\_sp.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

geo.Tor.sp <- lapply(Tor\_sp.ls,ppp.to.geoR.fn,quad.size=20,xmin,xmax,ymin,ymax,method="total.ba")

CoDisp\_PC1\_HomP <- vector("list",nsim)

CoDisp\_PC1\_RLM <- vector("list",nsim)

CoDisp\_PC1\_Tor <- vector("list",nsim)

CoDisp\_PC2\_HomP <- vector("list",nsim)

CoDisp\_PC2\_RLM <- vector("list",nsim)

CoDisp\_PC2\_Tor <- vector("list",nsim)

for(j in 1:nsim){

# Run codispersion analysis on null model data

print(paste("CoDisp\_HomP\_sp, i j",i,j))

CoDisp\_PC1\_HomP[[j]] <- codisp.fn(geo.pc1,geo.HomP.sp[[j]],k=k,max.window.size=max.window.size)

CoDisp\_PC2\_HomP[[j]] <- codisp.fn(geo.pc2,geo.HomP.sp[[j]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_RLM\_sp, i j",i,j))

CoDisp\_PC1\_RLM[[j]] <- codisp.fn(geo.pc1,geo.RLM.sp[[j]],k=k,max.window.size=max.window.size)

CoDisp\_PC2\_RLM[[j]] <- codisp.fn(geo.pc2,geo.RLM.sp[[j]],k=k,max.window.size=max.window.size)

print(paste("CoDisp\_Tor\_sp, i j",i,j))

CoDisp\_PC1\_Tor[[j]] <- codisp.fn(geo.pc1,geo.Tor.sp[[j]],k=k,max.window.size=max.window.size)

CoDisp\_PC2\_Tor[[j]] <- codisp.fn(geo.pc2,geo.Tor.sp[[j]],k=k,max.window.size=max.window.size)

} # end simulations loop

# Convert output lists to array objects

CoDisp\_PC1\_HomP\_ary <- list2ary(CoDisp\_PC1\_HomP)

CoDisp\_PC1\_RLM\_ary <- list2ary(CoDisp\_PC1\_RLM)

CoDisp\_PC1\_Tor\_ary <- list2ary(CoDisp\_PC1\_Tor)

CoDisp\_PC2\_HomP\_ary <- list2ary(CoDisp\_PC2\_HomP)

CoDisp\_PC2\_RLM\_ary <- list2ary(CoDisp\_PC2\_RLM)

CoDisp\_PC2\_Tor\_ary <- list2ary(CoDisp\_PC2\_Tor)

## Save the output objects

nam=paste("CoDisp\_PC1\_HomP\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC1\_HomP\_ary)

nam=paste("CoDisp\_PC1\_RLM\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC1\_RLM\_ary)

nam=paste("CoDisp\_PC1\_Tor\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC1\_Tor\_ary)

nam=paste("CoDisp\_PC2\_HomP\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC2\_HomP\_ary)

nam=paste("CoDisp\_PC2\_RLM\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC2\_RLM\_ary)

nam=paste("CoDisp\_PC2\_Tor\_ary",spe,sep="\_")

assign(nam,CoDisp\_PC2\_Tor\_ary)

} # end species loop

save.image("TY\_PC1PC2\_null\_199.RData")

load("TY\_PC1PC2\_null\_199.RData")

source("CoDisp\_functions.R")

binwidth = 0.1

dataset="TY"

gtitle = "TY"

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

## Figures for paper

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

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

binwidth = 0.1

n.spp = 5 # number of species

n.vars = 2 # number of variables

n.mods = 3 # number of null models

png("TY\_CodispNull\_Tor\_Figure.png",width=1600,height=(n.spp\*n.vars\*120))

grid.newpage()

pushViewport(viewport(layout=grid.layout((n.spp\*n.vars),14)))

# Loop through each species and plot the observed CoDisp graphs

obs.codisp.ls <- list(CoDisp\_env\_PC1\_fracar[[1]],CoDisp\_env\_PC2\_fracar[[1]],CoDisp\_env\_PC1\_linben[[1]],CoDisp\_env\_PC2\_linben[[1]],CoDisp\_env\_PC1\_quealb[[1]],CoDisp\_env\_PC2\_quealb[[1]],CoDisp\_env\_PC1\_querub[[1]],CoDisp\_env\_PC2\_querub[[1]],CoDisp\_env\_PC1\_quevel[[1]],CoDisp\_env\_PC2\_quevel[[1]])

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

# Observed graphs

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

print(g1, vp=vplayout(i,1:2))

}

codisp.obj.ls <- list(CoDisp\_env\_PC1\_fracar[[1]],CoDisp\_env\_PC1\_fracar[[1]],CoDisp\_env\_PC1\_fracar[[1]],CoDisp\_env\_PC2\_fracar[[1]],CoDisp\_env\_PC2\_fracar[[1]],CoDisp\_env\_PC2\_fracar[[1]],CoDisp\_env\_PC1\_linben[[1]],CoDisp\_env\_PC1\_linben[[1]],CoDisp\_env\_PC1\_linben[[1]],CoDisp\_env\_PC2\_linben[[1]],CoDisp\_env\_PC2\_linben[[1]],CoDisp\_env\_PC2\_linben[[1]],CoDisp\_env\_PC1\_quealb[[1]],CoDisp\_env\_PC1\_quealb[[1]],CoDisp\_env\_PC1\_quealb[[1]],CoDisp\_env\_PC2\_quealb[[1]],CoDisp\_env\_PC2\_quealb[[1]],CoDisp\_env\_PC2\_quealb[[1]],CoDisp\_env\_PC1\_querub[[1]],CoDisp\_env\_PC1\_querub[[1]],CoDisp\_env\_PC1\_querub[[1]],CoDisp\_env\_PC2\_querub[[1]],CoDisp\_env\_PC2\_querub[[1]],CoDisp\_env\_PC2\_querub[[1]],CoDisp\_env\_PC1\_quevel[[1]],CoDisp\_env\_PC1\_quevel[[1]],CoDisp\_env\_PC1\_quevel[[1]],CoDisp\_env\_PC2\_quevel[[1]],CoDisp\_env\_PC2\_quevel[[1]],CoDisp\_env\_PC2\_quevel[[1]])

null.ary.ls <- list( CoDisp\_PC1\_HomP\_ary\_fracar,CoDisp\_PC1\_RLM\_ary\_fracar,CoDisp\_PC1\_Tor\_ary\_fracar,CoDisp\_PC2\_HomP\_ary\_fracar,CoDisp\_PC2\_RLM\_ary\_fracar,CoDisp\_PC2\_Tor\_ary\_fracar,CoDisp\_PC1\_HomP\_ary\_linben,CoDisp\_PC1\_RLM\_ary\_linben,CoDisp\_PC1\_Tor\_ary\_linben,CoDisp\_PC2\_HomP\_ary\_linben,CoDisp\_PC2\_RLM\_ary\_linben,CoDisp\_PC2\_Tor\_ary\_linben,CoDisp\_PC1\_HomP\_ary\_quealb,CoDisp\_PC1\_RLM\_ary\_quealb,CoDisp\_PC1\_Tor\_ary\_quealb,CoDisp\_PC2\_HomP\_ary\_quealb,CoDisp\_PC2\_RLM\_ary\_quealb,CoDisp\_PC2\_Tor\_ary\_quealb,CoDisp\_PC1\_HomP\_ary\_querub,CoDisp\_PC1\_RLM\_ary\_querub,CoDisp\_PC1\_Tor\_ary\_querub,CoDisp\_PC2\_HomP\_ary\_querub,CoDisp\_PC2\_RLM\_ary\_querub,CoDisp\_PC2\_Tor\_ary\_querub,CoDisp\_PC1\_HomP\_ary\_quevel,CoDisp\_PC1\_RLM\_ary\_quevel,CoDisp\_PC1\_Tor\_ary\_quevel,CoDisp\_PC2\_HomP\_ary\_quevel,CoDisp\_PC2\_RLM\_ary\_quevel,CoDisp\_PC2\_Tor\_ary\_quevel )

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

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

row.loop.no <- rep(1:(n.spp\*n.vars),each=n.mods)

g2.col.loop.no <- rep(c(3,7,11),(n.spp\*n.vars))

g3.col.loop.no <- rep(c(5,9,13),(n.spp\*n.vars))

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()

## stats for paper

round(tapply(dat$ba,dat$sp,sum),2)

round(range(CoDisp\_env\_PC1\_fracar[[1]]$Codispersion),2)

round(range(CoDisp\_env\_PC2\_fracar[[1]]$Codispersion),2)

round(range(CoDisp\_env\_PC1\_linben[[1]]$Codispersion),2)

round(range(CoDisp\_env\_PC2\_linben[[1]]$Codispersion),2)

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

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

##### Required functions for codispersion analysis

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

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

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

### 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

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

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

# Inputs are the null model input array object and the observed CoDisp result list

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()

)

t1.no.leg\_lab <-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)

)

## 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 anisotropic point patterns

# # dimensions of the plot

xmin=0

xmax=300

ymin=0

ymax=300

grid.points=5 # the distance between points on the underlying grid (must divide evenly into the plot dimensions)

env.func = "uniform"

ppp.model="Thomas" # or "CSR"

kappa=20

sigma=0.5

mu=10

lambda=200

pattern.method="abundance" # or "quant.marks"

marks.method="uniform"

minmark=1

maxmark=80

sp.pattern= "random" # "decreasing.x","increasing.x","decreasing.xy","increasing.xy","bivariate.normal" # the distribution pattern of the species

sp.maxab=15

ntrees = 1500 #the number of trees you want

app.sim.fn <- function(grid.points = grid.points,env.func = c("uniform","CSR","decreasing.x","increasing.x","decreasing.xy","increasing.xy","bivariate.normal"),pattern.method=c("quant.marks","abundance"),ppp.model=c("CSR","Thomas"),marks.method=c("random","decreasing.x","increasing.x","decreasing.xy","increasing.xy","bivariate.normal"),sp.pattern=c("random","decreasing.x","increasing.x","decreasing.xy","increasing.xy","bivariate.normal"),ntrees=ntrees,sp.maxab=sp.maxab,xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax,minmark=minmark,maxmark=maxmark,kappa=kappa,sigma=sigma,mu=mu,lambda=lambda,Print=c("TRUE","FALSE")){ # begin function

# 1. 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)

# 2. Create a set of marks to use as values for the environmental variable based on the 'env.func' argument

if(env.func=="uniform"){Z <- jitter(rep(50,(length(X)\*length(Y)))) }

if(env.func=="CSR"){Z <- rnorm(n=length(gridxy$x),mean=50,sd=15) }

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

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

if(env.func=="decreasing.xy"){

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

}

if(env.func=="increasing.xy"){

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

}

if(env.func=="bivariate.normal"){

Z <- 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

epp.df <- data.frame(x=gridxy$x,y=gridxy$y,Z=Z)

epp.sim <- as.ppp(epp.df,marks=Z,W=owin(c(xmin,xmax),c(ymin,ymax)))

#plot(epp.sim)

# 3. Marked point pattern

# 3a. Create a ppp of trees using the selected model

if(pattern.method=="quant.marks"){

if(ppp.model=="CSR"){ temp <- rpoispp(lambda=lambda,win=owin(c(xmin/100,xmax/100),c(ymin/100,ymax/100)))

mpp.sim <- temp[1:ntrees] } # generate 2000 trees in the plot

if(ppp.model=="Thomas"){ temp <- rThomas(kappa=kappa,sigma=sigma,mu=mu,win=owin(c(xmin/100,xmax/100),c(ymin/100,ymax/100)))

mpp.sim <- temp[1:ntrees] }# generate 2000 trees in the plt

# 3b. Assign the marks to the point pattern using the selected method of codispersion

if(marks.method=="random"){ mrks <- rtlnorm(mpp.sim$n,meanlog=log(maxmark/2),sdlog=log(maxmark/15),lower=minmark,upper=maxmark) } # generate a random set of marks drawn from a lognormal distribution

if(marks.method=="decreasing.x"){ temp <- -18\*mpp.sim$x+5

mrks <- temp+abs(min(temp)) }

if(marks.method=="increasing.x"){ mrks <- 18\*mpp.sim$x+5 }

if(marks.method=="decreasing.xy"){

temp <- -((mpp.sim$x+1)^2+(mpp.sim$y+1)^2)

mrks <- temp+abs(min(temp)) } # (x-u)^2+(y-v)^2

if(marks.method=="increasing.xy"){

mrks <- ((mpp.sim$x+1)^2+(mpp.sim$y+1)^2) } # (x-u)^2+(y-v)^2

if(marks.method=="bivariate.normal"){ mrks <- bivariate(((mpp.sim$x-min(mpp.sim$x))/(max(mpp.sim$x)-min(mpp.sim$x))\*3)-2,((mpp.sim$y-min(mpp.sim$y))/(max(mpp.sim$y)-min(mpp.sim$y))\*4)-2) } # bivariate.normal

mrks1 <- (mrks-min(mrks))/(max(mrks)-min(mrks)) # scale marks between 0 and 1

mpp.sim$marks <- minmark+mrks1/max(mrks1)\*(maxmark-minmark) # spread marks between max and min mark

mpp.sim$window <- owin(c(xmin,xmax),c(ymin,ymax)) # rescale window to metres

mpp.sim$x <- mpp.sim$x\*100 # rescale x and y values to metres

mpp.sim$y <- mpp.sim$y\*100

} # end quant.marks

# 4. Generate a species point pattern using the selected method

# 4a. First generate a grid of values in the selected pattern.

if(pattern.method=="abundance"){

if(sp.pattern=="random"){ ab <- runif(n=length(c(gridxy$x)),min=0,max=sp.maxab) }

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

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

if(sp.pattern=="decreasing.xy"){ ab <- 1+rev(((gridxy$x+1)^2+(gridxy$y+1)^2)/3000) } # (x-u)^2+(y-v)^2

if(sp.pattern=="increasing.xy"){ ab <- 1+((gridxy$x+2)^2+(gridxy$y+1)^2)/3000 } # (x-u)^2+(y-v)^2

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

ab <- 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

AB <- round(ab/max(ab)\*sp.maxab,0) # scale abundance to maximum number of individuals per grid cell

mpp.df <- data.frame(x=gridxy$x,y=gridxy$y,ab=AB)

mpp.sim <- as.ppp(mpp.df,marks=ab,W=owin(c(xmin,xmax),c(ymin,ymax)))

} # end abundance loop

# 5. Put both the environment ppp object and the species2 ppp object into an output list object

app.sim <- vector("list")

app.sim[[1]] <- epp.sim

app.sim[[2]] <- mpp.sim

# 6. Print map of points if desired

if(Print=="TRUE"){

par(mfrow=c(1,2))

(plot(epp.sim,main=paste("env.func =",env.func),cex.main=0.7))

if(pattern.method=="quant.marks"){

(plot(mpp.sim,main=paste("mrks =",marks.method,mpp.sim$n),cex.main=0.7)) }

if(pattern.method=="abundance"){

(plot(mpp.sim,main=paste("species =",sp.pattern,mpp.sim$n),cex.main=0.7)) }

} # end Print loop

return(app.sim)

} # end function

#app.sim <- app.sim.fn(grid.points=5,env.func="increasing.x",pattern.method="abundance",sp.pattern="increasing.x",sp.maxab=20,xmin=0,xmax=200,ymin=0,ymax=200,Print="TRUE")

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

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

##### End of source file code

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

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