**METHODS**

*Simulations to compare the performance of genetic distance metrics*

Although we were able to compare the results of resistance surface optimization using different metrics, and identify the ones with the highest fit, we cannot be certain that highest fit represents the most accurate metrics. Specifically, we cannot know for sure whether the highest fit indicates models that most appropriately assign values and ranks to different resistance surfaces. In order to evaluate the performance of different metrics more rigorously, we conducted spatially-explicit demo-genetic simulations using CDPOP (Landguth & Cushman, 2010) based on the lizard dataset (Beninde et al., 2016). In those simulations, we input resistance values for each level of the categorical variable and created a cost distance matrix based on this input landscape. Finally, we ran simulations where individual dispersal was constrained based on this cost distance matrix (Landguth & Cushman, 2010).

We respected the biology of *Podarcis* *muralis* in several aspects in the simulations. For example, we ranked the resistance values of different land cover categories meaningfully according to the general patterns found in the analysis of the empirical dataset. We constrained available habitat to land cover categories where they were originally sampled in large numbers (< median resistance value). Thus, we excluded river and very urbanized areas from carrying lizards in the lizards in the simulations, but lizards can potentially move through them. We also input reasonable parameters for reproductive ability (Ji & Braña, 2000) and density (Avery & Perkins, 1989; Barbault & Mou, 1988). We simulated 16 loci with 12 alleles each to mimic the original dataset (16 loci and an average of 11.8125 alleles per locus). We also subsampled the simulated lizards and only kept those present in the cells where the real lizards were caught (Sup. Mat. 3).

To evaluate how the ecological and demographic contexts affect inference we considered two factors in the simulations: isolation-by-distance (IBD) strength (Wright, 1943), and the maximum absolute resistances values. We calculated IBD by relating Loiselle’s kinship (Loiselle et al., 1995) to the logarithm of the geographic distances between simulated individuals (Rousset, 2000). The slope in the original dataset was -0.013. We considered strong IBD scenarios to be more than 30% lower (larger absolute value) and weak IBD scenarios to be more than 30% higher (smaller absolute value) than the observed value. We achieved varying IBD by changing dispersal ability through several CDPOP parameters (Landguth & Cushman, 2010) in our simulated lizards. We set the resistance values for five levels of a categorical land cover variable in a biologically meaningful way (e.g., river is a barrier), following previous literature (Beninde et al., 2016) and our previous results on the empirical dataset. We designed low resistance scenarios to present two weakly resisting land cover categories, and high resistance scenarios to present two strongly and two weakly resisting land cover categories. In high resistance scenarios, resistance values were set as 20 for non-urban impervious areas, 10 for forests and pastures, 70 for heavily urbanized areas, 80 for river, and 1 for transport infrastructure. In low resistance scenarios, resistance values were set as 1 for non-urban impervious areas, 1 for forests and pastures, 8 for heavily urbanized areas, 10 for river, and 1 for transport infrastructure. High IBD scenarios IBD slopes ranged from -0.019 to -0.025. Low IBD scenarios IBD slopes ranged from -0.007 to -0.009.

We increased the spatial grain of the landscape by a factor of three prior to simulations to achieve workable computation times for simulations and for subsequent optimizations. This should not affect our results much because spatial grain (i.e., cell size in our case) has been shown to have only limited effects on landscape genetics inference (Cushman & Landguth, 2010; McRae et al., 2008; McRae & Beier, 2007). This is especially true as we made sure to preserve the continuity of linear features in the landscape during the coarsening of the landscape grain (Sup. Mat. 2).

**SUPPLEMENTARY MATERIAL**

All the code and data are available from the following repository:

<https://github.com/julian-wittische/BestDistance>

The main pieces of code are also available below.

***1 - Functions used to simulate and explore simulation results***

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

########## Julian Wittische - November 2021 - Simulating connectivity ##########

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

### Acknowledgements:

# This is based on previous work by William Peterman and Kristopher Winiarski

# Thanks also to Dr Erin Landguth who welcomed me for a short visit in her lab

#' @author Bill Peterman/Julian Wittische

#' @description Function to run CDPOP from R

#'

#' @param CDPOP.py

#' @param sim\_name Name for simulation results. Defaults to 'output'

#' @param pts Spatial points object

#' @param sim\_dir Directory where simulation results will be written

#' @param resist\_rast Resistance surface

#' @param agefilename Path to age file. Default will create and use a non-overlapping generations file.

#' @param mcruns Default = 1

#' @param looptime Default = 401; Number generations to conduct simulation

#' @param output\_years Default = 50; Interval to write out simulation results

#' @param gridformat Default = 'genepop'; c('genepop', 'genalex', 'structure', 'cdpop')

#' @param cdclimgentime Default = 0. To initiate the CDClimate module, this is the generation/year that the next effective distance matrix will be read in at. You can specify multiple generations by separating each generation to read in the next cost distance matrix by ‘|’. Then in the following surface columns, a separate file can be given for each generation.

#' @param matemoveno Default = 2; Uses Inverse Square (1 / (Cost Distance^2)). This function gets rescaled to min and threshold of the inverse square cost distance.

#' @param matemoveparA Not used with inverse square movement

#' @param matemoveparB Not used with inverse square movement

#' @param matemoveparC Not used with inverse square movement

#' @param matemovethresh Default = 'max'; The maximum movement is the maximum resistance distance

#' @param output\_matedistance

#' @param sexans Default = 'N'; No selfing

#' @param Freplace Default = 'N'; Females mate without replacement

#' @param Mreplace Default = 'N'; Males mate without replacement

#' @param philopatry Default = 'N';

#' @param multiple\_paternity Default = 'N'; No philopatry of Males or Females

#' @param selfans Default = 'N'; No selfing

#' @param Fdispmoveno Default = NULL. Will be set equal to matemoveno

#' @param FdispmoveparA Not used with inverse square movement

#' @param FdispmoveparB Not used with inverse square movement

#' @param FdispmoveparC Not used with inverse square movement

#' @param Fdispmovethresh Default = NULL. Will be set to matemovethresh

#' @param Mdispmoveno Default = NULL. Will be set equal to matemoveno

#' @param MdispmoveparA Not used with inverse square movement

#' @param MdispmoveparB Not used with inverse square movement

#' @param MdispmoveparC Not used with inverse square movement

#' @param Mdispmovethresh Default = NULL. Will be set to matemovethresh

#' @param offno Default = 2; Poisson draw around ‘mean fecundity’

#' @param MeanFecundity Default = 5; Specifies mean fecundity in age variable file.

#' @param Femalepercent Default = 50

#' @param EqualsexratioBirth Default = 'N'

#' @param TwinningPercent Default = 0

#' @param popModel Default = 'exp'

#' @param r Population growth rate. No applicable when using exponential growth rate

#' @param K\_env Equal to the number of individuals simulated

#' @param subpopmortperc Default = 0|0|0|0; Not using subpopulation features

#' @param muterate Default = 0.0005

#' @param mutationtype Default = 'forward'

#' @param loci Default = 1000; For simulating SNP-like markers

#' @param intgenesans Default = 'random'; Random initiation of alleles

#' @param allefreqfilename Default = 'N'

#' @param alleles Default = 2; For simulating SNP-like markers

#' @param mtdna Default = 'N'

#' @param startGenes Default = 0;

#' @param cdevolveans Default = 'N'; No loci are under selection

#' @param startSelection Default = 0; No selection

#' @param betaFile\_selection Default = 'N'; No selection

#' @param epistasis Default = 'N'; No epigenetics

#' @param epigeneans Default = 'N'; No epigenetics

#' @param startEpigene Default = 0; No epigenetics

#' @param betaFile\_epigene Default = 'N'; No epigenetics

#' @param cdinfect Default = 'N'; No epigenetics

#' @param transmissionprob Default = 0; No epigenetics

#'

#'

#'

cdpopJW <- function(CDPOP.py,

sim\_name = 'output\_',

pts,

sim\_dir,

resist\_rast,

resist\_mat = NULL,

agefilename = NULL,

mcruns = 1,

looptime = 400,

output\_years = 50,

gridformat = 'genepop',

cdclimgentime = 0,

matemoveno = 5,

matemoveparA = 1,

matemoveparB = 5,

matemoveparC = 0,

matemovethresh = 'max',

output\_matedistance = 'N',

sexans = 'Y',

Freplace = 'N',

Mreplace = 'N',

philopatry = 'N',

multiple\_paternity = 'N',

selfans = 'N',

Fdispmoveno = NULL,

FdispmoveparA = 0,

FdispmoveparB = 0,

FdispmoveparC = 0,

Fdispmovethresh = NULL,

Mdispmoveno = NULL,

MdispmoveparA = 0,

MdispmoveparB = 0,

MdispmoveparC = 0,

Mdispmovethresh = NULL,

offno = 2,

MeanFecundity = 5,

Femalepercent = 50,

EqualsexratioBirth = 'N',

TwinningPercent = 0,

popModel = 'exp',

r = 1,

K\_env = length(pts),

subpopmortperc = 0,

muterate = 0.0005,

mutationtype = 'random',

loci = 1000,

intgenesans = 'random',

allefreqfilename = 'N',

alleles = 2,

mtdna = 'N',

startGenes = 0,

cdevolveans = 'N',

startSelection = 0,

betaFile\_selection = 'N',

epistasis = 'N',

epigeneans = 'N',

startEpigene = 0,

betaFile\_epigene = 'N',

cdinfect = 'N',

transmissionprob = 0){

# Install / Load Libraries ------------------------------------------------

list.of.packages <- c("gdistance",

"adegenet",

"readr",

"raster")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages)) install.packages(new.packages)

library(raster)

library(gdistance)

library(adegenet)

library(readr)

# Create directories ------------------------------------------------------

if(!dir.exists(sim\_dir)) dir.create(sim\_dir, recursive = TRUE)

suppressWarnings(

dir.create(paste0(sim\_dir,"/data/"), recursive = TRUE)

)

data\_dir <- paste0(sim\_dir,"/data/")

# Fill NULL ---------------------------------------------------------------

if(matemoveno == 9){

if(class(resist\_rast) == "RasterLayer"){

stop('Specify a probability matrix instead of a raster layer!')

}

write.table(resist\_rast,

paste0(data\_dir, "move\_prob.csv"),

sep = ",",

row.names = FALSE,

col.names = FALSE)

cdmat <- 'move\_prob'

# write.table(matemoveno, paste0(data\_dir, 'DispProb.csv'),

# sep = ",",

# row.names = F)

# matemoveno <- 'DispProb'

}

if(is.null(Fdispmoveno)){

Fdispmoveno <- matemoveno

}

if(is.null(Mdispmoveno)){

Mdispmoveno <- matemoveno

}

if(is.null(Fdispmovethresh)){

Fdispmovethresh <- matemovethresh

}

if(is.null(Mdispmovethresh)){

Mdispmovethresh <- matemovethresh

}

# Age file ----------------------------------------------------------------

if(is.null(agefilename)){

age\_df <- data.frame(`Age class` = c(0,1),

Distribution = c(0,1),

`Male Mortality` = c(0,100),

`Female Mortality` = c(0,100),

`Mean Fecundity` = c(0,MeanFecundity),

`Std Fecundity` = c(0,0),

`Male Maturation` = c(0,1),

`Female Maturation` = c(0,1),

check.names = F)

write.table(age\_df, paste0(data\_dir, 'AgeVars.csv'),

sep = ",",

row.names = F)

# age\_file <- paste0(data\_dir, 'AgeVars.csv')

age\_file <- 'AgeVars.csv'

}

# XY File -----------------------------------------------------------------

xyFile\_df <- data.frame(Subpopulation = rep(1, length(pts)),

XCOORD = pts@coords[,1],

YCOORD = pts@coords[,2],

ID = paste0('initial',1:length(pts) - 1),

sex = sample(c(0,1),

replace = T,

size = length(pts)),

Fitness\_AA = rep(0, length(pts)),

Fitness\_Aa = rep(0, length(pts)),

Fitness\_aa = rep(0, length(pts)),

Fitness\_AABB = rep(0, length(pts)),

Fitness\_AaBB = rep(0, length(pts)),

Fitness\_aaBB = rep(0, length(pts)),

Fitness\_AABb = rep(0, length(pts)),

Fitness\_AaBb = rep(0, length(pts)),

Fitness\_aaBb = rep(0, length(pts)),

Fitness\_AAbb = rep(0, length(pts)),

Fitness\_Aabb = rep(0, length(pts)),

Fitness\_aabb = rep(0, length(pts))

)

write.table(xyFile\_df, paste0(data\_dir, 'xyFile.csv'),

sep = ',',

row.names = F)

# xyFile <- paste0(data\_dir, 'xyFile')

xyFile <- 'xyFile'

# Resistance Distance -----------------------------------------------------

if(matemoveno == 9){

write.table(resist\_mat,

paste0(data\_dir, "resist\_mat.csv"),

sep = ",",

row.names = FALSE,

col.names = FALSE)

cdmat <- 'resist\_mat'

}

if(matemoveno != 9){

if(!is.null(resist\_mat)){

write.table(resist\_mat,

paste0(data\_dir, "resist\_mat.csv"),

sep = ",",

row.names = FALSE,

col.names = FALSE)

cdmat <- 'resist\_mat'

} else {

print("Calculating resistance distance with `gdistance`...")

trans <- transition(x = resist\_rast,

transitionFunction = function(x) 1 / mean(x),

directions = 8)

trR <- geoCorrection(trans, "r", scl = T)

resist\_mat <- as.matrix(commuteDistance(trR, pts) / 1000)

## Check file format, row/col names?

write.table(resist\_mat,

paste0(data\_dir, "resist\_mat.csv"),

sep = ",",

row.names = FALSE,

col.names = FALSE)

cdmat <- 'resist\_mat'

}

}

# CDPOP input ----------------------------------------------------------

cdpop\_df <- data.frame(xyfilename = xyFile,

agefilename = age\_file,

mcruns = mcruns,

looptime = looptime,

output\_years = output\_years,

gridformat = gridformat,

cdclimgentime = cdclimgentime,

matecdmat = cdmat,

dispcdmat = cdmat,

matemoveno = matemoveno,

matemoveparA = matemoveparA,

matemoveparB = matemoveparB,

matemoveparC = matemoveparC,

matemovethresh = matemovethresh,

output\_matedistance = output\_matedistance,

sexans = sexans,

Freplace = Freplace,

Mreplace = Mreplace,

philopatry = philopatry,

multiple\_paternity = multiple\_paternity,

selfans = selfans,

Fdispmoveno = Fdispmoveno,

FdispmoveparA = FdispmoveparA,

FdispmoveparB = FdispmoveparB,

FdispmoveparC = FdispmoveparC,

Fdispmovethresh = Fdispmovethresh,

Mdispmoveno = Mdispmoveno,

MdispmoveparA = MdispmoveparA,

MdispmoveparB = MdispmoveparB,

MdispmoveparC = MdispmoveparC,

Mdispmovethresh = Mdispmovethresh,

offno = offno,

Femalepercent = Femalepercent,

EqualsexratioBirth = EqualsexratioBirth,

TwinningPercent = TwinningPercent,

popModel = popModel,

r = r,

K\_env = K\_env,

subpopmortperc = subpopmortperc,

muterate = muterate,

mutationtype = mutationtype,

loci = loci,

intgenesans = intgenesans,

allefreqfilename = allefreqfilename,

alleles = alleles,

mtdna = mtdna,

startGenes = startGenes,

cdevolveans = cdevolveans,

startSelection = startSelection,

betaFile\_selection = betaFile\_selection,

epistasis = epistasis,

epigeneans = epigeneans,

startEpigene = startEpigene,

betaFile\_epigene = betaFile\_epigene,

cdinfect = cdinfect,

transmissionprob = transmissionprob,

check.names = F)

write.table(cdpop\_df,

paste0(data\_dir, "CDPOP\_inputs.csv"),

sep = ",",

row.names = FALSE,

col.names = TRUE,

quote = F)

# Run CDPOP ---------------------------------------------------------------

print("Running CDPOP...")

system(paste("python", CDPOP.py, data\_dir, "CDPOP\_inputs.csv", sim\_name))

# Import Results ----------------------------------------------------------

fi <- file.info(list.files(path = sim\_dir,

pattern = "grid",

recursive = T,

full.names = T))

## Get latest simulation results

newest\_sim <- dirname(rownames(fi)[which.max(fi$mtime)])

grid\_dir <- list.files(path = newest\_sim,

pattern = "grid",

recursive = T,

full.names = T)

read.grid <- function(grid,

pops = NULL){

suppressWarnings(

cdpop\_out <- read\_csv(grid,

col\_types = cols(Subpopulation = col\_skip(),

#XCOORD = col\_skip(), YCOORD = col\_skip(),

sex = col\_skip(), age = col\_skip(),

infection = col\_skip(), DisperseCDist = col\_skip(),

hindex = col\_skip()))

)

#geogr <- read\_csv(grid)[which(cdpop\_out$ID != "OPEN"),c("XCOORD", "YCOORD")]

occ\_pop <- which(cdpop\_out$ID != "OPEN")

if(!is.null(pops)) {

return(occ\_pop)

} else {

cd\_df <- as.data.frame(cdpop\_out[occ\_pop,c(-1,-2,-3)])

cd\_df[,ncol(cd\_df)] <- gsub(",","",cd\_df[,ncol(cd\_df)])

cd\_df <- apply(as.matrix(cd\_df),2,as.numeric)

# fakedf <- data.frame(matrix(rep(paste(paste0("A", rep\_len(0:(alleles-1),length.out = nrow(cd\_df))),

# paste0("A", rep\_len(0:(alleles-1),length.out = nrow(cd\_df))),

# sep="/"), loci), ncol=loci))

fakedf<-data.frame(matrix(rep(paste(LETTERS[rep\_len(1:alleles,length.out = nrow(cd\_df))],

LETTERS[rep\_len(1:alleles,length.out = nrow(cd\_df))],

sep="/"), loci), ncol=loci))

colnames(fakedf) <- paste0("L",1:loci)

ncode <- 1

gi <- adegenet::df2genind(fakedf, ploidy=2, sep="/", type="codom")

gi@tab <- cd\_df

propertabnames <- character(0)

for (i in 1:loci){

propertabnames <- c(propertabnames, paste(names(gi$all.names)[i],

unlist(gi$all.names)[1:alleles],

sep="."))

}

colnames(gi@tab) <- propertabnames

gi@other$xy <- cdpop\_out[occ\_pop, c(1,2)]

gi@tab <- apply(gi@tab, 2, as.integer)

return(gi)

}

}

grid\_list <- lapply(grid\_dir, read.grid)

pop\_list <- lapply(grid\_dir, read.grid, pops = TRUE)

gens <- basename(grid\_dir) %>% sub('.csv', '', .) %>% # <

sub('grid', '',.) %>% as.numeric()

grid\_list <- grid\_list[order(gens)]

pop\_list <- pop\_list[order(gens)]

names(pop\_list) <- names(grid\_list) <- paste0('gen\_', sort(gens))

# Wrap-up -----------------------------------------------------------------

out <- list(grid\_list = grid\_list,

pop\_list = pop\_list)

return(out)

}

# PCA dist -------------------------------------------------------

pca\_dist <- function(gi,

n\_axes = 64){

a\_tab <- adegenet::tab(gi)

pc <- prcomp(a\_tab)

pc\_dist <- as.matrix(dist(pc$x[,1:n\_axes]))

return(pc\_dist)

}

# Random Samples ----------------------------------------------------------

## Randomly select populations and individuals from within populations

gi\_samp <- function(gi,

n\_ind = 100) {

ind\_samp <- sort(sample(1:nInd(gi), n\_ind))

gi\_s <- gi[ind\_samp]

out <- list(genind = gi\_s,

pop\_samp = ind\_samp)

}

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

########## Julian Wittische - November 2021 - Simulating connectivity ##########

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

# This is based on previous work by William Peterman and Kristopher Winiarski

# Simulations based on empirical data

empir.sim <- function(catraster = catraster,

geosites = geosites,

habitat = 0.5,

RMexpvar\_r = 1,

RMexpscale\_r = 15,

n\_ind = 10000,

n\_samplepoints = 250,

start = 1,

seed = 1,

iters = 1,

parallel =3,

method = 'standard',

maxiter = 100,

JULIA\_HOME = "C:/Users/jwittische/AppData/Local/Programs/Julia-1.6.3/bin/",

#JULIA\_HOME = "C:/Users/Utilisateur/AppData/Local/Programs/Julia-1.7.1/bin/",

CDPOP.py = 'C:/Users/jwittische/Desktop/Projects/BestDistance/CDPOP-master/src/CDPOP.py',

#CDPOP.py = 'C:/Users/Utilisateur/Desktop/Projects/BestDistance/CDPOP-master/src/CDPOP.py',

sim\_name = 'output\_',

sim\_dir = "C:/Users/jwittische/Desktop/Projects/BestDistance/cdpop\_sim\_TEST/",

#sim\_dir = "C:/Users/Utilisateur/Desktop/Projects/BestDistance/cdpop\_sim\_TEST/",

looptime = 101,

output\_years = 100,

gridformat = 'cdpop',

loci = 17,

alleles = 20,

matemoveno = 1, ## 1 = Linear, 2 = Inv sq; 9 = custom prob matrix

matemovethresh = 0.1,

matemoveparA = 1,

matemoveparB = 5,

MeanFecundity = 4,

n\_axes = 64)

{

if(is.null(start)) {

stop("Specify integer `start` value!!!")

}

# Main Function -----------------------------------------------------------

for(z in start:iters){

# >> Make Random Surfaces -------------------------------------------------

# \* Empirical surface ------------------------------------------------------------

# Load ordinal variable template

orig <- catraster #not the same as Copernicus! (3035)

# \* Random surface --------------------------------------------------------

# coo <- dim(orig)

# bb <- extent(orig)

# model <- RMexp(var=RMexpvar\_r, scale=RMexpscale\_r)

# rf.sim <- RFsimulate(model = model, x=1:nrow(orig), y=1:ncol(orig), grid=TRUE)

# rand <- raster(scale(as.matrix(rf.sim)))

# rand <- setExtent(rand, bb, snap= TRUE)

# random\_1 <- rand

# >> Create Directory -----------------------------------------------------

dir.create(paste0(sim\_dir, "Results/",

'iter\_\_', z),

recursive = TRUE)

out <- paste0(sim\_dir, "Results/",

'iter\_\_', z, "/")

# >> Create Truth ---------------------------------------------------------

Resist <- catraster

Resist[Resist==0] <- 20 # Remaing built-up

Resist[Resist==1] <- 10# Forest and open areas

Resist[Resist==2] <- 70 # Urban areas

Resist[Resist==3] <- 80 # River

Resist[Resist==4] <- 1 # Transport infrastructure

# Load sampling sites ------------------------------------------------------

pts <- unique(floor(cbind(runif(100000, extent(catraster)[1], extent(catraster)[2]),

runif(100000, extent(catraster)[3], extent(catraster)[4]))))

sample.thresh <- as.numeric(quantile(Resist, habitat))

sample.extract <- extract(Resist, pts)

sample.suit <- pts[sample.extract <= sample.thresh,]

sample.suit <- na.omit(sample.suit)

pts <- SpatialPoints(sample.suit[sample(nrow(sample.suit), n\_ind, replace = F),])

###

# >> Calculate cost distance -----------------------------------------------

jl.inputs <- jl.prep(n.Pops = length(pts),

CS\_Point.File = pts,

JULIA\_HOME = JULIA\_HOME)

## Pairwise resistance distances

r.dist <- Run\_CS.jl(jl.inputs = jl.inputs,

r = Resist,

full.mat = T)

plot(Resist)

plot(pts, add = T, pch = 19)

m\_thresh <- quantile(lower(r.dist), matemovethresh)

cdpop\_sim <- cdpopJW(CDPOP.py = CDPOP.py,

sim\_name = sim\_name,

pts = pts,

resist\_rast = Resist,

resist\_mat = r.dist,

sim\_dir = out,

looptime = looptime,

output\_years = output\_years,

gridformat = gridformat,

loci = loci,

alleles = alleles,

K\_env = 10000,

matemoveno = matemoveno, ## 1 = Linear, 5 = Neg exp; 9 = custom prob matrix

matemovethresh = m\_thresh,

matemoveparA = matemoveparA,

matemoveparB = matemoveparB,

MeanFecundity = MeanFecundity)

saveRDS(cdpop\_sim, paste0(out,"cdpop\_sim.rds"))

writeRaster(Resist,

paste0(out, "true\_resist.asc"), overwrite=TRUE)

} # end iteration loop (z)

} # end function

***2 - Code used to create the coarser version of the input raster while preserving the continuity of linear features***

# Geographical coordinates

geosites <- SpatialPoints(read.table("Data/geo.txt", header=TRUE)[,2:3],

CRS(SRS\_string = "EPSG:3044"))

# CDPOP does not allow for a K limit PER cell so I have to change my strategy

catraster\_SA <- raster("Data/study\_area5.asc", crs = "EPSG:3044")

catraster\_SA <- reclassify(catraster\_SA, cbind(c(3,4),c(4,3)))

# plot(catraster\_SA)

# plot(geosites, add=TRUE, pch = 19)

# plot(catraster\_SA, colNA="blue")

garbage\_120m <- aggregate(catraster\_SA, fact=3, fun=max, na.rm=TRUE) #max

# Resample original using ngb

set.seed(1) #ngb is random so final result may differ between runs

catraster\_SA\_resamp <- resample(catraster\_SA, garbage\_120m, method = "ngb")

catbrick\_SA <- layerize(catraster\_SA, falseNA=FALSE)

catbrick\_SA <- catbrick\_SA\*c(1,2,3,4,5)

catbrick\_SA\_agg <- aggregate(catbrick\_SA, fact=3, fun=max, na.rm=TRUE)

catbrick\_SA\_agg$X0[catbrick\_SA\_agg$X0==0] <- NA

catbrick\_SA\_agg$X1[catbrick\_SA\_agg$X1==0] <- NA

catbrick\_SA\_agg$X2[catbrick\_SA\_agg$X2==0] <- NA

catbrick\_SA\_agg$X3[catbrick\_SA\_agg$X3==0] <- NA

catbrick\_SA\_agg$X4[catbrick\_SA\_agg$X4==0] <- NA

catbrick\_SA\_agg <- catbrick\_SA\_agg-1

lol\_SA <- mosaic(catbrick\_SA\_agg$X3, catbrick\_SA\_agg$X4, fun=max)

catraster\_SA\_coarser <- merge(lol\_SA,catraster\_SA\_resamp)

catraster\_SA\_coarser\_cropped <- crop(catraster\_SA\_coarser,

extent(geosites)+c(-360,360,-360,360))

catraster\_SA\_coarser\_cropped <- reclassify(catraster\_SA\_coarser\_cropped,

cbind(c(3,4),c(4,3)))

***3 - Code used to subsample simulated individuals from the same cells where lizards were actually caught***

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

### Loading the simulation data

cdpop\_sim1 <- readRDS("cdpop\_sim\_TEST/Results/iter\_\_1/cdpop\_sim.rds")

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

### Loading the Podarcis muralis data

# Geographical coordinates

geosites <- SpatialPoints(read.table("Data/geo.txt", header=TRUE)[,2:3],

CRS(SRS\_string = "EPSG:3044"))

empir\_geo\_dist <- as.matrix(dist(read.table("Data/geo.txt", header=TRUE)[,2:3]))

# Genetic data

lizgen <- read.csv("Data/TR\_NA\_header.csv", row.names="ID", na.strings="NA")

lizgen[lizgen==95] <- "095"

lizgen[lizgen==97] <- "097"

lizgen[lizgen==99] <- "099"

lizgen.df <- lizgen[,1:(ncol(lizgen)/2)]

for (i in seq(1,ncol(lizgen),2)){

lizgen.df[ ,(i+1)/2] <- apply( lizgen[ , i:(i+1) ] , 1, paste , collapse = "/" )

}

colnames(lizgen.df) <- paste0("L", 1:ncol(lizgen.df))

lizgen.df[lizgen.df=="NA/NA"] <- NA

lizgen.genind <- df2genind(lizgen.df, NA.char=NA,

ploidy=2, type="codom", sep = "/", check.ploidy=TRUE)

empirLoiselle\_EcoGenetics <- eco.kin.loiselle(genind2ecogen(lizgen.genind))

mantel.randtest(as.dist(empir\_geo\_dist), as.dist(1-empirLoiselle\_EcoGenetics))

empir\_geo\_dist2 <- empir\_geo\_dist

empir\_geo\_dist2[empir\_geo\_dist2==0] <- NA

IBD <- lm(c(as.dist(empirLoiselle\_EcoGenetics))~log(c(as.dist(empir\_geo\_dist2))))

summary(IBD)

plot(log(empir\_geo\_dist2), empirLoiselle\_EcoGenetics)

abline(IBD, col="red")

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

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

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

# sim\_geo\_dist <- as.matrix(dist(cdpop\_sim1$grid\_list$gen\_101@other$xy))

# sim\_geo\_dist[sim\_geo\_dist==0] <- NA

#

# simLoiselle\_EcoGenetics <- eco.kin.loiselle(genind2ecogen(cdpop\_sim1$grid\_list$gen\_101))

#

# mantel.randtest(as.dist(sim\_geo\_dist), as.dist(1-simLoiselle\_EcoGenetics))

# IBDsim <- lm(c(as.dist(simLoiselle\_EcoGenetics))~log(c(as.dist(sim\_geo\_dist))))

# summary(IBDsim)

# plot(log(sim\_geo\_dist), simLoiselle\_EcoGenetics)

# abline(IBDsim, col="red")

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

# Find the empirical distribution of lizard abundance at our new res

lizpercell <- as.matrix(table(extract(catraster\_SA\_coarser\_cropped, geosites,

cellnumbers = TRUE,

fun=length)[,1]))

lizgrid <- catraster\_SA\_coarser\_cropped

lizgrid[] <- 0

lizcellrowcol <- rowColFromCell(lizgrid, as.numeric(names(lizpercell[,1])))

lizgrid[lizcellrowcol] <- lizpercell

plot(lizgrid)

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

# Let us try to subsample to get a similar sampling as the empirical dataset

sim\_geosites <- SpatialPoints(cdpop\_sim1$grid\_list$gen\_101@other$xy,

CRS(SRS\_string = "EPSG:3044"))

lizgridno0 <- lizgrid

lizgridno0[lizgridno0==0] <- NA

#subs <- extract(lizgridno0, sim\_geosites, cellnumber-TRUE, sp=TRUE)

library(spatialEco)

lizgridno0poly <- rasterToPolygons(lizgridno0)

plot(lizgridno0poly)

subs <- erase.point(sim\_geosites, lizgridno0poly, inside=FALSE)

"%IN%" <- function(x, y) interaction(x) %in% interaction(y)

index <- cdpop\_sim1$grid\_list$gen\_101@other$xy %IN% as.data.frame(subs)

sim\_subs\_genind <- cdpop\_sim1$grid\_list$gen\_101[index]