**Run and analyse a metacommunity model**

Today we are going to run a metacommunity model and analyse the outputs for different values of dispersal ability and niche width, to explore the process-pattern relationship. Rather than coding the metacommunity model from scratch (which would take way more than 2 hours!), we will use the MCSim R package, which is available on GitHub. This model allows to model dispersal and abiotic niche selection for multiple species in at different sites, where each site is characterised by an environmental variable.

A tutorial to use this package is available from: <http://rstudio-pubs-static.s3.amazonaws.com/159425_80725873417e42fdb13821c10a198281.html>. I have adapted and tailored it for this course. In the future, you may need to implement your own model accounting for the specificities of your study system (different dispersal processes, multiple niche dimensions, species interactions, multiple trophic levels, etc.), but this will familiarise you with how to explore the parameter space and analyse this kind of model in general.

**Implement and run the model using the MCSim package**

**First things first**

Let’s start by installing the MCSim package. It is not available on CRAN, we need to install it directly from GitHub. You need to install the devtools package first, which is available on CRAN. You can then type:

devtools::install\_github('sokole/MCSim')

Then, let’s load all the packages you will need:

library(MCSim)

library(tidyverse)

library(fields)

library(vegan)

library(betapart)

And to make the simulations reproducible, let’s define a seed, which determine the starting point of random generators and allow to get the same results every time despite being “random”:

set.seed(1234)

You can remove this or use other seeds to get other results.

**Model initialisation**

We will simulate 10 species over 50 discrete sites. We will first define the sites and their characteristics. Take the time to look at the code (and comments) and the variables it generates in your workspace to understand what it is doing. Ask if there is anything unclear.

nsites <- 50 ##number of sites

# Let's generate a data frame with coordinates for nsites sites, randomly distributed in a 10 x 10 environment

xy.coordinates <- data.frame(

x = runif(nsites)\*10,

y = runif(nsites)\*10

)

xy.coordinates <- xy.coordinates[order(xy.coordinates$x,xy.coordinates$y),]

print(xy.coordinates)

plot(y ~ x, xy.coordinates)

##The following creates a list with information on sites that is later needed by the function running the model

my.landscape <- MCSim::fn.make.landscape(

site.coords = xy.coordinates, ## site coordinates

m = rep(0.05,nsites),#rep(0.01,10), ##immigration rate for each site from the species pool

Ef = sort(runif(nsites)), ## Environmental variable value for each site - since the sites are ordered on the x-y axise, using sort() introduces an environmental gradient along the x axis – remove sort for a more random environment

JM = 1000000 ## metacommunity size (sensu Hubbell 2001) – i.e. species pool size

)

print(my.landscape$site.info)

We can plot the sites again, colouring them according to their environment:

plot(y ~ x, my.landscape$site.coords, pch=16, col=tim.colors(n=64)[ceiling(my.landscape$site.info$Ef\*64)])

We will then define the 10 species, including their niches along the environmental axis. Each niche is modelled as a Normal distribution, which is typical for Hutchinsonian niches, as we saw in class.

# niche positions, niche breadths, and relative abundances for 10 species

niche.positions <- runif(10) ##niche optima are distributed randomly along the environmental axis

niche.breadths <- rep(0.1,10) ##all species have the same niche width of 0.1 (keep in mind the environment varies between [0,1])

regional.rel.abund <- rep(1/10,10) ##each species will be initialised with the same number of individual in each site

####Plot niches

# -- function for plotting bell curves (the curve of a normal distribution)

fn\_norm\_curve <- function(sigma=1, mu = 0,...) {

curve(

(1/sigma \* sqrt(2 \* pi)) \* exp((-1 \*(x - mu)^2) / (2 \* sigma^2)), ...) #formula for bell curve

}

# -- Initialize plot

plot(1,1,

xlim = c(-0.2,1.2),

ylim = c(0, (1/niche.breadths[1]\* sqrt(2 \* pi))),

type = 'n',

xlab = 'Environmental gradient',

ylab = 'Prob. dens.',

main = 'Niche positions and niche widths')

mypal <- tim.colors(length(niche.positions))

# -- loop to plot each species' habitat preference

for (i.spp in 1:length(niche.positions)){

fn\_norm\_curve(

mu = niche.positions[i.spp],

sigma = niche.breadths[i.spp],

add = TRUE,

col = mypal[i.spp],

n=1000)

}

# -- plot sites along the x-axis

rug(my.landscape$site.info$Ef)

legend(x = 'topleft',

legend = 1:10,

lty = 1,

col = mypal,

cex = .75,

ncol = 4)

**Run the model**

We will run simulations over 100 timesteps. If you play with the model (i.e. if you increase simulation length and look at the different community patterns), you will see this is not always enough for the model to reach equilibrium depending on the parameters, but this is enough for this practical while keeping simulation times reasonable.

n.timestep <- 100

sim.result <- MCSim::fn.metaSIM(

landscape = my.landscape,

trait.Ef = niche.positions,

trait.Ef.sd = niche.breadths,

gamma.abund = regional.rel.abund,

W.r = 2, ##slope of exponential dispersal kernel – the higher the value, the more restricted the dispersal. 0 means infinite dispersal

nu = 0, ##probability that a novel species will appear during a recruitment event

n.timestep = n.timestep,

sim.ID = "my\_test\_sim",

output.dir.path = "my\_sim\_output\_directory")

head(sim.result$J.long)

**Analyse the model outputs**

If you look at sim.result$J.long, which is the main output of the function, it contains the species composition of all sites over all the simulated timesteps. You therefore need to extract the part of the data frame corresponding to the last time step. Once this is done, you need to transform this data frame into a species-by-site data frame, as usual. This can be done with the following code:

MC.fin <- sim.result$J.long[which(sim.result$J.long$timestep==n.timestep),]

MC.fin.mat <- MC.fin %>%

pivot\_wider(names\_from=spp,values\_from=c(count)) ##this data frame contains abundance data

MC.fin.mat <- MC.fin.mat[,-c(1,2)]

MC.fin.mat.pa <- as.data.frame((MC.fin.mat>0)\*1) ##we convert the abundance data frame into a presence-absence one to compute some indices afterwards

Using these data frames, compute gamma diversity, alpha diversity (richness-based), the Shannon index, incidence-based beta diversity from the presence-absence data (using Sørensen and Simpson indices), and the abundance-based beta diversity using the Bray-Curtis index (whis is the equivalent of Sørensen for abundance data):

Gamma <- length(which(colSums(MC.fin.mat)>0))

#Alpha

#rowSums(MC.fin.mat.pa)

Alpha <- mean(rowSums(MC.fin.mat.pa))

#Shannon

#diversity(MC.fin.mat)

Shannon <- mean(diversity(MC.fin.mat))

#Beta

#beta.pair(MC.fin.mat.pa)

#beta.pair.abund(MC.fin.mat)$beta.bray

beta <- lapply(beta.pair(MC.fin.mat.pa),mean)

beta.Sim <- beta$beta.sim

beta.Sor <- beta$beta.sor

beta.ab <- mean(beta.pair.abund(MC.fin.mat)$beta.bray)

**Check if there is convergence**

Run the following code at the end of the previous one. What does this code do, and can you consider that the simulation has reached equilibrium?

MC.fin.mat.pa <- array(NA,c(n.timestep,50,10))

MC.fin.mat <- array(NA,c(n.timestep,50,10))

for(i in 1:n.timestep){

MC.fin <- sim.result$J.long[which(sim.result$J.long$timestep==i),]

MC.fin.mat.temp <- MC.fin %>%

pivot\_wider(names\_from=spp,values\_from=c(count)) ##this data frame contains abundance data

MC.fin.mat[i,,] <- as.matrix(MC.fin.mat.temp[,-c(1,2)])

MC.fin.mat.pa[i,,] <- as.matrix(as.data.frame((MC.fin.mat.temp[,-c(1,2)]>0)\*1)) ##we convert the abundance data frame into a presence-absence one to compute some indices afterwards

}

oc.time <- apply(MC.fin.mat.pa,c(1),colSums)

plot(1:n.timestep,oc.time[1,],type="l",col=tim.colors(n=10)[1],ylim=c(0,max(oc.time)))

for(i in 2:10){

lines(1:n.timestep,oc.time[i,],col=tim.colors(n=10)[i])

}

ab.time <- apply(MC.fin.mat,c(1),colSums)

plot(1:n.timestep,ab.time[1,],type="l",col=tim.colors(n=10)[1],ylim=c(0,max(ab.time)))

for(i in 2:10){

lines(1:n.timestep,ab.time[i,],col=tim.colors(n=10)[i])

}

**Run multiple simulations for different parameter values**

We want to explore the influence of dispersal ability and niche width on the model outputs and the community patterns described above. You will then run your script within two for loops, to increment these two parameters. These for loop will go through the following two vectors that you need to initialise:

niche.width <- c(0.01,0.1,2,5,1000) ##vector of niche widths

disp.ab <- c(0,2,5,10,1000,5e6) ##vector of dispersal abilities – as mentioned above, the higher the value, the more restricted the dispersal. 0 means infinite dispersal

Parameters niche.breadths and W.r will take these values in the code above.

What kind of model seen in class does a niche width of 1000 approximate? Conversely, what kind of model does a dispersal ability parameter of 0, i.e. infinite dispersal, correspond to?

Store the values of gamma diversity, alpha diversity, Shannon index and the different beta diversity in 5 x 6 matrices. Store also MC.fin.mat in a list of list, as it may help you to better understand how the community patterns change.

Gamma <- Alpha <- Shannon <- beta.Sim <- beta.Sor <- beta.ab <- matrix(NA,length(niche.width),length(disp.ab))

MC.fin.mat.store <- list()

You now just need to add the two for() loops to your code, and the indices to the patterns, and run the model. It should take about 7 minutes, or you can load the Simulation\_results.RData file instead.

**Analyse and discuss the results**

For different niche widths, how does changing dispersal ability affects the different community patterns? Is this consistent for the different niche widths?

For different dispersal abilities, how does changing the niche width affects the different community patterns? Is this consistent for the different values of dispersal abilities?

Are these changes monotonic or linear?

What does it mean for model fitting using community patterns only (i.e. inverse modelling or retro-fitting)?

**Additional work if you have extra time:**

**Null models**

Following last practical, you can use null models to explore if that changes some of the patterns. Look at the permatfull() help file in the vegan package. How would you randomise the matrices? Do this for one set of parameter values (e.g. for a niche width of 0.1 and a dispersal ability of 5), compute the different patterns, and compare the distributions of the pattern values for the randomised matrices to the observed pattern values.

**Influence of initial conditions**

Change the initial population sizes in the different sites. For example, initialise the populations relative abundances as:

regional.rel.abund <- (1:100)/sum(1:100)

You can load the file Simulation\_results\_different\_abundances.RData.

What does this do, and how does this change the results?

**Fitting the model to real data**

Here we used a theoretical model to explore how changing the dispersal and niche width parameters. But you may also parameterise the model so that these parameter values reflect real ones. Dispersal can be hard to estimate in real systems, as it can be hard to track individuals or seeds. For niche optimum and width, we have statistical models that we can use, such as SDMs, that you saw in class with Ally.

The online tutorial at <http://rstudio-pubs-static.s3.amazonaws.com/159425_80725873417e42fdb13821c10a198281.html> shows a way to do this for the mite data in vegan, estimating species niches using the ade4 package, in sectin “Seeding a simulation with observed data”. You can run this code to explore how they do this. The main differences are the following:

* The site locations are initialised from the real data
* The environmental variable is estimated from real data, after using a PCA on multiple environmental variables to summarise this information, using the dudi.pca()function from the ade4 R package.
* The niches are computed using the dudi.pca() and niche() functions from the ade4 R package.
* The initial population sizes are estimated from the real data: the J.t0 parameter replaces the parameter gamma.abund in function fn.metaSIM().
* In the tutorial, the author only considers very limited dispersal, with W.r = 5e6, which may be an appropriate approximation for mites with low dispersal abilities.