Running the CoevoMutAntag simulations

Lucas Arantes Camacho

01/10/2019

Introduction

This is a guide to show step-by-step how to run the coevolution simulations in mutualistic networks with cheater exploitation outcomes inserted. You will se that the process use different functions from the *R/functions* folder depending on what we wanna do in our scripts (which are saved in the *R/scripts* folder). Also, you wil see that the scripts base structure have a organization pattern which helps in the understanding of the code steps in the specific function that it's being realized.

Basic structure

We start setting our work directory that must contain the *.txt* file with the empirical matrices that you interested in work with. Also, we gonna load some packages and functions that we gonna need in our script. Remember of install the required packages before runing the script.

```r
# loading packages and functions
setwd("~/Dropbox/Master/Code/coevo_mut_antag/R/scripts/")

library(ggplot2)
library(reshape2)
library(cowplot)

source("~/Dropbox/Master/Code/coevo_mut_antag/R/functions/Antagonize.R")
source("~/Dropbox/Master/Code/coevo_mut_antag/R/functions/Counting.R")
source("~/Dropbox/Master/Code/coevo_mut_antag/R/functions/FindInteractors.R")
source("~/Dropbox/Master/Code/coevo_mut_antag/R/functions/SpDegree.R")
source("~/Dropbox/Master/Code/coevo_mut_antag/R/functions/CoevoMutAntNet.R")
```

Ok, we have our packages and functions. Now we can think about the matrix we wanna work with. In this example I will construct a fully connected bipartite adjacency matrix. If you wanna run the scripts with a empirical data matrix just read the matrix with *read.table* function and remember of use the *Square.Matrix* (provided in the *R/functions folder*) if this is a retangular matrix.

Let's first define our antprob value (p) which is the probability of the elements from our mutualistic matrix M pass from mutualism to cheater exploitation. Also, let's construct our M matrix defining the number of species (animals + plants) that we wanna in our matrix. It's important to note that we are not interested in intraespecific interactions so diagonal of M is zeroed.

```r
# initial parameters
antprob = 0.8 # current probability value
n_sp = 10 # defining number of species
M = matrix(1, ncol = n_sp, nrow = n_sp) # building matrix M of positive outcomes
diag(M) = 0 # no intraespecific interactions
```

We already have our adjacency matrix M of positive effects. Know let's apply our first function in the M matrix: Antagonize. This functions creates the V matrix and transform positive in negative effects follow the antprob value. The functions return the M and V matrices as a list which is used to define M and V to the next step.

```r
# Antagonize M (transform positive links in negative)
antagonize = Antagonize(M, antprob)
M = antagonize[[1]]
V = antagonize[[2]]
```

We have the two layers of our initial matrix, the positive effects layer matrix M and negative effects layer matrix V. With those matrices in hand we can apply some functions to get some informations about the numbers of different outcomes interactions in the community and how much each species are interaction with mutualists or cheaters. Let's apply three functions: Counting, FindInteractors and SpDegree. The Counting functions first:

```r
c = Counting(M, V)
c
```

```
## [[1]]
## [1] 0
##
## [[2]]
## [1] 40
##
## [[3]]
## [1] 5
```

The Counting function provides the counting of the number of interaction outcomes in our adjacency matrix after the Antagonize process. You see a list with three values: [1] number of double negative effects which are not the focus of this work and never reproduced by the Antagonize function, [2] number of cheater exploitation outcomes and [3] number of mutualism outcomes. Usefull to do some consistency tests about the frequency of interaction outcomes in the network. Next function, the FindInteractors:

```r
f = FindInteractors(M, V)
f
```

```
## [[1]]
## integer(0)
##
```

```
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## [[3]]
## [1]  1  2  7  8  9 10
```

The FindInteractors function provides a list with the identity of species that has mutualism and/or cheater exploitation in the community. Due to te way Antagonize works, a single species could be in the cheaters group [[2]] and mutualism group [[3]] at te same time. Usefull when we wanna separate species in groups depending on the interaction outcome they have. Finally, the last function before we run the coevolution process, the SpDegree:

```r
sp = SpDegree(M, V)
sp
```

```
##    AA AM MM
## 1   0  7  2
## 2   0  8  1
## 3   0  9  0
## 4   0  9  0
## 5   0  9  0
## 6   0  9  0
## 7   0  8  1
## 8   0  6  3
## 9   0  8  1
## 10  0  7  2
```

The SpDegree function just count the interactions of each species separating by outcomes. With that we could know how much each species have of double negative (AA, but not considered in this paper), AM as cheaters exploitation and MM as mutualisms. Usefull to idenfify assimetrys of outcomes between species.

Ok, until know we just explore our matrices of interaction outcomes. Let's work with coevolution know. First, let's define our coevolution model parameters

```r
# coevolutionary model parameters
phi = 0.2
alpha = 0.2
theta = runif(n_sp, 0, 5)
init = runif(n_sp, 0, 5)
p = 0.1
epsilon = 3
eq_dif = 0.0001
t_max = 1000
```

After that we gonna use the last (and maybe most important) function that we load in the beggining of this tutorial: the CoevoMutAntNet which runs the coevolution process in the community. You can understand the functioning of this function in the file Guide_CoevoMutAntag from my GitHub account.

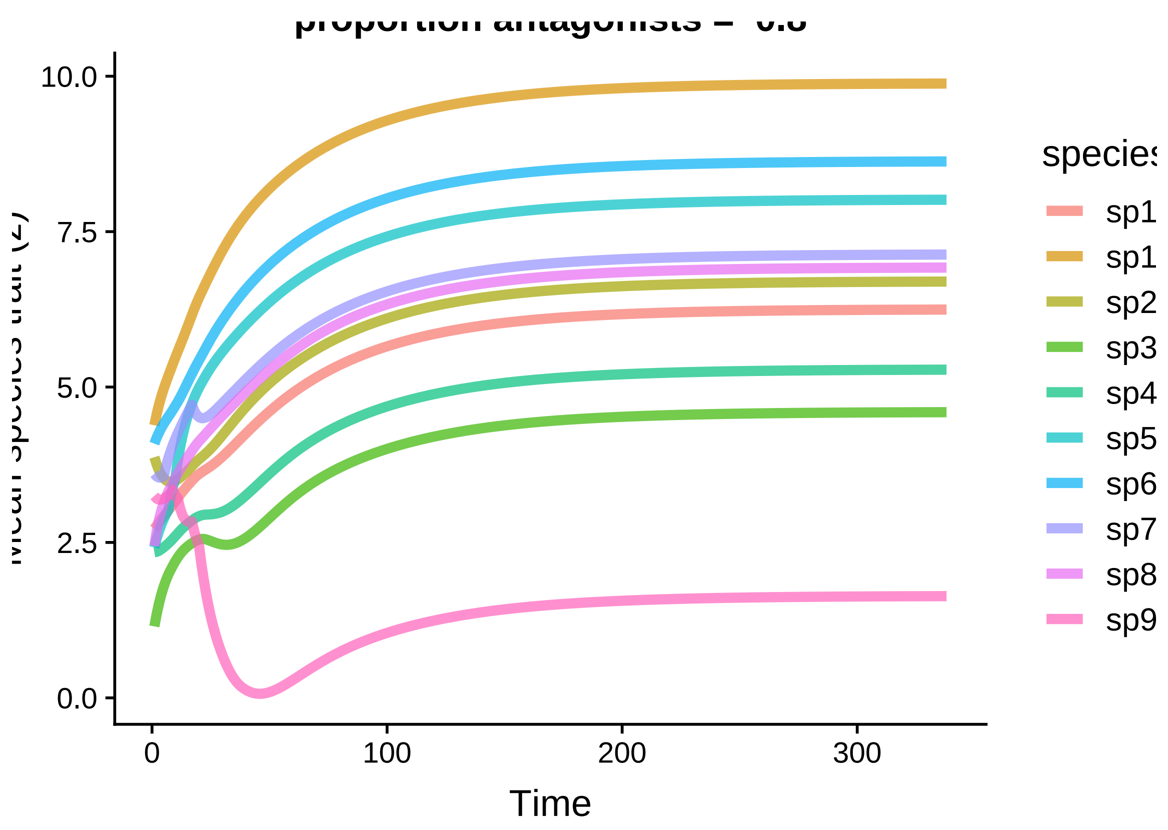
```
# running coevolution simulation
z_mat = CoevoMutAntNet(n_sp, M, V, phi, alpha, theta, init, p, epsilon, eq_dif, t_max)
```

The CoevoMutAntNet function provides an final matrix here called z_mat which has the species in the collums and the timesteps in the rows which each element of this matrix shows the trait values of species. With this matrix we can plot the species traits in time using the reshape2, ggplot2 and cowplot packages to help us. This is the end of the script.

```
# building data frame to plot the results
traits = as.data.frame(z_mat)
n_sp = ncol(traits)
traits_vec = c(as.matrix(traits))
traits_df = data.frame(species = rep(paste("sp", 1:n_sp, sep = ""), each = nrow(traits)
                       time = rep(1:nrow(traits), times = n_sp),
                       trait = traits_vec)

# plotting traits through time
plotar = ggplot(traits_df, aes(x = time, y = trait, color = species)) +
  geom_path(size = 1.8, alpha = 0.7) +
  ggtitle(paste("proportion antagonists = ", antprob)) +
  xlab("Time") +
  ylab("Mean species trait (z)") +
  theme(axis.text.x = element_text(size = 11),
        axis.text.y = element_text(size = 11),
        axis.title = element_text(size = 14),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 12))

plotar
```



Final considerations

You will see that the scripts used in this paper are just variations of this script-base structure that I show in this small tutorial. For example you could use the CentralAntagonize instead of Antagonize to transform only the central mutualistic species in cheaters or use some of the ConDep functions to considerer the temporal variatio of species outcomes and their influence on coevolution.

Finally, if you have some doubts or suggestions please let me know in lucas.camacho@usp.br.

Good bye.