Guide to CoevoMutAntag model

Lucas Arantes Camacho

27/06/2018

Introduction

I write this guide to help the understanding of the model and explain the code to run the simulations. For that, we gonna start speaking a little bit about our model, how it works and after I will explain step-by-step what the R function CoevoMutAntNet does. Hope it helps.

The model

We are trying to describe how a certain average trait Z of a specie i change in time due to coevolution and other selective pressures (abiotic conditions for example). For that, we have our model.

$$Z_i^{(t+1)} = Z_i^{(t)} + \varphi_i\{(1-\gamma_i)[\sum_j^N m_{ij}^{(t)}(Z_j^{(t)} - Z_i^{(t)}) + \sum_j^N v_{ij}^{(t)}(Z_j^{(t)} \pm \epsilon_{ij} - Z_i^{(t)})] + \gamma_i(\theta_i - Z_i^{(t)})\}$$

This trait Z of specie i in the timestep $t+1$ depends of Z one timestep further Z_i^t. The traits change also depends on the heritability and additive genetic variance of Z which is given by the φ_i parameter. After that we have 3 components changing the trait Z: the mutualism, the exploitative interactions and the environmental selection. Let's see the mutualism component first.

$$\sum_j m_{ij}^t(Z_j^t - Z_i^t)$$

In this component we have m_{ij}^t that is the mutualistic evolutionary effects of a species j to the specie i. These component favours the trait matching between Z_i and Z_j. Thus, this values Z_i and Z_j tend to become more similar in time. The second component is the exploitative component

$$\sum_j v_{ij}^t(Z_j^t \pm \epsilon_{ij} - Z_i^t)$$

Similar to mutualism, we also have an evolutionary exploitation effect of j in i which is v_{ij}^t. Note that we have a parameter ϵ, which is our "trait barrier" or the maximum differente of Z_i and Z_j for this exploitative interaction happen. Also, here we classify an victim i and an

explorer j, in which the adaptive peak of i is just beeing smaller or greater than the explorer j Finally, we have the environment component

$$\gamma_i(\theta_i - Z_i^t)$$

which θ_i are the vector with the environmental trait optima for each specie.

For the exploitation and mutualism component, we have the evolutionary effects of these interactions, that we called v_{ij}^t and m_{ij}^t. These evolutionary effects are defined as a relative effect of specie j in i normalized by all the other interactions that i has with other species. We have

$$m_{ij}^{(t)} = M_{ij}^{(t)} \frac{e^{-\alpha(Z_j^{(t)} - Z_i^{(t)})^2}}{\sum_{k,i\neq k}^{N} e^{-\alpha(Z_k^{(t)} - Z_i^{(t)})^2}}$$

$$v_{ij}^{(t)} = V_{ij}^{(t)} \frac{e^{-\alpha(Z_j^{(t)} - Z_i^{(t)})^2}}{\sum_{k,i\neq k}^{N} e^{-\alpha(Z_k^{(t)} - Z_i^{(t)})^2}}$$

where $M_{ij}^{(t)}$ and $V_{ij}^{(t)}$ are the elements from the adjacency matrix that give us the species interactions and α is the sensitivity of selection to trait matching.

The code

Now, let's construct a code in R that simulates the coevolution process following these model. First, we start by set our parameters of interest and construct our matrices of positive (M) and negative (V) effects.

```r
# fixate the random parameters (necessary only for this guide)
set.seed(42)
# number of species
n_sp = 5
# probability of an effect become negative
antprob = 0.5
# herdability and additive genetic variance parameter
phi = 0.2
# sensitivity of evolution due to trait matching
alpha = 0.2
# vector with optimas of Z`s to environment
theta = runif(n_sp, 0, 10)
# vector of initial Z values
init = runif(n_sp, 0, 10)
# importance of environment to changes in Z
p = 0.1
```

```
 # value of max difference between Z`s (barrier)
epsilon = 5
# difference between Z to stop simulation
eq_dif = 0.0001
# max timesteps of simulation
t_max = 1000

# Positive effects adjacency matrix
M = matrix(1, ncol = n_sp, nrow = n_sp)
# no intraespecific interactions
diag(M) = 0
# Negative effects adjacency matrix filled with 0
V = M * 0
```

Great, we have our parameters defined, a matrix of positive effects completely connected and an empty negative effects matrix. Now, we have to transform some links of our M network in negative links. For each interaction in our M matrix, we are going to sample a number between 0 and 1 and following the probability of a certain link become exploitative which is our *antprob* parameter, transform or not the positive effect in negative.

```
  # define the matrix upper triangle
  index = which(upper.tri(M), arr.ind = TRUE)

  for(i in 1:nrow(index)){ # for each element in index
    if(M[index[i,][1], index[i,][2]] == 1){ # if this element is equal to 1
      p = runif(1, 0, 1) # sample a number between 0 and 1
      if(p <= antprob){ # if this number is equal or lower than antprob...
        M[index[i,][2], index[i,][1]] = 0 # M[j,i] element is 0
        V[index[i,][2], index[i,][1]] = 1 # V[j,i] element is 1
      }
    }
  }

  # no intraespecific interactions
  diag(V) = 0
```

Let's look to our matrices and see which interactions become negative

```
M
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    1    1    1
## [2,]    0    0    1    1    1
## [3,]    1    1    0    1    1
## [4,]    0    0    1    0    1
## [5,]    1    0    0    1    0
```

```
V
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    1    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    1    1    0    0    0
## [5,]    0    1    1    0    0
```

Great, but there a couple of things we have to see at this point. First, when we transform an interaction from mutualism to exploitative, we are defining an explorer and a victim. In these scenario, we introduce an negative effect of j in i, but not an negative effect of i in j. The V matrix tell us the negative effects of species in the collums to the species in the rows. Second (and more intuitive), the V and M matrix are complementar. The sum of V and M gives us all the interactions in the network.

Ok, we have our interactions, now we can start thinking in the simulations. For start, let's create a empty matrix called *z_mat* which is our results matrix. In the matrix *z_mat* we have the species in the collums and the timesteps of simulation in the rows. The first line of *z_mat* is the initial trait values.

```
  # matrix to store z values
  z_mat = matrix(NA, nrow = t_max, ncol = n_sp)
  # initial trait values
  z_mat[1, ] = init
```

Ok, know we are entering in a loop, which means that these process will happend several times. Let's start by just saying to R which are our initial values of Z. After this, we get all the interactions that we have in our network just getting the sum of M and V.

```
    # current z values
    z = z_mat[1, ]
    # matrix with all positive and negative effects
    A = M + V
```

We start our simulation by calculating the evolutionary effects between the species in the network. That's our famous Q matrix. To get the Q matrix, we gonna calculate the trait difference between all species in the network. See how our differences of traits matrix (*z_dif*) looks like.

```
    # matrix with all trait differences
    z_dif = t(A * z) - A * z
    z_dif
```

```
##           [,1]      [,2]      [,3]       [,4]       [,5]
## [1,]  0.000000 2.1749237 -3.844294  1.3789634  1.8596883
## [2,] -2.174924 0.0000000 -6.019217 -0.7959602 -0.3152353
## [3,]  3.844294 6.0192172  0.000000  5.2232569  5.7039819
```

```
## [4,] -1.378963 0.7959602 -5.223257  0.0000000  0.4807249
## [5,] -1.859688 0.3152353 -5.703982 -0.4807249  0.0000000
```

Ok, now we calculate the $e^{-\alpha(Z_j^{(t)}-Z_i^{(t)})^2}$ component using the trait differences that we already have. We are not interested in intraespecific interactions, so we gonna insert "0" in the diagonal of the matrix.

```
    # matrix Q
    Q = A * (exp(-alpha * (z_dif ^ 2)))
    Q
```

```
##            [,1]         [,2]         [,3]        [,4]       [,5]
## [1,] 0.00000000 0.3882683397 0.0520412538 0.683650800 0.50073005
## [2,] 0.38826834 0.0000000000 0.0007128817 0.880988645 0.98032154
## [3,] 0.05204125 0.0007128817 0.0000000000 0.004268544 0.00149283
## [4,] 0.68365080 0.8809886454 0.0042685436 0.000000000 0.95483255
## [5,] 0.50073005 0.9803215395 0.0014928297 0.954832551 0.00000000
```

```
    # intraespecific effects are not allowed
    diag(Q) = 0
    Q
```

```
##            [,1]         [,2]         [,3]        [,4]       [,5]
## [1,] 0.00000000 0.3882683397 0.0520412538 0.683650800 0.50073005
## [2,] 0.38826834 0.0000000000 0.0007128817 0.880988645 0.98032154
## [3,] 0.05204125 0.0007128817 0.0000000000 0.004268544 0.00149283
## [4,] 0.68365080 0.8809886454 0.0042685436 0.000000000 0.95483255
## [5,] 0.50073005 0.9803215395 0.0014928297 0.954832551 0.00000000
```

Now we gonna normalize our Q matrix and get $\frac{e^{-\alpha(Z_j^{(t)}-Z_i^{(t)})^2}}{\sum_{k,i\neq k}^{N} e^{-\alpha(Z_k^{(t)}-Z_i^{(t)})^2}}$

```
    # normalizing the Q matrix
    Q_n = Q / apply(Q, 1, sum)
    Q_n
```

```
##           [,1]       [,2]         [,3]       [,4]       [,5]
## [1,] 0.0000000 0.23897989 0.0320314889 0.42078834 0.30820028
## [2,] 0.1725414 0.00000000 0.0003167953 0.39149980 0.43564204
## [3,] 0.8893583 0.01218278 0.0000000000 0.07294722 0.02551169
## [4,] 0.2708879 0.34908051 0.0016913559 0.00000000 0.37834022
## [5,] 0.2054381 0.40220350 0.0006124739 0.39174595 0.00000000
```

Great, we already have our normalized matrix of evolutionary effects between the interacting species in the network. But we have a parameter that says how much the environment is important to trait changes (γ_i). If we are considering γ_i as the importance of environment, $(1-\gamma_i)$ should be the importance of interactions to trait changes. We gonna multiply these importance (in the code, called p) of interactions to our Q matrix.

```
    # multiplying each row i of matrix Q by (1 - p)
    Q_m = Q_n * (1 - p)
    Q_m
```

```
##             [,1]       [,2]         [,3]      [,4]       [,5]
## [1,] 0.00000000 0.10507163 0.0140831968 0.1850069 0.13550557
## [2,] 0.07586079 0.00000000 0.0001392845 0.1721296 0.19153754
## [3,] 0.39102172 0.00535637 0.0000000000 0.0320725 0.01121666
## [4,] 0.11910054 0.15347927 0.0007436338 0.0000000 0.16634381
## [5,] 0.09032440 0.17683571 0.0002692847 0.1722379 0.00000000
```

Ok, we are ready to calculate our selection differentials. Basically, calculate how much the Z trait will change. We start doing it for the environment. For that, we compute $\gamma_i(\theta_i - Z_i^t)$

```
    # response to selection related to the environment
    r_env = phi * p * (theta - z)
```

Great, now the mutualism part. First we get only the mutualistic interactions and multiply the evolutionary effects by the trait differences between interacting species. We are just computing $\sum_j m_{ij}^t(Z_j^t - Z_i^t)$.

```
    # calculating selection differentials to mutualism
    sel_dif_mut = M * Q_m * z_dif
    # response to selection related to mutualism
    r_mut = phi * apply(sel_dif_mut, 1, sum)
```

Ok, finally we compute the exploitative part of trait changes, but before that, we have some conditions. First, the trait difference between i and j has to be minor than our parameter ϵ. The second condition tell us that if the trait of i is minor than j, we have the sum of ϵ in the trait differences $(Z_j^t + \epsilon_{ij} - Z_i^t)$. Otherwise, we have the subtract of ϵ in trait differences $Z_j^t - \epsilon_{ij} - Z_i^t$. Thus, in one situation we are increasing the trait values for the victins and another situation we are decreasing these trait values.

```
    # create V_m to use in exploitative selection differential
    V_m = V
    # excluding interactions of traits that are larger than the barrier
    V_m[abs(z_dif) > epsilon] = 0
    # matrix with barrier (epsilon) values
    epsilon_plus = (z_dif < 0) * matrix(epsilon, n_sp, n_sp)
    # matrix wih -epsilon values
    epsilon_minus = (z_dif > 0) * matrix(-epsilon, n_sp, n_sp)
    # adding barrier values to trait differences
    z_dif_a = z_dif + epsilon_plus + epsilon_minus
```

See in the code above how we create a matrix for each situation of increasing and decreasing the trait value of the victim, them we just apply these computations to our trait differences. Ok, now we are ready to calculate the selection differentials due to exploitative interactions

using our V matrix, the Q matrix and the trait differences between species.

```
    # calculating selection differentials
    sel_dif_ant = V_m * Q_m * z_dif_a
    # response to selection related to exploitative interactions
    r_ant = phi * apply(sel_dif_ant, 1, sum)
```

We have our selection differentials. Now we update our initial z values. The next line of our result matrix is equal to the sum of initial z values and our selection differentials.

```
    # updating z values
    z_mat[2, ] = z + r_env + r_mut + r_ant
```

Remember we are on a loop? These process will be repeated several times. There's a certain moment in the simulation where the traits doesn't change too much. We can define a certain moment, where the changing in Z is very low that we can just stop our simulations, the traits is in equilibrium. We define in this way

```
    # computing the mean difference between old and new z values
    dif = mean(abs(z - z_mat[2, ]))
    if (dif < eq_dif) # if the difference in very low
      break # stop the simulation
```

Of course, maybe these equilibrium state doesn't happend, so the simulation will go until we reach the maximum time of simulations ($t\_max = 1.000$)

Ok, now we just gonna say to R show to us our resulting matrix z_mat, now with a second line of trait values.

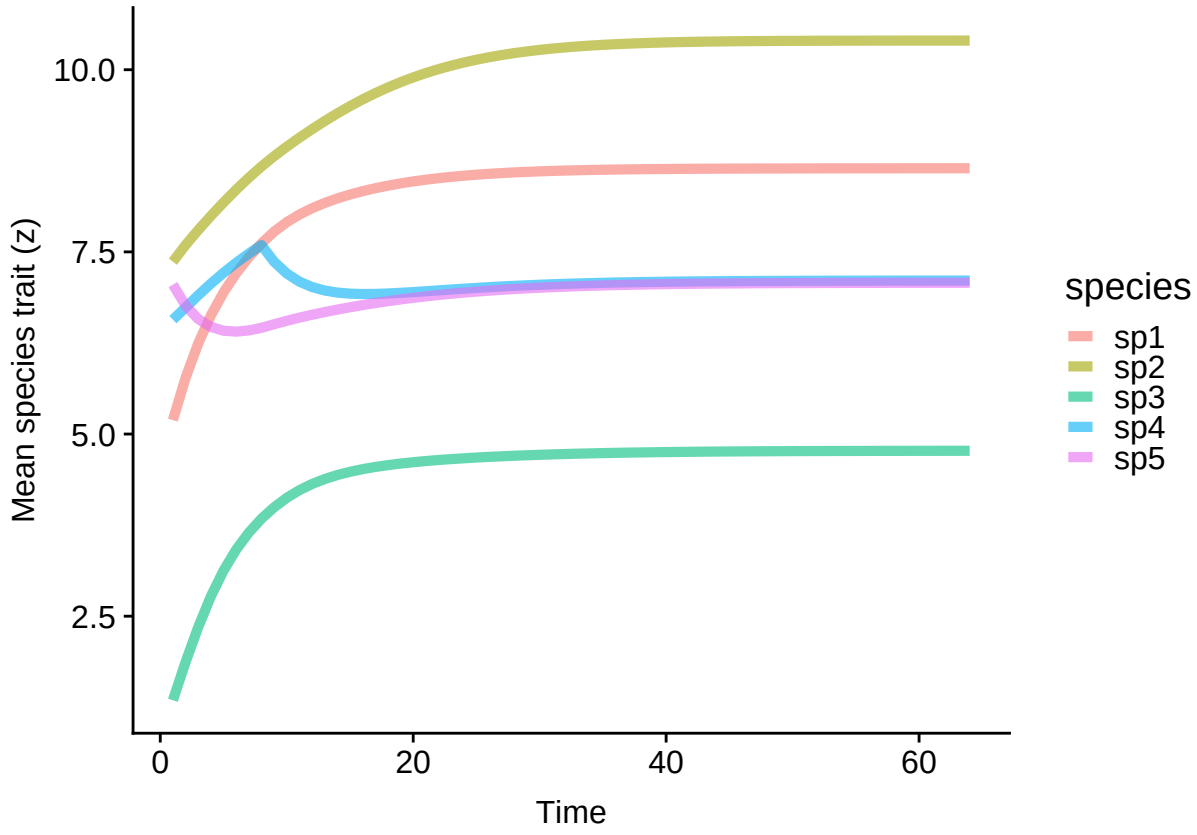
```
 # return z_matrix
 return(z_mat[1:2, ])
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 5.190959 7.365883 1.346666 6.569923 7.050648
## [2,] 5.770718 7.593779 1.869806 6.736731 6.763847
```

And that's end our simulation process. We just simulate one timestep, these process will happend until reach our t_max already defined in the parameters or will stop when the traits become very similar. Below we have our complete code inside a *for* loop and the final graph showing the timesteps in the x axis and the trait values in y axis.

```
  # simulation runs for a maximum of t_max timesteps
  for (r in 1:(t_max - 1)) {
    # current z values
    z = z_mat[r, ]
    # matrix with all positive and negative effects
    A = M + V
    # matrix with all trait differences
    z_dif = t(A * z) - A * z
```

```
  # matrix Q
  Q = A * (exp(-alpha * (z_dif ^ 2)))
  # intraespecific effects are not allowed
  diag(Q) = 0
  # normalizing the matrix
  Q_n = Q / apply(Q, 1, sum)
  # multiplying each row i of matrix Q by (1 - p[i])
  Q_m = Q_n * (1 - p)

  # response to selection related to the environment
  r_env = phi * p * (theta - z)

  # calculating selection differentials to mutualism
  sel_dif_mut = M * Q_m * z_dif
  # response to selection related to mutualism
  r_mut = phi * apply(sel_dif_mut, 1, sum)

  V_m = V
  # excluding interactions of traits that are larger than the barrier
  V_m[abs(z_dif) > epsilon] = 0

  # matrix with barrier (epsilon) values
  epsilon_plus = (z_dif < 0) * matrix(epsilon, n_sp, n_sp)
  # matrix wih -epsilon values
  epsilon_minus = (z_dif > 0) * matrix(-epsilon, n_sp, n_sp)
  # adding barrier values to trait differences
  z_dif_a = z_dif + epsilon_plus + epsilon_minus
  # calculating selection differentials
  sel_dif_ant = V_m * Q_m * z_dif_a
  # response to selection related to exploitation
  r_ant = phi * apply(sel_dif_ant, 1, sum)

  # updating z values
  z_mat[r+1, ] = z + r_env + r_mut + r_ant

  # computing the mean difference between old and new z values
  dif = mean(abs(z - z_mat[r+1, ]))
  if (dif < eq_dif)
    break
}

return(z_mat[1:(r+1), ])
```



Final statements

That's all for now. Please, let me know if you find some error in my code or model by sending an e-mail to lucas.camacho@usp.br. You can find the R code in my GITHUB repository, in the CoevoMutAntNet.R file from *R/functions* folder.

Also, you can find in this repository all the others functions and scripts that I made.

Good Bye