Tutorial: coevolution in mutualistic networks

Lucas P Medeiros and Paulo R Guimarães Jr

November, 2018

Contents

1 Introduction **1**

2 Mutualistic networks **2**
2.1 Architecture of mutualistic networks 2

3 Coevolution in mutualistic networks **11**
3.1 Mathematical model of coevolution in mutualistic networks 11

4 Conclusions **33**

References **33**

1 Introduction

Coevolution is the reciprocal evolutionary change in interacting species. Research on coevolution has primarily focused on simple systems in which two or three species interact with each other. Well-known examples of coevolving species include garter snakes that prey upon toxic newts (Hanifin *et al.* 2008) and *Greya* moths that pollinate while laying their eggs on *Lithophragma* plants (Thompson *et al.* 2017). However, even in these specialized systems involving few species, other species may participate in the interaction. In the *Greya*-*Lithophragma* interaction, for instance, several other species of insects may pollinate *Lithophragma* species at some localities (Thompson & Cunningham 2002). The observation that most interacting assemblages include more than a handful of species suggests that coevolution is often a multispecies phenomenon and, therefore, we need specific tools to understand how coevolution operates in species-rich systems.

In this tutorial, we present a theoretical approach to study coevolution in species-rich interactions. In particular, we will focus on mutualisms by showing some recent results on how coevolution proceeds in species-rich mutualistic networks (Guimarães Jr *et al.* 2017). We divided this tutorial in two parts. In the first part, we introduce the idea of mutualistic networks and work with some examples in order to show how these networks vary in the organization of interactions among species. In the second part, we present a mathematical model of coevolution and show how coevolution is expected to operate in different kinds of mutualistic networks.

Throughout this tutorial we will use the **R** environment to perform calculations, visualize networks, introduce models, and run simulations. The idea is that the reader should be able to reproduce the results contained here by using **R**. Although the objective here is not to teach how to code in **R**, we will explain our code whenever we think that it goes beyond a basic understanding. We hope you enjoy this tutorial and please let us know if you have any questions about or suggestions for this tutorial!

2 Mutualistic networks

Species rarely interact in small groups with two or three species. Different types of interactions (e.g., predation, mutualism) frequently form large networks of interacting species in ecological communities. Food webs, for instance, depict who eats whom in ecological communities (McCann 2011). Mutualistic networks, on the other hand, represent interactions that benefit both partners (e.g., pollinators and plants) (Bascompte & Jordano 2013). In what follows, we will mathematically introduce mutualistic networks and then describe different patterns in the organization of species interactions that have been reported for these networks.

2.1 Architecture of mutualistic networks

Species interaction networks may be represented by a matrix $\mathbf{A}$, in which an entry a_{ij} contains information on the interaction between species i and species j (Bascompte & Jordano 2013). Mutualisms usually contain two distinct sets of interacting species (e.g., pollinators and plants) and mutualistic networks are therefore represented by bipartite networks in which interactions are only allowed to occur across sets. In a bipartite matrix, rows represent one set of species (e.g., plant species) whereas columns represent the other set (e.g., pollinator species). The entries a_{ij} in bipartite matrices may indicate whether species i and j interact or not (i.e., $a_{ij} = 1$ or $a_{ij} = 0$) or it may represent the strength of interaction between species i and j (e.g., total number of visits of pollinator species j to plant species i).

Now that we have introduced the basic terminology and concepts in mutualistic networks, let us work with some examples in **R**. First, we will build a hypothetical bipartite network containing 3 plant species, which will be placed in the rows of matrix $\mathbf{A}$, and 4 pollinator species, which will be placed in the columns. To keep things simple, we will use only 0's and 1's for the entries a_{ij}. In addition, we will randomly choose who interacts with whom by using the `sample` function to sample 1's in the matrix with probability $p = 0.5$.

```r
set.seed(20)
n_plant <- 3
n_pol <- 4
p <- 0.5
(A <- matrix(data = sample(x = c(0, 1), size = n_plant*n_pol,
                           replace = TRUE, prob = c(1 - p, p)),
             nrow =  n_plant, ncol = n_pol))
```

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

The most obvious characteristics that we can use to describe a matrix are its size and shape. The size of a bipartite ecological network is the *species richness*, which is given by the number of species in the rows plus the number of species in the columns ($N = N_1 + N_2$). In the example above, the species richness is $N = N_1 + N_2 = 3 + 4 = 7$. The difference in the number of rows and columns gives us the overall matrix shape. If those numbers are equal (i.e., $N_1 = N_2$) we will have a square matrix. Otherwise, if they are different (i.e., $N_1 < N_2$ or $N_1 > N_2$), we will have a matrix with a rectangular shape, meaning that there are more pollinators than plants or vice versa. Another important property of a mutualistic network is its *connectance*, which tells us what the overall density of interactions in the network is (Bascompte & Jordano 2013). More specifically, the connectance corresponds to the fraction of all possible interactions that actually occur in the network and is calculated as $C = \frac{I}{N_1 N_2}$, where I represents the total number of interactions in the network ($I = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} a_{ij}$). Let us now visualize two networks with different connectance values by varying the probability that a link occurs between two species. To do so, we will create matrices with $N_1 = 30, N_2 = 20$ and use either $p = 0.1$ or $p = 0.9$ as the probability that $a_{ij} = 1$.

```r
n_plant <- 30
n_pol <- 20
p_low <- 0.1
A_low <- matrix(data = sample(x = c(0, 1), size = n_plant*n_pol,
                              replace = TRUE, prob = c(1 - p_low, p_low)),
                nrow = n_plant, ncol = n_pol)
# computing connectance
(connect_A_low <- sum(A_low)/(n_pol*n_plant))
```

```
## [1] 0.1116667
```

```r
p_high <- 0.9
A_high <- matrix(data = sample(x = c(0, 1), size = n_plant*n_pol,
                               replace = TRUE, prob = c(1 - p_high, p_high)),
                 nrow = n_plant, ncol = n_pol)
# computing connectance
(connect_A_high <- sum(A_high)/(n_pol*n_plant))
```

```
## [1] 0.8916667
```

Now that we have generated our two matrices `A_low` and `A_high` and calculated their connectance values, let us use the package `ggplot2` to plot the two bipartite matrices and observe the difference in the density of interactions.

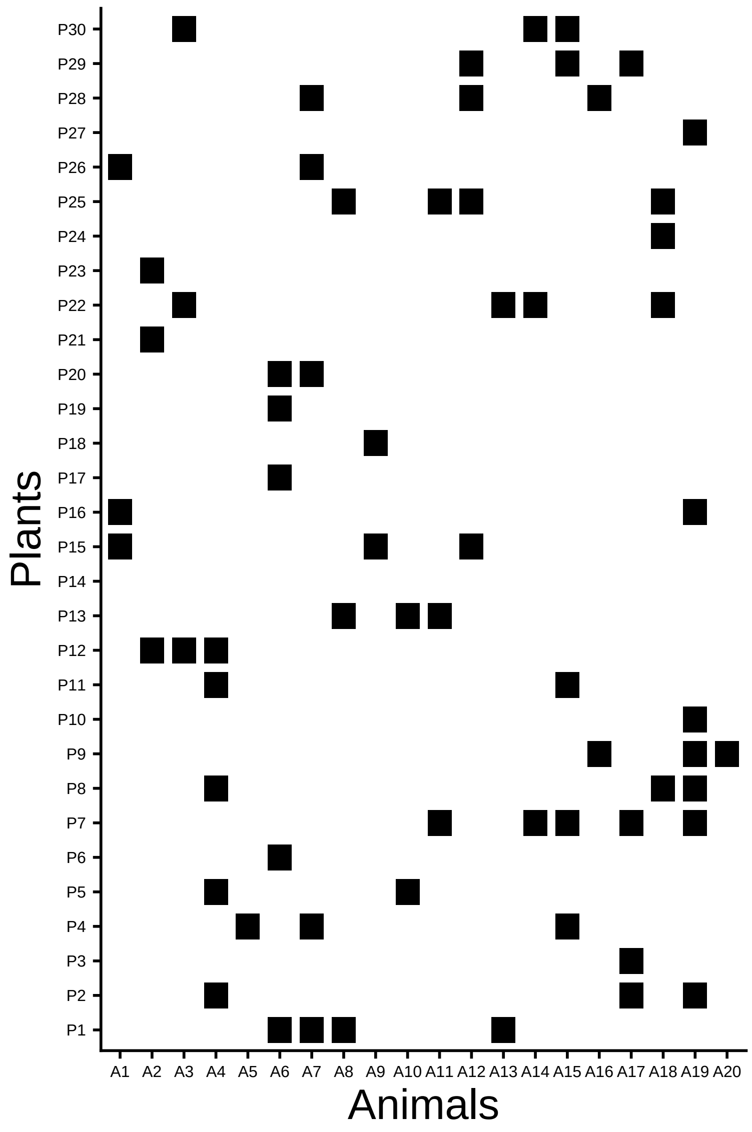
```r
# ---------- Installing and loading packages ----------
# installing packages if not installed yet
if (!("ggplot2" %in% installed.packages()[ ,"Package"]))
  install.packages("ggplot2")
if (!("cowplot" %in% installed.packages()[ ,"Package"]))
  install.packages("cowplot")
if (!("viridis" %in% installed.packages()[ ,"Package"]))
  install.packages("viridis")
if (!("reshape" %in% installed.packages()[ ,"Package"]))
  install.packages("reshape")
# loading packages
library(ggplot2)
library(cowplot)
library(viridis)
library(reshape)
```

```r
# ---------- Plotting network with low connectance ----------
# adding labels to rows and columns
rownames(A_low) <- paste("P", 1:n_plant, sep = "")
colnames(A_low) <- paste("A", 1:n_pol, sep = "")
# creating a list of interactions
melted_A_low <- melt(A_low)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

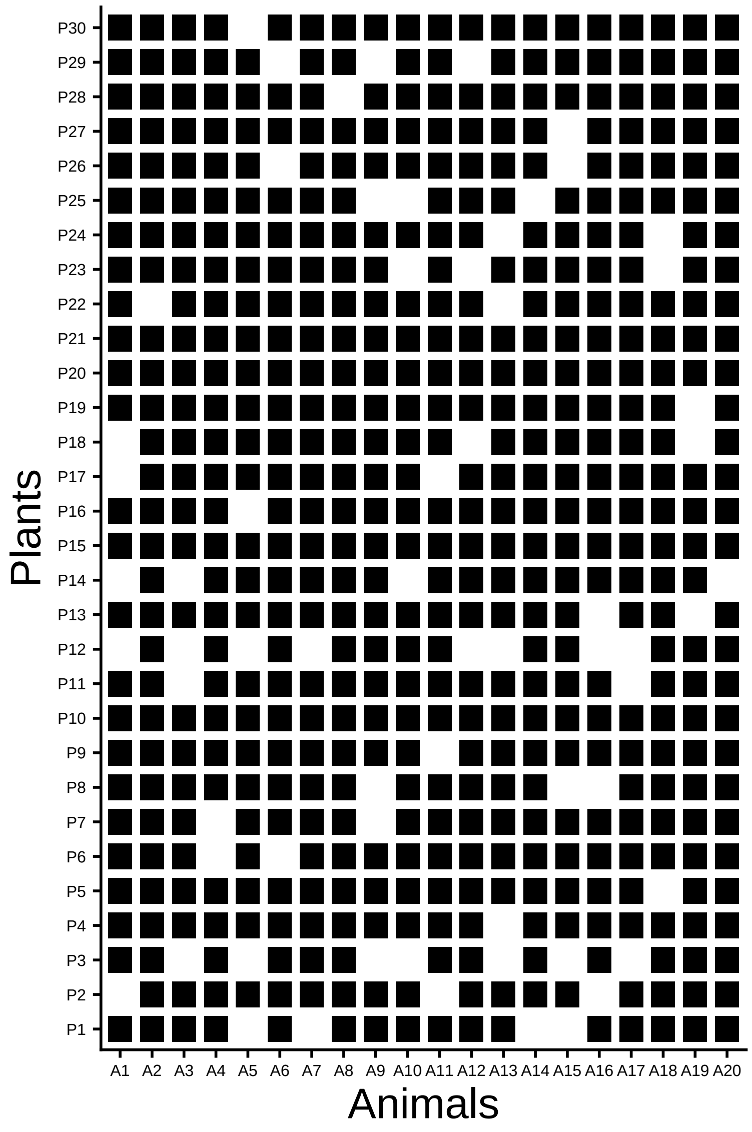
```r
melted_A_low$color[melted_A_low$value == 1] <- "black"
melted_A_low$color[melted_A_low$value == 0] <- "white"
# plotting
ggplot(data = melted_A_low, aes(x = X2, y = X1, fill = value,
                                    height = 0.75, width = 0.75)) +
  geom_tile(fill = melted_A_low$color) +
  ylab("Plants") +
  xlab("Animals") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 6),
        axis.text.y = element_text(size = 6),
        axis.title = element_text(size = 16))
```



```
# ---------- Plotting network with high connectance ----------
# adding labels to rows and columns
rownames(A_high) <- paste("P", 1:n_plant, sep = "")
colnames(A_high) <- paste("A", 1:n_pol, sep = "")
# creating a list of interactions
melted_A_high <- melt(A_high)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
melted_A_high$color[melted_A_high$value == 1] <- "black"
melted_A_high$color[melted_A_high$value == 0] <- "white"
# plotting
ggplot(data = melted_A_high, aes(x = X2, y = X1, fill = value,
                                 height = 0.75, width = 0.75)) +
  geom_tile(fill = melted_A_high$color) +
  ylab("Plants") +
  xlab("Animals") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 6),
        axis.text.y = element_text(size = 6),
        axis.title = element_text(size = 16))
```



The first matrix depicted above is closer to what we usually find in empirical datasets. Indeed, ecological networks usually have very low values of connectance, meaning that many interactions are not observed

(Bascompte & Jordano 2013). Metrics such as connectance describe general patterns of interaction that do not take into account the identity of interacting species. It tells us about the density of interactions but not about their organization in the network. We will now present two other metrics that aim to capture how interactions are organized in mutualistic networks: *nestedness* and *modularity*.

Let us first explore nestedness. Nestedness occurs when specialists (i.e., species with few interactions) interact with a proper subset of the species that generalists (i.e., species with few interactions) interact with (Bascompte *et al.* 2003; Bascompte & Jordano 2013). To visualize and compute nestedness, let us first load an empirical pollination network from Northeastern Brazil that contains 13 plant species (matrix rows) and 13 pollinator species (matrix columns) (Bezerra *et al.* 2009). This network was downloaded directly from the Web of Life database.

```
# loading data
A <- as.matrix(read.csv("empirical_networks/M_PL_059.csv",
                        header = TRUE, row.names = 1))
n_plant <- nrow(A)
n_pol <- ncol(A)
# changing labels
rownames(A) <- paste("P", 1:n_plant, sep = "")
colnames(A) <- paste("A", 1:n_pol, sep = "")
A
```

```
##       A1   A2   A3  A4  A5  A6  A7  A8  A9 A10 A11 A12 A13
## P1  1368 1364  740 460 416 256   0 328 364 368   0   0   0
## P2   924  320 2108 464 284   0   0   0  28   0   0   0   0
## P3   396  468  108 140 272 652 912 364  44   0 368 164  84
## P4   764  680  528 308 404 300   0  28   0   0   0  76   0
## P5   740  656  528 332   0 324   0 116 116   0   0   0   0
## P6   556  512  356 132 524   0   0   0   0   0   0   0   0
## P7   604  452  432 200   0   0   0   0   0   0   0   0   0
## P8   504  816    0   0   0   0   0   0   0   0   0   0   0
## P9   292  300  244 116   0   0   0   0   0   0   0   0   0
## P10  228  224  124 120   0   0   0   0   0   0   0   0   0
## P11  240  164   68   0 196   0   0   0   0   0   0   0   0
## P12  268  196    0   0 164   0   0   0   0   0   0   0   0
## P13  188  244    0  96   0   0   0   0   0   0   0   0   0
```

As we can see above, each entry a_{ij} in matrix $\mathbf{A}$ contains the total number of visits of pollinator j to plant i. To simplify our nestedness analysis, let us first convert all values greater than 0 to 1 in order to obtain a binary interaction matrix.

```
A[A > 0] <- 1
A
```

```
##     A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13
## P1   1  1  1  1  1  1  0  1  1   1   0   0   0
## P2   1  1  1  1  1  0  0  0  1   0   0   0   0
## P3   1  1  1  1  1  1  1  1  1   0   1   1   1
## P4   1  1  1  1  1  1  0  1  0   0   0   1   0
## P5   1  1  1  1  0  1  0  1  1   0   0   0   0
## P6   1  1  1  1  1  0  0  0  0   0   0   0   0
## P7   1  1  1  1  0  0  0  0  0   0   0   0   0
## P8   1  1  0  0  0  0  0  0  0   0   0   0   0
## P9   1  1  1  1  0  0  0  0  0   0   0   0   0
## P10  1  1  1  1  0  0  0  0  0   0   0   0   0
## P11  1  1  1  0  1  0  0  0  0   0   0   0   0
## P12  1  1  0  0  1  0  0  0  0   0   0   0   0
## P13  1  1  0  1  0  0  0  0  0   0   0   0   0
```

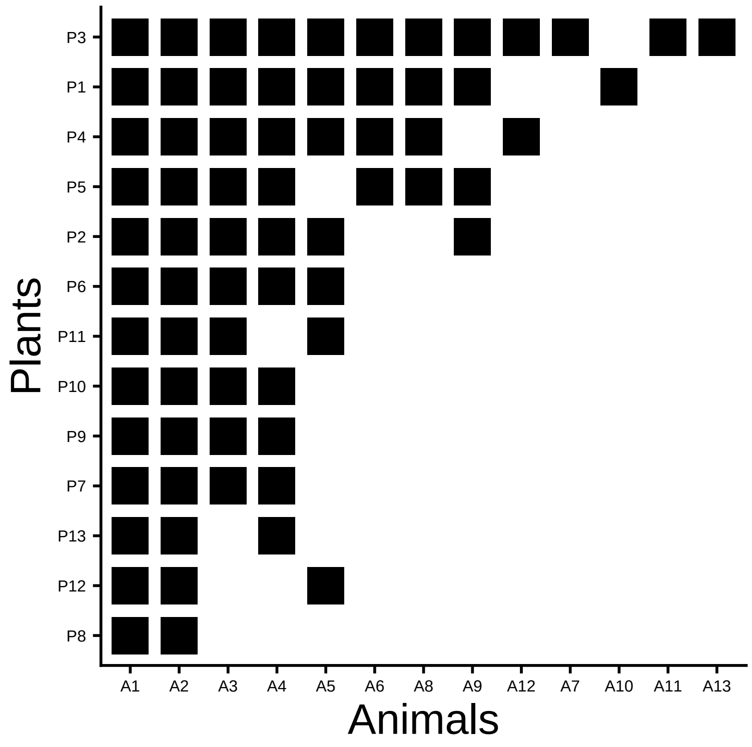
Let us now visualize this matrix using the `ggplot2` package. Before we do so, we will order our list of interactions so that species in the interaction matrix are ordered by their *degree*, which is the total number of interactions of a species in the network. The degree of a species is calculated as $k_i = \sum_{j=1}^{N_2} a_{ij}$ for species in the rows or as $k_j = \sum_{i=1}^{N_1} a_{ij}$ for species in the columns of the matrix. Ordering species by degree will help us to visualize the nested pattern.

```r
# computing species degree
k_plant <- apply(A, 1, sum)
k_pol <- apply(A, 2, sum)
# creating a list of interactions
melted_A <- melt(A)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```r
melted_A$color[melted_A$value == 1] <- "black"
melted_A$color[melted_A$value == 0] <- "white"
# reordering the levels of the factors X1 and X2 according to species degree
melted_A$X1 <- factor(melted_A$X1,
                      levels = names(sort(k_plant, decreasing = FALSE)))
melted_A$X2 <- factor(melted_A$X2,
                      levels = names(sort(k_pol, decreasing = TRUE)))
# plotting
ggplot(data = melted_A, aes(x = X2, y = X1, fill = value,
                              height = 0.75, width = 0.75)) +
  geom_tile(fill = melted_A$color) +
  ylab("Plants") +
  xlab("Animals") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 6),
        axis.text.y = element_text(size = 6),
        axis.title = element_text(size = 16))
```



It is clear that, in the network above, the interactions of specialists are proper subsets of the interactions of generalists. In other words, as we move down in the interaction matrix (i.e., from generalists to specialists), most interactions are nested within the interactions of the species above. But, how can we compute the value of nestedness of a given matrix? Although there are several metrics available to calculate nestedness, here we will focus on a metric called Nestedness based on Overlap and Decreasing Fill (*NODF*) (Almeida-Neto *et al.* 2008). The rationale behind this metric is that nestedness is related to a high overlap in the interactions of species from one set (e.g., plant species) as we move from generalist to specialist species in the matrix (i.e., decreasing fill). The formula to calculate *NODF* in a bipartite matrix is:

$$NODF = \frac{\sum_{i<j}^{N_1} M_{ij} + \sum_{i<j}^{N_2} M_{ij}}{[\frac{N_1(N_1-1)}{2}] + [\frac{N_2(N_2-1)}{2}]}$$

in which the sum on the left is over all pairs of species in the rows (e.g., plants) and the sum on the right is over all pairs of species in the columns (e.g., pollinators). These sums correspond to the comparisons that we make between species pairs as we move down (or left) in the interaction matrix. For each pair of species i and j from the same set, M_{ij} is defined in the following way:

$$M_{ij} = \begin{cases} \frac{n_{ij}}{\min(k_i,k_j)}, & \text{if } k_i \neq k_j \\ 0, & \text{if } k_i = k_j \end{cases}$$

in which n_{ij} is the number of common interactions between i and j (i.e., number of shared 1's between i and j) and k_i and k_j are the degrees of these species. The denominator in the NODF equation represents the total number of species pairs for which we calculate M_{ij}. We use this denominator to compute the mean value of M_{ij} for the whole network. Now that we have shown how nestedness is calculated let us use the function `nested` from the `bipartite` package to compute the value of nestedness of the pollination network presented above.

```r
# installing package if not installed yet
if (!("bipartite" %in% installed.packages()[ ,"Package"]))
  install.packages("bipartite")
```

```
# loading package
library(bipartite)
# computing nestedness
(NODF <- nested(A, method = "NODF2"))
```

```
##    NODF2
## 84.93106
```

The value of nestedness of the network above is very high, given that *NODF* ranges from 0 (no nestedness) to 100 (perfect nestedness). Although it is desirable to compare such value against a distribution of *NODF* values generated according to a null model, we will not perform this analysis here and the interested reader may refer to Bascompte *et al.* (2003).

Now let us move on to explore modularity in mutualistic networks. This will be our final incursion into the structural patterns of mutualistic networks before we begin analyzing coevolution in these networks. Modularity occurs when the network can be divided in different subsets of species, such that there are more interactions within than between different subsets (Olesen *et al.* 2007). Such subsets are called modules. To visualize and compute modularity, we will use a network containing the interactions between protective ants and their host myrmecophyte plants in the Peruvian rain forest (Davidson *et al.* 1989). This network contains 8 plant species and 16 ant species and was downloaded directly from the Web of Life database.

```
# loading data
A <- as.matrix(read.csv("empirical_networks/M_PA_001.csv",
                        header = TRUE, row.names = 1))
# placing plants in rows and ants in columns
A <- t(A)
n_plant <- nrow(A)
n_ant <- ncol(A)
# changing labels
rownames(A) <- paste("P", 1:n_plant, sep = "")
colnames(A) <- paste("A", 1:n_ant, sep = "")
A
```

```
##    A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16
## P1 35  0  0  0  0  0  0  4  0   0   0   0   2   0   1   1
## P2  0 30  0  0  0  0  0  0  0   0   0   0   0   0   0   0
## P3  0  0 19  0  0  0  0  0  0   5   0   0   0   0   0   0
## P4  0  0  0  7  8  0  0  0  0   0   0   0   0   0   0   0
## P5  0  0  0  0  0  0 10  0  0   0   4   0   0   0   0   0
## P6  0  0  0  0  0 12  0  0  0   0   0   2   0   0   0   0
## P7  0  0  0  6  5  0  0  0  0   0   0   0   0   0   0   0
## P8  0  0  0  0  0  0  0  2  5   0   0   0   0   1   0   0
```

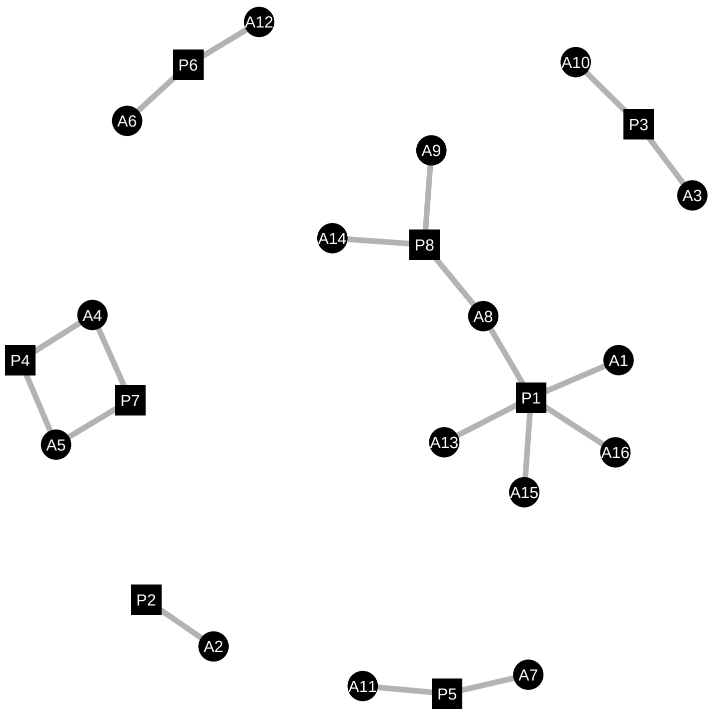
Just like the pollination network that we have analyzed previously, this ant-plant network also contains quantitative data on the presence of each ant species on each plant species. To continue our modularity analysis we will again simplify this dataset by setting all a_{ij} values greater than 0 to 1.

```
A[A > 0] <- 1
A
```

```
##    A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16
## P1  1  0  0  0  0  0  0  1  0   0   0   0   1   0   1   1
## P2  0  1  0  0  0  0  0  0  0   0   0   0   0   0   0   0
## P3  0  0  1  0  0  0  0  0  0   1   0   0   0   0   0   0
## P4  0  0  0  1  1  0  0  0  0   0   0   0   0   0   0   0
## P5  0  0  0  0  0  0  1  0  0   0   1   0   0   0   0   0
## P6  0  0  0  0  0  1  0  0  0   0   0   1   0   0   0   0
## P7  0  0  0  1  1  0  0  0  0   0   0   0   0   0   0   0
## P8  0  0  0  0  0  0  0  1  1   0   0   0   0   1   0   0
```

Now, let us visualize this network to have a better idea of the modular pattern. To do so, we will now take a different approach and plot this network using nodes and links. Each square will represent a plant species, each circle will represent an ant species, and each link will represent an interaction between two species. We will use the function `ggnet2` from the package `GGally` in order to plot this network.

```r
# installing package if not installed yet
if (!("GGally" %in% installed.packages()[ ,"Package"]))
  install.packages("GGally")
# loading package
library(GGally)
# creating network object for plotting
net <- network(A, matrix.type = "bipartite",
               directed = FALSE, ignore.eval = FALSE)
# plotting
ggnet2(net, node.size = 5, color = "black", edge.color = "gray70",
       edge.size = 1, mode = "kamadakawai",
       label = TRUE, label.color = "white", label.size = 2,
       shape = c(rep(15, n_plant), rep(16, n_ant)))
```



Using the above representation of this ant-plant network, it is clear how the network is structured as disconnected subsets of species that only interact among each other. Now let us compute a value of modularity for this ant-plant network. To compute modularity we will use the metric Q, developed by Barber (2007), which is well suited for bipartite networks. For a given partition of the network in modules, the formula for metric Q is given by the following equation:

$$Q = \sum_{i=1}^{n} \left(\frac{e_i}{I} - \frac{d_i^{N_1}}{L} \frac{d_i^{N_2}}{I} \right)$$

where n is the number of modules, e_i is the total number of interactions within module i, I is the total number of interactions in the network, and $d_i^{N_1}$ ($d_i^{N_2}$) is the sum of the degrees of species from the first (or second) set within module i (Barber 2007; Bascompte & Jordano 2013). This modularity metric is based on the fact that modularity will be higher if there are many interactions within modules and few interactions between modules. In this sense, Q computes, for every module, the difference between the fraction of links within module i (first term in the equation) and the expected fraction of links within module i (second term in the equation). Let us now use the function `computeModules` from the `bipartite` package to compute Q for our ant-plant network. Note that `computeModules` computes a modularity metric for a weighted interaction matrix, which reduces to Q if the input matrix is binary (Beckett 2016).

```
# computing modularity
Q <- computeModules(A)
Q@likelihood
```

```
## [1] 0.7783934
```

The modularity value above is very high, given that Q varies from 0 (no modularity) to $1 - \frac{1}{n}$ (maximum modularity). Differently from nestedness, modularity measures need to be optimized given that different organizations of the network in modules will render different modularity values. Thus, the Q value presented above represents the maximum value for this network obtained via an optimization algorithm. Just like we mentioned for nestedness, modularity values need to be compared against values obtained via a null model so that we can conclude if a network is significantly modular. More details about the different structural metrics presented here can be found in the book by Bascompte & Jordano (2013). Our aim here was to give an overview of a few metrics and introduce ways to compute and visualize those patterns using **R**. In the next section we will present an approach to study coevolutionary dynamics in mutualistic networks. After presenting the mathematical model we will perform some simulations and analyses that take into consideration the different network architectures explored here.

3 Coevolution in mutualistic networks

The idea that interacting species undergo reciprocal evolutionary change is now a general consensus in ecology and evolution. Several approaches ranging from experimental coevolution (Brockhurst & Koskella 2013) to phylogenetic studies spanning large temporal and spatial scales (Gómez *et al.* 2010) have been used to study how coevolution shapes biodiversity. A particular problem that has been receiving more and more attention in recent years is how coevolution proceeds in assemblages of interacting species containing dozens or even hundreds of species (Guimarães Jr *et al.* 2011, 2017; Nuismer *et al.* 2013, 2018; Andreazzi *et al.* 2017; Medeiros *et al.* 2018). In this section of the tutorial, we take a theoretical approach to this problem by introducing a mathematical model of coevolution in mutualistic networks and running simulations and performing an analytical study of the model. Using this approach, we will show how network architecture is expected to shape the patterns of adaptation and coadaptation between interacting species in mutualistic networks. Furthermore, we will show how evolutionary effects between non-interacting species (i.e., indirect effects) are expected to emerge in different kinds of networks.

3.1 Mathematical model of coevolution in mutualistic networks

Let us start our incursion into coevolution in mutualistic networks by introducing a recently developed mathematical model (Guimarães Jr *et al.* 2017). The starting point for this model is the classical equation of phenotypic evolution by Lande (1976). This equation relates phenotypic change in one generation with the three basic components of evolution by natural selection: heritability, genotypic/phenotypic variation, and fitness differences:

$$z_i^{(t+1)} = z_i^{(t)} + h_i^2 \sigma_i^2 \frac{\partial \ln(W_i)}{\partial z_i^{(t)}},$$

where $z_i^{(t)}$ is the mean trait value of population of species i at generation t, h_i^2 is the heritability of trait z_i, σ_i^2 is the phenotypic variance of trait z_i, and $\frac{\partial \ln(W_i)}{\partial z_i^{(t)}}$ is the selection gradient of trait z_i at generation t. The selection gradient captures how the mean fitness of the population (W_i) is affected by changes in the mean trait value (z_i) and, therefore, defines an adaptive landscape for the trait. The simplest way to define such adaptive landscape is to think that $\frac{\partial \ln(W_i)}{\partial z_i^{(t)}}$ varies linearly with $z_i^{(t)}$:

$$\frac{\partial \ln(W_i)}{\partial z_i^{(t)}} = \rho(z_{i,p}^{(t)} - z_i^{(t)})$$

where ρ is related to the slope of the adaptive landscape and $z_{i,p}^{(t)}$ defines the peak of the adaptive landscape at generation t. Thus, the farthest $z_i^{(t)}$ is from $z_{i,p}^{(t)}$, the stronger the selection.

The next step in the development of this model is to define the adaptive peak $z_{i,p}^{(t)}$, which we assume to be related to mutualistic interactions and to other selective forces present in the local environment. Regarding the mutualism, we assume that individuals with stronger trait matching with their interacting partners have higher fitness, which is expected for many different types of mutualism (Santamaría & Rodríguez-Gironés 2007; Nuismer *et al.* 2010; Guimarães Jr *et al.* 2017). In this sense, the optimum trait value of a species i in relation to its mutualistic interaction with species j is the mean trait value of this mutualistic partner, $z_j^{(t)}$. If species i has many mutualistic partners, its optimum trait value is given by a combination of the trait values of its interacting parners ($z_j^{(t)}$) weighted by the evolutionary importance of each species j at time t: $\sum_{j=1}^{N} q_{ij}^{(t)} z_j^{(t)}$, where $q_{ij}^{(t)}$ is the relative evolutionary effect of species j to species i in relation to all other mutualistic partners of species i and N is the total number of species in the network. Thus, $q_{ij}^{(t)}$ ranges from 0 to 1 and $q_{ij}^{(t)} = 0$ if species i and j do not interact in the network. Now that we have defined mutualistic selection, let us consider how other selective forces may drive the evolution of $z_i^{(t)}$. To do so, we will assume that species i has a trait value (θ_i) that maximizes its fitness in a given environment. Such environment may include different factors that also shape the evolution of trait z_i, such as abiotic conditions. By combining these two distinct components, we may define the adaptive peak of trait z_i at generation t as:

$$z_{i,p}^{(t)} = \sum_{j=1}^{N} q_{ij}^{(t)} z_j^{(t)} + (1 - m_i)\theta_i$$

where m_i measures the relative importance of the mutualism for the trait evolution of species i ($0 < m_i < 1, \sum_{j=1}^{N} q_{ij}^{(t)} = m_i$). By combining Lande (1976) equation with our definition of the selection gradient, we arrive at a general model of coevolution in multispecies mutualisms:

$$z_i^{(t+1)} = z_i^{(t)} + \varphi_i \Big[\sum_{j=1}^{N} q_{ij}^{(t)} (z_j^{(t)} - z_i^{(t)}) + (1 - m_i)(\theta_i - z_i^{(t)}) \Big]$$

where $\varphi_i = h_i^2 \sigma_i^2 \rho$ is a compound parameter the combines the trait heritability, the phenotypic variance, and the slope of the adaptive landscape. The final step in developing this model is to define how the term $q_{ij}^{(t)}$ changes through time. To do so, we assume that $q_{ij}^{(t)}$ depends on different traits that are likely to govern mutualistic selection. First, $q_{ij}^{(t)}$ depends on a suite of traits not explicitly modeled by us that define if the interaction between two species can or cannot occur. Second, $q_{ij}^{(t)}$ depends on the traits $z_i^{(t)}$ and $z_j^{(t)}$, reflecting the fact that if those traits are similar to each other, the interaction should be functionally relevant and selection should be stronger. Combining these two components, we arrive at our equation for $q_{ij}^{(t)}$:

$$q_{ij}^{(t)} = m_i \frac{a_{ij} e^{-\alpha (z_j^{(t)} - z_i^{(t)})^2}}{\sum_{k=1}^{N} a_{ik} e^{-\alpha (z_k^{(t)} - z_i^{(t)})^2}}$$

where m_i represents the overall strength of the mutualism as a selective pressure, a_{ij} indicates whether the interaction between i and j is or is not allowed to occur ($a_{ij} = 0$ or $a_{ij} = 1$), and α controls the sensitivity of $q_{ij}^{(t)}$ to differences between traits. Note that the denominator of the equation above normalizes $q_{ij}^{(t)}$, so that $0 < q_{ij}^{(t)} < m_i, \sum_{j=1}^{N} q_{ij}^{(t)} = m_i$. Also note that, although $q_{ij}^{(t)}$ is defined for any two species in the network, this term will always be 0 if, for example, i and j are two plant species in a pollination network.

Now that we have a mathematical model, there are two basic ways to explore how species traits evolve in mutualistic networks. First, we may simulate the coevolutionary dynamics for a given choice of parameters (e.g., m_i, a_{ij}, θ_i, φ_i, α) in order to gain insight into how these parameters affect coevolution. In the next section, we will use **R** to perform these numerical simulations using different parameterizations and different empirical networks. Second, we may perform an analytical study to obtain some general conclusions about the model. In general, an analytical study is only possible as an approximation of the full model and we will perform this analysis by making the assumption that $q_{ij}^{(t)}$ is fixed over time (i.e., $q_{ij}^{(t)} = q_{ij}$). After investigating our model through simulations, we will also use **R** to explore the consequences of our analytical approximation of the model.

3.1.1 Numerical simulations

Let us start our exploration of the coevolutionary model using a simple theoretical network containing 3 plant species and 2 pollinator species.

```r
n_plant <- 3
n_pol <- 2
n_sp <- n_plant + n_pol
A <- matrix(c(1, 1, 1, 0, 1, 0),
            nrow = n_plant,
            ncol = n_pol,
            byrow = TRUE)
# changing labels
rownames(A) <- paste("P", 1:n_plant, sep = "")
colnames(A) <- paste("A", 1:n_pol, sep = "")
A
```

```
##    A1 A2
## P1  1  1
## P2  1  0
## P3  1  0
```

Now, let us expand this matrix so that it contains all the 5 species in the rows and in the columns. This new matrix will be well suited for our simulations, because it contains all interactions among all 5 species in the network. Therefore, it will be straightforward to parameterize the a_{ij} values of our model using this matrix.

```r
# creating two null matrices
zero_plant <- matrix(0, n_plant, n_plant)
zero_pol <- matrix(0, n_pol, n_pol)
A <- rbind(cbind(zero_plant, A),
           cbind(t(A), zero_pol))
A
```

```
##          A1 A2
## P1 0 0 0  1  1
## P2 0 0 0  1  0
## P3 0 0 0  1  0
## A1 1 1 1  0  0
## A2 1 0 0  0  0
```

The next step is to define the parameter values to be used in the simulation. We will define these parameter values using vectors containing the values for all the species in the network.

```
# defining m values
(m <- rep(0.5, n_sp))
```

```
## [1] 0.5 0.5 0.5 0.5 0.5
```

```
# defining phi values
(phi <- rep(0.5, n_sp))
```

```
## [1] 0.5 0.5 0.5 0.5 0.5
```

```
# defining alpha value
(alpha <- 0.2)
```

```
## [1] 0.2
```

```
# sampling theta values from uniform distribution
(theta <- runif(n_sp, min = 0, max = 10))
```

```
## [1] 4.550256 9.198272 6.734361 1.230797 5.154373
```

```
# sampling initial trait values from uniform distribution
(init <- runif(n_sp, min = 0, max = 10))
```

```
## [1] 2.785922 8.673749 2.185099 3.837514 4.312578
```

```
# defining trait values to be the sampled values
z <- init
```

Now that we have all the key ingredients of our model, let us perform the calculations for a single timestep of the model. We will use vectors and matrices containing the data for all species in the network in order to perform the computations faster. First, we can build a matrix where each entry contains the trait difference $z_j^{(t)} - z_i^{(t)}$:

```
(z_dif <- t(A * z) - A * z)
```

```
##           P1       P2        P3        A1       A2
##     0.000000 0.000000  0.000000  1.051591 1.526655
##     0.000000 0.000000  0.000000 -4.836236 0.000000
##     0.000000 0.000000  0.000000  1.652415 0.000000
## A1 -1.051591 4.836236 -1.652415  0.000000 0.000000
## A2 -1.526655 0.000000  0.000000  0.000000 0.000000
```

Now we can easily create a matrix containing all $q_{ij}^{(t)}$ values, which we will call the **Q** matrix.

```
(Q <- A * (exp( - alpha * (z_dif^2))))
```

```
##                                                A1        A2
## P1 0.0000000 0.000000000 0.0000000 0.801581356 0.6274224
## P2 0.0000000 0.000000000 0.0000000 0.009299125 0.0000000
## P3 0.0000000 0.000000000 0.0000000 0.579207320 0.0000000
## A1 0.8015814 0.009299125 0.5792073 0.000000000 0.0000000
## A2 0.6274224 0.000000000 0.0000000 0.000000000 0.0000000
```

```
# normalizing the Q matrix
(Q <- Q / apply(Q, 1, sum))
```

```
##                                               A1        A2
## P1 0.0000000 0.000000000 0.0000000 0.5609372 0.4390628
## P2 0.0000000 0.000000000 0.0000000 1.0000000 0.0000000
## P3 0.0000000 0.000000000 0.0000000 1.0000000 0.0000000
## A1 0.5766408 0.006689596 0.4166696 0.0000000 0.0000000
## A2 1.0000000 0.000000000 0.0000000 0.0000000 0.0000000
```

```
# multiplying each row i of matrix Q by m[i]
(Q <- Q * m)
```

```
##                                               A1        A2
## P1 0.0000000 0.000000000 0.0000000 0.2804686 0.2195314
## P2 0.0000000 0.000000000 0.0000000 0.5000000 0.0000000
## P3 0.0000000 0.000000000 0.0000000 0.5000000 0.0000000
## A1 0.2883204 0.003344798 0.2083348 0.0000000 0.0000000
## A2 0.5000000 0.000000000 0.0000000 0.0000000 0.0000000
```

Now that we have computed the **Q** matrix, we can calculate the mutualistic and the environmental components of the model. To do so, we will compute a new matrix containing the mutualistic component of the model ($q_{ij}^{(t)}(z_j^{(t)} - z_i^{(t)})$) in each entry. After that, we will sum over all partners j of each species i to obtain a vector containing the complete mutualistic component ($\sum_{j=1}^{N} q_{ij}^{(t)}(z_j^{(t)} - z_i^{(t)})$) for each species i in the network.

```
(Q_dif <- Q * z_dif)
```

```
##                                                   A1        A2
## P1  0.0000000 0.00000000  0.0000000  0.2949383 0.3351488
## P2  0.0000000 0.00000000  0.0000000 -2.4181178 0.0000000
## P3  0.0000000 0.00000000  0.0000000  0.8262073 0.0000000
## A1 -0.3031952 0.01617623 -0.3442555  0.0000000 0.0000000
## A2 -0.7633276 0.00000000  0.0000000  0.0000000 0.0000000
```

```
# summing the mutualistic selection of all partners j of species i
(mut_sel <- apply(Q_dif, 1, sum))
```

```
##         P1         P2         P3         A1         A2
##  0.6300871 -2.4181178  0.8262073 -0.6312744 -0.7633276
```

Finally, let us calculate the environmental component of trait evolution. To do so, we will simply compute a vector containing the difference $\theta_i - z_i^{(t)}$ for each species i and then multiply each element of this vector by $(1 - m_i)$ to obtain $(1 - m_i)(\theta_i - z_i^{(t)})$.

```
(env_sel <- (1 - m) * (theta - z))
```

```
## [1]  0.8821667  0.2622614  2.2746307 -1.3033582  0.4208978
```

If we add these two components for each species i and multiply them by φ_i, we will obtain the amount by which the mean trait $z_i^{(t)}$ will change from generation t to $t + 1$. Thus, we can update the vector `z` with the new mean trait values (i.e., $z_i^{(t+1)}$).

```
(z <- z + phi * (mut_sel + env_sel))
```

```
##       P1       P2       P3       A1       A2
## 3.542049 7.595821 3.735518 2.870197 4.141363
```

The simulation of a single timestep of trait evolution described above can now be used as the basic block of an **R** function that simulates the coevolutionary dynamics until equilibrium is reached. The function that simulates the coevolutionary model will contain as arguments all the parameters explored above plus two additional arguments. These two additional arguments will tell the function when to stop the simulation. The first argument, `epsilon`, contains the threshold value that must be reached for the simulation to stop. If the mean value of $|z_i^{(t+1)} - z_i^{(t)}|$ over all species i is smaller than `epsilon`, than the simulation will stop. The second argument, `t_max`, contains the maximum number of timesteps allowed. Therefore, if the threshold value `epsilon` is not reached, but the simulation reaches `t_max` timesteps, then the simulation also stops. Note that all vectors used as arguments in the function below need to contain the values for all species in the network in the same order as those species are present in the full, $N \times N$, interaction matrix $\mathbf{A}$.

```r
coevo_mut_net <- function(n_sp, A, m, phi, alpha, theta, init, epsilon, t_max) {
  # creating matrix to store z values
  z_mat <- matrix(NA, nrow = t_max, ncol = n_sp)
  # adding initial trait values
  z_mat[1, ] <- init
  for (t in 1:(t_max - 1)) {
    # defining current z values
    z <- z_mat[t, ]
    # creating matrix with all trait differences
    z_dif <- t(A * z) - A * z
    # calculating matrix Q
    Q <- A * (exp( - alpha * (z_dif^2)))
    # normalizing matrix Q
    Q <- Q / apply(Q, 1, sum)
    # multiplying each row i of matrix Q by m[i]
    Q <- Q * m
    # multiplying each entry in Q by corresponding trait difference
    Q_dif <- Q * z_dif
    # calculating mutualistic selection for each species i
    mut_sel <- apply(Q_dif, 1, sum)
    # calculating environmental selection for each species i
    env_sel <- (1 - m) * (theta - z)
    # storing z values of next timestep
    z_mat[t + 1, ] <- z + phi * (mut_sel + env_sel)
    # calculating the mean difference between old and new z values
    dif <- mean(abs(z - z_mat[t + 1, ]))
    if (dif < epsilon)
      break
  }
  return(z_mat[1:(t + 1), ])
}
```

We now have an **R** function that simulates the coevolutionary dynamics in a mutualistic network, given a set of parameter values. Our next step in this tutorial will be to use this function to investigate how two parameters, mutualistic selection (m_i) and network architecture (a_{ij}), affect coevolution in mutualistic networks. To do so, we will use different values of m_i and different empirical networks to parameterize the a_{ij} values of the coevolutionary model in order to verify how these parameters affect trait evolution.

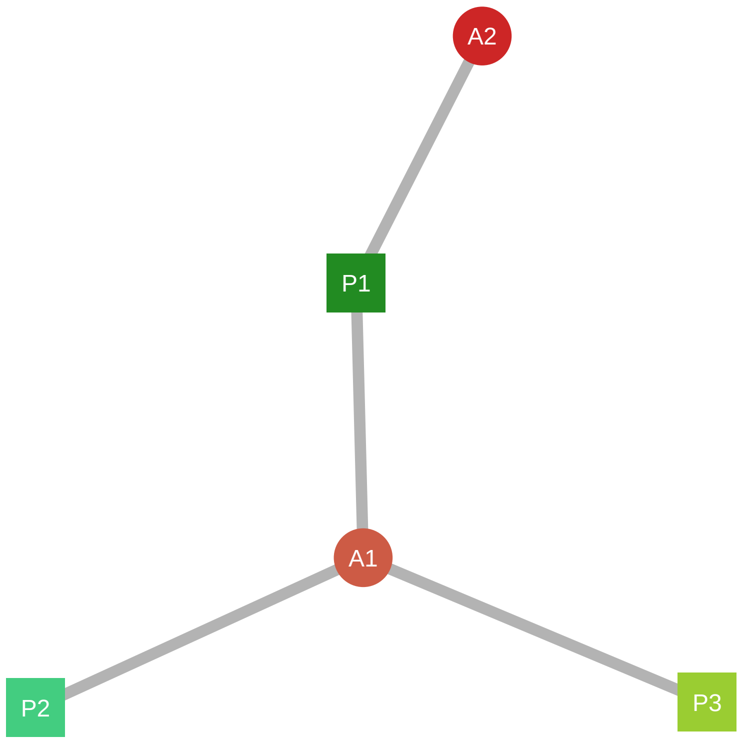
Let us start by performing one simulation of the coevolutionary model as an example, using the small pollination network with 5 species created above. We will also use all the parameter values already defined above. In addition, we will set `epsilon` to 0.01 and `t_max` to 1,000.

```r
(z_mat <- coevo_mut_net(n_sp, A, m, phi, alpha, theta, init,
                        epsilon = 0.01, t_max = 1000))
```

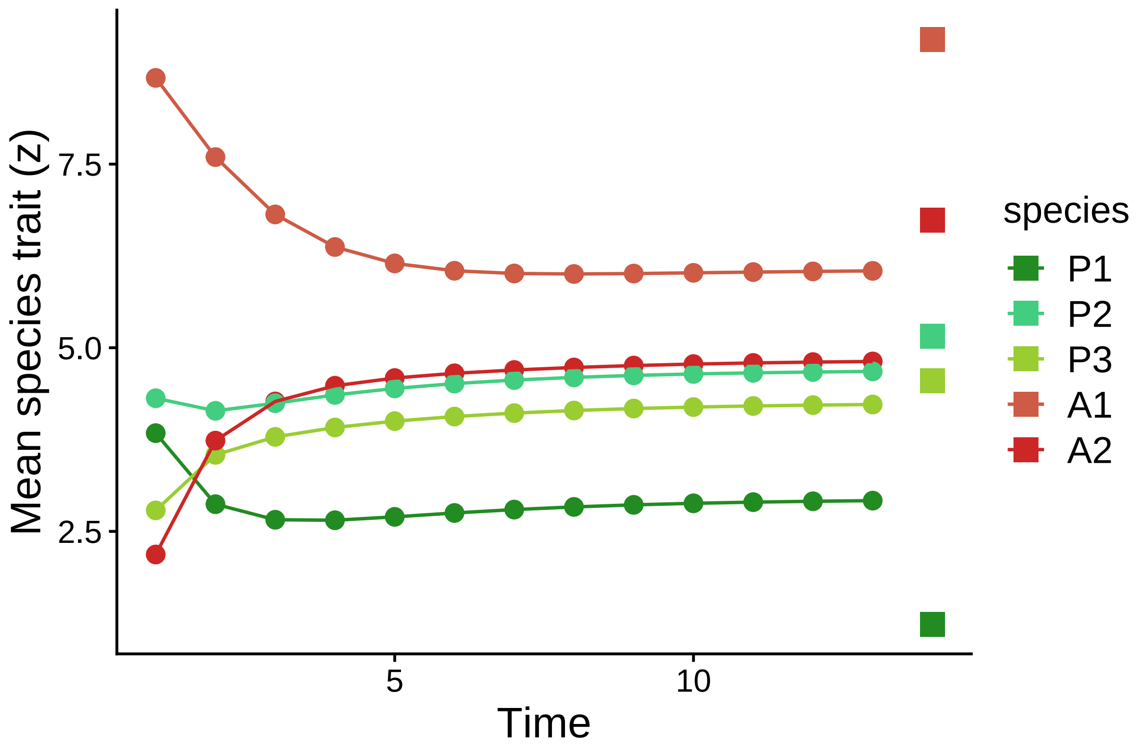
```
##          [,1]     [,2]     [,3]     [,4]     [,5]
##  [1,] 2.785922 8.673749 2.185099 3.837514 4.312578
##  [2,] 3.542049 7.595821 3.735518 2.870197 4.141363
##  [3,] 3.786499 6.815028 4.268899 2.658143 4.244787
##  [4,] 3.914686 6.371618 4.482575 2.651638 4.357612
##  [5,] 4.000706 6.148286 4.587787 2.697651 4.446071
##  [6,] 4.063413 6.048124 4.651896 2.750568 4.511805
##  [7,] 4.110329 6.011272 4.697180 2.796575 4.560349
##  [8,] 4.145674 6.004348 4.731324 2.833093 4.596350
##  [9,] 4.172322 6.010015 4.757525 2.861055 4.623187
## [10,] 4.192397 6.019839 4.777617 2.882146 4.643267
## [11,] 4.207505 6.030024 4.792935 2.897955 4.658326
## [12,] 4.218864 6.039069 4.804546 2.909775 4.669633
## [13,] 4.227401 6.046546 4.813307 2.918607 4.678126
```

The object `z_mat` returned by the function `coevo_mut_net` is a matrix containing the trait value of species j for timestep i as each entry. We can see that this simulation only lasted for 13 timesteps. In order to better visualize and understand this simulation, let us create a plot of the mean trait value of each species ($z_i^{(t)}$) over time. Before we do so, we will make a plot of our mutualistic network with 5 species and use the same color for each species in both plots. Tones of green will correspond to plant species and tones of red to pollinator species.

```r
# obtaining bipartite matrix
B <- A[1:n_plant, (n_plant + 1):(n_plant + n_pol)]
# creating network object for plotting
net <- network(B, matrix.type = "bipartite",
               directed = FALSE, ignore.eval = FALSE)
# defining color for each species
col <- c("forestgreen", "seagreen3", "olivedrab3",
         "coral3", "firebrick3")
# plotting
ggnet2(net, node.size = 10, color = col, edge.color = "gray70",
       edge.size = 2, mode = "kamadakawai",
       label = TRUE, label.color = "white", label.size = 3,
       shape = c(rep(15, n_plant), rep(16, n_pol)))
```



```
# transforming matrix to vector
z_vec <- c(z_mat)
# extracting value of final timestep
t_final <- nrow(z_mat)
# creating data frame with trait values through time
z_df <- data.frame(species = rep(rownames(A), each = t_final),
                   time = rep(1:t_final, times = n_sp),
                   trait = z_vec)
# creating data frame with theta values
theta_df = data.frame(theta = theta, x = rep((t_final + 1), length(theta)),
                      species = rownames(A))
# plotting
ggplot(z_df, aes(x = time, y = trait, color = species, group = species)) +
  scale_color_manual(values = col, label = rownames(A)) +
  geom_point(size = 2.8, shape = 19) +
  geom_line(size = 0.6) +
  geom_point(data = theta_df, aes(x = x, y = theta, color = species),
             size = 4, shape = 15) +
  xlab("Time") +
  ylab("Mean species trait (z)") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



In the plot above, the circles and lines represent species traits ($z_i^{(t)}$) trajectories and the squares in the right side represent the environmental optimum for each species (θ_i). We can see that species traits change a lot in the beginning of the simulation, but then reach an equilibrium. It is also interesting to note that the equilibrium trait values, denoted by z_i^*, are different from the environmental trait values (θ_i) for each species. This shows that mutualistic selection is shaping trait evolution by generating trait matching among mutualistic partners.

This coevolutionary model has many different parameters and it is not the goal of this tutorial to analyze the effect of all of them. Extensive analyzes of the effects of model parameters have been performed previously by Guimarães Jr *et al.* (2017) and by Medeiros *et al.* (2018). Our aim here is just to give a few examples of how mutualistic selection (m_i) and network architecture (a_{ij}) affect coevolution. To evaluate the effect of m_i on trait evolution, let us now simulate the coevolutionary dynamics using 5 different values of m_i: 0.1, 0.3, 0.5, 0.7, and 0.9. To do so, we will use the function `lapply` to run the function `coevo_mut_net` 5 times using a different m_i value each time. The rest of the parameters will be the same as used in the previous simulation.

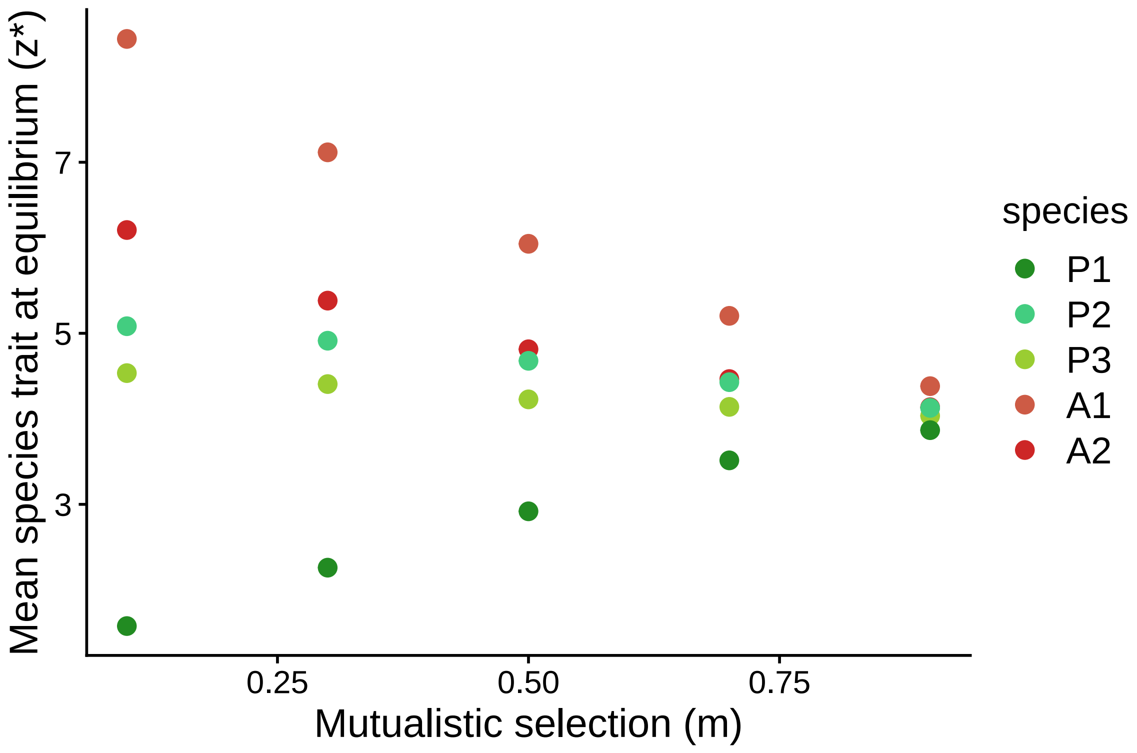
```
# defining list with m values
m_list <- list(rep(0.1, n_sp), rep(0.3, n_sp),
               rep(0.5, n_sp), rep(0.7, n_sp),
               rep(0.9, n_sp))
# running simulations
z_list <- lapply(m_list, coevo_mut_net, n_sp = n_sp,
                 A = A, phi = phi, alpha = alpha,
                 theta = theta, init = init,
                 epsilon = 0.01, t_max = 1000)
```

The object `z_list` above is a list in which each element contains a matrix with the results of a single simulation. To visualize the effect of mutualistic selection, we will create a plot with the equilibrium trait values for each m_i value. The colors of the points correspond to the same colors of the previous plot.

```
# extracting the equilibrium vector for each simulation
z_eq_list <- lapply(z_list, function(x) x[nrow(x), ])
# putting vectors in a data frame
```

```r
z_eq_df <- data.frame(trait = unlist(z_eq_list),
                      species = rep(rownames(A),
                                    length(m_list)),
                      m = unlist(m_list))

# plotting
ggplot(z_eq_df, aes(x = m, y = trait, color = species)) +
  scale_color_manual(values = col, label = rownames(A)) +
  geom_point(size = 2.8, shape = 19) +
  xlab("Mutualistic selection (m)") +
  ylab("Mean species trait at equilibrium (z*)") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        axis.title = element_text(size = 15),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



The plot above clearly shows that the effect of mutualistic selection is to increase trait matching in the mutualistic network. When m_i is low, species traits are mostly driven by environmental selection and the equilibrium value of each species i becomes close to θ_i. On the other hand, when m_i is high, species traits are driven towards a strong coupling with their mutualistic partners and trait values of different species become more similar to each other.

Our next step in simulating our coevolutionary model will be to test the effect of the structure of interactions. To do so, we will look at the pairwise values of trait matching at equilibrium ($e^{-\alpha(z_j^*-z_i^*)^2}$) in two simulations. The first simulation will be parameterized by the nested pollination network and the second simulation by the modular ant-plant network. These two networks were introduced in the previous section of the tutorial. In order to simplify our analysis, we will only use one value of mutualistic selection: $m_i = 0.7$. Let us first load the two networks into **R**.

```r
# ---------- Nested network ----------
# loading data
A_nested <- as.matrix(read.csv("empirical_networks/M_PL_059.csv",
                               header = TRUE, row.names = 1))
n_plant_nested <- nrow(A_nested)
n_anim_nested <- ncol(A_nested)
n_sp_nested <- n_plant_nested + n_anim_nested
# changing labels
rownames(A_nested) <- paste("P", 1:n_plant_nested, sep = "")
colnames(A_nested) <- paste("A", 1:n_anim_nested, sep = "")
# changing values to 1 or 0
A_nested[A_nested > 0] <- 1
# creating expanded matrix for simulations
A_nested <- rbind(cbind(matrix(0, n_plant_nested, n_plant_nested), A_nested),
                  cbind(t(A_nested), matrix(0, n_anim_nested, n_anim_nested)))
```

```r
# ---------- Modular network ----------
# loading data
A_modular <- as.matrix(read.csv("empirical_networks/M_PA_001.csv",
                                header = TRUE, row.names = 1))
# placing plants in rows and ants in columns
A_modular <- t(A_modular)
n_plant_modular <- nrow(A_modular)
n_anim_modular <- ncol(A_modular)
n_sp_modular <- n_plant_modular + n_anim_modular
# changing labels
rownames(A_modular) <- paste("P", 1:n_plant_modular, sep = "")
colnames(A_modular) <- paste("A", 1:n_anim_modular, sep = "")
# changing values to 1 or 0
A_modular[A_modular > 0] <- 1
# creating expanded matrix for simulations
A_modular <- rbind(cbind(matrix(0, n_plant_modular, n_plant_modular), A_modular),
                   cbind(t(A_modular), matrix(0, n_anim_modular, n_anim_modular)))
```

Now, we can simulate the coevolutionary dynamics for each network. All parameters of the model will be the same for both simulations, except for the interaction coefficients a_{ij}, which will be parameterized with the empirical data.

```r
# ---------- Nested network ----------
# defining list with m values
m <- rep(0.7, n_sp_nested)
# defining phi values
phi <- rep(0.5, n_sp_nested)
# defining alpha value
alpha <- 0.2
# sampling theta values from uniform distribution
theta <- runif(n_sp_nested, min = 0, max = 20)
# sampling initial trait values from uniform distribution
init <- runif(n_sp_nested, min = 0, max = 20)
# running simulations
z_nested <- coevo_mut_net(n_sp_nested, A_nested, m, phi, alpha, theta, init,
                          epsilon = 0.0001, t_max = 1000)
# extracting equilibrium values
z_nested_eq <- z_nested[nrow(z_nested), ]
```

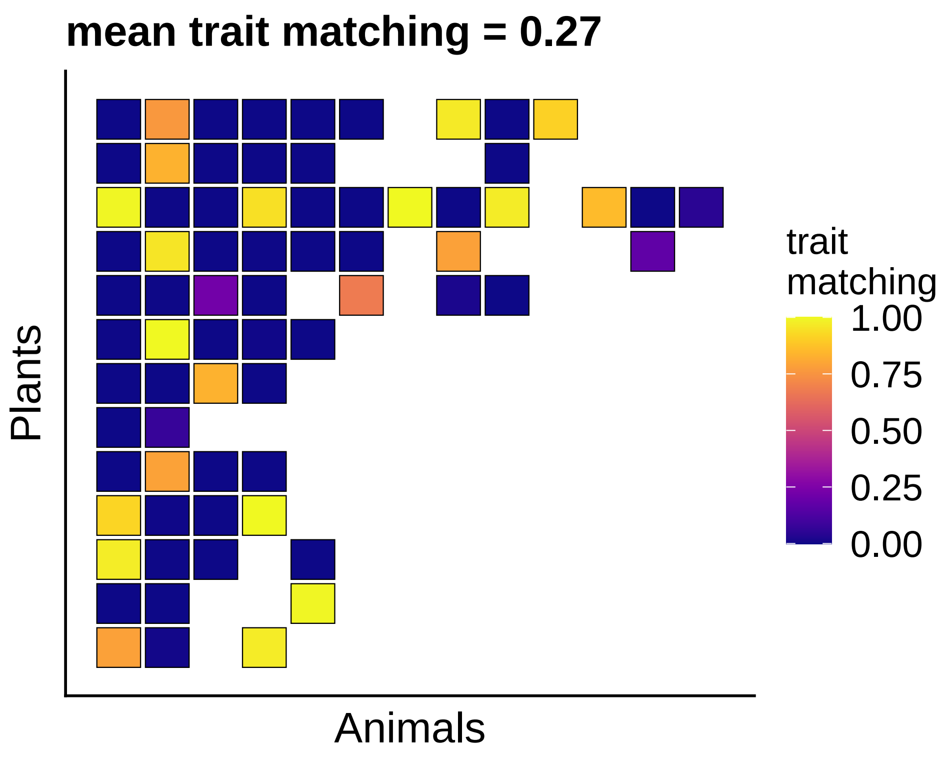
```r
# ---------- Modular network ----------
# defining list with m values
m <- rep(0.7, n_sp_modular)
# defining phi values
phi <- rep(0.5, n_sp_modular)
# defining alpha value
alpha <- 0.2
# sampling theta values from uniform distribution
theta <- runif(n_sp_modular, min = 0, max = 20)
# sampling initial trait values from uniform distribution
init <- runif(n_sp_modular, min = 0, max = 20)
# running simulations
z_modular <- coevo_mut_net(n_sp_modular, A_modular, m, phi, alpha, theta, init,
                           epsilon = 0.0001, t_max = 1000)
# extracting equilibrium values
z_modular_eq <- z_modular[nrow(z_modular), ]
```

We now have two vectors `z_nested_eq` and `z_modular_eq` that contain the equilibrium trait values of a single simulation using the nested network and the modular network, respectively. The next step is to calculate the trait matching value for each pair of interacting species and to plot these values. To do so, we will first create a function that computes the trait matching for every pair of species in a network. The inputs for this function are: `n_sp`, the total number of species; `n_row`, the number of species in the rows of the original bipartite matrix; `n_col`, the number of species in the columns of the original bipartite matrix; `A`, the full $N \times N$ interaction matrix $\mathbf{A}$; `z`, the trait values with species ordered in the same way as in the matrix $\mathbf{A}$; and `alpha` the parameter α of the trait matching formula.

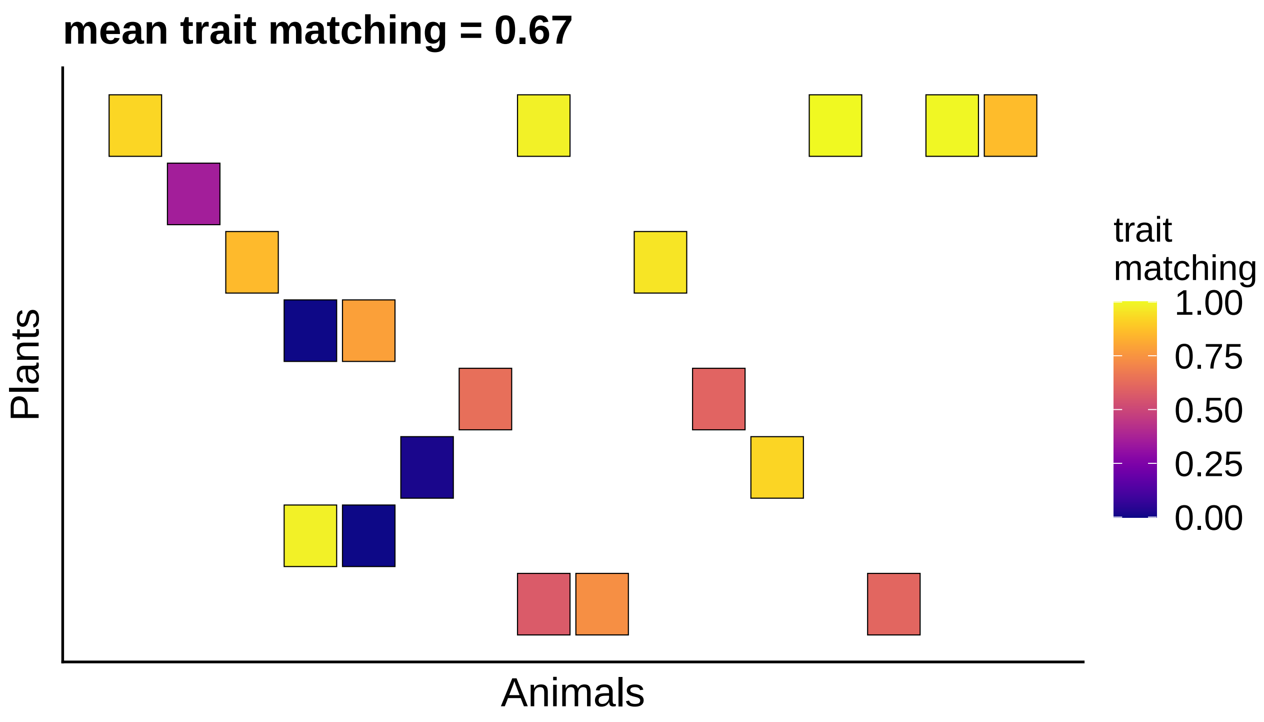
```r
trait_matching <- function(n_sp, n_row, n_col, A, z, alpha) {
  # matrix with trait values
  A_traits <- A * z
  # vectorizing the transpose matrix
  t_vec_traits <- c(t(A_traits))
  # row sp trait values
  row_sp_traits <- t_vec_traits[1:(n_sp * n_row)]
  row_sp_traits <- row_sp_traits[row_sp_traits > 0]
  vec_traits <- c(A_traits)
  # col sp trait values
  col_sp_traits <- vec_traits[1:(n_sp * n_row)]
  col_sp_traits <- col_sp_traits[col_sp_traits > 0]
  # pairwise trait differences
  trait_diff <- row_sp_traits - col_sp_traits
  # labels for rows and columns
  row_names <- paste("R", 1:n_row, sep = "")
  col_names <- paste("C", 1:n_col, sep = "")
  # getting codes for links
  names_rep <- rep(c(row_names, col_names), n_sp)
  names_node_1 <- names_rep[which(A_traits != 0)]
  # node degrees
  k <- apply(A, 1, sum)
  names_node_2 <- rep(c(row_names, col_names), k)
  # total number of links
  n_links <- sum(A) / 2
  # putting results in data frame
  results_df <- data.frame(column = names_node_1[1:n_links],
                           row = names_node_2[1:n_links],
                           matching = rep(NA, n_links))
  # computing trait matching
  results_df$matching <- exp(- alpha * (trait_diff)^2)
  return(results_df)
}
```

This function returns a data frame with each row representing a pair of interacting species in the network. The first column contains the label for the first species, the second column contains the label for the second species, and the third column contains the trait matching value. Now, let us use this function to compute equilibrium pairwise trait matching values for our two simulations. After computing trait matching, we will plot the results on the original bipartite interaction matrix. Above each plot, we will also show the mean trait matching across all interacting species.

```r
# ---------- Nested network ----------
# computing trait matching
trait_match_df <- trait_matching(n_sp_nested, n_plant_nested, n_anim_nested,
                                 A_nested, z_nested_eq, alpha = 0.2)
# computing mean trait matching
mean_trait_match <- mean(trait_match_df$matching)
# changing order of species in the data frame
trait_match_df$row <- n_plant_nested -
  as.numeric(substr(as.character(trait_match_df$row), start = 2,
                    stop = nchar(as.character(trait_match_df$row)))) + 1
trait_match_df$column <- as.numeric(substr(as.character(trait_match_df$column), start = 2,
                                           stop = nchar(as.character(trait_match_df$column))))
# plotting
ggplot(data = trait_match_df, aes(x = column, y = row,
                                  fill = matching,
                                  height = 0.9, width = 0.9)) +
  geom_tile(color = "black", size = 0.2) +
  scale_fill_viridis(name = "trait\nmatching", option = "plasma", limits = c(0, 1)) +
  ylab("Plants") +
  xlab("Animals") +
  ggtitle(paste("mean trait matching =", round(mean_trait_match, 2))) +
  theme_cowplot(12) +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



```
# ---------- Modular network ----------
# computing trait matching
trait_match_df <- trait_matching(n_sp_modular, n_plant_modular, n_anim_modular,
                                 A_modular, z_modular_eq, alpha = 0.2)
# computing mean trait matching
mean_trait_match <- mean(trait_match_df$matching)
# changing order of species in the data frame
trait_match_df$row <- n_plant_modular -
  as.numeric(substr(as.character(trait_match_df$row), start = 2,
                    stop = nchar(as.character(trait_match_df$row)))) + 1
trait_match_df$column <- as.numeric(substr(as.character(trait_match_df$column), start = 2,
                                           stop = nchar(as.character(trait_match_df$column))))
# plotting
ggplot(data = trait_match_df, aes(x = column, y = row,
                                  fill = matching,
                                  height = 0.9, width = 0.9)) +
  geom_tile(color = "black", size = 0.2) +
  scale_fill_viridis(name = "trait\nmatching", option = "plasma", limits = c(0, 1)) +
  ylab("Plants") +
  xlab("Animals") +
  ggtitle(paste("mean trait matching =", round(mean_trait_match, 2))) +
  theme_cowplot(12) +
  theme(plot.title = element_text(size = 16, face = "bold"),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



Although the pairwise trait matching can vary a lot within each network, we can see from the plots above that trait matching is higher on average for the modular network than for the nested network. This happens because the disconnected structure of the modular network allows species in isolated groups to evolve high trait matching with each other. On the other hand, species in the nested network are faced with multiple conflicting selection pressures, which inhibits the evolution of trait matching. Although we only performed one simulation for each network, it has been shown that these conclusions about the coevolutionary effects of network structure hold when many simulations are performed using many empirical networks (Guimarães Jr *et al.* 2017; Medeiros *et al.* 2018).

Now that we have explored the coevolutionary model through simulations, let us move on to the last section of this tutorial, in which we will explore an analytical approximation to the coevolutionary model.

3.1.2 Analytical study

In the previous section, we have used numerical simulations to gain insight into how species coevolve in mutualistic networks. We have analyzed how different model parameters (m_i and a_{ij}) affect trait evolution. Importantly, we have explored how the structure of interactions contained in the entries a_{ij} of the interaction matrix $\mathbf{A}$ may shape coevolution. In the present section of this tutorial, we will use an analytical approximation of the model to understand how coevolutionary selection operates in mutualistic networks (Guimarães Jr *et al.* 2017). In particular, we will explore how m_i and a_{ij} affect trait evolution through direct and indirect coevolutionary effects in the network.

The first goal of our analytical study is to find the mathematical equation that describes the equilibrium of the coevolutionary model. More specifically, we want to find the equation that satisfies $z_i^{(t+1)} = z_i^{(t)} = z_i^*$, which means that the mean trait value of every species i in the network is fixed over time. To be able to solve for equilibrium, we have to make the assumption that $q_{ij}^{(t)}$ is fixed over time (i.e., $q_{ij}^{(t)} = q_{ij}$). By assuming that pairwise evolutionary effects are fixed, we can solve our model for equilibrium, given that it is now a linear dynamical system. So, our goal is to solve the following equation for z_i^*:

$$0 = \varphi_i \Big[\sum_{j=1}^{N} q_{ij}(z_j^* - z_i^*) + (1 - m_i)(\theta_i - z_i^*) \Big]$$

Because $\varphi_i \neq 0$, we arrive at the following equation:

$$0 = \sum_{j=1}^{N} q_{ij} z_j^* - z_i^* \sum_{j=1}^{N} q_{ij} + (1 - m_i)\theta_i - (1 - m_i) z_i^*$$

Now we can use that fact that $\sum_{j=1}^{N} q_{ij} = m_i$ and simplify the above equation further to obtain:

$$0 = \sum_{j=1}^{N} q_{ij} z_j^* + (1 - m_i)\theta_i - z_i^*$$

The next step is to write the equation above using vectors and matrices. To do so, let us first define the following vectors and matrices:

$$\boldsymbol{z^*} = \begin{bmatrix} z_1^* \\ z_2^* \\ \vdots \\ z_N^* \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}, \boldsymbol{Q} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1N} \\ q_{21} & q_{22} & \cdots & q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & q_{NN} \end{bmatrix}, \boldsymbol{\Psi} = \begin{bmatrix} 1 - m_1 & 0 & \cdots & 0 \\ 0 & 1 - m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 - m_N \end{bmatrix}$$

We can now write the previous equation using these vectors and matrices:

$$\mathbf{0} = \boldsymbol{Q z^*} + \boldsymbol{\Psi \theta} - \boldsymbol{z^*}$$

where $\mathbf{0}$ is a vector of dimension N filled with zeros. Finally, we perform two additional modifications to the equation above to arrive at the final equilibrium expression of the coevolutionary model:

$$\boldsymbol{z^*} - \boldsymbol{Q z^*} = \boldsymbol{\Psi \theta}$$

$$\boldsymbol{z^*} = (\boldsymbol{I} - \boldsymbol{Q})^{-1} \boldsymbol{\Psi \theta}$$

where $\boldsymbol{I}$ is the identity matrix of dimension N and $(\boldsymbol{I} - \boldsymbol{Q})^{-1}$ is the inverse of the matrix $(\boldsymbol{I} - \boldsymbol{Q})$. The above equation describes how we obtain the equilibrium trait value of each species (z_i^*) only by knowing some of the parameters of the model (q_{ij}, m_i, and θ_i). This mathematical analysis tells us that, if q_{ij} values are fixed over time, we can simply obtain the final trait values without the need of performing a numerical simulation. In this sense, we can now explore coevolution in mutualistic networks by using the analytical equilibrium expression above. To do so, let us first create a simple **R** function that performs the analytical computation.

```r
coevo_equil <- function(n_sp, Q, Psi, theta) {
  I <- diag(1, n_sp)
  z <- solve(I - Q) %*% Psi %*% theta
  return(c(z))
}
```

Let us now use this function to compute equilibrium trait values for the nested pollination network that we used previously. We will use specific values for m_i and θ_i and empirical data to parameterize q_{ij}. Our assumption here is that the interaction strength between species i and j (i.e., total number of visits of pollinator species j to plant species i) can be used to estimate the evolutionary effect that species j imposes on species i. The first step is to load the pollination network in **R** and build the following matrices and vectors: $\boldsymbol{Q}$, $\boldsymbol{\Psi}$, and $\boldsymbol{\theta}$. Then, we will use the function `coevo_equil` to compute equilibrium trait values.

```r
# ---------- Nested network ----------
# loading data
A_nested <- as.matrix(read.csv("empirical_networks/M_PL_059.csv",
                               header = TRUE, row.names = 1))
n_plant_nested <- nrow(A_nested)
n_anim_nested <- ncol(A_nested)
n_sp_nested <- n_plant_nested + n_anim_nested
# changing labels
rownames(A_nested) <- paste("P", 1:n_plant_nested, sep = "")
colnames(A_nested) <- paste("A", 1:n_anim_nested, sep = "")
# creating expanded matrix A
A_nested <- rbind(cbind(matrix(0, n_plant_nested, n_plant_nested), A_nested),
                  cbind(t(A_nested), matrix(0, n_anim_nested, n_anim_nested)))
# defining mutualistic selection
m <- 0.5
# normalizing matrix to build matrix Q
Q_nested <- A_nested / apply(A_nested, 1, sum)
# multiplying each row by m
Q_nested <- Q_nested * m
# building matrix Psi
Psi_nested <- diag(0.5, n_sp_nested)
# sampling theta values
theta <- runif(n_sp_nested, min = 0, max = 10)
# computing equilibrium
(z_nested_eq <- coevo_equil(n_sp_nested, Q_nested, Psi_nested, theta))
```

```
##  [1] 7.177157 6.906130 6.591404 7.163608 3.313810 5.369259 5.973178 5.098904
##  [9] 7.217718 3.568307 4.268193 6.093459 6.839388 4.363979 4.101387 4.269014
## [17] 7.601478 3.772481 3.277478 5.085355 5.185913 7.360350 4.902636 5.974002
## [25] 6.727450 6.336000
```

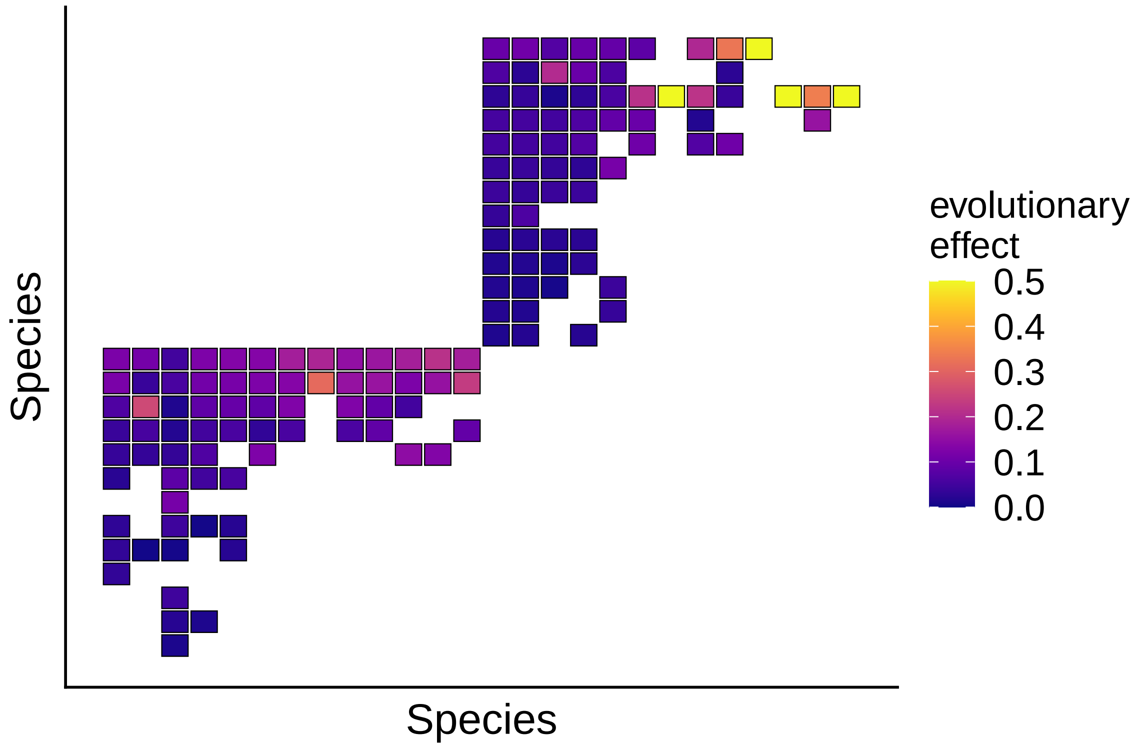
The vector `z_nested_eq` above gives us the equilibrium trait values for all the species in the network. In addition to allowing us to compute the equilibrium traits without the need of performing a simulation, the analytical solution allows us to explore how selection operates in mutualistic networks. To do so, let us look at the structure of the matrix $\boldsymbol{T} = (\boldsymbol{I} - \boldsymbol{Q})^{-1}\boldsymbol{\Psi}$ by varying two parameters of the model: the strength of mutualistic selection (m_i) and the interaction network (a_{ij}). Because $\boldsymbol{T}$ is the matrix that connects the environmental optima ($\boldsymbol{\theta}$) with traits at equilibrium ($\boldsymbol{z}^*$), our hope is that its structure will tell us something about how coevolution operates in mutualistic networks. Before we start exploring $\boldsymbol{T}$, let us look at the structure of $\boldsymbol{Q}$, the matrix of evolutionary effects.

```r
# ---------- Matrix Q ----------
# adding labels to Q
rownames(Q_nested) <- 1:n_sp_nested
colnames(Q_nested) <- 1:n_sp_nested
# transforming matrix into data frame
Q_nested_df <- melt(Q_nested)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
# plotting
ggplot(data = subset(Q_nested_df, value > 0),
         aes(x = X1, y = X2, fill = value, height = 0.9, width = 0.9)) +
  geom_tile(color = "black", size = 0.2) +
  scale_fill_viridis(name = "evolutionary\neffect", option = "plasma",
                     limits = c(0, 0.5)) +
  ylab("Species") +
  xlab("Species") +
  scale_y_reverse() +
  theme_cowplot(12) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



We can see that only certain elements of the matrix $\boldsymbol{Q}$ are different from zero, given that $q_{ij} = 0$ if i and j do not interact in the network (e.g., i and j are both plant species). Now, let us compare the structure of matrix $\boldsymbol{Q}$ with the structure of matrix $\boldsymbol{T}$.

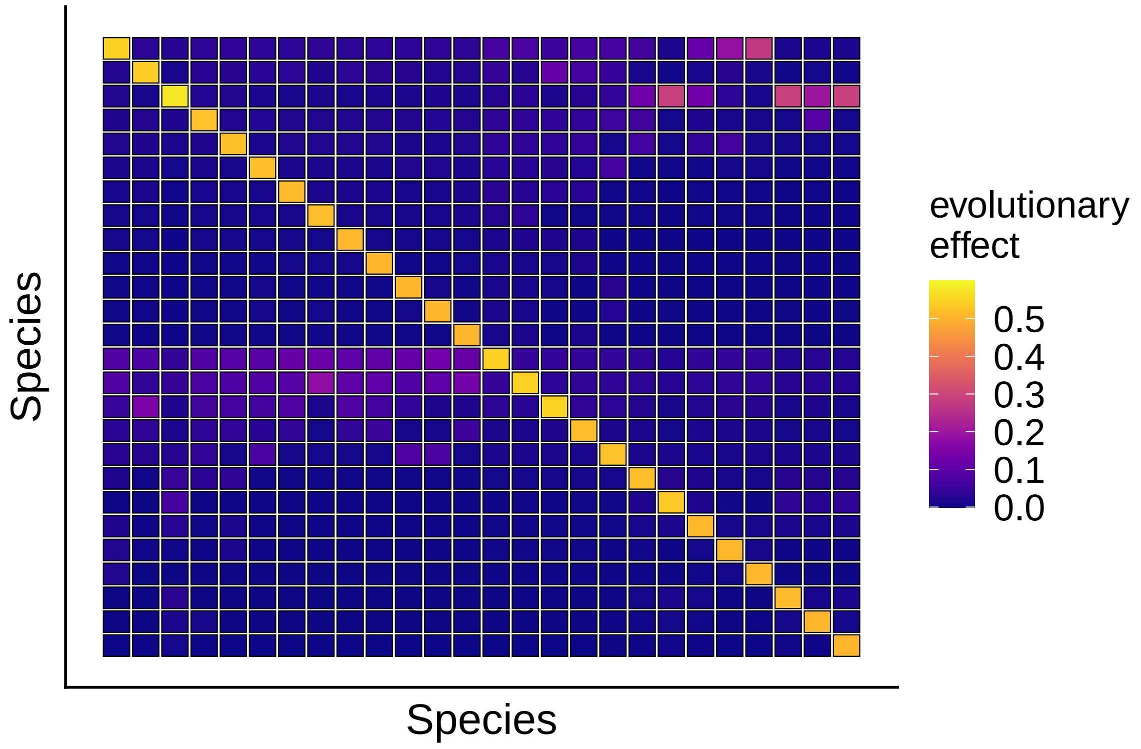
```
# ---------- Matrix T ----------
# computing T
T_nested <- solve(diag(1, n_sp_nested) - Q_nested) %*% Psi_nested
# adding labels to T
rownames(T_nested) <- 1:n_sp_nested
colnames(T_nested) <- 1:n_sp_nested
# transforming matrix into data frame
```

```
T_nested_df <- melt(T_nested)
```

```
## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE

## Warning in type.convert.default(X[[i]], ...): 'as.is' should be specified by the
## caller; using TRUE
```

```
# plotting
ggplot(data = subset(T_nested_df, value > 0),
         aes(x = X1, y = X2, fill = value, height = 0.9, width = 0.9)) +
  geom_tile(color = "black", size = 0.2) +
  scale_fill_viridis(name = "evolutionary\neffect", option = "plasma",
                     limits = c(0, 0.6)) +
  ylab("Species") +
  xlab("Species") +
  scale_y_reverse() +
  theme_cowplot(12) +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



In contrast with matrix $\boldsymbol{Q}$, matrix $\boldsymbol{T}$ is completely filled with nonzero values. Therefore, the plot above shows that matrix $\boldsymbol{T}$ contains evolutionary effects between species that are not directly connected in the network, that is, indirect evolutionary effects. This means that the equilibrium trait value of each species i in the network is affected by every other species j in the network. This result suggests that indirect effects are an important feature of coevolution in mutualistic networks (Guimarães Jr *et al.* 2017).

Our final analysis will be to investigate how important are indirect effects for trait evolution. To do so, we will explore how m_i and a_{ij} affect the indirect coevolutionary effects present in matrix $\boldsymbol{T}$. Following Guimarães Jr *et al.* (2017), we will measure the contribution of indirect effects to trait evolution as the proportion of all evolutionary effects of matrix $\boldsymbol{T}$ that are indirect:

$$\kappa = \frac{\sum_i^N \sum_{j,j\neq i}^N (1 - a_{ij}) t_{ij}}{\sum_i^N \sum_{j,j\neq i}^N t_{ij}}$$

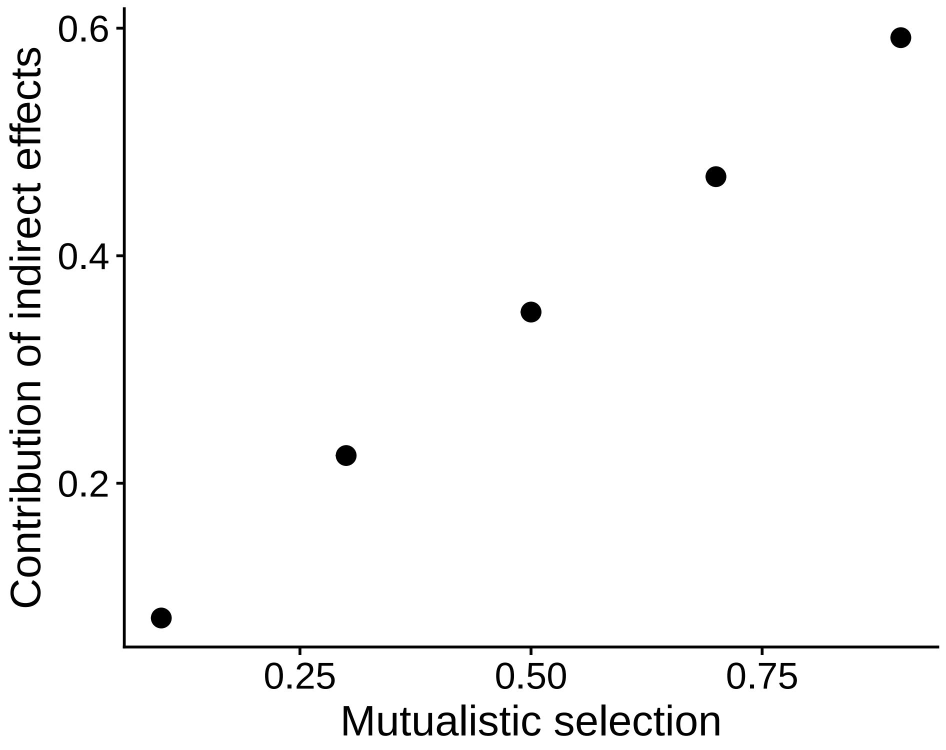
where t_{ij} is an element of matrix $\boldsymbol{T}$. Let us first build a function in **R** to compute κ.

```r
prop_ind_effects <- function(T_mat, A) {
  diag(T_mat) <- NA
  kappa <- sum(T_mat[A == 0], na.rm = TRUE) / sum(T_mat, na.rm = TRUE)
  return(kappa)
}
```

Now let us compute the contribution of indirect effects to trait evolution in the nested pollination network for multiple values of m_i. All other parameters will be the same as used before.

```r
# ---------- Nested network ----------
# loading data
A_nested <- as.matrix(read.csv("empirical_networks/M_PL_059.csv",
                               header = TRUE, row.names = 1))
n_plant_nested <- nrow(A_nested)
n_anim_nested <- ncol(A_nested)
n_sp_nested <- n_plant_nested + n_anim_nested
# changing labels
rownames(A_nested) <- paste("P", 1:n_plant_nested, sep = "")
colnames(A_nested) <- paste("A", 1:n_anim_nested, sep = "")
# creating expanded matrix A
A_nested <- rbind(cbind(matrix(0, n_plant_nested, n_plant_nested), A_nested),
                  cbind(t(A_nested), matrix(0, n_anim_nested, n_anim_nested)))
# defining mutualistic selection
m_list <- list(0.1, 0.3, 0.5, 0.7, 0.9)
# normalizing matrix to build matrix Q
Q_nested <- A_nested / apply(A_nested, 1, sum)
# bulding one matrix Q for each m value
Q_list <- lapply(m_list, function(mat, m) mat * m,
                 mat = Q_nested)
# building one matrix Psi for each m value
Psi_list <- lapply(m_list, function(n_sp, m) diag(m, n_sp),
                   n_sp = n_sp_nested)
# obtaining binary matrix A
A_nested[A_nested > 0] <- 1
# computing T matrices for each m value
T_list <- mapply(function(mat1, mat2, n_sp) {
  solve(diag(1, n_sp) - mat1) %*% mat2
  }, mat1 = Q_list, mat2 = Psi_list, MoreArgs = list(n_sp = n_sp_nested),
  SIMPLIFY = FALSE)
# computing kappa for each T matrix
kappa <- lapply(T_list, prop_ind_effects, A = A_nested)
# build data frame for plotting
kappa_nested_df <- data.frame(kappa = unlist(kappa), m = unlist(m_list))
```

```r
# plotting
ggplot(data = kappa_nested_df, aes(x = m, y = kappa)) +
  geom_point(size = 3) +
  ylab("Contribution of indirect effects") +
  xlab("Mutualistic selection") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 14),
        axis.text.y = element_text(size = 14),
        axis.title = element_text(size = 16))
```

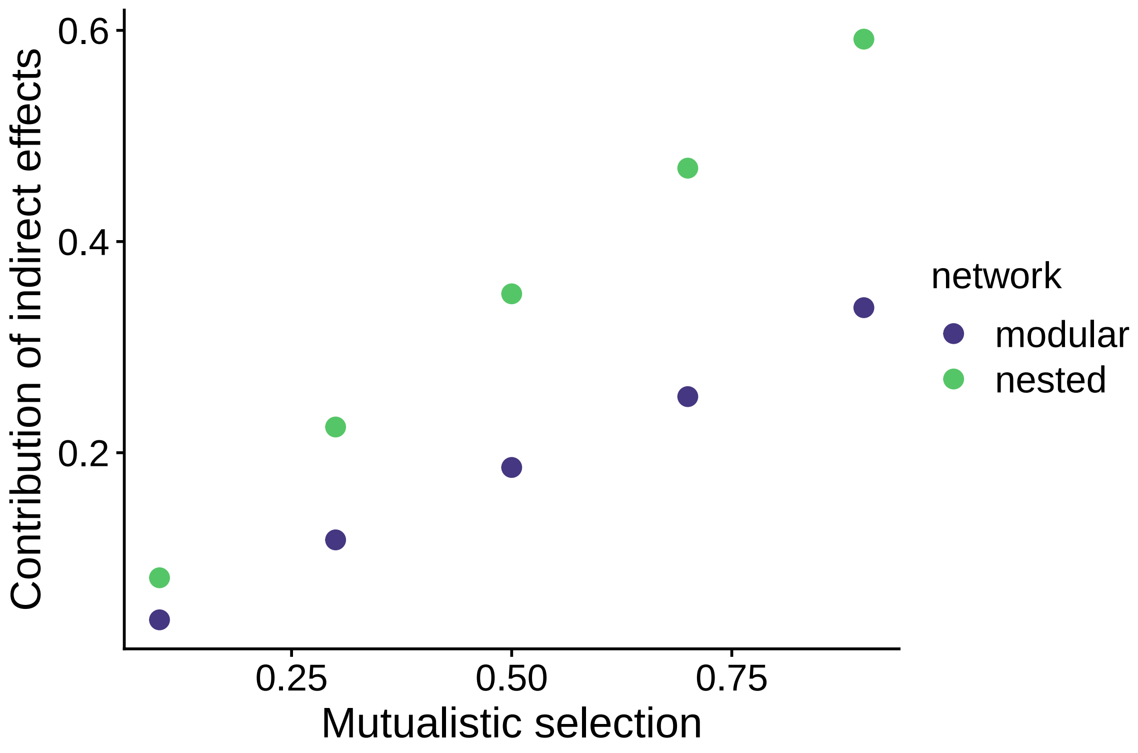


The plot above shows that as we increase the importance of mutualistic selection, we increase the contribution of indirect coevolutionary effects to trait evolution (Guimarães Jr *et al.* 2017). Therefore, this result based on our analytical approximation of the model provides a mechanism for how mutualistic selection favors trait matching in mutualistic networks. It does so by fueling the indirect coevolutionary effects between species in the network (Medeiros *et al.* 2018).

We can now perform the same analysis for the modular ant-plant network and check if different network structures vary in how they generate indirect evolutionary effects. To do so, let us add the points corresponding to the modular network to the plot above.

```r
# ---------- Modular network ----------
# loading data
A_modular <- as.matrix(read.csv("empirical_networks/M_PA_001.csv",
                                header = TRUE, row.names = 1))
n_plant_modular <- nrow(A_modular)
n_anim_modular <- ncol(A_modular)
n_sp_modular <- n_plant_modular + n_anim_modular
# changing labels
rownames(A_modular) <- paste("P", 1:n_plant_modular, sep = "")
colnames(A_modular) <- paste("A", 1:n_anim_modular, sep = "")
# creating expanded matrix A
```

```r
A_modular <- rbind(cbind(matrix(0, n_plant_modular, n_plant_modular), A_modular),
                  cbind(t(A_modular), matrix(0, n_anim_modular, n_anim_modular)))
# defining mutualistic selection
m_list <- list(0.1, 0.3, 0.5, 0.7, 0.9)
# normalizing matrix to build matrix Q
Q_modular <- A_modular / apply(A_modular, 1, sum)
# bulding one matrix Q for each m value
Q_list <- lapply(m_list, function(mat, m) mat * m,
                 mat = Q_modular)
# building one matrix Psi for each m value
Psi_list <- lapply(m_list, function(n_sp, m) diag(m, n_sp),
                   n_sp = n_sp_modular)
# obtaining binary matrix A
A_modular[A_modular > 0] <- 1
# computing T matrices for each m value
T_list <- mapply(function(mat1, mat2, n_sp) {
  solve(diag(1, n_sp) - mat1) %*% mat2
  }, mat1 = Q_list, mat2 = Psi_list, MoreArgs = list(n_sp = n_sp_modular),
  SIMPLIFY = FALSE)
# computing kappa for each T matrix
kappa <- lapply(T_list, prop_ind_effects, A = A_modular)
# build data frame for plotting
kappa_modular_df <- data.frame(kappa = unlist(kappa), m = unlist(m_list))
# merging the two data frames
kappa_df <- rbind(kappa_nested_df, kappa_modular_df)
kappa_df$network <- c(rep("nested", nrow(kappa_nested_df)),
                      rep("modular", nrow(kappa_modular_df)))
# plotting
ggplot(data = kappa_df, aes(x = m, y = kappa, color = network)) +
  geom_point(size = 3) +
  scale_color_manual(values = c("#453781FF", "#55C667FF")) +
  ylab("Contribution of indirect effects") +
  xlab("Mutualistic selection") +
  theme_cowplot(12) +
  theme(axis.text.x = element_text(size = 14),
        axis.text.y = element_text(size = 14),
        axis.title = element_text(size = 16),
        legend.key.size = unit(0.6, "cm"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 14))
```



The plot above provides further evidence that coevolution is likely to operate differently in different types of network. Coevolution in nested networks should be more influenced by indirect effects than coevolution in modular networks. This is probably the mechanism behind the different degrees of trait matching that we observed previously for these two networks. Extensive analyzes on how different types of mutualism vary in their contribution to indirect evolutionary effects have been performed by Guimarães Jr *et al.* (2017).

4 Conclusions

The goal of this tutorial was to introduce an approach to study coevolution in mutualistic networks using the **R** environment. We first introduced the concept of mutualistic networks and explored their structural patterns. Then, we worked with a mathematical model of coevolution and showed how it can be explored with simulations and with analytical tools. We have shown that different mutualisms (e.g., pollination, ant-plant interactions) produce very different types of network. Furthermore, these different networks are expected to produce different coevolutionary dynamics. We also showed how the importance of mutualistic selection can affect coevolution in mutualistic networks. We hope that this tutorial was useful and please let us know if you have any questions about or suggestions for this tutorial!

References

Almeida-Neto, M., Guimaraes, P., Guimaraes Jr, P.R., Loyola, R.D. & Ulrich, W. (2008). A consistent metric for nestedness analysis in ecological systems: Reconciling concept and measurement. *Oikos*, 117, 1227–1239.

Andreazzi, C.S., Thompson, J.N. & Guimarães Jr, P.R. (2017). Network structure and selection asymmetry drive coevolution in species-rich antagonistic interactions. *The American Naturalist*, 190, 99–115.

Barber, M.J. (2007). Modularity and community detection in bipartite networks. *Physical Review E*, 76, 066102.

Bascompte, J. & Jordano, P. (2013). *Mutualistic networks.* Princeton University Press.

Bascompte, J., Jordano, P., Melián, C.J. & Olesen, J.M. (2003). The nested assembly of plant–animal mutualistic networks. *Proceedings of the National Academy of Sciences*, 100, 9383–9387.

Beckett, S.J. (2016). Improved community detection in weighted bipartite networks. *Royal Society open science*, 3, 140536.

Bezerra, E.L., Machado, I.C. & Mello, M.A. (2009). Pollination networks of oil-flowers: A tiny world within the smallest of all worlds. *Journal of Animal Ecology*, 78, 1096–1101.

Brockhurst, M.A. & Koskella, B. (2013). Experimental coevolution of species interactions. *Trends in Ecology & Evolution*, 28, 367–375.

Davidson, D.W., Snelling, R.R. & Longino, J.T. (1989). Competition among ants for myrmecophytes and the significance of plant trichomes. *Biotropica*, 64–73.

Gómez, J.M., Verdú, M. & Perfectti, F. (2010). Ecological interactions are evolutionarily conserved across the entire tree of life. *Nature*, 465, 918.

Guimarães Jr, P.R., Jordano, P. & Thompson, J.N. (2011). Evolution and coevolution in mutualistic networks. *Ecology Letters*, 14, 877–885.

Guimarães Jr, P.R., Pires, M.M., Jordano, P., Bascompte, J. & Thompson, J.N. (2017). Indirect effects drive coevolution in mutualistic networks. *Nature*, 550, 511.

Hanifin, C.T., Brodie Jr, E.D. & Brodie III, E.D. (2008). Phenotypic mismatches reveal escape from arms-race coevolution. *PLoS Biology*, 6, e60.

Lande, R. (1976). Natural selection and random genetic drift in phenotypic evolution. *Evolution*, 30, 314–334.

McCann, K.S. (2011). *Food webs*. Princeton University Press.

Medeiros, L.P., Garcia, G., Thompson, J.N. & Guimarães Jr, P.R. (2018). The geographic mosaic of coevolution in mutualistic networks. *Proceedings of the National Academy of Sciences*.

Nuismer, S.L., Gomulkiewicz, R. & Ridenhour, B.J. (2010). When is correlation coevolution? *The American Naturalist*, 175, 525–537.

Nuismer, S.L., Jordano, P. & Bascompte, J. (2013). Coevolution and the architecture of mutualistic networks. *Evolution*, 67, 338–354.

Nuismer, S.L., Week, B. & Aizen, M.A. (2018). Coevolution slows the disassembly of mutualistic networks. *The American Naturalist*, 192, 490–502.

Olesen, J.M., Bascompte, J., Dupont, Y.L. & Jordano, P. (2007). The modularity of pollination networks. *Proceedings of the National Academy of Sciences*, 104, 19891–19896.

Santamaría, L. & Rodríguez-Gironés, M.A. (2007). Linkage rules for plant–pollinator networks: Trait complementarity or exploitation barriers? *PLoS Biology*, 5, e31.

Thompson, J.N. & Cunningham, B.M. (2002). Geographic structure and dynamics of coevolutionary selection. *Nature*, 417, 735.

Thompson, J.N., Schwind, C. & Friberg, M. (2017). Diversification of trait combinations in coevolving plant and insect lineages. *The American Naturalist*, 190, 171–184.