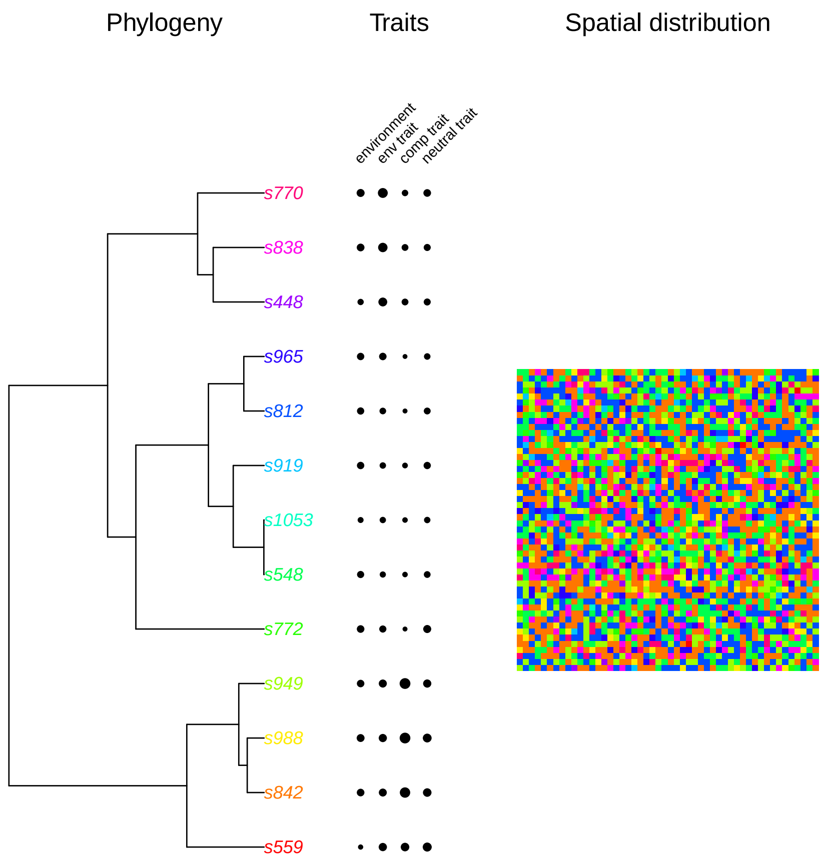
THE PHYLOSIM R PACKAGE



Tutorial for the PhyloSim package. Comments and contact: Florian Hartig / University of Freiburg / Germany.

Warning for the user : consider this package to be in a beta-stage.

To obtain the package: The package is not officially realeased yet. You can install by typing the following code in R. This code contains a private token - please do not pass it on!

```
# install.packages(c("devtools","Rcpp")) # I case you don't have them installed
library(devtools)
library(Rcpp)

install_url("https://dl.dropboxusercontent.com/s/zkdofob5b523qxt/PhyloSim_0.3.tar.gz")

?PhyloSim
browseVignettes("PhyloSim")
```

To cite the package: use the citation information provided with the package

```
citation('PhyloSim')
```

```

To cite package 'PhyloSim' in publications use:

  Paul Bauche, Florian Hartig, with contributions from Christian
  Buschbeck and Stefan Paul (2015). PhyloSim: Analysis of
  Phylogenetic and Biogeographic patterns, inferring species
  community assembly. R package version 0.3.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {PhyloSim: Analysis of Phylogenetic and Biogeographic patterns, inferring species
community assembly},
    author = {Paul Bauche and Florian Hartig and with contributions from Christian Buschbeck and Stefan Paul},
    year = {2015},
    note = {R package version 0.3},
  }

ATTENTION: This citation information has been auto-generated from
the package DESCRIPTION file and may need manual editing, see
'help("citation")'.
```

Contents

1
Model description and running the model

1.1 Overview

The PhyloSim package is a modelling environment for neutral and non-neutral biogeographic simulations. It generates a) Spatial distribution and diversity metrics b) phylogeny and c) traits from four basic processes: i) dispersal, ii) (adaptive) envirnomental preferences, iii) (adaptive) competitive traits, and iv) speciation in a stochastic simulation.

The package permits to choose between a number of options for each of these processes, and provides function for analyzing the results, including standard biogeographic and phylgenetic summaries such as species-area curves, rank-abundance curves or phylogentic balance and diversity.

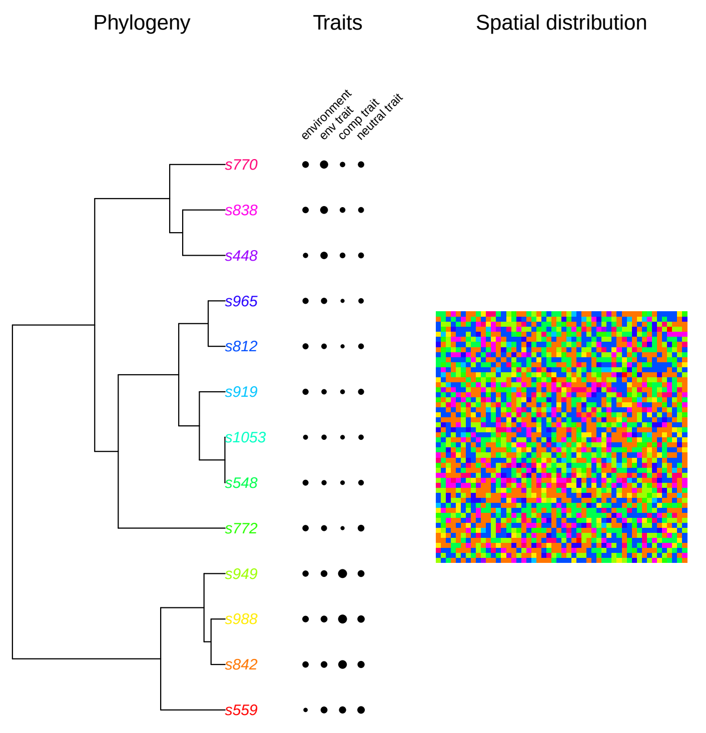


Figure 1.1: The basic output of the PhyloSim simulation: spatial distribution, phylogeny and traits in a biogeographic region

1.2 State variables

The grid: The simulation runs on a spatially discrete grid. Each cell is occupied by one individual at any time. Additionally an environmental gradient can be included. To avoid boundary effects the boundaries of the grid are warped.

Indivuduals: Each individual has three traits: an environmental trait, a competition trait, and a neutral trait. The properties of these traits are described in more detail in section 1.3.

Species and phylogeny: Each individual belongs to a species. Species properties are described by the mean of the respective traits of all individuals of the species. The species traits serve as an attractor in the individual trait evolution proces. Further, each species contains information about its parent species and the species that emerged from it, which gives rise to the overall phylogeny.

1.3 Ecological processes

The simulation runs over a defined number of generations. In each generation, for each individual, the following ecological processes occurr:

1. Mortality / reproduction If an individual dies, it is replaced by a surrounding individual. What surrounding means depends on the dispersal kernel that is chosen. Dispersal can be either global, or it may occurr with an exponential kernel, which creates a weighting w_d according to

In the base version, individuals have identical mortalities of one per time step. However, this can be modified towards fitness-dependent mortality rates

$$w_d = exp - \frac{d}{\delta} \tag{1.1}$$

with a typical dispersal distance δ that can be set by the user. To increase the computational speed, there is a dispersal cutoff at 2δ, meaning that individuals disperse only in a distance of 2δ

2. Evolution The second step in the simulation of one generation is the trait evolution. Trait evolution is modelled by a small gaussian random variable that is added on top of the parental trait values. Additional to the random step, a directed attraction towards the species' mean value is implemented

$$newtrait = (1 - w_s \cdot p) + (w_s \cdot s) + (w_r \cdot r) \tag{1.2}$$

3. Fitness calculation After reproduction, the new individual will calculate it's fitness according to its competition and environmental traits. For example, the closer the values of the competition traits in the surrounding of an individual, the stronger its fitness is reduced. Numerically it can be written as the sum of all absolute competition trait differences in a certain area around the individual

$$r_i = \frac{\sum_{j=1}^{n} \parallel (c_i - c_j) \parallel}{n} \tag{1.3}$$

1.4 Speciation mechanisms

After all individuals have reproduced according to 1-3, speciation occurs. The default speciation mechanism is point speciation. Their frequency of these events is controlled by the speciation rate in the parameter settings. Each generation a number of new species is introduced in the system replacing randomly chosen individuals. The traits of these individuals are calculated from the parent's traits and a randomly generated value as follows:

$$newtrait = (w_p \cdot p) + (w_r \cdot r) \tag{1.4}$$

There are several more speciation mechanisms implemented in the model:

- Point Mutation: A randomly chosen individual becomes a new species
- Fission Type 1: A species is randomly chosen and every second indiviual of the species becomes part of a new species.
- Fission Type 2: A species is randomly chosen and geographically split in two parts. Whereas one part evolves tw a new species, the other part is not affected.
- Protracted Speciation: Based on [1] species undergoes an 'incipient' stage before becomming a new species.
- Red Queen: Based on [2] New species have advantages over old species due to their novelty alone. The effect vanishes over time.

[1] Rosindell, J., S. J. Cornell, S. P. Hubbell, and R. S. Etienne (2010). Protracted speciation revitalizes the neutral theory of biodiversity. *Ecology Letters 13*(6), 716–727

[2] O'Dwyer, J. P. and R. Chisholm (2014). A mean field model for competition: from neutral ecology to the red queen. *Ecology letters 17*(8), 961–969

1.5 Running the model in R

The first thing to do is obviously loading the package:

```
library(PhyloSim)
```

Running the model consists of two consecutive steps. First a list of parameters is generated based on the users settings.

```
par ← createCompletePar(x = 50, y = 50, dispersal = "global"
    , runs = c(500,1000), density = 1, environment = 0.5 ,
    specRate = 1, type="base")
```

In this example, an area of 50*50 cells provides the basis for one simulation with outputs recorded after 500 and 1000 generations. Also the results after 500 generations will be stored in the output. The dispersal is set to global, meaning each individual could reproduce in every cell of the grid. Further, the density as well as the environment have an influence on the individuals. For further explanations and more settings see:

```
?createCompletePar
```

The list of parameters is now being used to execute the simulation

```
simu ← runSimulation(par)
```

The output is saved as an object of type "Phylosim". Each Phylosim object contains two lists, $Output and $Model. All model results are saved in the Output list.

In this example, the output consists of two results, the simulation after 500 and 1000 generations. They contain the spatial species matrix, the (environmental) trait matrix, the environmental matrix (representing the environment), the competition matrix and the neutral matrix as well as the phlogeny in two formats.

To acces them, the easiest way is to use indices

```
Output1 ← simu$Output[[1]]
str(Output1)
```

```
List of 7
 $ specMat  : int [1:50, 1:50] 9138 9176 8251 8694 8251 8251 8849 7721
    7721 7022 ...
 $ traitMat : num [1:50, 1:50] 0.698 0.466 0.695 0.464 0.674 ...
 $ envMat   : num [1:50, 1:50] 0 0 0 0 0 0 0 0 0 0 ...
 $ compMat  : num [1:50, 1:50] 0.998 0.182 0.99 0.165 0.979 ...
 $ neutMat  : num [1:50, 1:50] 0.702 0.559 0.711 0.569 0.721 ...
 $ phylogeny:List of 6
  ..$ edge       : int [1:4288, 1:2] 2146 2147 2148 2149 2150 2151 2152
    2153 2154 2155 ...
  ..$ Nnode      : int 2144
  ..$ tip.label  : chr [1:2145] "s1" "s2295" "s1742" "s1668" ...
  ..$ edge.length: num [1:4288] 2 1 1 1 2 3 4 1 1 2 ...
  ..$ node.label : chr [1:2144] "s1" "s1" "s1" "s1" ...
  ..$ root.edge  : num 1
  ..- attr(*, "class")= chr "phylo"
  ..- attr(*, "order")= chr "cladewise"
 $ phyloTXT : chr "((((((((((((((((((((((((s1:10,s2295:15)s1:30,s1742
    :27)s1:2,s1668:23)s1:2,s1599:13)s1:8,s1521:16)s1:5,s1460:11)s1:13,
    s1100:13)"| __truncated__
```

The simulation settings are saved in the model object.

```
str(simu$Model)
```

```
List of 22
 $ x                          : num 50
 $ y                          : num 50
 $ dispersal                  : chr "global"
 $ runs                       : num [1:2] 500 1000
 $ specRate                   : num 1
 $ density                    : logi TRUE
 $ environment                : logi TRUE
 $ fitnessActsOn              : chr "mortality"
 $ fitnessBaseMortalityRatio: num 10
 $ densityCut                 : num 1
 $ seed                       : int 153
 $ type                       : chr "base"
 $ scenario                   : NULL
 $ fission                    : num 1
 $ redQueen                   : num 0
 $ redQueenStrength           : num 0
 $ protracted                 : num 0
 $ airmatR                    : num 1
 $ soilmatR                   : num 1
 $ compStrength               : num 1
 $ envStrength                : num 0.5
 $ runtime                    : num 3.64
```

The Phylosim object serves as the basis of all further analysis. It can directly be passed to all other funktions. If the Phylsim object contains multiple results, they can be accessed by the which.result argument in most functions. By default, the last result is used.

Often, users will want to run several simulations with different or equal parameters. For this purpose, the runSimulationBatch function allows to run a list of parameter combinations. The function can make use of parallel computing and returns an object of class PhylosimList, which is essentially a list of phylosim classes. In the following example, two slightly different parameter sets are created and processed using parallel computing on two cores.

```
par1 <- createCompletePar(x = 50, y = 50, dispersal = "global"
    , runs = 1000, density = 1, environment = 0.5, specRate =
     1, type="base")

par2 <- createCompletePar(x = 50, y = 50, dispersal = "global"
    , runs = 1000, density = 1, environment = 2, specRate =
    1, type="base")

parmBatch <- list(par1, par2)

simuBatch <- runSimulationBatch(parmBatch, parallel = 2)
```

The simuBatch object contains two Phylosim objects saved in a list. You can easily access them seperately by indexing. For example:

```
simu1 <- simuBatch[[1]]
```

A summary of the basic classes and functions to run simulations is provided in Fig. 1.2.

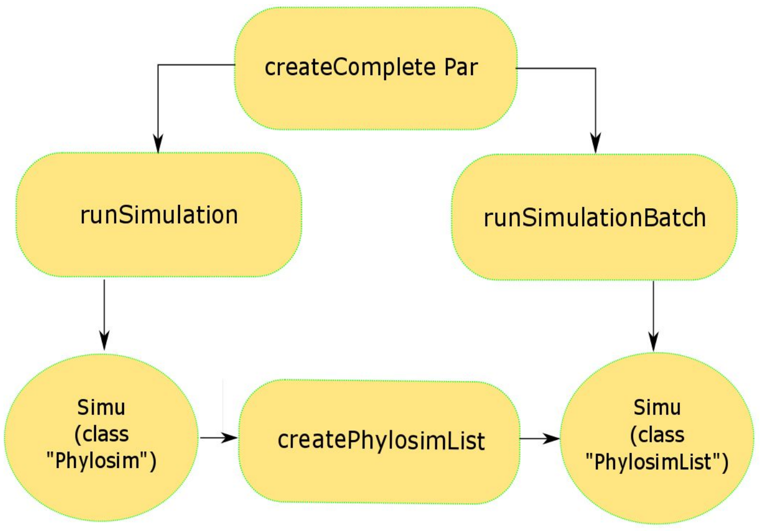


Figure 1.2: Overview about the basic model functions in the PhyloSim package. Functions are represented by bubbles, objects by circles. Objects of the class PhylosimList can be indexed if a single Phylosim object is needed.

2
Plots and summary statistics

After the description of the simulation environment in the previous chapter, the present explains plots and summary statistics that can be created from the PhyloSim and PhyloSimList.

2.1 Plot Functions

The phylosim package contaions multiple functions to visualize the results of the simulation, which are visualized in Fig. 2.8 at the end of this chapter.

plotSpatialPhylo: The easiest way to get an overview of the results of one simulation is the plotSpatialPhylo function, which is also the default plot function for a phylosim object:

```
plotSpatialPhylo(simu, plot = "both", plotTraits = T,
    which.result = NULL)
```

Species-area curve (SAC): The SAC shows the accumulated species richness as a function of area. It is calculated by creating plots of different sizes in the meta-community

```
sac(simu, which.result = NULL)
```

Rank-abundace curve (RAC): The rank-abundance curve shows the abundance of each species within the meta-community

```
rac(simu, which.result = NULL)
```

Trait plots: For a closer look at the traits you can visualize them using the plotTraitDistribution function.

```
plotTraitDistribution(simu = simu, which.result = NULL, type =
    "all")
```

Reconstructed Phylogenies: The function phyloReconstruct uses the species' traits to construct a phylogeny for the given community. These phylogenies can then be compared to the real phylogeny of the community:

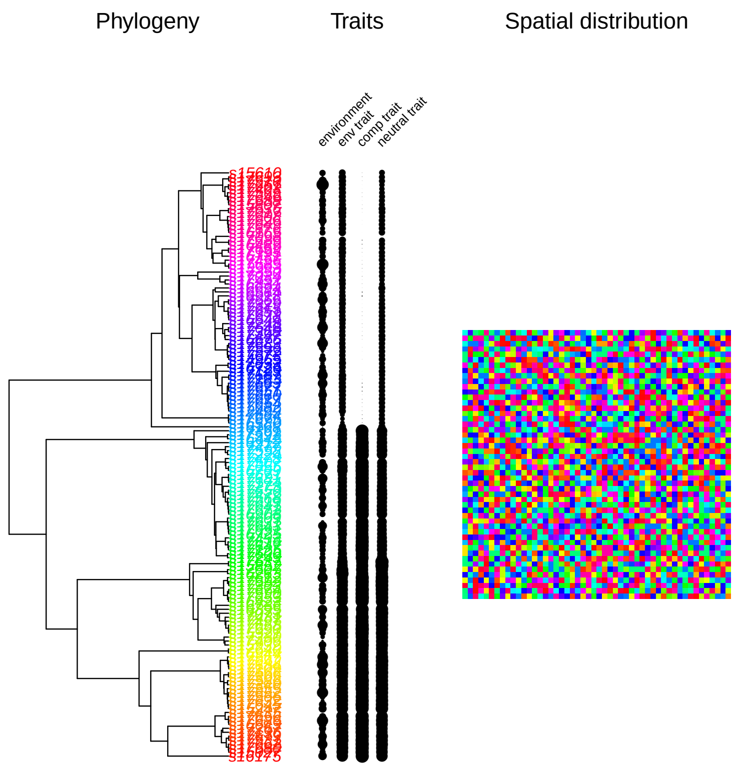


Figure 2.1: The plotSpatialPhylo plot consist of three different figures, a phylogeny that shows the evolutionary history of the species, a trait plot that shows the traits values for each of the species (coded by size), and the spatial distribution of individuals, with colors matching the colors in the phylogeny. Which of these figures are plotted can be triggered with the arguments *plot* and *plotTraits* (see help)

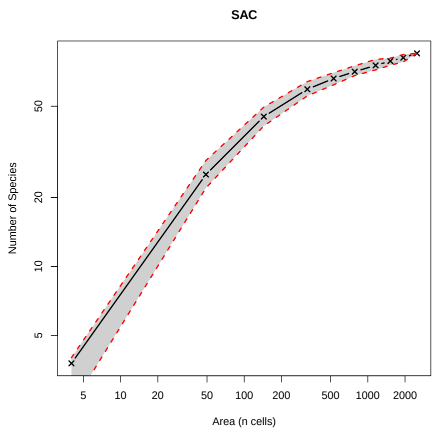


Figure 2.2: Shows the species richness in dependency to the plotsize of a local community. A positively bent curve indicates clustering of a species community. An increase in plot size leads to an increase in specis richness. A negatively bent curve indicates a more neutral distribution of species within the community.

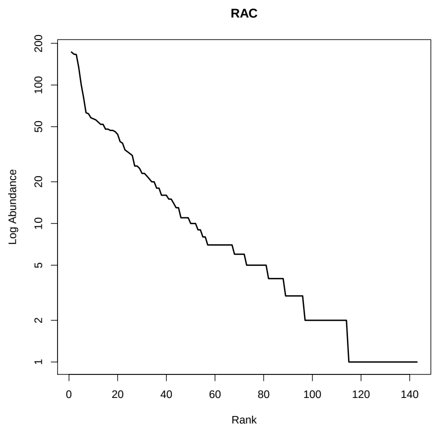


Figure 2.3: Shows the rank-abundance curve (RAC). For that, each species is given a rank according to its abundance (highest abundance = rank 1). Then the abundance is plotted in dependency to the species' rank. RACs display the amount of equally abundand species that the community can support. A linear curve indicates a less stable or neutral community supporting only a few highly abundand species, whereas an S-shaped curve indicates a more stable community. In the latter case several species of the same abundance can be supported.

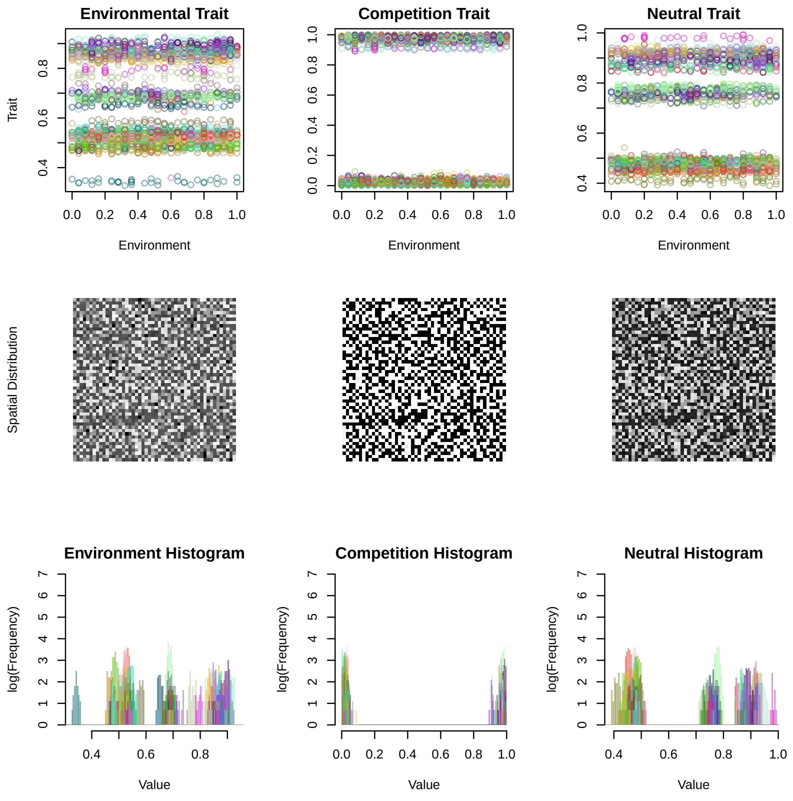


Figure 2.4: The plotTraitDistribution plot consists of three different figures for the three traits in the model (environmental trait, competition trait, neutral trait). The upper panel illustrates the trait's magnitude in dependency of the environment for different species. The middle panel illustrates the spatial distribution of the given trait. The lower panel shows a histogram of the given trait.

```
Phyl    ← ape::drop.fossil(simu$Output[[2]]$phylogeny)
rePhyl ← phyloReconstruct(simu)

par(mfrow=c(1,2))
plot(rePhyl, main="reconstructed Phylogeny")
plot(Phyl, main="real Phylogeny")
```

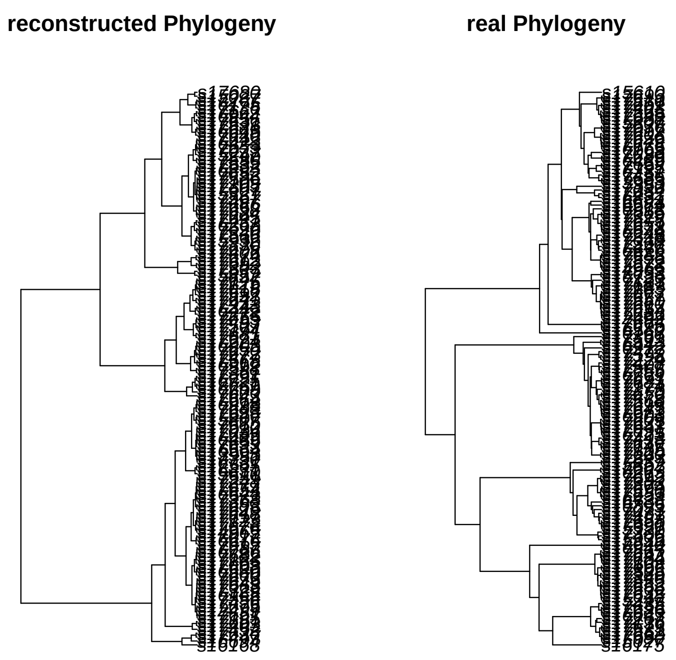


Figure 2.5: Shows the reconstructed as well as the real phylogeny. It is visible that there are differences in the speces that make up the resulting communities as well as in their evolution / assembly

Highlight Local Communities hihglightLocalPhylo highlights a phylogeny of a local community within a metacommunity.

```
highlightLocalPhylo(simu, size = 10,n = 1)
```

```
highlightLocalPhylo(simu, size = 5, n = 1)
```

Both figures show the phylogeny of the same metacommunity. But they highlight different local plots with different plotsizes (10 and 5). Obviously the larger plot is richer in species than the smaller one.

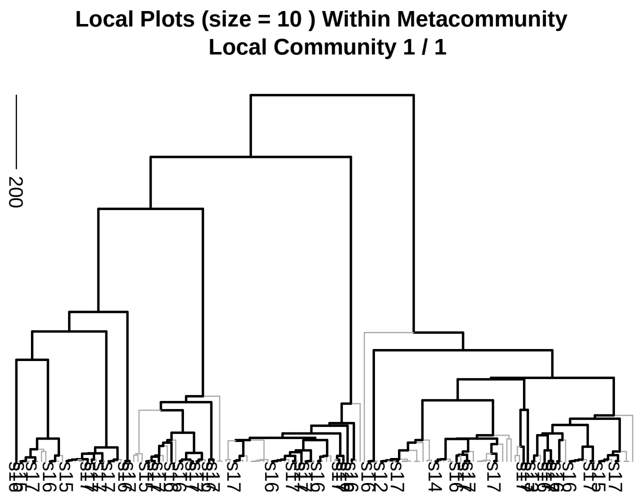


Figure 2.6: Highlights a relatively large local community

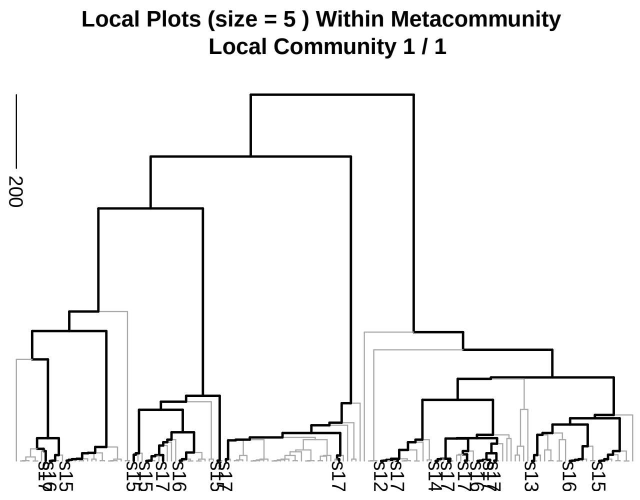


Figure 2.7: Highlights a relatively large local community

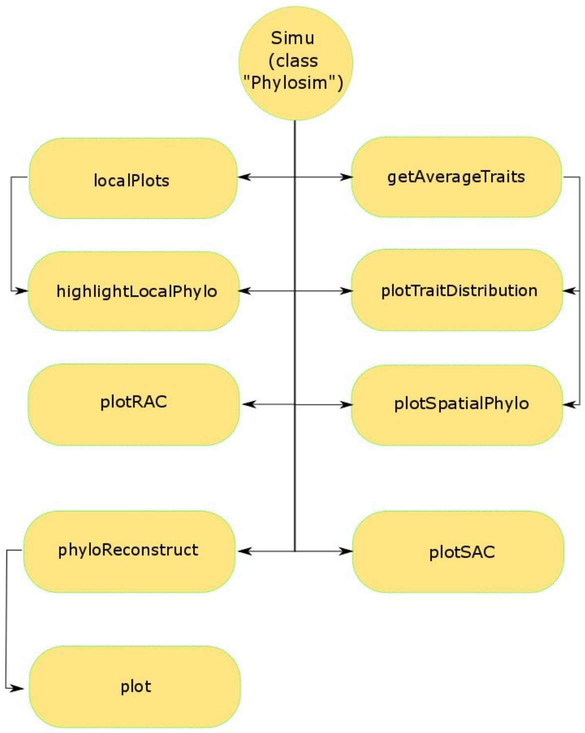


Figure 2.8: Overview about the plotting functions in the Phylosim package. Functions are represented by bubbles, objects by circles. The arrows illustrate what is used as input parameter for the functions

2.2 *Summary statistics*

Quite a few of the functions suc as species-area curves, local phylogenies are based on the localPlots function, a simple observation model that creates random subplots of a given size within the metacommunity

```
localPlots(simu, size = 10, n = 2)
```

```
$subPlots
$subPlots[[1]]
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]
 [1,] 16903 16903 15027 17556 17493 17472 12499 15321 16769 17123 16903
 [2,] 16643 15807 16263 17551 13993 13993 17607 15807 13993 17305 14965
 [3,] 12499 16903 16165 16263 16676 14965 17125 13993 16731 15247 17403
 [4,] 17652 14965 16643 14965 16944 15027 16564 16944 17368 16944 17109
 [5,] 14965 15807 17248 15247 15807 15247 13993 17267 17243 16580 16643
 [6,] 17645 17403 16903 16423 15807 16423 16165 17195 15807 17125 17633
 [7,] 16903 15247 17329 16731 16175 16918 16769 16165 17431 17368 17109
 [8,] 15807 16643 17617 16643 17195 17368 15610 16644 14965 17631 17447
 [9,] 16903 16580 16903 17654 16903 16175 16165 16643 17273 12499 16903
[10,] 17567 16694 17248 16903 16903 17368 15807 17447 17017 16423 16731
[11,] 17556 15321 16263 13993 15807 17273 16903 15610 17125 15807 16944

$subPlots[[2]]
      [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11]
 [1,] 16263 17572 17586 16175 16903 16263 17497 16165 14965 14965 12499
 [2,] 16903 17109 15807 17592 17017 17646 17368 15027 17109 16644 16165
 [3,] 12499 13993 17619 16165 17452 16903 12499 12499 15807 16903 17617
 [4,] 16903 14965 16644 17305 17125 12499 16165 17174 12499 17174 16165
 [5,] 16643 15807 16769 17017 17551 12499 17472 16165 17607 14965 15321
 [6,] 17531 16944 15610 16903 17329 17303 13993 17653 12499 15852 17125
 [7,] 15807 12499 17607 16903 16903 17267 16903 15807 12499 17368 16725
 [8,] 16725 17195 15247 17329 14965 17195 17017 17456 17125 17472 17305
 [9,] 15852 15610 16580 17472 17329 17598 13993 16769 17273 15807 15321
[10,] 17646 17632 16903 12499 17477 13993 16769 17368 12499 16423 12499
[11,] 15321 17565 17303 16165 16903 14965 16643 16731 15247 14965 17248


$envPlots
list()
```

2.3 *Null Models*

To compare patterns in the results of the simualtions with random expectations, the package comprises a set of null models. The null model creates subplots from the metacommunity and tests them against random expecations.

```
pValues ← nullModel(simu = simu, abundance = FALSE,
    localPlotSize = 10, numberOfPlots = 20, repetitions =
    100, fun="mpd")
```

The calculatePhylogeneticDispersion function: The calculatePhylogeneticDispersion function and its associated plot functions are a convenience functions for a rather specialized application - the function calculates null models for all runs in a PhyloBatch object and for different plot sizes. Which null model is used is determined by the types argument with the following options.

- "PhylMeta" equivalent to the nullModell function with abundace = FALSE
- "PhylSample" equivalent to the nullModell function with abundace = TRUE
- "PhylPool" uses picante::ses.mpd with null.model = "phylogeny.pool"
- "SamplePool" uses picante::ses.mpd with null.model = "sample.pool"

```
pValuesBatch ← calculatePhylogeneticDispersion(simuBatch,
    plotlength=20, plots=20, replicates=20, type="PhylMeta")
```

The result can be visualized with the plotPhylogeneticDispersion function:

```
data("pPD.pvalues")
data("pPD.positions")

plotPhylogeneticDispersion(pvalues = pPD.pvalues, positions =
    pPD.positions, title = "Null Meta")
```

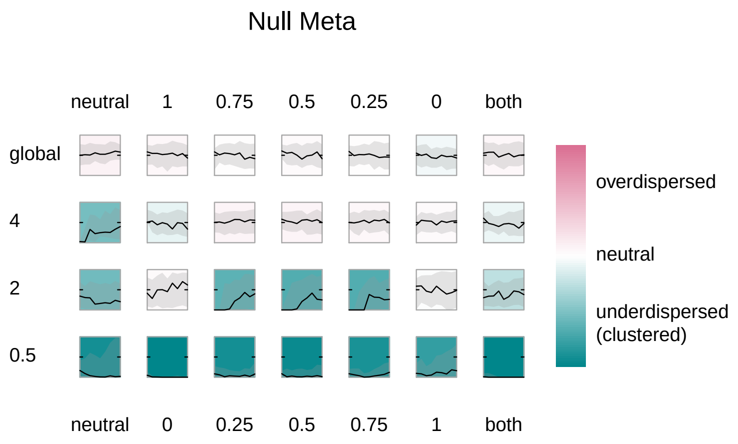


Figure 2.9: Shows different combinations of dispersal limitation (y-axis), environmental trait(upper x-axis) and density dependnce (lower x-axis). The respective colour shows the phylogenetic pattern (over- or underdispersed).

3
Bibliography

O'Dwyer, J. P. and R. Chisholm (2014). A mean field model for competition: from neutral ecology to the red queen. *Ecology letters 17*(8), 961–969.

Rosindell, J., S. J. Cornell, S. P. Hubbell, and R. S. Etienne (2010). Protracted speciation revitalizes the neutral theory of biodiversity. *Ecology Letters 13*(6), 716–727.