THE PHYLOSIM PACKAGE

1
Abstract

The question whether or to which degree community assembly processes can be determined by examining their resulting spatial and phylogenetic patterns is of central importance in ecology. It is these processes that create and maintain local assemblages.

Currently there are many different explanations as to how species communities are shaped based on the emerging spatial or phylogenetic patterns. The phylosim package represents an expansion of previous models by combining both, the spatial and the phylogenetic aspect, into one model.

The models' simulation can include different mechanisms of community assembly such as dispersal limitation, environmental preferences and local competition, whereas the outcome consists of phylogenetic and spatial patterns. The mechanisms can be activated seperately or in combination. This allows the user to identify biogeographic or phylogenetic patterns exclusively correlated with a certain mechanism or combination of mechanisms.

In addition the central model the package comprises a set of functions to illustrate and analyze the results.

Contents

1 *Abstract* 2

2 *Run the Model* 4

2.1 *The grid* 4

2.2 *The indivuduals* 4

2.3 *Processes* 4

2.4 *Implementation in R* 5

2.5 *Run the model* 6

3 *Plots and summary statistics* 9

3.1 *Plot Functions* 9

3.2 *plotSpatialPhylo* 10

3.3 *Trait plots* 11

3.4 *Phylogeny Trees* 12

3.5 *Spatial Patterns* 14

4 *Null Models* 15

4.1 *nullModel function* 16

4.2 *Several model runs testet against null models* 16

4.3 *Illustrate and compare different scenarios* 17

5 *Appendix* 18

2
Run the Model

2.1 The grid

The basis for the simualtion is a grid of spatially discrete cells. Each cell is inhabited by one individual at any time. Additionally an environmental gradient can be included. To avoid boundary effects the boundaries of the grid are warped.

2.2 The indivuduals

Each individual belongs to a species. The properties of the individual are represented in two traits ranging between zero and one. The competition trait represents the individual's preferences apart from the environment. The more similar the traits of two individuals, the stronger they compete with each other.

The environmental trait simulates the individual's interaction with the environment. It is defined as the mean of a Gaussian Distribution with σ set to 0.047. This represents a niche width covering ten percent of the total space of the respective resource (approch see: (**?**)).

Further each individual possesses a neutral trait. This trait evolves over time but not is influencing the individual's properties.

Apart from the individual's traits each species has trait values. These are not independent but defined as 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.

2.3 Processes

The simulation runs over a defined number of generations. In each generation the simulation consists of three consecutive steps.

First the individuals reproduce and spread over the grid. All cells are sequentially chosen at random and the individuals are replaced by the offspring of other individuals.

The individual which produces its offspring and disperses into an empty cell is chosen by an multinomial distribution which consists of all cells within the dispersion kernel. This kernel represents the distance to the cell to be populated and is defined as:

$$w_d = exp\frac{1}{-\Delta_s/2} \tag{2.1}$$

w_d represents the weight derived from the euclidean distance Δ_s between the cell to be populated and the potential parent. An increasing distance to the cell leads to an exponential decline in reproductive fitness. The dispersal kernel can be cut at a certain radius.

The density dependence is based on the competition traits of the indivuduals. The closer the values of the competition traits of two individuals, the stronger they compete. Numerically it can be written as the sum of all absolute competition trait differences in a certain area around the individual. The normalized value is now included in the multinomial distribution as follows:

$$r_i = \frac{\sum_{j=1}^{n} \parallel (c_i - c_j) \parallel}{n} \tag{2.2}$$

The second step in the simulation of one generation is the trait evolution. This step represents genetic recombination in the real world. The trait is modelled as the mean of a Gaussian Distribution (see above). In each generation this value is slightly changed. The new trait is mosty influenced by the parent's value. Further, a weighted random value and the attraction of the specie's value have an influence on the new trait. The trait values during trait evolution are calculated as follows:

$$newtrait = (1 - w_s \cdot p) + (w_s \cdot s) + (w_r \cdot r) \tag{2.3}$$

The last step in a generation is speciation. Here it describes the introduction of a new species via point mutations. These mutations can either become more abundand or go extinct (**?**).The 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:

$$newtrait = (w_p \cdot p) + (w_r \cdot r) \tag{2.4}$$

2.4 *Implementation in R*

The basic structure of how the model is applied is shown in figure 2.1.

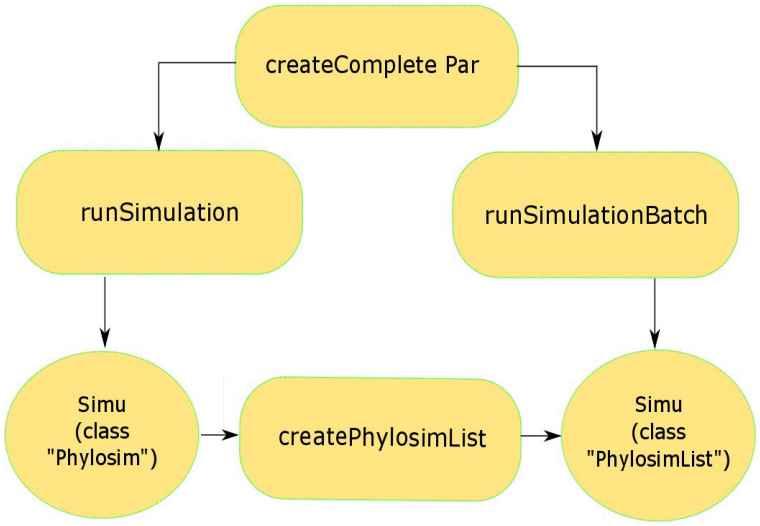

Figure 2.1: Overview about the basic model functions in the PhyloSim package. Functions are represented by bubbles, objects by circles. Objects of the class PhylosimList can be indexed if a single Phylosim object is needed.

2.5 *Run the model*

The first thing to do is obviously loading the package:

```
library(PhyloSim)
```

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

In the following example an area of 50*50 cells provides the basis for one simulation with 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.

The package can run three types of models. The default model, a Leipzig model and a neutral model purely written in R. The last one is to be seen only for test and teaching purpose. To be used in practice it is far too slow.

For further explanations and more settings see:

```
?createCompletePar
```

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

The list of parameters is now being used to exectute the simulation:

```
simu <- runSimulation(par)
```

The output is saved as an object of type "Phylosim". Each Phylosim object consists of 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. To acces them the easiest way is to use indexes.

```
Output1 <- simu$Output[[1]]
```

There are several objects in this list.

```
str(Output1)
```

```
List of 7
 $ specMat  : int [1:50, 1:50] 289 207 366 304 289 304 366 366 304 304
    ...
 $ traitMat : num [1:50, 1:50] 0.474 0.404 0.478 0.345 0.473 ...
 $ envMat   : num [1:50, 1:50] 0 0 0 0 0 0 0 0 0 0 ...
 $ compMat  : num [1:50, 1:50] 0.565 0.395 0.612 0.335 0.57 ...
 $ neutMat  : num [1:50, 1:50] 0.462 0.5 0.475 0.515 0.449 ...
 $ phylogeny:List of 6
  ..$ edge       : int [1:44, 1:2] 24 25 26 27 28 28 27 29 30 31 ...
  ..$ Nnode      : int 22
  ..$ tip.label  : chr [1:23] "s1" "s188" "s118" "s453" ...
  ..$ edge.length: num [1:44] 27 29 29 64 282 67 181 124 18 55 ...
  ..$ node.label : chr [1:22] "s1" "s1" "s1" "s1" ...
  ..$ root.edge  : num 37
  ..- attr(*, "class")= chr "phylo"
  ..- attr(*, "order")= chr "cladewise"
 $ phyloTXT : chr "(((((s1:282,s188:67)s1:64,(((s118:55,(s453:5,s485:5)
    s453:50)s118:18,s434:73)s118:124,s325:21)s118:181)s1:29,((s94:8,s483
    :8)s94:"| __truncated__
```

These are the 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.

The parameter settings are only saved once. Additionaly, the run time is saved with these settings.

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

```
List of 16
 $ 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 5238
 $ type                     : chr "base"
 $ scenario                 : NULL
 $ compStrength             : num 1
 $ envStrength              : num 0.5
 $ runtime                  : num 4.81
```

In the package, 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.simulation argument in most functions. By default always the last result is used.

Because the simulations have a high computational cost, the package comprises a funktion to make use of parallel computing. This function can be seen as an extension to the runSimulation funktion in order to accelerate the simulation of multiple scenarios.

As input serves a list of parameter settings as created by createCompletePar, whereas the output is an object of tyoe PhyloBatch. Basically this object is a list of Phylosim objects.

In the following example two slightly different parameter sets are created and processed using parallel computing.

```
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]]
```

3
Plots and summary statistics

3.1 Plot Functions

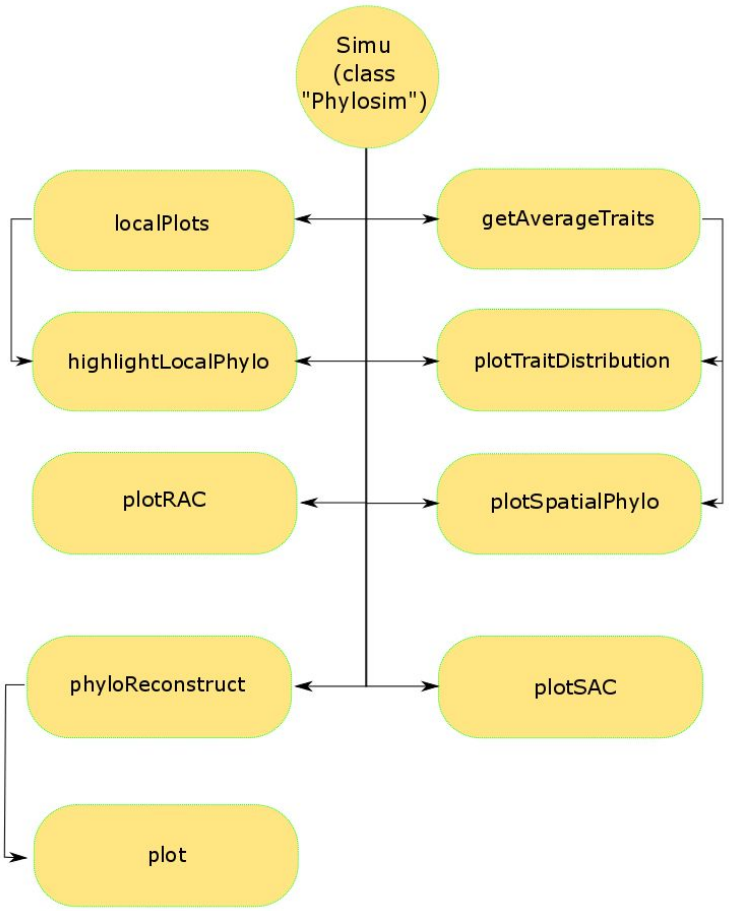

Figure 3.1: Overview about the plotting functions in the Phylosim package. Functions are represented by bubbles, objects by circles. The arrows illustrate what is used as input parameter for the functions

3.2 *plotSpatialPhylo*

The phylosim package contaions multiple functions to visualize the results of the simulation.

The easiest way to get an overview of the results of one simulation is the plotSpatialPhylo function.

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

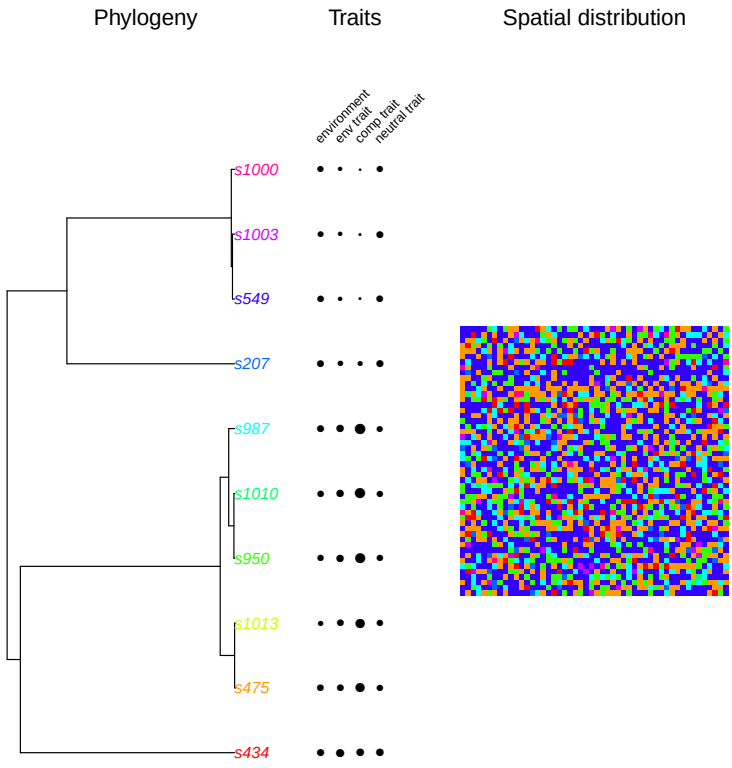

Figure 3.2: Shows the plot created by plotSpatialPhylo with all three possibilites (phylogeny tree, traits, and spatial distribution)

The plotSpatialPhylo plot consist of three different figures. The phylogeny tree that shows the phylogeny of the modeled community with colored tip labels. A traitplot that shows the magnitude of different traits for each of the species within the community. And a plot that shows the spatial abundance of the species within the community. Here the species have the same color as the tip labels for recognition. Which of these figures are plotted can be triggered with the arguments "plot" and "plotTraits" (see help)

3.3 *Trait plots*

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

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

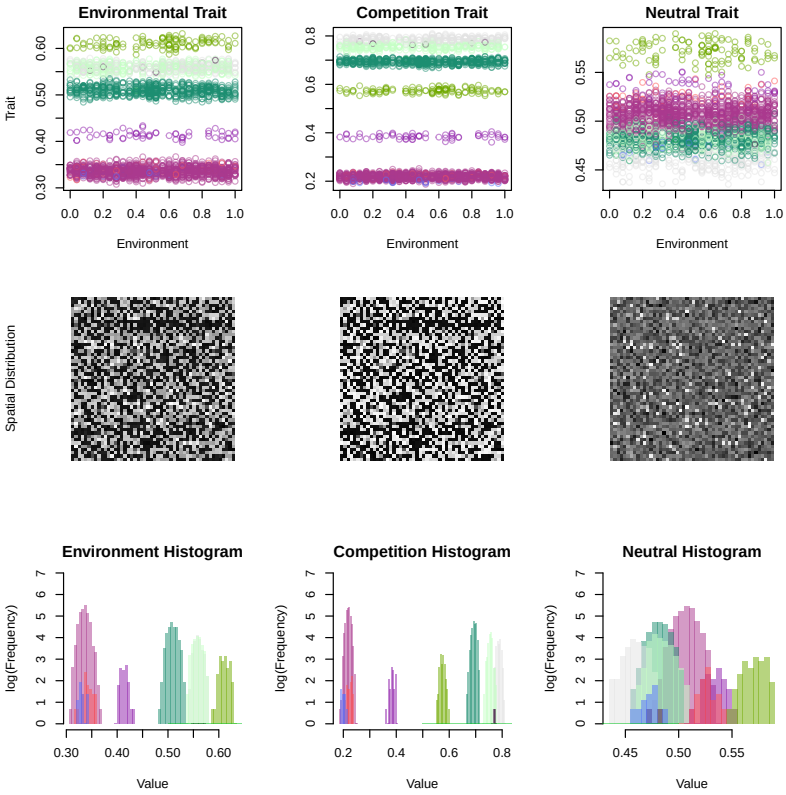

Figure 3.3: Shows the different traits in dependency to the environment (upper), their spatial distribution (middle) and the frequency of the traits' values per species.

The plotTraitDistribution plot consists of three different figures for the three traits in the model (environmental trait, competition trait, neutral trait). The first illustrates the trait's magnitude in dependency of the environment for different species. The second illustrates the spatial distribution of the given trait. The third shows a histogram of the given trait. As mentioned above each individual has a neutral trait. This trait evolved over time but is not influencing the individual's fitness

3.4 *Phylogeny Trees*

When it comes to analizing the phylogenies of species, it is often useful to look at their phylogeny trees to get an idea how the community evolved. Despite the plotSpatialPhylo function (which includes a phylogeny tree to get an overview rather than for analization) there are two more functions that create phylogeny trees. These are called phyloReconstruct and hihglightLocalPhylo.

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:

```
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")
```

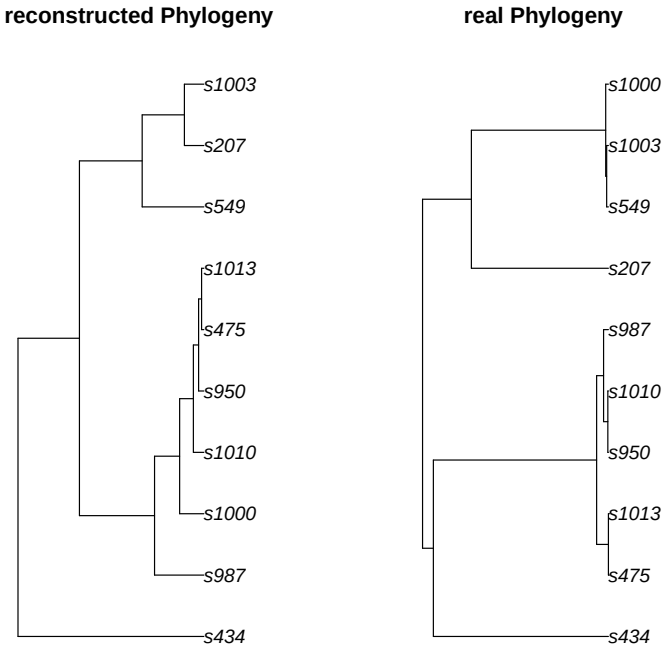

Figure 3.4: Shows the reconstructed as well as the real phylogeny. It is visible that there are differences in the speces that make up the resulting communities as well as in their evolution / assembly

Highlight Local Communities

Further, the package comprises a plot function to highlight the phylogeny of a local community within a metacommunity. This function is called hihglightLocalPhylo.

```
highlightLocalPhylo(simu, size = 50, n = 1)
```

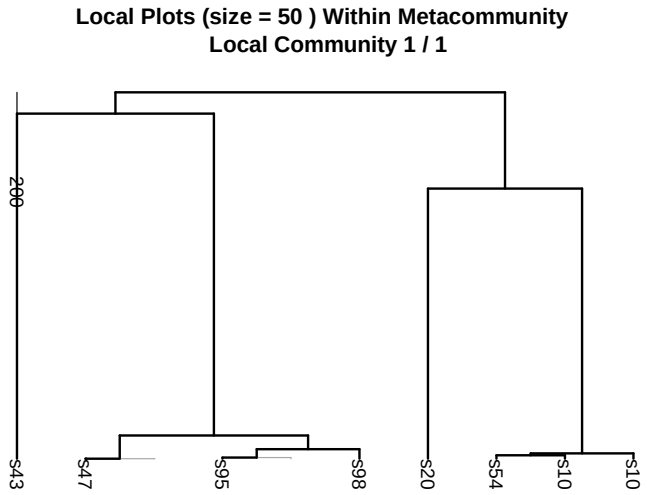

Figure 3.5: Highlights a relatively large local community

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

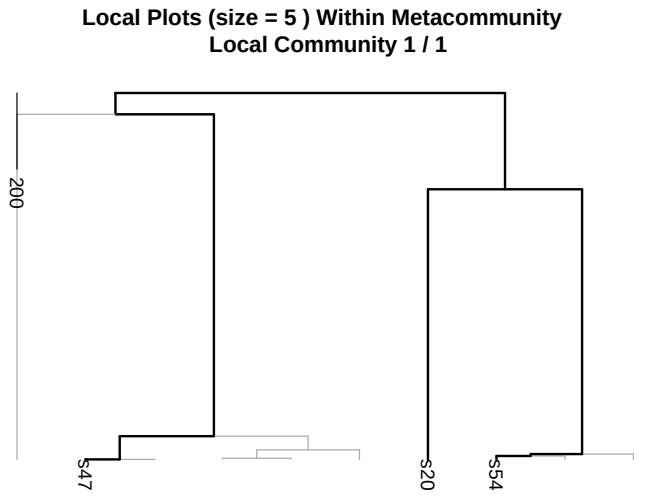

Figure 3.6: Highlights a relatively small local community

Both figures show the phylogeny of the same metacommunity. But they highlight different local plots with different plotsizes (50 and 5). Obviously the larger plot is richer in species than the smaller one.

3.5 *Spatial Patterns*

The package contains two functions to analyze the spatial patterns of the results. The species-area curve (SAC) and the rank-abundace curve (RAC). The SAC shows the accumulated species richness as a function of the plot size while the RAC is an indicator for the stability of a community.

SAC

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

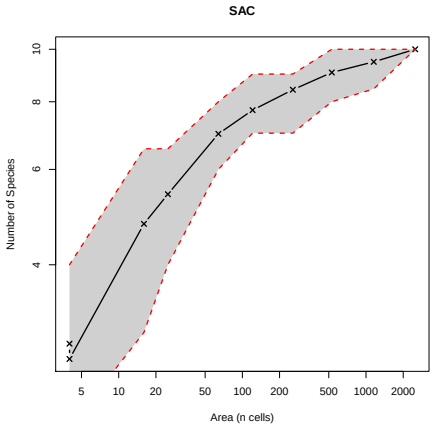

Figure 3.7: Shows the species richness in dependency to the plotsize of a local community. A positively bent curve indicates clustering of a species community. An increase in plot size leads to an increase in specis richness. A negatively bent curve indicates a more neutral distribution of species within the community.

RAC

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

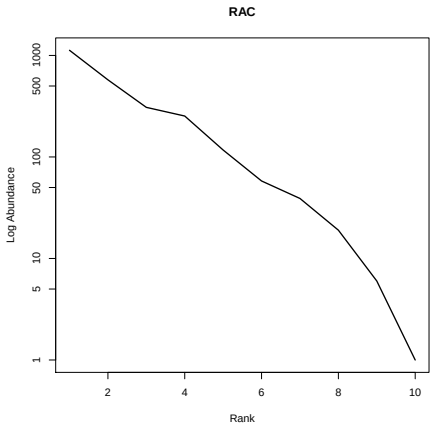

Figure 3.8: Shows the rank-abundance curve (RAC). For that, each species is given a rank according to its abundance (highest abundance = rank 1). Then the abundance is plotted in dependency to the species' rank. RACs display the amount of equally abundand species that the community can support. A linear 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.

4
Null Models

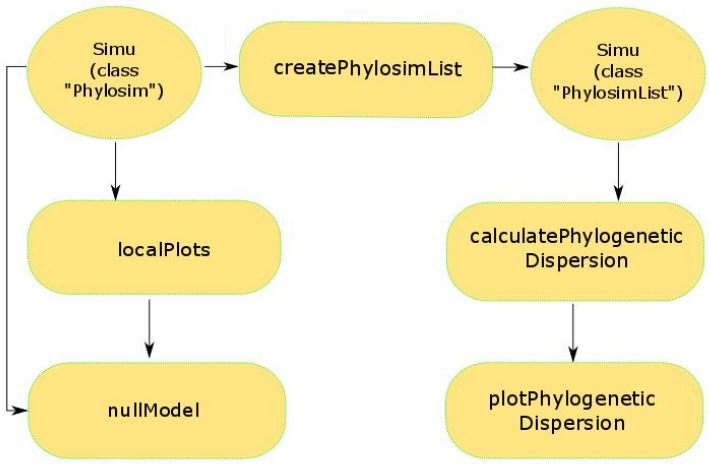

Figure 4.1: Overview about the basic nullmodel functions in the PhyloSim package. Functions are represented by bubbles, objects by circles. Objects of the class PhylosimList can be indexed if a single Phylosim object is needed. The functions "calculatePhylogeneticDispersion" and "plotPhylogeneticDispersion" only make sence for objects of the class PhylosimList because they compare several model runs.

To tackle the question, whether there are patterns in the results of the simualtions, the package comprises a set of null models to test the observed results against. For objects of type Phylosim these models are brought together in the nullModel function.

In the function subplots of the results are created based on a Phylosim object. These subplots can also be created seperately for

different purposes.

```
subPlots ← localPlots(simu = simu, size=100, n=10)
```

These subplots can now be tested against randomly collected plots.

4.1 *nullModel function*

You can choose, whether the null model should be based on the abundance distribution of the sample plots or not.

If the model is not based on the abundance distribution, there are two possible measures of the phylogeny that are used to compare the observed results to the null model. These are the Mean Pairwise Distance (mpd) or Fait's Phylogenetic Diversity (pd).

The Mean Pairwise Distance and Fait's Phylogenetic Diversity have their origin in the picante package. For more information see the help pages in the picante package.

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

4.2 *Several model runs testet against null models*

If you want to analyze larger datasets in an PhyloBatch object the calculatePhylogeneticDispersion function is used. It can be seen as an extension to the nullModel function that deals with lists of Phylosim objects and can also calculate two external null models. 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")
```

4.3 *Illustrate and compare different scenarios*

With the function plotPhylogeneticDispersion it is possible to illustrate and compare different scenarios of community assembly that have different dispersal-, environmental trait- and density dependence attributes. For that, the calculated pvalues of the null models (pvaluesBatch) are used. Because our simuBatch object here only contains two model runs it is more descriptive to just load some pre made data for the plotPhylogeneticDisperion function.

For the exact explanation how to use this function check the plotPhylogeneticDispersion help file.

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

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

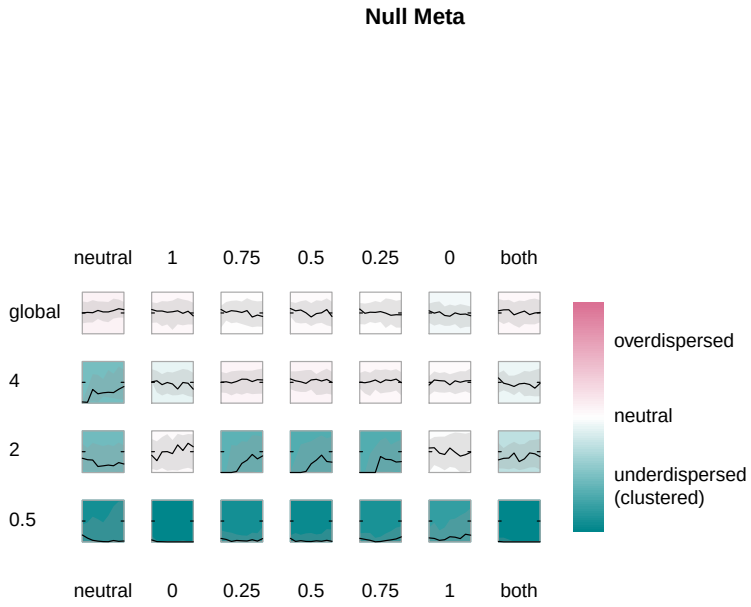

Figure 4.2: Shows different combinations of dispersal limitation (y-axis), environmental trait(upper x-axis) and density dependnce (lower x-axis). The respective colour shows the phylogenetic pattern (over- or underdispersed).

5
Appendix