Table of contents

1. The Packages

2. The BDTT function

3. Example

3.1. Computing BDTT

3.2. Testing the statistical link between BDTT and metadata

1. The needed R packages

Before running the BDTT analysis, you will need to install and load the following R packages:

- ape
- castor
- matrix
- abind

```r
library(ape)
library(castor)
library(abind)
library(Matrix)
```

2. The BDTT function

The BDTT function

requires the follwing inputs:

similarity_slices:

the slices (i.e. the multiple phylogenetic resolutions) at which you want to aggregate the tips of the phylogeny and compute corresponding beta-diversity. 0 corresponds to no aggregation, i.e use the raw tips of the phylogeny as microbial units. Values >0 will aggregate the tips of the phylogeny according to the given value to create aggregated microbial units and compute corresponding beta-diversity. Use the 'getHnodes' function to have an idea of the resolution slices you can explore (see below).

tree:

the species (or OTUs, or sequence variants) phylogeny (the names of the tips must match those in the site*species matrix)

sampleOTUs:

samples * species (or OTUs, or sequence variants) matrix

onlyBeta:

Putting "TRUE" (default) will make the function return beta-diversity dissimilary matrices only Putting "FALSE" will make the function return beta-diversity dissimilary matrices + matrix detailling the relationship between tips and the aggregated units.

metric:

Beta-diversity metric chosen; we provide Jaccard ("Jac") its true turnover component ("Jac_TT") and Bray-Curtis ("Bray").

The function requires the following input:

tree:

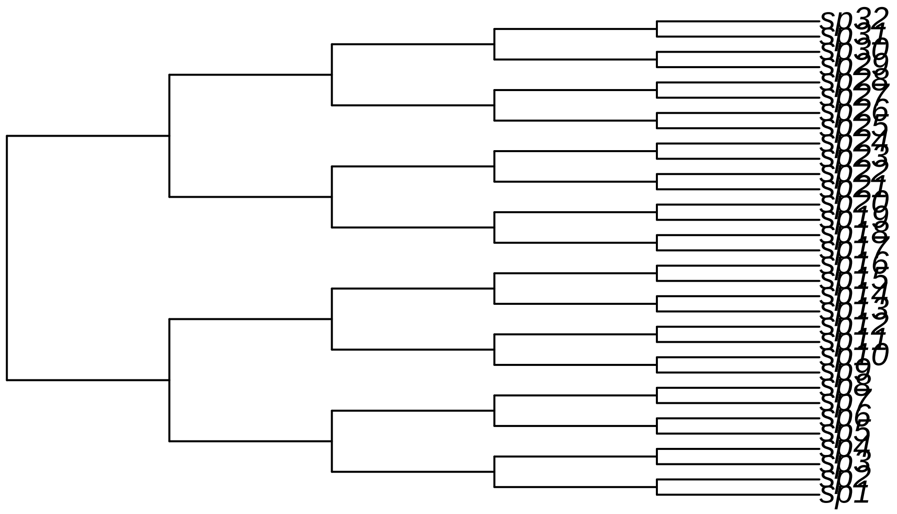
the species (or OTUs, or sequence variants) phylogeny

2. Examples

Computing BDTT

```
library(picante)

data(phylocom)
TreeExample=phylocom$phylo
plot(TreeExample)
```

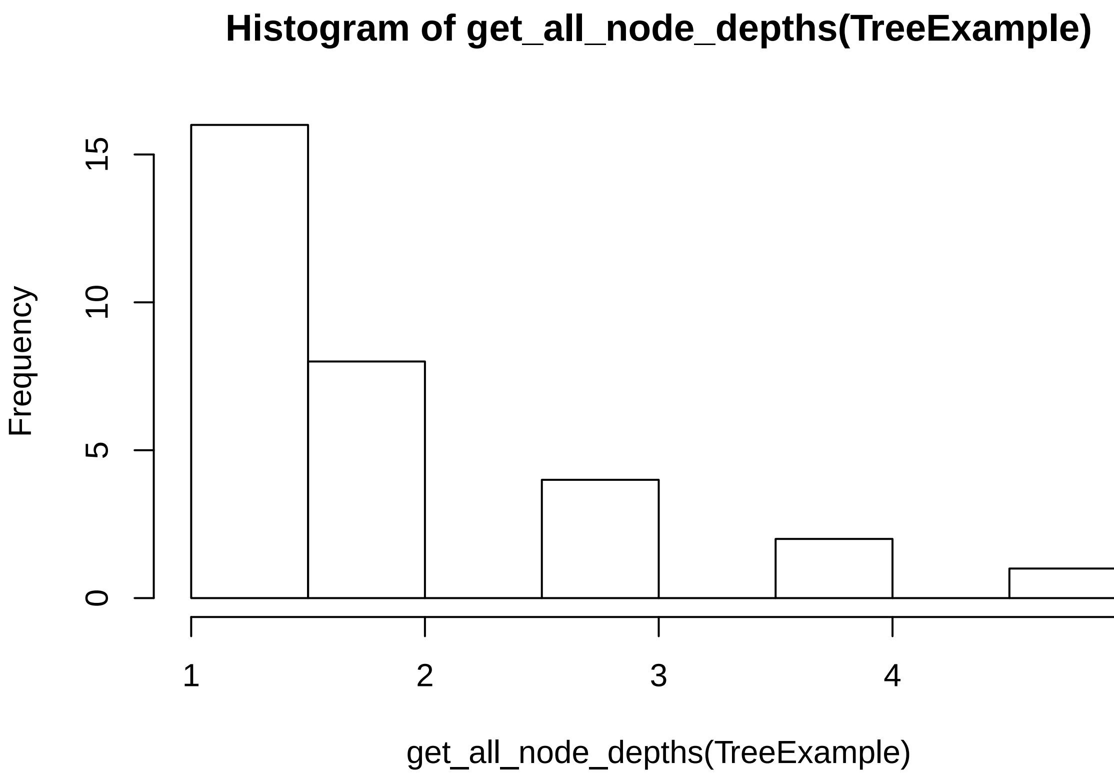


```
SiteSpExample=t(phylocom$sample)
SiteSpExample
```

```
##      clump1 clump2a clump2b clump4 even random
## sp1       1       1       1      1    1      0
## sp10      0       2       0      1    0      0
## sp11      0       2       0      0    0      0
## sp12      0       2       0      0    0      1
## sp13      0       0       0      0    1      0
## sp14      0       0       0      0    0      4
## sp15      0       0       0      0    0      2
## sp17      0       0       2      2    1      3
## sp18      0       0       2      2    0      0
## sp19      0       0       2      0    0      0
## sp2       1       1       1      1    0      1
## sp20      0       0       2      0    0      0
## sp21      0       0       0      0    1      0
## sp22      0       0       0      0    0      1
## sp24      0       0       0      0    0      2
## sp25      0       0       0      2    1      0
## sp26      0       0       0      2    0      0
## sp29      0       0       0      0    1      0
## sp3       1       1       1      0    0      0
## sp4       1       1       1      0    0      0
## sp5       1       0       0      0    1      2
## sp6       1       0       0      0    0      0
## sp7       1       0       0      0    0      0
## sp8       1       0       0      0    0      0
## sp9       0       2       0      1    1      0
```

```
source("BDTT_functions.R")

hist(get_all_node_depths(TreeExample))
```



```
slices=c(0:3)
Betas=BDTT(similarity_slices = slices,tree = TreeExample,sampleOTUs = (SiteSpExample))
```

```
## [1] "0 similarity provides 32 total new OTUs"
## [1] "1 similarity provides 16 total new OTUs"
## [1] "2 similarity provides 8 total new OTUs"
## [1] "3 similarity provides 4 total new OTUs"
```

Linking BDTT with environement / metadata

Create random metada catageory

```
Meta=sample(x=c("Condition_1","Condition_2"),size=dim(SiteSpExample)[2],replace = T)
names(Meta)=colnames(SiteSpExample)
```

```
Meta
```

```
##        clump1       clump2a       clump2b        clump4          even 
## "Condition_2" "Condition_2" "Condition_2" "Condition_1" "Condition_2" 
##        random 
## "Condition_1"
```

Test statistically the link between metadata and BDTT using PERMANOVA

Load vegan to be able to use adonius function

```
library(vegan)
```

Example of the test for a given resolution (0) and a given metric (Jaccard); make sure that samples are in the same order

```
samples=names(Meta)
adonis(Betas["0","Jac",samples,samples]~Meta[samples])
```

```
## 
## Call:
## adonis(formula = Betas["0", "Jac", samples, samples] ~ Meta[samples]) 
## 
## Permutation: free
## Number of permutations: 719
## 
## Terms added sequentially (first to last)
## 
##               Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## Meta[samples]  1   0.29478 0.29479 0.94203 0.19062    0.8
## Residuals      4   1.25170 0.31293         0.80938       
## Total          5   1.54649                 1.00000
```

Construct table to store results in a ready-to-use format for ggplot

```
predictors="Conditions1_2"
StatsRes=expand.grid(similarity_slices=as.character(slices),predictors=predictors,metric=c("Jac","Bray")
StatsRes[["F.Model"]]=StatsRes[["R2"]]=StatsRes[["Pr(>F)"]]=NA
head(StatsRes)
```

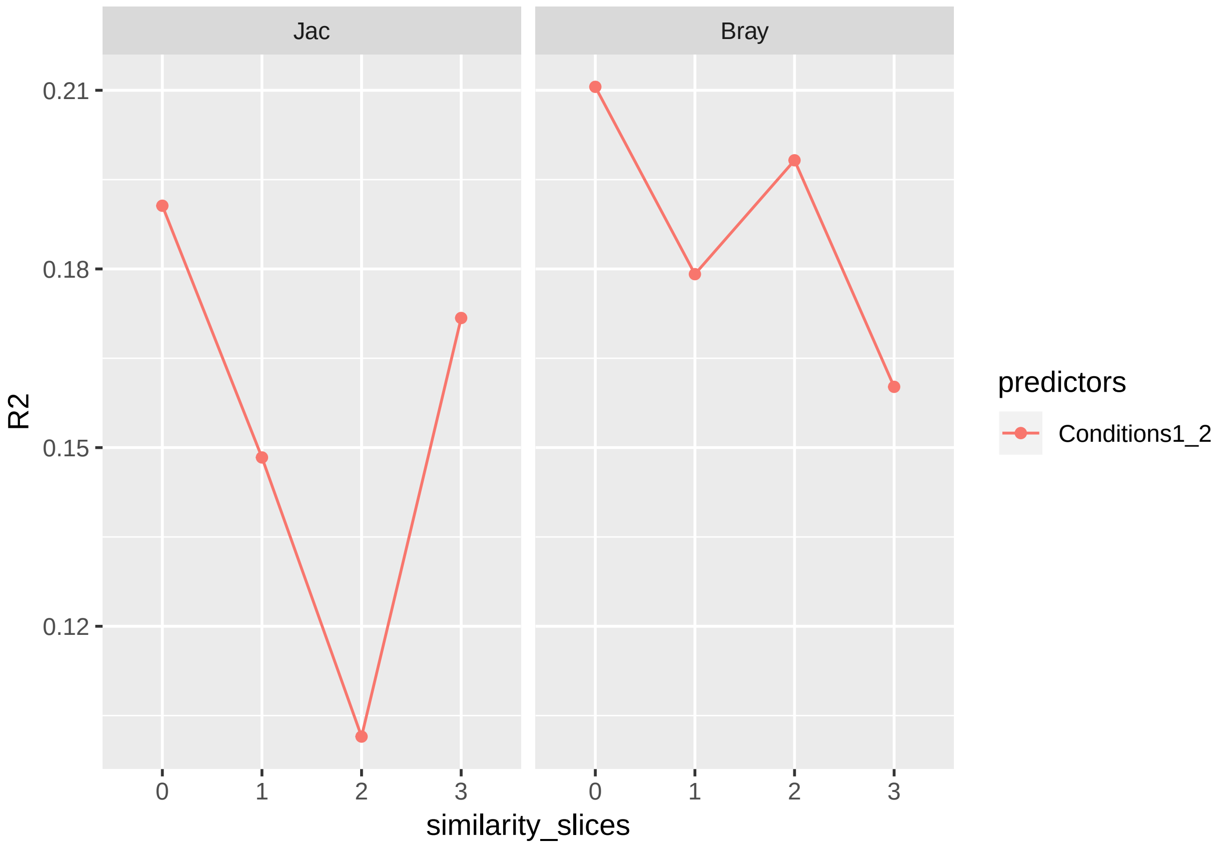
```
##   similarity_slices    predictors metric Pr(>F) R2 F.Model
## 1                 0 Conditions1_2    Jac     NA NA      NA
## 2                 1 Conditions1_2    Jac     NA NA      NA
## 3                 2 Conditions1_2    Jac     NA NA      NA
## 4                 3 Conditions1_2    Jac     NA NA      NA
## 5                 0 Conditions1_2   Bray     NA NA      NA
## 6                 1 Conditions1_2   Bray     NA NA      NA
```

Run multiple PERMANOVA across phylogenetic resolution and store results in a table ready to use for ggplot

```
for (i in as.character(slices))
{
   res=unlist(adonis(formula =Betas[i,"Jac",samples,samples]~Meta[samples])$aov.tab[1,c(6,5,4)])
   StatsRes[(StatsRes$metric=="Jac")&(StatsRes$similarity_slices==i),4:6]=res
   res=unlist(adonis(formula =Betas[i,"Bray",samples,samples]~Meta[samples])$aov.tab[1,c(6,5,4)])
   StatsRes[(StatsRes$metric=="Bray")&(StatsRes$similarity_slices==i),4:6]=res
}
```

We can then plot the profiles of R2 along the phylogenetic time scale:

```
library(ggplot2)
ggplot(aes(y=R2,x=similarity_slices,colour=predictors,group=factor(predictors)),data=StatsRes)+geom_poi
```



or just the profile for the significant effects (not run cause nothing is significant)

```
#ggplot(aes(y=R2,x=similarity_slices,colour=predictors,group=factor(predictors)),data=StatsRes[StatsRes
```