Branch length evaluation for Phylogenetic Diversity: a worked example

Daniel R. Miranda-Esquivel

2020 - 10 - 11

Four taxa and two areas

Preparing the data space

First, we load the required libraries:

```
## cleaning
rm(list = ls())

## libraries

## installing and loading the package

##install.packages("../../blepd_0.mainRev.minorRev.tar.gz", repos = NULL, type="source")

library(blepd)
```

```
## Loading required package: ape
## Loading required package: picante
## Loading required package: vegan
## Loading required package: permute
##
## Attaching package: 'permute'
## The following object is masked from 'package:devtools':
##
##     check
## Loading required package: lattice
## This is vegan 2.5-6
## Loading required package: nlme
```

```
packageVersion("blepd")
```

```
## [1] '0.1.6.2020.10.11'
```

```
## To plot trees you can use ggtree, ape or phytools. The example is based on
## ggtree as a matter of choice.

## More about ggtree
```

```
## https://guangchuangyu.github.io/ggtree-book/chapter-ggtree.html

## If you want to install ggtree go to
## https://bioconductor.org/packages/release/bioc/html/ggtree.html

library(ggplot2)

library(ggtree)
```

```
## Registered S3 method overwritten by 'treeio':
##   method     from
##   root.phylo ape
```

```
## ggtree v2.2.4  For help: https://yulab-smu.github.io/treedata-book/
##
## If you use ggtree in published research, please cite the most appropriate paper(s):
##
## [36m-[39m Guangchuang Yu. Using ggtree to visualize data on tree-like structures. Current Protocols i
## [36m-[39m Guangchuang Yu, Tommy Tsan-Yuk Lam, Huachen Zhu, Yi Guan. Two methods for mapping and visua
## [36m-[39m Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, Tommy Tsan-Yuk Lam. ggtree: an R packag
```

```
##
## Attaching package: 'ggtree'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse
```

```
## The following object is masked from 'package:ape':
##
##     rotate
```

```
library(gridExtra)

library(RColorBrewer)
```

Now, we load the data included in the package: tree and distribution.

```
#data(package = "blepd")

## trees

data(tree)

str(tree)
```

```
## List of 5
##  $ edge       : int [1:6, 1:2] 5 6 6 5 7 7 6 1 2 7 ...
##  $ edge.length: num [1:6] 1 1 1 1 1 1
##  $ Nnode      : int 3
##  $ tip.label  : chr [1:4] "t1" "t2" "t3" "t4"
##  $ root.edge  : num 1
##  - attr(*, "class")= chr "phylo"
##  - attr(*, "order")= chr "cladewise"
```

```
initialTree <- tree
```

```
## distributions

data(distribution)

str(distribution)
```

```
##  int [1:2, 1:4] 1 0 0 1 1 0 0 1
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2] "A1" "A2"
##   ..$ : chr [1:4] "t1" "t2" "t3" "t4"
```

```
dist4taxa <- distribution

## distribution to XY to be able to plot the data, otherwise skip the next step

distXY <- matrix2XY(dist4taxa)


## plotting

## the tree

plotTree <-  ggtree(initialTree, ladderize=TRUE,
                    color="black", size=1, linetype="dotted") +
             geom_tiplab(size=6, color="black") +
             theme_tree2() +
             ggtitle("Four terminals, equal branch length")


##print(plotTree)



## the distribution

plotDistrib <- ggplot(data=distXY,
                      aes(x= Area, y= Terminal),
                      size =11) +
               geom_point() +
               labs(title = "Distributions",
                    y = "",
                    x = "Area")

##print(plotDistrib)

cowplot::plot_grid(plotTree, plotDistrib, ncol=2)
```

We check whether names in both objects: initialTree and dist4taxa, are the same.

```
all(colnames(dist4taxa) == initialTree$tip.label)
```

```
## [1] TRUE
```

We report the branch length, and calculate the PD values.

```
initialTree$edge.length
```

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

```
initialPD <- PDindex(tree=initialTree, distribution = dist4taxa)

initialPD
```

```
## [1] 4 4
```

As expected there is a tie between both areas.

branch swap

An initial approach, is to evaluate the effect in PD when internal and terminal branch lengths are swapped. In this case it is not the sensitivity to the branch length as a parameter, but the stability to the actual branch lengths.

The function to perform the analysis is *swapBL*, that has four parameters: the tree, the distribution, the model to evaluate (valid models are "simpleswap", "allswap" -default value- and "uniform"), the number of times to swap (default value = 100), and branch to swap ("terminals" (default) or "internals").

Uning the default parameters we get.

```
swapBL(tree = initialTree,
       distribution = dist4taxa)
```

```
## model to test allswap reps 100
```

```
## $initialPD
## [1] 4 4
##
## $bestInitialArea
## [1] "A1A2"
##
## $bestModifiedArea
##   AreaSelected Freq
## 1         A1A2  100
##
## $tree
##
## Phylogenetic tree with 4 tips and 3 internal nodes.
##
## Tip labels:
##   t1, t2, t3, t4
##
## Rooted; includes branch lengths.
##
## $distribution
##    t1 t2 t3 t4
## A1  1  0  1  0
## A2  0  1  0  1
##
## $model
## [1] "allswap"
##
## $nTimes
## [1] 100
```

```
##
## $root
## [1] TRUE
##
## $index
## [1] "PD"
##
## $branch
## [1] "terminals"
##
## attr(,"class")
## [1] "blepd"
```

As this is a tree where all branches are equal, there is no impact when the branch lengths are swapped.

Or we could use the random uniform branch length model.

```
swapBL(tree = initialTree,
       distribution = dist4taxa,
       model = "uniform")
```

```
## model to test uniform reps 100
```

```
## $initialPD
## [1] 4 4
##
## $bestInitialArea
## [1] "A1A2"
##
## $bestModifiedArea
##   AreaSelected Freq
## 1         A1A2  100
##
## $tree
##
## Phylogenetic tree with 4 tips and 3 internal nodes.
##
## Tip labels:
##   t1, t2, t3, t4
##
## Rooted; includes branch lengths.
##
## $distribution
##    t1 t2 t3 t4
## A1  1  0  1  0
## A2  0  1  0  1
##
## $model
## [1] "uniform"
##
## $nTimes
## [1] 100
##
## $root
## [1] TRUE
##
```

```
## $index
## [1] "PD"
##
## $branch
## [1] "terminals"
##
## attr(,"class")
## [1] "blepd"
```

This is a tree where all branches are equal, therefore min and max are equal. There is no impact when the branch lengths are swapped, and areas A1A2 are selected.

Function to evaluate a single terminal

To test the effect of changing the branch length in a single terminal ("t1"), we will use the function **evalTerminal**. This function uses four parameters: tree, distribution, tipToEval (label of the tip), approach (two options: "lower"/"upper", to evaluate from 0 to the actual length or from the actual length to the sum of all branch lengths).

The function reports a S3 object, printable with **print.blepd**.

```
terminalT1 <- evalTerminal(tree = initialTree,
              distribution = dist4taxa,
              tipToEval = "t1",
              approach = "lower" )

print.blepd(terminalT1)
```

```
##
## BestInitial:A1A2     Selected:   [1] "A2"
```

```
terminalT1$delta
```

```
## [1] 0.9999
```

The lower limit reported when we change the branch length for terminal t1 is 0.99 [stored as *object**$delta**], therefore, this branch length will modify the area selected from A1A2 to A2, as the tie between the path between terminals t1/t3 (area A1) vs t2/t4 (area A2) will be solved in favour of t2/t4 when A1 is shorter.

In the same way, if we chage t1 to 1.01 the will break in favour of the area A1.

Tree evaluation function

branch length

The function to test all terminals at the same time is *evalTree*, with two parameters: the tree and the distribution. The function returns a data.frame object with 14 fields: labelTerminal, lowerBranchLength, InitialArea, lowerFinalArea, initialLength, upperBranchLength, upperFinalArea, changeLower, changeUpper, deltaUpper, deltaLower, deltaPD, areaDelta, and abDelta.

```
finalResults <-   evalTree(tree = initialTree, distribution = dist4taxa)
```

```
## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 3 has 2 rows to replace 1 rows
```

```
## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 4 has 4 rows to replace 1 rows
```

```
## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 7 has 2 rows to replace 1 rows
```

```
## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 8 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## provided 13 variables to replace 4 variables

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(6, 7), value = list(maxPD = 0, :
## replacement element 2 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 3 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 4 has 4 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 7 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 8 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## provided 13 variables to replace 4 variables

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(6, 7), value = list(maxPD = 0, :
## replacement element 2 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 3 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 4 has 4 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 7 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 8 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## provided 13 variables to replace 4 variables

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(6, 7), value = list(maxPD = 0, :
## replacement element 2 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 3 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 4 has 4 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 7 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## replacement element 8 has 2 rows to replace 1 rows

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(2:5), value = structure(list(:
## provided 13 variables to replace 4 variables

## Warning in `[<-.data.frame`(`*tmp*`, counter, c(6, 7), value = list(maxPD = 0, :
## replacement element 2 has 2 rows to replace 1 rows
```

```
## Warning in evalTree(tree = initialTree, distribution = dist4taxa): NAs
## introduced by coercion

## Warning in evalTree(tree = initialTree, distribution = dist4taxa): NAs
## introduced by coercion

## Warning in evalTree(tree = initialTree, distribution = dist4taxa): NAs
## introduced by coercion
```

```
## todo: errors to correct evalTree

finalResults
```

```
##   labelTerminal InitialArea initialLength lowerFinalArea lowerBranchLength
## 1            t1      0.9999            NA             A1                 0
## 2            t2      0.9999            NA             A1                 0
## 3            t3      0.9999            NA             A1                 0
## 4            t4      0.9999            NA             A1                 0
##   changeLower deltaLower upperFinalArea upperBranchLength changeUpper
## 1          A1         NA             A1                 0          A1
## 2          A1         NA             A1                 0          A1
## 3          A1         NA             A1                 0          A1
## 4          A1         NA             A1                 0          A1
##   deltaUpper deltaPD    areaDelta abDelta
## 1         NA       0 L:_A1_/U:_A1       0
## 2         NA       0 L:_A1_/U:_A1       0
## 3         NA       0 L:_A1_/U:_A1       0
## 4         NA       0 L:_A1_/U:_A1       0
```

The extreme sensitivity of the PD results to the terminal branch length is seen in the column absolute length difference (=abDelta), as any length change -larger than 0-, will modify the area selected.

We plot the results to see the effect in each terminal, as a table.

```
plotResults <- ggplot(data=finalResults, aes(x= labelTerminal, y= initialLength,
                       shape="Actual",
                       colour=InitialArea)) +
               geom_point(size= 7) +
               geom_point(aes(x= labelTerminal, y= lowerBranchLength,
                              colour=lowerFinalArea,
                              shape="Lower_limit"), size=7) +
               labs(title = "Branch length change, lower limits. Equal branches.",
                    colour = "Area selected",
                    shape = "Terminal branch length value",
                    y = "Terminal branch length",
                    x = "Terminal")

print(plotResults)
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```


or plotted as a simple table.

```
countFreqChanges <- table(finalResults$areaDelta)
```

```r
countFreqChanges <- as.data.frame(countFreqChanges, ncol=1)


colnames(countFreqChanges) <- c("Area change","Freq")


row.names(countFreqChanges) <- NULL


countFreqChanges
```

```
##    Area change Freq
## 1 L:_A1_/U:_A1    4
```

or plotted into the tree:

```r
theTitle <- paste("Initial area selected:",finalResults$InitialArea[1])

p0 <-    ggtree(initialTree, layout="slanted", ladderize=TRUE,
               color=c("red","blue","red","blue","black","black","black"),
                size=0.8 ) +
         theme(legend.position="right") +
         labs(title = theTitle)


p <- p0 %<+% finalResults + geom_tiplab(aes(color=areaDelta), size =6) +
          scale_colour_brewer("Area change", palette="Dark2")


print(p)
```

For terminals t1/t3, a change from 1 to 0.99 in branch length -the lower limit (=L)- will change the initial area selected (A1A2) to A2; or a change from 1 to 1.01 in branch length -the upper limit(=U)-, will change the area to A1.