

Testing a simple example for the inapplicability problem

Thomas Guillaume
t.guillerm@imperial.ac.uk

March 31, 2016

This vignette aims to test which algorithms will pick up the signal of a really simple matrix with a clear signal. We are going to test a simple 12 taxa matrix with a minimum parsimony of 18 steps and a “stairs” topology (i.e. (t1, (t2, (t3, etc.)))).

We are going to test with `tnt` inapplicable R package and `phangorn` R package.

1 Before starting

We are also going to need the `IterativeAlgo` “package” that contains some utilities functions.

```
if(!require(devtools)) install.packages("devtools")
install_github("TGuillerm/Parsimony_Inapplicable/IterativeAlgo")
library(IterativeAlgo)
```

These are the required packages in R.

```
if(!require(ape)) install.packages("ape")
if(!require(phangorn)) install.packages("phangorn")

## Loading required package: phangorn

if(!require(phyclust)) install.packages("phyclust")

## Loading required package: phyclust

if(!require(versitree)) install.packages("versitree")

## Loading required package: versitree
## Loading required package: methods

if(!require(inapplicable)) install.packages("inapplicable")

## Loading required package: inapplicable
## Loading required package: parallel
```

2 Loading the data

The data is a simple 18 step matrix that should result in a ladderised tree. The matrix is available at this link https://github.com/TGuillerm/Parsimony_Inapplicable/blob/master/Simple_example/18step_matrix.csv.

```
## Reading in the matrix
matrix <- as.matrix(read.csv("18step_matrix.csv", header = F, row.names=1))

## Reading in the expected tree
expected_tree <- read.tree(
  text = "(t1,(t2,(t3,(t4,(t5,(t6,(t7,(t8,(t9,(t10,(t11,t12))))))))))";")
```

3 Setting up the data to be used by inapplicable

A couple of transformations have to be made in order to feed the data to the inapplicable package:

```
## Creating the contrast matrix (function from IterativeAlgo)
contrast_matrix <- get.contrast.matrix(matrix)

## Error in eval(expr, envir, enclos): could not find function "get.contrast.matrix"

## Create the phyDat object
matrix_phyDat <- phyDat(matrix, type = 'USER', contrast = contrast_matrix)

## Error in phyDat.default(data, levels = levels, return.index = return.index, : object 'contrast_matrix' not found

## Optimise the data for the inapplicable package analysis
matrix_inap <- prepare.data(matrix_phyDat)

## Error in prepare.data(matrix_phyDat): object 'matrix_phyDat' not found

## Specify the names of the outgroup taxa
outgroup <- c('t1')

## Set random seed for generating the tree
set.seed(1)

## Generate a random tree
rand_tree <- rtree(nrow(matrix), tip.label = rownames(matrix), br = NULL)

## Generate a NJ tree (with 0 branch length)
nj_tree <- root(nj(dist.hamming(matrix_phyDat)), outgroup, resolve.root = TRUE)

## Error in dist.hamming(matrix_phyDat): object 'matrix_phyDat' not found

nj_tree$edge.length <- NULL

## Error in nj_tree$edge.length <- NULL: object 'nj_tree' not found

## Generate a list of trees
trees <- list(expected_tree, rand_tree, nj_tree)
class(trees) <- "multiPhylo"
```

4 Calculating parsimony

We can then calculate the parsimony score using the simple Fitch algorithm from phytools and the one from inapplicable using the expected tree, the random one and the NJ calculated one.

```

## Calculate the normal fitch parsimony score using the 3 trees
parsimony_fitch <- unlist(lapply(trees, parsimony, matrix_phyDat,
  method = "fitch"))

## Error in FUN(X[[i]], ...): object 'matrix_phyDat' not found

## Calculate the parsimony score using the inapplicable algorithm
parsimony_inapp <- unlist(lapply(trees, parsimony.inapp, matrix_inap))

## Error in fitch.inapp(tree, data): object 'matrix_inap' not found

## Displaying the results
matrix(c(parsimony_fitch, parsimony_inapp), 2, 3, byrow = TRUE,
  dimnames = list(c("Fitch parsimony", "Inapp parsimony"),
    c("Expected_tree", "Random_tree", "NJ_tree")))

## Error in matrix(c(parsimony_fitch, parsimony_inapp), 2, 3, byrow = TRUE, : object 'parsimony_fitch'
not found

```

We can also use the tree search algorithms from the inapplicable package. These functions do not seem to work properly (at least on the version of the inapplicable package I have, they crash R).

```

## Search for a better tree
better_tree <- tree.search(expected_tree, matrix_inap, outgroup)

## Try a sectorial search (Goloboff, 1999)
better_tree <- sectorial.search(rand_tree, matrix_inap)

## Try the parsimony ratchet (Nixon, 1999)
better_tree <- pratchet.inapp(expected_tree, matrix_inap, outgroup)

```