Generating discrete morphological matrices with R

Thomas Guillerme
t.guillerme@imperial.ac.uk

February 25, 2016

This is just an informal note on what I've been working on on the Inapplicable data project and is now properly running. Basically, prior to testing any algorithms dealing with inapplicable characters as we discussed last time, I spend some time developing some tools that you might find handy. The following functions are stored in a R "package" clumsily called `IterativeAlgo` (that ought to be changed in the near future). They allow to generate some simulated yet realistic discrete morphological matrices and introduce some inapplicable characters. These matrices just need a tree as an input and allow a certain number of parameters to be set by the user.

1 Before starting

The "package" is stored on my github page:

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

A couple of packages are needed for running the functions and should install automatically. If not, they can be installed from here

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

2 Quick go through

This is a quick example on how it works

```r
suppressWarnings(library(IterativeAlgo))

## Setting the starting tree with 15 taxa as a random coalescent tree
my_tree <- rcoal(15)

## Generating a matrix with 100 characters (85% binary and 15% three states) and
## an equal rates model with a gamma rates distribution (0.5, 1) with no
## invariant characters.
my_matrix <- make.matrix(tree = my_tree, characters = 100, states = c(0.85,
    0.15), rates = c(rgamma, 0.5, 1), invariant = FALSE)
```

```
## Checking the matrix properties with a quick Maximum Parsimony tree search
check.matrix(my_matrix, my_tree)

## Most parsimonious tree search:
## Final p-score 172 after  0 nni operations
##
## Maximum parsimony        172.0000000
## Consistency index          0.6395349
## Retention index            0.7987013
## Robinson-Foulds distance   4.0000000

## Adding 10 inapplicable characters (5 from characters and 5 from phylogeny)
matrix_inap <- apply.inapplicables(my_matrix, inapplicables = c(rep("character",
    5), rep("clade", 5)), my_tree, invariant = FALSE)

## 3 characters are now invariant due inapplicable data.
```

It is then possible to save this matrix in classic nexus format:

```
write.nexus.data(matrix_inap, file = "test.nex")
```

3 Details

Details can be found for the three functions in their manuals:

```
?make.matrix
?check.matrix
?apply.inapplicables
```

Basically, `make.matrix` is really flexible and intakes a lot of different arguments to allow to simulate realistic matrix. It has two implemented models: `"ER"` for Equal Rates (M*k* model) or the `"HKY"` one which is the molecular HKY model but transforms pyrines in 0's and purimidines in 1's. It can also intake some specific distribution for the rates or the substitutions, allowing to modify these basic models.

The `check.matrix` runs a quick MP tree using the `phangorn` parsimony algorithm. It quickly calculates the parsimony score, the consistency and retention indices and, if a tree is provided (e.g. the tree used to generate the matrix) it calculates the Robinson-Foulds distance between the MP tree and the provided tree.

Finally, the interesting bit, the `apply.inapplicables` function transforms some of the characters states in the matrix into inapplicable tokens. There are two way the algorithm replaces the states by a inapplicable token:

1. By selecting a "pattern character" in the matrix and linking it to a "target character" next to it. The taxa with a randomly chosen state in the "pattern character" (e.g. 0) will then have an inapplicable character token for the "target character". This is done by using the `"character"` argument and is meant to simulate data inapplicability that originates from the coder (i.e. I first coded the character "tail" - the "pattern character" - and then I coded "tail color" - the "target character").

2. By selecting a random clade or it's associated grade (i.e. everything that's not in the clade) and attributing inapplicable tokens for a given character to all the taxa present in that clade/grade. This is done by using the `"clade"` argument and is meant to simulate data inapplicability that originates from the phylogeny (i.e. the deer's antlers that is present only in that clade).

I was thinking it could be interesting to play with these two different character inapplicability "origin" and see if that influences anything.

4 Good parameters for a good matrix

I was testing the effect of different parameters on the ability to generate a “realistic” matrix (meaning not to different from the input tree with a consistency and retention index close to what’s in the literature). The matrix seems the most “realistic” with a starting coalescent tree, equal rates model with 0.85 binary characters and 0.15 three states characters (by the way, these proportion are extracted from observed data my Total Evidence paper [1]), a gamma distribution with a shape parameter (α) of 0.5 and no scaling (β = 1).

Something like that should give a decent matrix:

```r
## tree
my_tree <- rcoal(15)
## matrix
morpho_mat <- make.matrix(my_tree, characters = 100, model = "ER",
    rates = c(rgamma, 0.5, 1), invariant = TRUE)
##
check.matrix(morpho_mat, my_tree)

## Most parsimonious tree search:
## Final p-score 90 after  1 nni operations
##
## Maximum parsimony        90.0000000
## Consistency index         0.6000000
## Retention index           0.7352941
## Robinson-Foulds distance 10.0000000
```

4.1 Testing the effect of the input tree

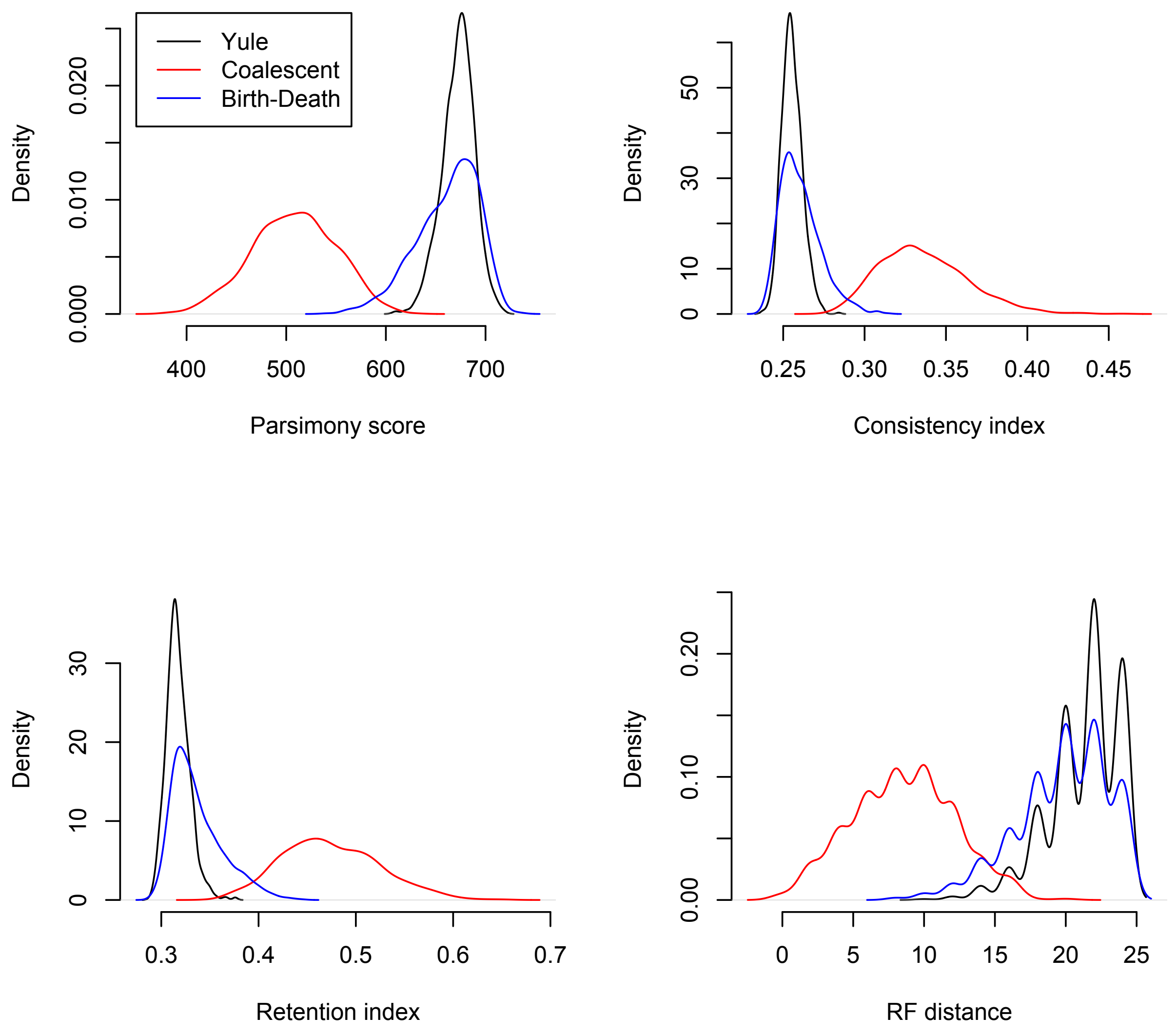

Figure 1: Effect of a yule, coalescent or birth-death (with random λ and μ) starting tree. The other parameters are fixed to be: `rates = c(rgamma, 0.5, 1)` and `model = "ER"`. Simulations where repeated 1000 times per tree type.

4.2 Testing the effect of the model

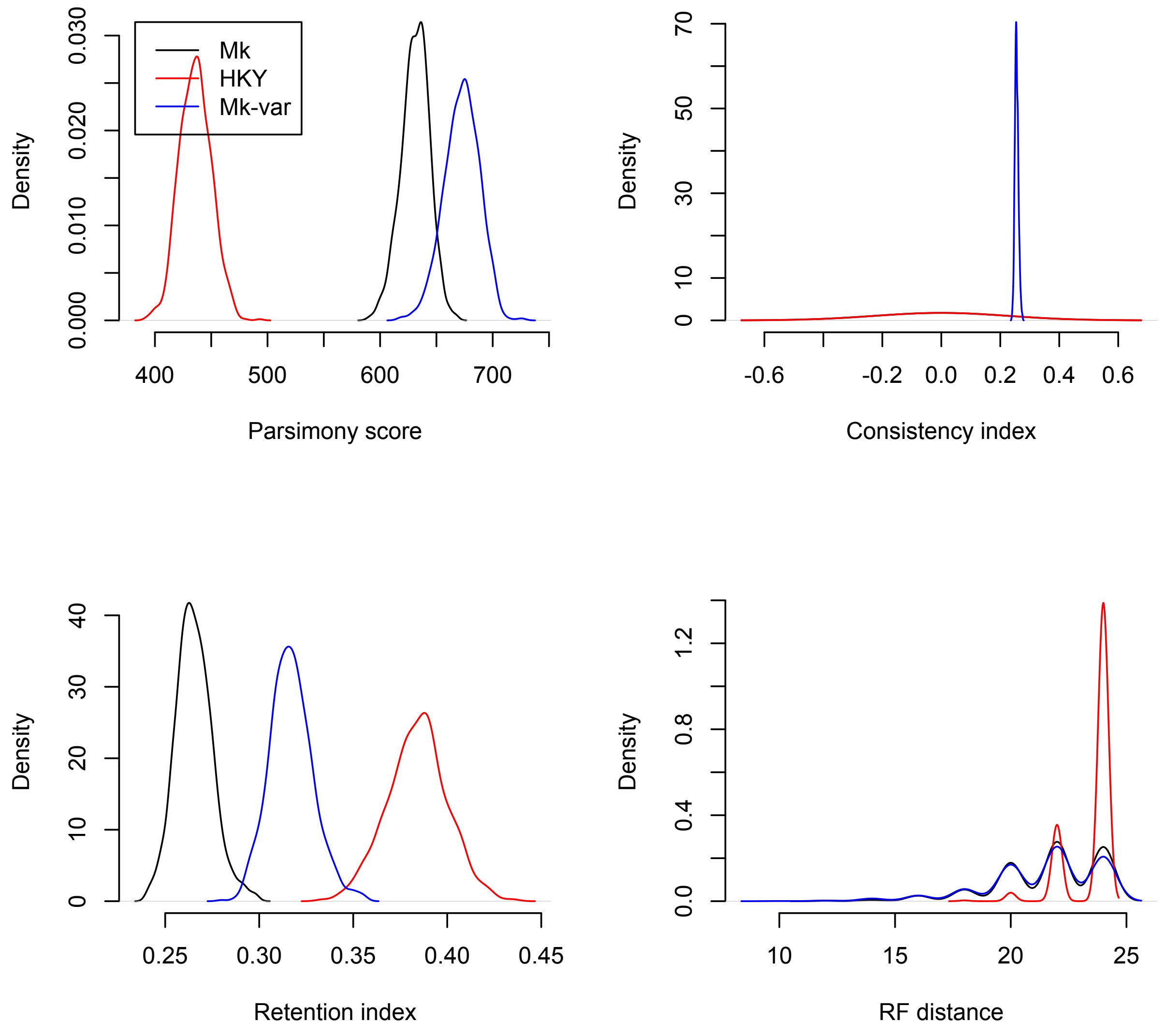

Figure 2: Effect of a Equal Rates (Mk), HKY (binary) or Mk with multiple states models. The other parameters are fixed to be: `rates = c(rgamma, 0.5, 1)` and `tree = coalescent`. Simulations where repeated 1000 times per model type.

4.3 Testing the effect of the rates distribution

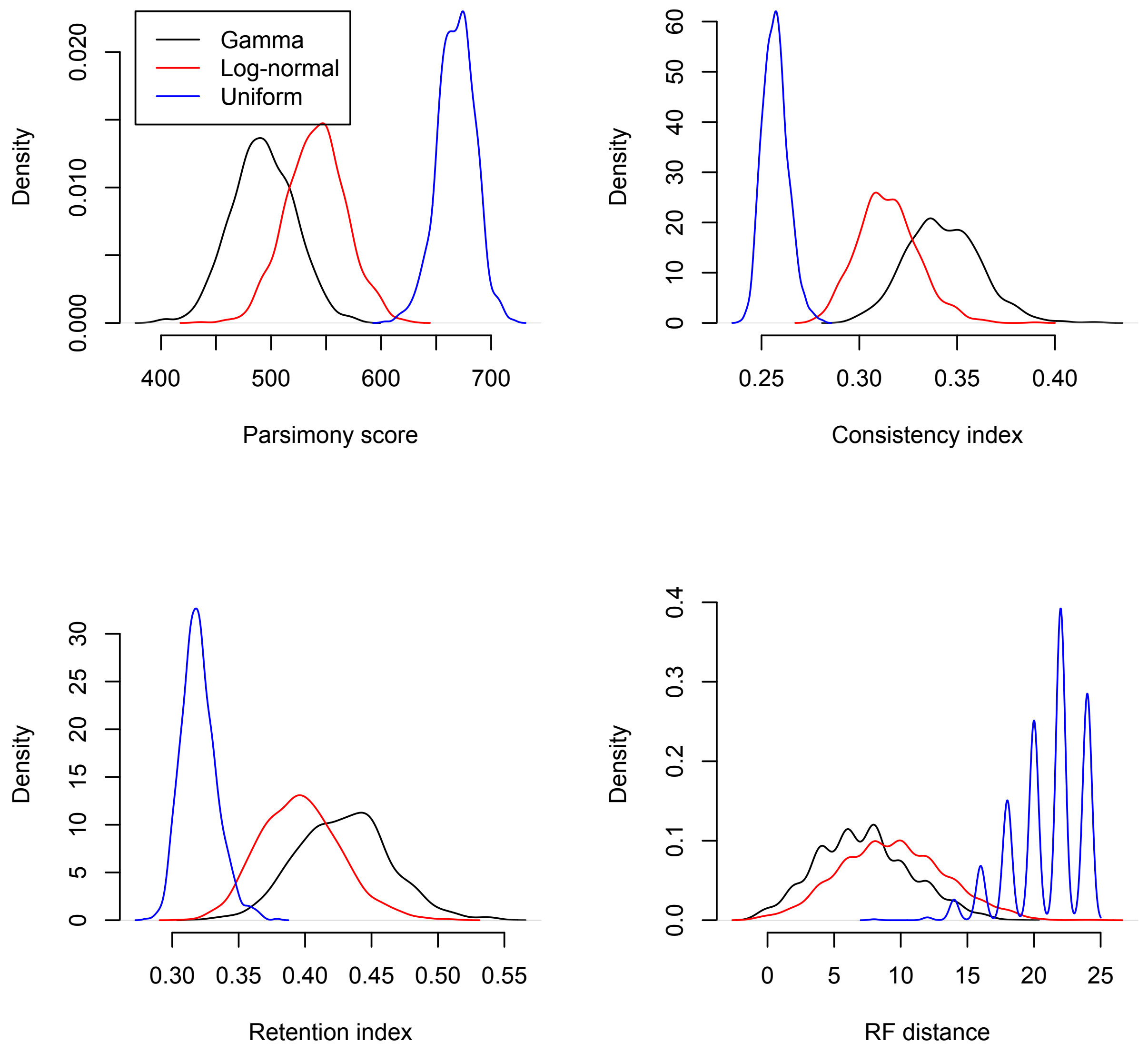

Figure 3: Effect of a Gamma, Log-normal or Uniform rates distribution. The other parameters are fixed to be: `model = "ER"` and `tree = coalescent`. Simulations where repeated 1000 times per distribution type.

4.4 Testing the effect of the rates distribution shape

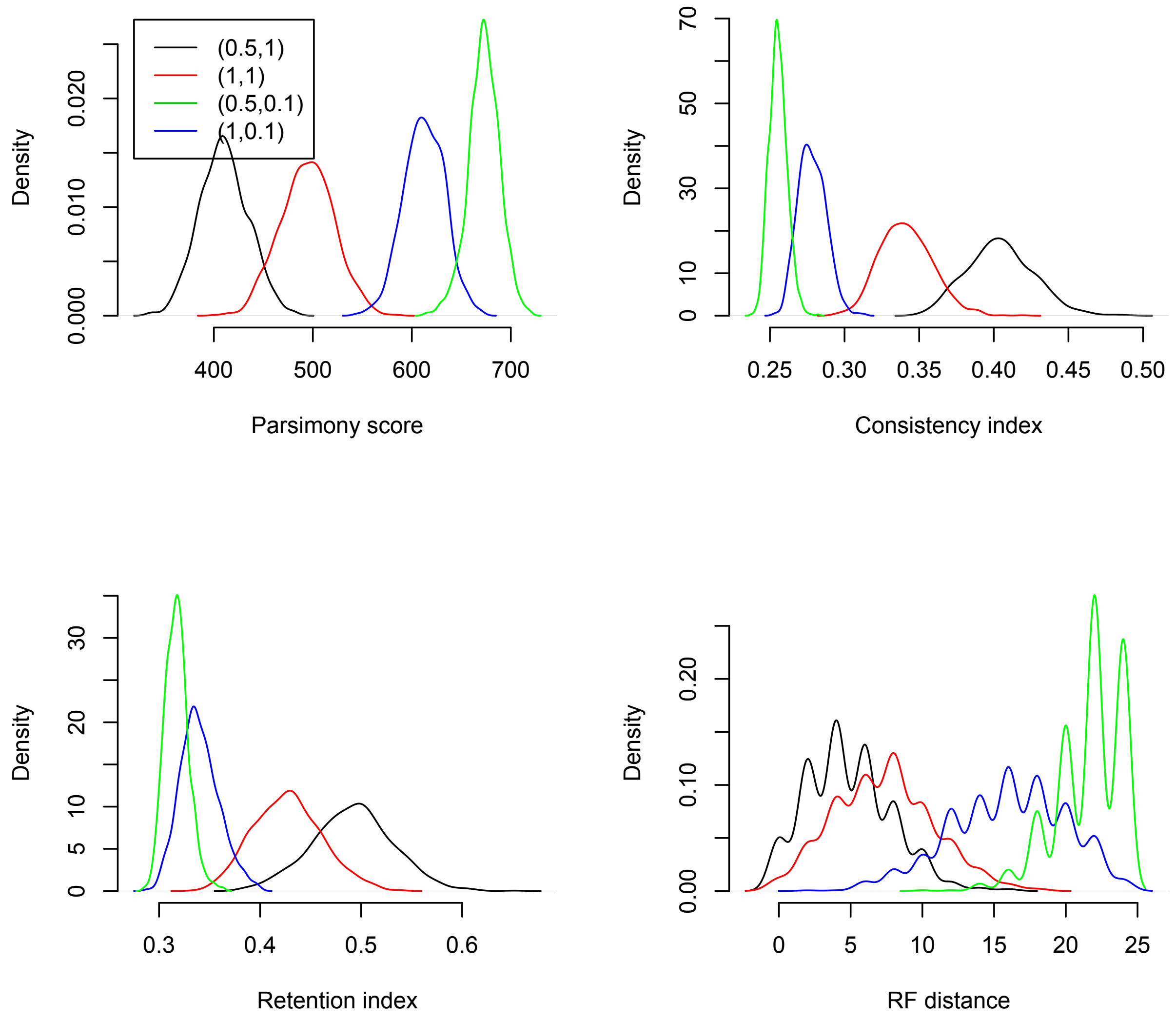

Figure 4: Effect of a Gamma distribution shape of α=0.5 and β=1; α=1 and β=1; α=0.5 and β=0.1; and α=1 and β=0.1. The other parameters are fixed to be: `model = "ER"` and `tree = coalescent`. Simulations where repeated 1000 times per shape type.

References

[1] Guillerme T, Cooper N. Effects of missing data on topological inference using a Total Evidence approach. Mol Phylogenet Evol. 2016;94, Part A:146 – 158. Available from: `http://www.sciencedirect.com/science/article/pii/S1055790315002547`.