Growing phylogenetic trees with TreeLine

Erik S. Wright

April 26, 2022

Contents

1 Introduction **1**

2 Performance Considerations **1**

3 Growing a Phylogenetic Tree **2**

4 Plotting Branch Support Values **5**

5 Ancestral State Reconstruction **11**

6 Session Information **13**

1 Introduction

This document describes how to grow phylogenetic trees using the `TreeLine` function in the DECIPHER package. `TreeLine` takes as input a set of aligned nucleotide or amino acid sequences and returns a phylogenetic tree (i.e., *dendrogram* object) as output. This vignette focuses on building maximum likelihood (ML) and maximum parsimony (MP) phylogenetic trees starting from sequences, but `TreeLine` can also be used to build additive trees from a distance matrix.

Why is the function called `TreeLine`? The goal of `TreeLine` is to find the most likely/parsimonious tree for a given sequence alignment. There are often many trees with nearly maximal likelihood/parsimony. Therefore, `TreeLine` seeks to find a tree as close as possible to the treeline, analogous to how no trees can grow above the treeline on a mountain.

Why use `TreeLine` versus other programs? The `TreeLine` function is designed to return an excellent phylogenetic tree with minimal user intervention. Many tree building programs have a large set of complex options for niche applications. In contrast, `TreeLine` simply builds a great tree when relying on its defaults. This vignette is intended to get you started and introduce additional options/functions that might be useful.

2 Performance Considerations

Finding a tree with very high likelihood/parsimony is no easy feat. `TreeLine` systematically optimizes hundreds to thousands of candidate trees before returning the best one. This takes time, but there are things you can do to make it go faster.

- Only use the sequences you need: `TreeLine` scales a bit worse than quadratically with the number of sequences. Hence, limiting the number of sequences is a worthwhile consideration. In particular, always eliminate redundant sequences, as shown below, and remove any sequences that are not necessary. This concern is shared for all tree building programs, and `TreeLine` is no exception.

- Set a timeout: The `maxTime` argument specifies the (approximate) maximum number of hours you are willing to let `TreeLine` run. If you are concerned about the code running too long then simply specify this argument.
- Compile with OpenMP support: Significant speed-ups can be achieved with multi-threading using OpenMP. See the "Getting Started DECIPHERing" vignette for how to do this on your computer platform. Then you only need to set the argument `processors=NULL` and `TreeLine` will use all available processors.
- Compile for SIMD support: `TreeLine` is configured to make use of SIMD operations, which are available on some processors. The easiest way to enable SIMD is to add " -O3 -march=native" to the end of `PKG_-CFLAGS` in the "DECIPHER/src/MAKEVARS" text file. This enables level-3 compiler optimization for your native computer architecture. Then, after recompiling, there can be an automatic speed-up on systems with SIMD support.

3 Growing a Phylogenetic Tree

`TreeLine` takes as input a multiple sequence alignment when constructing a maximum likelihood or maximum parsimony phylogenetic tree. Multiple sequence alignments can be constructed from a set of (unaligned) sequences using `AlignSeqs` or related functions. `TreeLine` will optimize trees for amino acid (i.e., `AAStringSet`) or nucleotide (i.e., `DNAStringSet` or `RNAStringSet`) sequences. Here, we are going to use a set of sequences that is included with DECIPHER. These sequences are from the internal transcribed spacer (ITS) between the 16S and 23S ribosomal RNA genes in several *Streptomyces* species.

```
> library(DECIPHER)
> # specify the path to your sequence file:
> fas <- "<<path to FASTA file>>"
> # OR find the example sequence file used in this tutorial:
> fas <- system.file("extdata", "Streptomyces_ITS_aligned.fas", package="DECIPHER")
> seqs <- readDNAStringSet(fas) # use readAAStringSet for amino acid sequences
> seqs # the aligned sequences
DNAStringSet object of length 88:
     width seq                                               names
 [1]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC supercont3.1 of S...
 [2]   627 NNNNCACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC supercont3.1 of S...
 [3]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC supercont1.1 of S...
 [4]   627 CGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC supercont1.1 of S...
 [5]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC supercont1.1 of S...
 ...   ... ...
[84]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC gi|297189896|ref|...
[85]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC gi|224581106|ref|...
[86]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC gi|224581106|ref|...
[87]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC gi|224581106|ref|...
[88]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC gi|224581108|ref|...
```

Many of these sequences are redundant or from the same genome. We can de-replicate the sequences to accelerate tree building:

```
> seqs <- unique(seqs) # remove duplicated sequences
> ns <- gsub("^.*Streptomyces( subsp\\. | sp\\. | | sp_)([^ ]+).*$", "\\2", names(seqs))
> names(seqs) <- ns # name by species
> seqs <- seqs[!duplicated(ns)] # remove redundant sequences from the same species
> seqs
```

```
DNAStringSet object of length 19:
     width seq                                               names
 [1]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC albus
 [2]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC clavuligerus
 [3]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC ghanaensis
 [4]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC griseoflavus
 [5]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC lividans
 ...   ... ...
[15]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC cattleya
[16]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC bingchenggensis
[17]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTTTCCGAATGGGGAAACC avermitilis
[18]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC C
[19]   627 TGTACACACCGCCCGTCA-CGTC...GGGGTGTCCGAATGGGGAAACC Tu6071
```

Now, it's time to find the most likely tree. Here, we will set a strict time limit to make this example faster, although longer time limits (e.g., 24 hours) are advised.

Note that `TreeLine` automatically selects a substitution model based on Akaike information criterion (by default). It is possible to specify specific model(s) (e.g., `model="GTR+G4"`) to limit the possible selections.

Also, since `TreeLine` is a stochastic optimizer, it is critical to always set the random number seed for reproducibility.

```
> set.seed(123) # set the random number seed
> tree <- TreeLine(seqs, reconstruct=TRUE, maxTime=0.05) # default is method="ML"
Optimizing model parameters:
JC69     -ln(L) = 5039, AICc = 10152, BIC = 10300
JC69+G4  -ln(L) = 4723, AICc = 9523, BIC = 9675
K80      -ln(L) = 4983, AICc = 10042, BIC = 10194
K80+G4   -ln(L) = 4413, AICc = 8905, BIC = 9061
F81      -ln(L) = 5020, AICc = 10121, BIC = 10281
F81+G4   -ln(L) = 4464, AICc = 9012, BIC = 9176
HKY85    -ln(L) = 4953, AICc = 9989, BIC = 10154
HKY85+G4 -ln(L) = 4371, AICc = 8829, BIC = 8997
T92      -ln(L) = 4965, AICc = 10010, BIC = 10166
T92+G4   -ln(L) = 4394, AICc = 8869, BIC = 9029
TN93     -ln(L) = 4942, AICc = 9969, BIC = 10138
TN93+G4  -ln(L) = 4371, AICc = 8831, BIC = 9004
SYM      -ln(L) = 4954, AICc = 9995, BIC = 10163
SYM+G4   -ln(L) = 4406, AICc = 8900, BIC = 9072
GTR      -ln(L) = 4929, AICc = 9951, BIC = 10132
GTR+G4   -ln(L) = 4368, AICc = 8832, BIC = 9017

The selected model was:  HKY85+G4

PHASE 1 OF 3: INITIAL TREES

1/3. Optimizing initial tree #1 of 10 to 100:
-ln(L) = 4369.3 (-0.048%), 1 Climb
1/3. Optimizing initial tree #2 of 10 to 100:
-ln(L) = 4392.0 (+0.518%), 2 Climbs
1/3. Optimizing initial tree #3 of 11 to 100:
-ln(L) = 4399.1 (+0.678%), 2 Climbs
1/3. Optimizing initial tree #4 of 12 to 100:
-ln(L) = 4394.3 (+0.570%), 1 Climb
1/3. Optimizing initial tree #5 of 13 to 100:
-ln(L) = 4394.3 (+0.570%), 1 Climb
1/3. Optimizing initial tree #6 of 14 to 100:
-ln(L) = 4384.4 (+0.345%), 2 Climbs
1/3. Optimizing initial tree #7 of 15 to 100:
-ln(L) = 4382.8 (+0.310%), 2 Climbs
1/3. Optimizing initial tree #8 of 16 to 100:
-ln(L) = 4377.6 (+0.190%), 3 Climbs

PHASE 2 OF 3: REGROW GENERATION 1 OF 10 TO 20

2/3. Optimizing regrown tree #1 of 10 to 100:
-ln(L) = 4369.2 (~0.000%), 1 Climb
2/3. Optimizing regrown tree #2 of 10 to 100:
-ln(L) = 4369.4 (+0.004%), 1 Climb
2/3. Optimizing regrown tree #3 of 10 to 100:
-ln(L) = 4369.2 (~0.000%), 1 Climb
2/3. Optimizing regrown tree #4 of 10 to 100:
-ln(L) = 4369.2 (0.000%), 0 Climbs
2/3. Optimizing regrown tree #5 of 10 to 100:
-ln(L) = 4369.2 (~0.000%), 0 Climbs
2/3. Optimizing regrown tree #6 of 10 to 100:
-ln(L) = 4369.2 (~0.000%), 1 Climb
2/3. Optimizing regrown tree #7 of 10 to 100:
-ln(L) = 4374.9 (+0.129%), 1 Climb
2/3. Optimizing regrown tree #8 of 10 to 100:
```

4 Plotting Branch Support Values

`TreeLine` automatically returns a variety of information about the tree that can be accessed with the `attributes` and `attr` functions:

```
> attributes(tree) # view all attributes
$members
[1] 19

$height
[1] 2.315969

$state
[1] "------CACCGCCCGTCA-CGTCACGAAAGTCGGTAACACCCGAAGCCGGTGGCCCAACCCCCCG-GGGAGGGAGCCGTCGAA

$class
[1] "dendrogram"

$siteLnLs
  [1]  -1.966395  -1.576672  -1.966395  -2.306336  -2.009135  -2.306336
  [7]  -2.158784  -2.460634  -2.158784  -2.158784  -1.712060  -2.158784
 [13]  -2.158784  -2.158784  -1.712060  -2.114982  -2.158784  -2.460634
 [19]   0.000000  -2.158784  -1.712060  -2.114982  -2.158784  -2.460634
 [25]  -4.647338  -1.712060  -2.460634  -2.460634  -2.460634  -1.712060
 [31]  -2.114982  -2.158784  -1.712060  -1.712060  -2.114982  -2.460634
 [37]  -5.912525  -4.647338  -5.912525  -2.158784  -2.158784  -2.158784
 [43]  -1.712060  -2.460634  -2.460634  -1.712060  -2.158784  -2.158784
 [49]  -4.964390  -4.428485  -2.114982  -1.712060  -1.712060  -2.158784
 [55]  -2.158784  -2.158784  -2.460634  -2.460634  -2.158784  -2.158784
 [61]  -2.158784 -12.729135  -7.350693 -16.136854 -13.934106  -6.828311
 [67]  -1.712060  -1.712060  -1.712060  -2.460634  -1.712060  -1.712060
 [73]  -1.712060  -2.460634  -1.712060  -8.757548 -11.941091  -1.712060
 [79]  -2.114982  -2.158784  -1.712060  -2.460634  -2.460634  -1.712060
 [85]  -1.712060  -2.114982  -1.712060  -1.712060  -1.712060  -2.460634
 [91]  -2.158784 -13.149243 -12.292466  -1.712060  -2.158784  -1.712060
 [97]  -2.460634  -2.114982  -2.114982  -1.712060  -1.712060  -1.712060
[103]  -2.460634  -2.158784  -1.712060  -2.460634  -2.460634  -1.712060
[109]  -2.114982  -2.158784  -1.712060  -2.114982  -2.460634  -2.460634
[115]  -2.158784  -2.460634  -2.460634  -1.712060  -1.712060  -2.114982
[121]  -2.460634  -1.712060  -2.158784  -2.158784  -1.712060  -2.114982
[127]  -2.460634  -2.158784  -2.158784  -1.712060  -1.712060  -2.460634
[133]  -2.460634  -1.712060  -1.712060  -2.114982  -1.712060  -2.158784
[139]  -1.712060  -1.712060  -2.158784  -2.114982  -1.712060  -1.712060
[145]  -2.460634  -2.114982  -2.158784  -2.460634  -2.158784  -2.158784
[151]  -2.114982  -2.158784  -2.158784  -2.114982  -2.114982  -2.114982
[157]  -2.158784  -2.114982  -2.460634  -2.460634  -1.712060  -1.712060
[163]  -2.460634  -1.712060  -2.158784  -2.460634 -15.503593 -19.546642
[169]  -5.542077 -11.996169 -12.246678 -13.664888 -17.124213 -13.376266
[175] -11.491655 -17.678210  -7.686737  -9.151623  -8.266815 -15.957651
[181] -22.610994 -19.624503  -7.829939 -12.310460 -11.135237 -11.004569
[187]  -8.839236 -12.921398  -9.802642 -11.419942 -14.309475 -11.177669
```

```
[193]  -7.174461  -7.327338  -5.962680 -10.158416  -9.296318  -8.764207
[199] -13.922975 -10.932027 -13.818976 -12.031583  -8.596898  -2.141517
[205]  -7.788024  -6.620083 -16.842530 -15.837964 -16.882686  -8.708737
[211] -16.134367 -10.164664 -14.446796 -14.402133  -6.596478  -2.158784
[217] -15.278404  -9.786629 -13.410018 -16.697923  -1.712060 -15.454284
[223] -17.364415 -15.828215 -16.593556 -16.434202 -18.254765 -12.472068
[229] -15.704205  -9.618319  -1.712060  -4.926292  -2.114982 -15.835139
[235]  -2.376085  -4.428485  -2.158784  -5.715703  -2.158784  -5.950623
[241] -11.610372  -1.712060  -4.964390  -4.428485  -4.631176  -1.712060
[247]  -4.964390  -4.717754  -4.717754  -4.647338  -1.712060  -2.114982
[253]  -2.114982  -1.712060  -5.950623 -17.268953  -5.069128  -4.717754
[259] -15.722814  -4.631176  -7.650236  -4.926292  -1.712060  -8.004095
[265] -10.363162 -15.715406 -19.083146 -19.383151 -13.165703 -21.670258
[271] -20.436455 -22.079010 -24.783914 -22.740155 -20.636688 -19.028944
[277] -22.682400 -25.458166 -23.382707 -26.241506 -21.418422 -24.207957
[283] -18.053977 -11.068876 -18.320749 -15.099123 -19.315979 -21.714881
[289]  -4.717754  -1.712060  -2.114982  -5.912525  -5.772577  -2.114982
[295]  -7.066854  -5.126002 -11.504587 -14.174805 -13.062412  -6.037830
[301]  -5.271972  -7.191604 -10.716736 -10.310486  -8.289520  -1.712060
[307] -13.747493  -7.308861  -9.678717  -1.712060 -10.766576  -9.171040
[313]  -8.949974 -25.456679 -17.415987 -16.242899  -1.456754  -1.418656
[319]  -1.418656  -7.819282 -21.315476 -23.920345 -22.381772 -22.450038
[325] -26.783112 -25.165857 -23.717952 -21.797613 -22.140251 -19.650187
[331] -13.208547  -3.790668 -18.929470 -23.887632 -23.781823 -18.431691
[337] -21.953241 -23.678930 -17.578357 -12.754442 -17.143819  -8.677828
[343] -17.352150  -5.126002  -9.001562  -1.712060  -1.712060  -5.772577
[349]  -8.367175  -2.158784  -7.635347 -12.635747  -4.631176  -1.712060
[355]  -5.715703  -5.069128  -1.712060  -1.456754  -1.712060  -1.712060
[361]  -5.715703 -13.540086 -12.229598  -2.158784  -2.114982  -1.712060
[367]  -4.717754 -10.634402  -1.712060  -1.712060 -23.840365 -12.358302
[373]  -7.511222  -1.712060 -11.161984 -17.217757  -4.533012 -16.888541
[379] -17.633103  -4.540369  -1.629417  -1.418656 -21.572242 -21.074674
[385] -23.621807  -8.582299 -20.413856 -12.603307 -23.179921 -12.984714
[391] -20.154821 -16.420261 -15.333493 -17.319907  -9.823467 -17.317278
[397] -24.420729 -16.367890  -1.887996  -1.456754  -1.418656  -1.418656
[403] -10.012175 -19.398981 -12.200049  -7.966801  -4.647338  -5.126002
[409]  -1.712060 -11.497992  -9.516886  -2.158784  -6.612800  -4.647338
[415] -17.817445  -8.249406  -7.563351 -17.190541 -32.638039  -2.460634
[421]   0.000000  -9.471173  -9.319333 -14.163273 -21.159693 -28.998350
[427] -21.251267 -19.629095 -21.760998 -17.664660 -17.659885 -17.668679
[433] -20.686803 -30.392306 -14.854243 -24.911298 -23.542422 -23.911419
[439] -22.532798 -14.063930 -19.655288 -15.827245  -1.712060  -7.426664
[445]  -8.544201 -15.941852 -18.855383  -4.926292  -1.712060  -1.712060
[451]  -7.426664  -1.712060  -5.622659 -15.599710  -6.685570  -1.576672
[457]  -1.576672  -1.966395  -4.647338  -1.712060  -5.069128  -2.114982
[463]  -1.712060 -13.043143  -2.114982  -2.114982  -1.712060  -2.460634
[469]  -1.712060  -2.460634  -2.460634  -2.158784  -9.656956 -10.697320
[475]  -4.647338  -2.460634 -12.045986  -2.460634  -1.712060  -2.114982
[481]  -1.712060  -1.712060  -2.460634  -4.647338  -1.712060  -2.158784
[487]  -8.731510  -2.460634  -1.712060  -2.158784  -2.460634  -2.114982
[493]  -2.158784  -2.114982  -1.418656  -1.418656  -1.712060  -2.114982
```

```
[499]  -1.712060  -1.712060  -4.647338  -2.158784  -2.460634  -2.460634
[505]  -1.712060  -2.114982  -2.114982  -5.069128  -2.114982  -2.114982
[511]  -2.460634  -2.460634  -1.712060  -1.712060  -1.712060  -2.158784
[517]  -4.428485  -2.158784  -2.460634  -4.647338  -1.712060  -1.712060
[523]  -2.114982  -1.712060  -1.712060  -2.460634  -2.114982  -1.712060
[529]  -4.647338  -2.158784  -2.114982  -2.114982  -1.712060  -1.712060
[535] -11.091759  -5.912525  -4.647338  -2.158784  -2.460634  -1.712060
[541]  -1.712060  -2.460634  -4.717754  -2.158784  -2.158784  -1.712060
[547]  -2.460634  -2.114982  -1.712060  -2.460634  -2.460634  -1.712060
[553]  -1.712060  -2.460634  -2.158784  -1.712060  -2.114982  -1.712060
[559]  -6.596478  -1.712060  -2.460634  -1.712060  -1.712060  -2.158784
[565]  -4.647338 -10.428334  -2.158784  -1.712060  -2.460634  -2.114982
[571]  -2.460634  -4.926292 -17.981392  -2.158784  -2.158784  -4.647338
[577]  -2.158784  -1.712060  -1.712060  -1.712060  -1.712060  -2.460634
[583]  -1.712060 -18.065739 -11.171538  -1.712060  -6.973804  -2.158784
[589]  -2.460634  -2.460634  -2.158784  -7.008025 -10.428334 -11.664001
[595]  -1.712060  -2.158784  -2.114982 -11.319198  -2.114982  -1.712060
[601]  -2.460634  -2.114982  -2.158784  -2.158784  -1.712060  -4.428485
[607]  -1.712060  -1.712060  -4.428485  -2.114982 -18.353138  -2.114982
[613]  -2.158784  -2.158784  -1.712060  -2.460634  -2.460634  -2.114982
[619]  -1.712060  -1.712060  -1.712060  -1.712060  -2.460634  -2.460634
[625]  -2.460634  -2.158784  -2.158784

$method
[1] "ML"

$model
[1] "HKY85+G4"

$parameters
    FreqA     FreqC     FreqG     FreqT     FreqI       A/G       C/T       A/C
0.1804644 0.2329914 0.3445050        NA        NA 3.7118172        NA        NA
      A/T       C/G    Indels     alpha
       NA        NA        NA 0.1891751

$score
[1] 4365.615

$midpoint
[1] 10.66797
> attr(tree, "score") # best score
[1] 4365.615
```

The tree is (virtually) rooted at its midpoint by default. For maximum likelihood trees, all internal nodes include aBayes branch support values [1]. These are given as probabilities that can be used in plotting on top of each edge. We can also italicize the species names.

```
> plot(dendrapply(tree,
        function(x) {
                s <- attr(x, "probability") # choose "probability" (aBayes) or "support
                if (!is.null(s) && !is.na(s)) {
                        s <- formatC(as.numeric(s), digits=2, format="f")
                        attr(x, "edgetext") <- paste(s, "\n")
                }
                attr(x, "edgePar") <- list(p.col=NA, p.lwd=1e-5, t.col="#CC55AA", t.cex=
                if (is.leaf(x))
                        attr(x, "nodePar") <- list(lab.font=3, pch=NA)
                x
        }),
        horiz=TRUE,
        yaxt='n')
> # add a scale bar
> arrows(0, 0, 0.4, 0, code=3, angle=90, len=0.05, xpd=TRUE)
> text(0.2, 0, "0.4 subs./site", pos=3, xpd=TRUE)
```

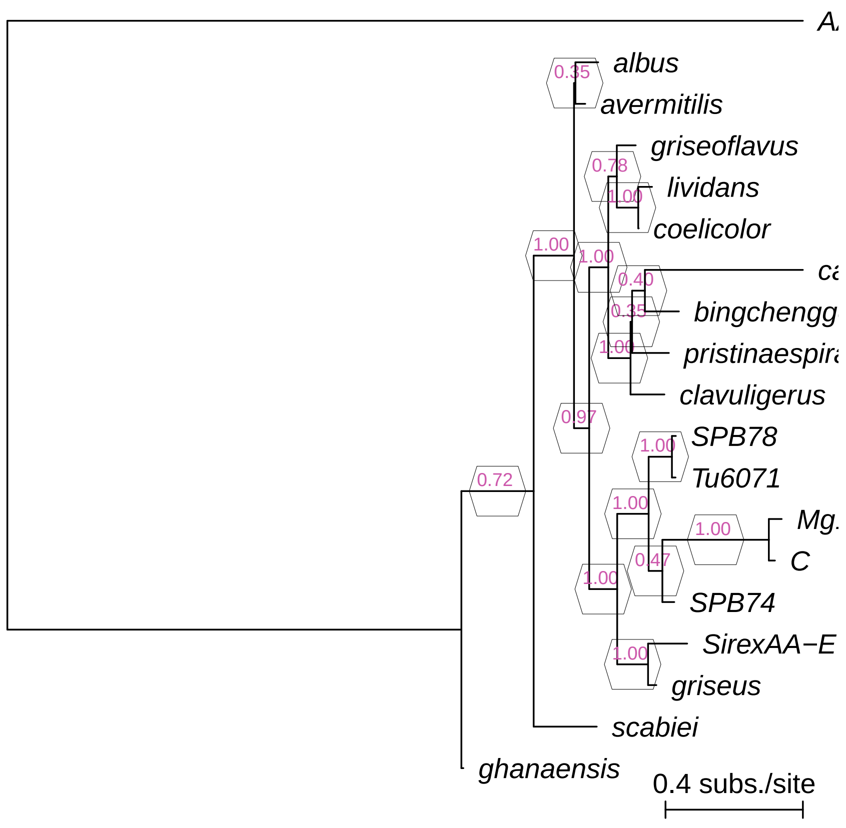

Figure 2: Tree with (aBayes) support probabilities at each internal node.

Maximum likelihood and maximum parsimony trees both provide branch supports in the form of the fraction of optimized trees that contained a given partition (branch). These are accessible from the “support” attribute. As expected, support values and (aBayes) probabilities are correlated, but support tends to be more conservative.

```
> getSupports <- function(x) {
          if (is.leaf(x)) {
                  NULL
          } else {
                  rbind(cbind(attr(x, "support"), attr(x, "probability")),
                          getSupports(x[[1]]), getSupports(x[[2]]))
          }
  }
> support <- getSupports(tree)
> plot(support[, 1], support[, 2], xlab="Support", ylab="aBayes probability", asp=1)
> abline(a=0, b=1, lty=2) # line of identity (y=x)
```

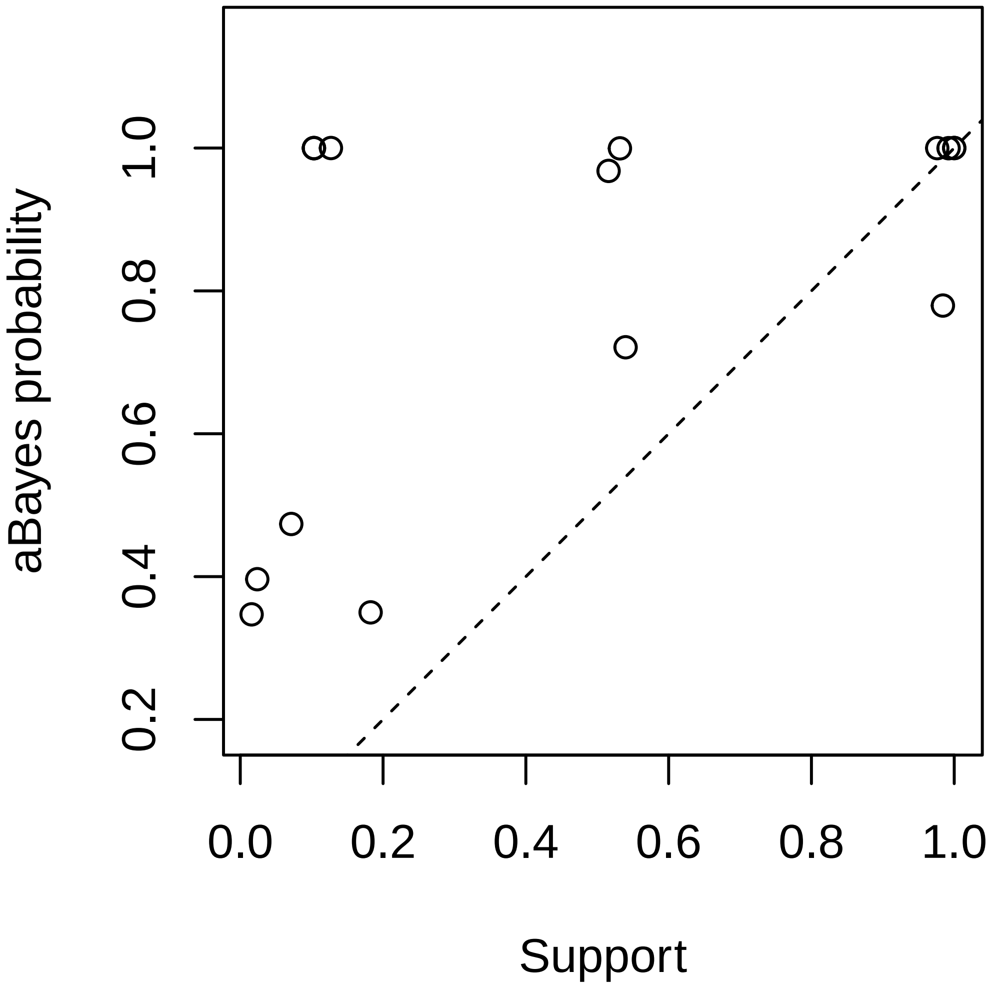

Figure 3: Comparison of aBayes probabilities and branch support values.

5 Ancestral State Reconstruction

One of the advantages of maximum likelihood and maximum parsimony tree building methods is that they automatically predict states at each internal node on the tree [2]. This feature is enabled when *reconstruct* is set to `TRUE`. These character states can be used by the function `MapCharacters` to determine state transitions along each edge of the tree.

```
> new_tree <- MapCharacters(tree, labelEdges=TRUE)
> plot(new_tree, edgePar=list(p.col=NA, p.lwd=1e-5, t.col="#55CC99", t.cex=0.7))
> attr(new_tree[[1]], "change") # state changes on first branch left of (virtual) root
 [1] "G65T"  "G168T" "G171T" "G172C" "G173A" "G180T" "G181C" "G182C" "G184T"
[10] "G185T" "G186A" "G201A" "G208T" "G209C" "G211A" "G220C" "G224C" "G227T"
[19] "G259C" "G271T" "G272C" "G276T" "G277C" "G280A" "G282A" "G287C" "G288C"
[28] "G302A" "G303A" "G314C" "G316C" "G321T" "G323A" "G324T" "G325C" "G326T"
[37] "G327T" "G328C" "G333C" "G337T" "G338T" "G339C" "G343C" "G371C" "G379C"
[46] "G385A" "G389C" "G393T" "G396T" "G397C" "G419C" "G435A" "G437C" "G440T"
[55] "G447C" "G584C"
```

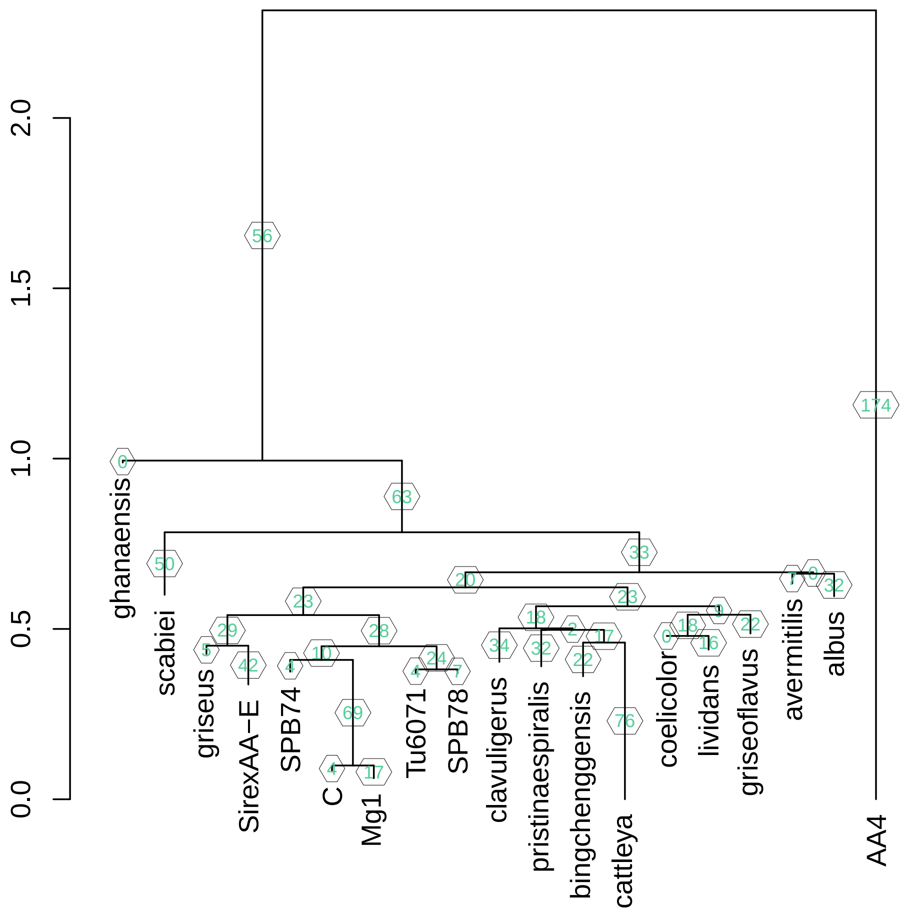

Figure 4: Edges labeled with the number of state transitions.

6 Session Information

All of the output in this vignette was produced under the following conditions:

- R version 4.2.0 RC (2022-04-19 r82224), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 20.04.4 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.15-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.15-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BiocGenerics 0.42.0, Biostrings 2.64.0, DECIPHER 2.24.0, GenomeInfoDb 1.32.0, IRanges 2.30.0, RSQLite 2.2.12, S4Vectors 0.34.0, XVector 0.36.0
- Loaded via a namespace (and not attached): DBI 1.1.2, GenomeInfoDbData 1.2.8, KernSmooth 2.23-20, RCurl 1.98-1.6, Rcpp 1.0.8.3, bit 4.0.4, bit64 4.0.5, bitops 1.0-7, blob 1.2.3, cachem 1.0.6, cli 3.3.0, compiler 4.2.0, crayon 1.5.1, fastmap 1.1.0, memoise 2.0.1, pkgconfig 2.0.3, rlang 1.0.2, tools 4.2.0, vctrs 0.4.1, zlibbioc 1.42.0

References

[1] Anisimova, M., Gil, M., Dufayard, J., Dessimoz, C., & Gascuel, O. Survey of branch support methods demonstrates accuracy, power, and robustness of fast likelihood-based approximation schemes. *Syst Biol.*, 60(5), 685-699.

[2] Joy, J., Liang, R., McCloskey, R., Nguyen, T., & Poon, A. Ancestral Reconstruction. *PLoS Comp. Biol.*, 12(7), e1004763.