Growing phylogenetic trees with TreeLine

Erik S. Wright

April 26, 2022

Contents

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) = 4721, AICc = 9520, BIC = 9672
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) = 4956, AICc = 9995, BIC = 10159
HKY85+G4 -ln(L) = 4371, AICc = 8829, BIC = 8997
T92      -ln(L) = 4963, AICc = 10005, BIC = 10161
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 (+0.571%), 1 Climb
1/3. Optimizing initial tree #5 of 13 to 100:
-ln(L) = 4394.3 (+0.571%), 1 Climb
1/3. Optimizing initial tree #6 of 14 to 100:
-ln(L) = 4384.4 (+0.346%), 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) = 4369.2 (~0.000%), 2 Climbs
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] 2.287712

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

$class
[1] "dendrogram"

$siteLnLs
  [1]  -1.952084  -1.545174  -1.952084  -2.267191  -1.970506  -2.267191
  [7]  -2.120389  -2.423394  -2.120389  -2.120389  -1.679343  -2.120389
 [13]  -2.120389  -2.120389  -1.679343  -2.101425  -2.120389  -2.423394
 [19]   0.000000  -2.120389  -1.679343  -2.101425  -2.120389  -2.423394
 [25]  -4.617713  -1.679343  -2.423394  -2.423394  -2.423394  -1.679343
 [31]  -2.101425  -2.120389  -1.679343  -1.679343  -2.101425  -2.423394
 [37]  -5.871690  -4.617713  -5.871690  -2.120389  -2.120389  -2.120389
 [43]  -1.679343  -2.423394  -2.423394  -1.679343  -2.120389  -2.120389
 [49]  -4.936127  -4.398675  -2.101425  -1.679343  -1.679343  -2.120389
 [55]  -2.120389  -2.120389  -2.423394  -2.423394  -2.120389  -2.120389
 [61]  -2.120389 -13.045011  -7.382174 -16.316933 -13.949193  -6.923982
 [67]  -1.679343  -1.679343  -1.679343  -2.423394  -1.679343  -1.679343
 [73]  -1.679343  -2.423394  -1.679343  -8.563764 -11.876671  -1.679343
 [79]  -2.101425  -2.120389  -1.679343  -2.423394  -2.423394  -1.679343
 [85]  -1.679343  -2.101425  -1.679343  -1.679343  -1.679343  -2.423394
 [91]  -2.120389 -13.832545 -13.212523  -1.679343  -2.120389  -1.679343
 [97]  -2.423394  -2.101425  -2.101425  -1.679343  -1.679343  -1.679343
[103]  -2.423394  -2.120389  -1.679343  -2.423394  -2.423394  -1.679343
[109]  -2.101425  -2.120389  -1.679343  -2.101425  -2.423394  -2.423394
[115]  -2.120389  -2.423394  -2.423394  -1.679343  -1.679343  -2.101425
[121]  -2.423394  -1.679343  -2.120389  -2.120389  -1.679343  -2.101425
[127]  -2.423394  -2.120389  -2.120389  -1.679343  -1.679343  -2.423394
[133]  -2.423394  -1.679343  -1.679343  -2.101425  -1.679343  -2.120389
[139]  -1.679343  -1.679343  -2.120389  -2.101425  -1.679343  -1.679343
[145]  -2.423394  -2.101425  -2.120389  -2.423394  -2.120389  -2.120389
[151]  -2.101425  -2.120389  -2.120389  -2.101425  -2.101425  -2.101425
[157]  -2.120389  -2.101425  -2.423394  -2.423394  -1.679343  -1.679343
[163]  -2.423394  -1.679343  -2.120389  -2.423394 -15.338209 -20.302830
[169]  -5.557969 -12.092339 -12.392385 -13.710906 -17.270008 -13.456621
[175] -11.575289 -17.723399  -7.785805  -9.154045  -8.266189 -15.780805
[181] -22.684919 -19.645341  -7.788416 -12.360150 -11.169477 -11.008020
[187]  -8.829030 -12.995958  -9.869057 -11.494430 -14.471847 -11.173395
```

```
[193]  -7.160322  -7.297620  -6.038625 -10.127272  -9.297403  -8.729793
[199] -14.086515 -11.010741 -13.921571 -12.121340  -8.937800  -2.100500
[205]  -8.044121  -6.826069 -17.558618 -15.427105 -16.874744 -11.415401
[211] -15.988565 -12.338557 -15.729624 -15.184410  -6.866717  -2.120389
[217] -15.331131  -9.782839 -13.502513 -16.282825  -1.679343 -12.723265
[223] -18.022047 -16.150323 -17.538411 -17.152568 -18.168835 -13.920033
[229] -15.658053  -9.175389  -1.679343  -4.919553  -2.101425 -15.986039
[235]  -2.344535  -4.398675  -2.120389  -5.696857  -2.120389  -5.888264
[241] -11.621819  -1.679343  -4.936127  -4.398675  -4.611239  -1.679343
[247]  -4.936127  -4.667833  -4.667833  -4.617713  -1.679343  -2.101425
[253]  -2.101425  -1.679343  -5.888264 -17.262850  -5.051495  -4.667833
[259] -15.584804  -4.611239  -7.944340  -4.919553  -1.679343  -8.059629
[265] -10.402169 -15.529683 -19.046143 -20.014294 -13.313761 -21.919133
[271] -20.259918 -22.087999 -24.716310 -22.868429 -20.682866 -18.969805
[277] -22.857468 -25.610857 -23.393284 -26.692622 -20.948455 -23.938655
[283] -18.124150 -11.182225 -18.436676 -15.218121 -19.233521 -21.690036
[289]  -4.667833  -1.679343  -2.101425  -5.871690  -5.720734  -2.101425
[295]  -7.121566  -5.075372 -11.409956 -13.937820 -13.044671  -6.083011
[301]  -5.337723  -7.230765 -10.743689 -10.355957  -8.276193  -1.679343
[307] -13.792246  -7.349470  -9.701720  -1.679343 -10.783739  -9.114447
[313]  -8.918157 -25.341629 -17.331316 -16.147070  -1.450730  -1.434157
[319]  -1.434157  -7.873812 -21.417440 -24.429270 -22.011263 -21.883046
[325] -27.043533 -25.093446 -23.636595 -22.573645 -22.424915 -19.962100
[331] -12.749846  -3.813802 -19.031495 -23.616671 -23.811064 -18.552676
[337] -21.758801 -24.041652 -17.036381 -12.595284 -17.171753  -8.717367
[343] -17.342486  -5.075372  -9.324507  -1.679343  -1.679343  -5.720734
[349]  -8.415392  -2.120389  -7.707056 -14.226447  -4.611239  -1.679343
[355]  -5.696857  -5.051495  -1.679343  -1.450730  -1.679343  -1.679343
[361]  -5.696857 -13.538381 -11.577329  -2.120389  -2.101425  -1.679343
[367]  -4.667833 -10.890608  -1.679343  -1.679343 -23.740548 -12.540781
[373]  -7.562801  -1.679343 -11.311775 -16.436244  -4.511835 -16.891125
[379] -18.406091  -4.576859  -1.638895  -1.434157 -21.891224 -21.742934
[385] -23.885631  -8.661633 -21.151584 -12.781990 -22.795722 -13.690141
[391] -21.178941 -16.510053 -15.232879 -17.203637 -11.192917 -17.208825
[397] -24.200420 -16.700721  -1.891511  -1.450730  -1.434157  -1.434157
[403]  -9.928808 -19.548117 -12.217233  -9.444072  -4.617713  -5.075372
[409]  -1.679343 -11.524240 -10.926228  -2.120389  -6.654560  -4.617713
[415] -17.729448  -8.291066  -7.642491 -16.991507 -33.181467  -2.423394
[421]   0.000000  -9.487815  -9.367824 -13.824717 -21.390014 -29.435969
[427] -20.571953 -20.147094 -21.603249 -17.373530 -17.633320 -17.583775
[433] -21.732673 -30.080095 -15.024780 -25.154053 -23.980659 -23.114939
[439] -22.631206 -14.689256 -19.514713 -15.738276  -1.679343  -7.489887
[445]  -8.645059 -16.206313 -18.540026  -4.919553  -1.679343  -1.679343
[451]  -7.489887  -1.679343  -5.672321 -15.413580  -6.752663  -1.545174
[457]  -1.545174  -1.952084  -4.617713  -1.679343  -5.051495  -2.101425
[463]  -1.679343 -12.518994  -2.101425  -2.101425  -1.679343  -2.423394
[469]  -1.679343  -2.423394  -2.423394  -2.120389  -9.431967 -11.000210
[475]  -4.617713  -2.423394 -12.071166  -2.423394  -1.679343  -2.101425
[481]  -1.679343  -1.679343  -2.423394  -4.617713  -1.679343  -2.120389
[487]  -8.494737  -2.423394  -1.679343  -2.120389  -2.423394  -2.101425
[493]  -2.120389  -2.101425  -1.434157  -1.434157  -1.679343  -2.101425
```

```
[499]  -1.679343  -1.679343  -4.617713  -2.120389  -2.423394  -2.423394
[505]  -1.679343  -2.101425  -2.101425  -5.051495  -2.101425  -2.101425
[511]  -2.423394  -2.423394  -1.679343  -1.679343  -1.679343  -2.120389
[517]  -4.398675  -2.120389  -2.423394  -4.617713  -1.679343  -1.679343
[523]  -2.101425  -1.679343  -1.679343  -2.423394  -2.101425  -1.679343
[529]  -4.617713  -2.120389  -2.101425  -2.101425  -1.679343  -1.679343
[535]  -9.000801  -5.871690  -4.617713  -2.120389  -2.423394  -1.679343
[541]  -1.679343  -2.423394  -4.667833  -2.120389  -2.120389  -1.679343
[547]  -2.423394  -2.101425  -1.679343  -2.423394  -2.423394  -1.679343
[553]  -1.679343  -2.423394  -2.120389  -1.679343  -2.101425  -1.679343
[559]  -6.866717  -1.679343  -2.423394  -1.679343  -1.679343  -2.120389
[565]  -4.617713  -8.968631  -2.120389  -1.679343  -2.423394  -2.101425
[571]  -2.423394  -4.919553 -17.432760  -2.120389  -2.120389  -4.617713
[577]  -2.120389  -1.679343  -1.679343  -1.679343  -1.679343  -2.423394
[583]  -1.679343 -17.850686  -8.915091  -1.679343  -7.275048  -2.120389
[589]  -2.423394  -2.423394  -2.120389  -7.289861  -8.968631  -9.120971
[595]  -1.679343  -2.120389  -2.101425 -11.410925  -2.101425  -1.679343
[601]  -2.423394  -2.101425  -2.120389  -2.120389  -1.679343  -4.398675
[607]  -1.679343  -1.679343  -4.398675  -2.101425 -17.937274  -2.101425
[613]  -2.120389  -2.120389  -1.679343  -2.423394  -2.423394  -2.101425
[619]  -1.679343  -1.679343  -1.679343  -1.679343  -2.423394  -2.423394
[625]  -2.423394  -2.120389  -2.120389

$method
[1] "ML"

$model
[1] "HKY85+G4"

$parameters
    FreqA     FreqC     FreqG     FreqT     FreqI       A/G       C/T       A/C
0.1818790 0.2347429 0.3459181        NA        NA 3.6683955        NA        NA
      A/T       C/G    Indels     alpha
       NA        NA        NA 0.1798327

$score
[1] 4368.291

$midpoint
[1] 10.08008
> attr(tree, "score") # best score
[1] 4368.291
```

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)
```

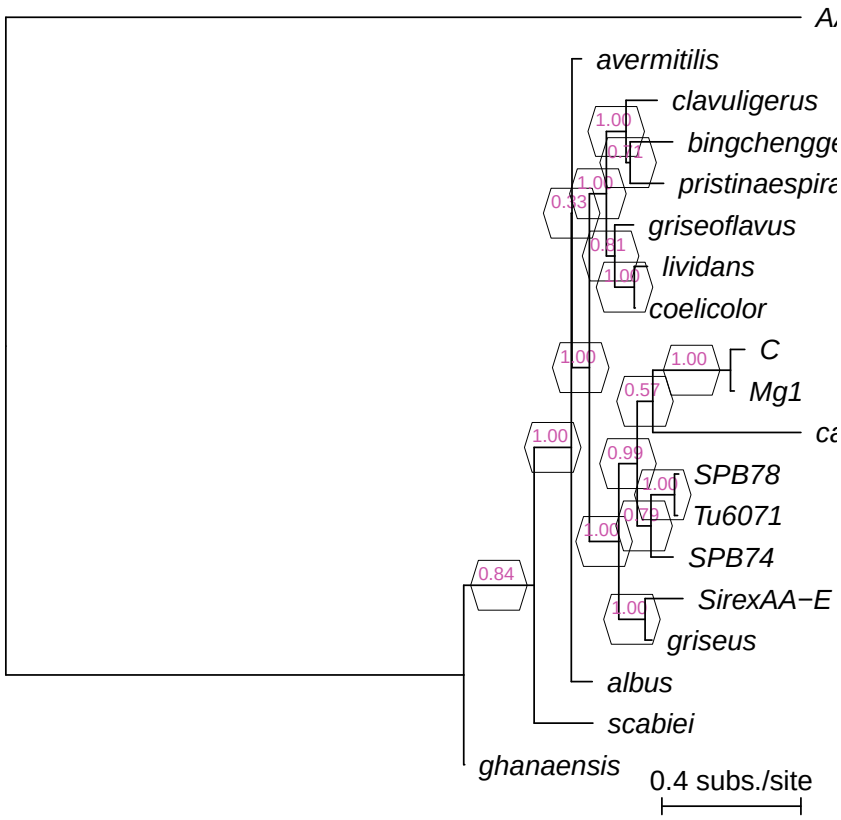

Figure 2: Tree with (aBayes) support probabilities at each internal node.

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)
```

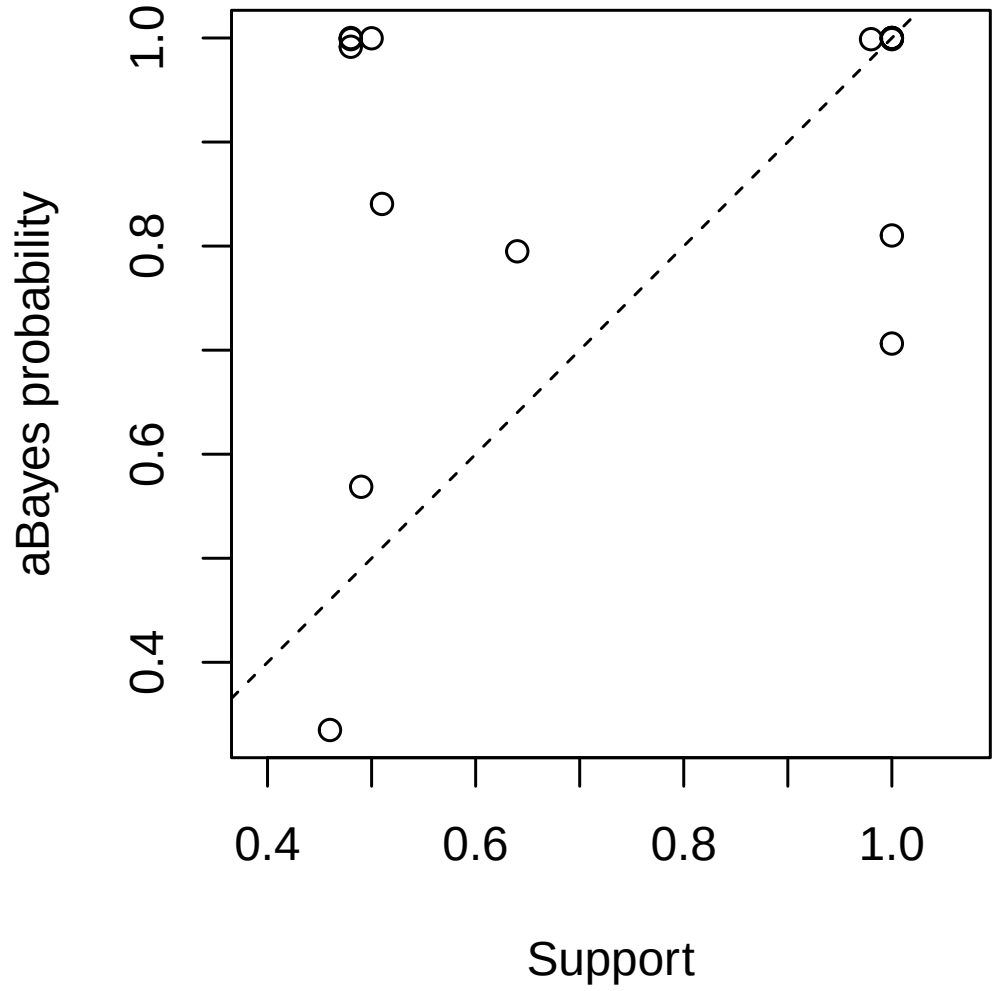

Figure 3: Comparison of aBayes probabilities and branch support values.

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] "G64C"  "G65T"  "G168T" "G171T" "G172C" "G173A" "G180T" "G181C" "G182C"
[10] "G184T" "G185T" "G186A" "G199A" "G201A" "G208T" "G209C" "G211A" "G213C"
[19] "G220C" "G223C" "G224C" "G226A" "G227T" "G229C" "G256C" "G259C" "G271T"
[28] "G272C" "G276T" "G277C" "G280A" "G282A" "G287C" "G288C" "G302A" "G303A"
[37] "G314C" "G316C" "G321T" "G323A" "G324T" "G325C" "G326T" "G327T" "G328C"
[46] "G333C" "G337T" "G338T" "G339C" "G343C" "G371C" "G379C" "G385A" "G389C"
[55] "G393T" "G394C" "G396T" "G397C" "G419C" "G435A" "G437C" "G440T" "G447C"
[64] "G584C"
```

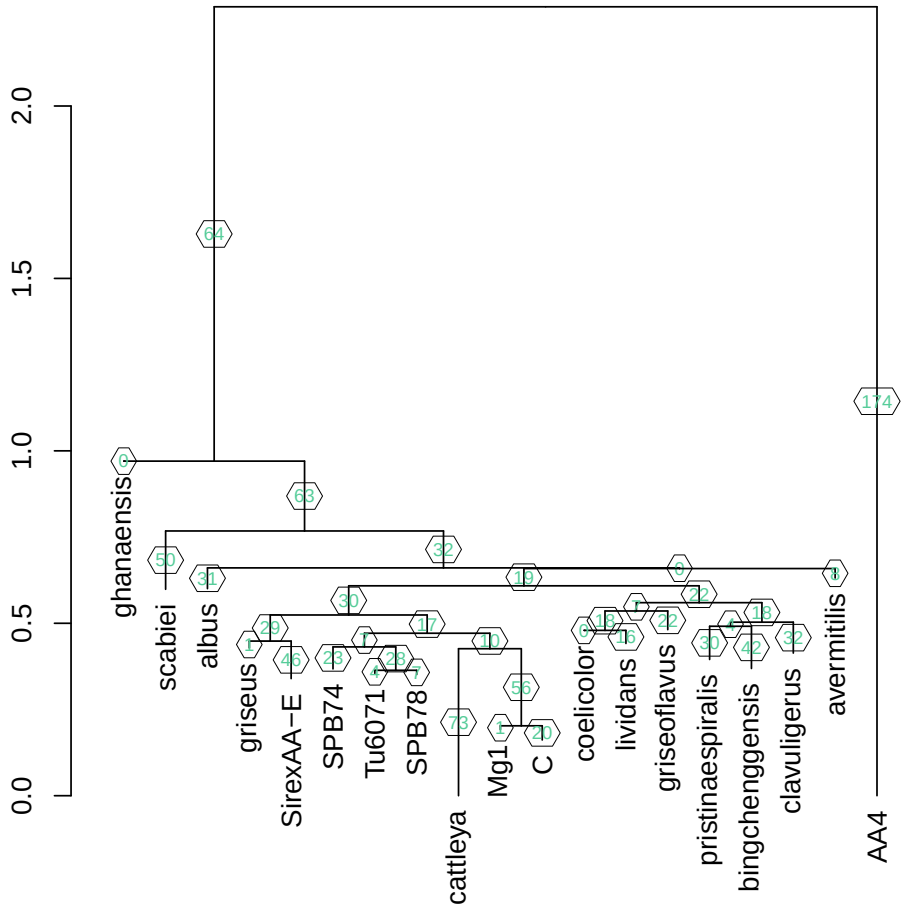

Figure 4: Edges labeled with the number of state transitions.

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 ucrt), `x86_64-w64-mingw32`
- Running under: `Windows Server x64 (build 20348)`
- Matrix products: default
- 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.