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) = 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) = 5022, AICc = 10126, BIC = 10287
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) = 4944, AICc = 9975, BIC = 10143
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

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%), 0 Climbs
2/3. Optimizing regrown tree #2 of 10 to 100:
-ln(L) = 4368.6 (-0.015%), 1 Climb
2/3. Optimizing regrown tree #3 of 12 to 100:
-ln(L) = 4369.3 (+0.015%), 0 Climbs
2/3. Optimizing regrown tree #4 of 12 to 100:
-ln(L) = 4369.3 (+0.015%), 0 Climbs
2/3. Optimizing regrown tree #5 of 12 to 100:
-ln(L) = 4369.3 (+0.015%), 1 Climb
2/3. Optimizing regrown tree #6 of 12 to 100:
-ln(L) = 4368.3 (-0.007%), 1 Climb
2/3. Optimizing regrown tree #7 of 16 to 100:
-ln(L) = 4368.3 (~0.000%), 0 Climbs
2/3. Optimizing regrown tree #8 of 16 to 100:
-ln(L) = 4368.3 (~0.000%), 0 Climbs
2/3. Optimizing regrown tree #9 of 16 to 100:
-ln(L) = 4368.6 (+0.007%), 0 Climbs
2/3. Optimizing regrown tree #10 of 16 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] 2.299925

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

$class
[1] "dendrogram"

$siteLnLs
  [1]  -1.951952  -1.545201  -1.951952  -2.266411  -1.970235  -2.266411
  [7]  -2.120210  -2.422676  -2.120210  -2.120210  -1.679460  -2.120210
 [13]  -2.120210  -2.120210  -1.679460  -2.101388  -2.120210  -2.422676
 [19]   0.000000  -2.120210  -1.679460  -2.101388  -2.120210  -2.422676
 [25]  -4.616826  -1.679460  -2.422676  -2.422676  -2.422676  -1.679460
 [31]  -2.101388  -2.120210  -1.679460  -1.679460  -2.101388  -2.422676
 [37]  -5.870634  -4.616826  -5.870634  -2.120210  -2.120210  -2.120210
 [43]  -1.679460  -2.422676  -2.422676  -1.679460  -2.120210  -2.120210
 [49]  -4.935809  -4.397679  -2.101388  -1.679460  -1.679460  -2.120210
 [55]  -2.120210  -2.120210  -2.422676  -2.422676  -2.120210  -2.120210
 [61]  -2.120210 -13.050612  -7.386197 -16.323536 -13.954900  -6.927968
 [67]  -1.679460  -1.679460  -1.679460  -2.422676  -1.679460  -1.679460
 [73]  -1.679460  -2.422676  -1.679460  -8.562384 -11.879330  -1.679460
 [79]  -2.101388  -2.120210  -1.679460  -2.422676  -2.422676  -1.679460
 [85]  -1.679460  -2.101388  -1.679460  -1.679460  -1.679460  -2.422676
 [91]  -2.120210 -13.833179 -13.213864  -1.679460  -2.120210  -1.679460
 [97]  -2.422676  -2.101388  -2.101388  -1.679460  -1.679460  -1.679460
[103]  -2.422676  -2.120210  -1.679460  -2.422676  -2.422676  -1.679460
[109]  -2.101388  -2.120210  -1.679460  -2.101388  -2.422676  -2.422676
[115]  -2.120210  -2.422676  -2.422676  -1.679460  -1.679460  -2.101388
[121]  -2.422676  -1.679460  -2.120210  -2.120210  -1.679460  -2.101388
[127]  -2.422676  -2.120210  -2.120210  -1.679460  -1.679460  -2.422676
[133]  -2.422676  -1.679460  -1.679460  -2.101388  -1.679460  -2.120210
[139]  -1.679460  -1.679460  -2.120210  -2.101388  -1.679460  -1.679460
[145]  -2.422676  -2.101388  -2.120210  -2.422676  -2.120210  -2.120210
[151]  -2.101388  -2.120210  -2.120210  -2.101388  -2.101388  -2.101388
[157]  -2.120210  -2.101388  -2.422676  -2.422676  -1.679460  -1.679460
[163]  -2.422676  -1.679460  -2.120210  -2.422676 -15.341022 -20.300003
[169]  -5.557028 -12.093274 -12.392676 -13.704229 -17.268871 -13.462853
[175] -11.577862 -17.723384  -7.785268  -9.154457  -8.266765 -15.785348
[181] -22.678707 -19.643346  -7.790352 -12.355401 -11.170231 -11.010178
[187]  -8.831076 -12.994366  -9.871490 -11.493866 -14.467781 -11.172061
```

```
[193]  -7.163086  -7.297468  -6.039414 -10.125920  -9.298786  -8.732447
[199] -14.091190 -11.010648 -13.919066 -12.121754  -8.940856  -2.100374
[205]  -8.048476  -6.823390 -17.557017 -15.427854 -16.884453 -11.416896
[211] -15.988533 -12.339455 -15.734268 -15.184520  -6.868237  -2.120210
[217] -15.332419  -9.785481 -13.502045 -16.287506  -1.679460 -12.723608
[223] -18.023829 -16.151883 -17.542418 -17.155984 -18.168784 -13.919167
[229] -15.663405  -9.174020  -1.679460  -4.919359  -2.101388 -15.993736
[235]  -2.343994  -4.397679  -2.120210  -5.695793  -2.120210  -5.887084
[241] -11.622489  -1.679460  -4.935809  -4.397679  -4.610398  -1.679460
[247]  -4.935809  -4.666673  -4.666673  -4.616826  -1.679460  -2.101388
[253]  -2.101388  -1.679460  -5.887084 -17.265248  -5.051128  -4.666673
[259] -15.588774  -4.610398  -7.943486  -4.919359  -1.679460  -8.060607
[265] -10.401101 -15.535896 -19.047354 -20.011840 -13.321402 -21.918845
[271] -20.259637 -22.081508 -24.712196 -22.864716 -20.676403 -18.972938
[277] -22.857783 -25.603814 -23.385923 -26.687306 -20.949565 -23.934488
[283] -18.117637 -11.184156 -18.442687 -15.217179 -19.232234 -21.689588
[289]  -4.666673  -1.679460  -2.101388  -5.870634  -5.719494  -2.101388
[295]  -7.122572  -5.074829 -11.411097 -13.939889 -13.052603  -6.084887
[301]  -5.338018  -7.229074 -10.746959 -10.356417  -8.277605  -1.679460
[307] -13.795166  -7.346486  -9.701069  -1.679460 -10.785934  -9.113127
[313]  -8.919976 -25.347061 -17.328893 -16.145070  -1.450715  -1.434265
[319]  -1.434265  -7.870746 -21.413053 -24.424125 -22.004668 -21.873241
[325] -27.031484 -25.086229 -23.628533 -22.569150 -22.421641 -19.958098
[331] -12.748042  -3.813824 -19.028620 -23.616173 -23.801499 -18.550950
[337] -21.763567 -24.037908 -17.042319 -12.600979 -17.180182  -8.721438
[343] -17.343073  -5.074829  -9.326963  -1.679460  -1.679460  -5.719494
[349]  -8.417300  -2.120210  -7.708727 -14.230000  -4.610398  -1.679460
[355]  -5.695793  -5.051128  -1.679460  -1.450715  -1.679460  -1.679460
[361]  -5.695793 -13.538887 -11.576230  -2.120210  -2.101388  -1.679460
[367]  -4.666673 -10.892171  -1.679460  -1.679460 -23.736450 -12.543379
[373]  -7.564211  -1.679460 -11.310463 -16.442250  -4.510849 -16.898487
[379] -18.403035  -4.577389  -1.639081  -1.434265 -21.893126 -21.738938
[385] -23.885924  -8.662237 -21.157446 -12.784003 -22.794104 -13.696101
[391] -21.171840 -16.507142 -15.231931 -17.204944 -11.191901 -17.210085
[397] -24.202736 -16.698009  -1.891691  -1.450715  -1.434265  -1.434265
[403]  -9.930223 -19.546525 -12.222150  -9.443687  -4.616826  -5.074829
[409]  -1.679460 -11.525769 -10.924174  -2.120210  -6.654837  -4.616826
[415] -17.730843  -8.290635  -7.640247 -16.992234 -33.191532  -2.422676
[421]   0.000000  -9.490695  -9.371051 -13.825042 -21.381533 -29.424732
[427] -20.571902 -20.146952 -21.598681 -17.373454 -17.635674 -17.584107
[433] -21.730957 -30.070429 -15.023006 -25.149574 -23.979571 -23.115116
[439] -22.626447 -14.697021 -19.522598 -15.745575  -1.679460  -7.492404
[445]  -8.645787 -16.208999 -18.541895  -4.919359  -1.679460  -1.679460
[451]  -7.492404  -1.679460  -5.671137 -15.414931  -6.753694  -1.545201
[457]  -1.545201  -1.951952  -4.616826  -1.679460  -5.051128  -2.101388
[463]  -1.679460 -12.519435  -2.101388  -2.101388  -1.679460  -2.422676
[469]  -1.679460  -2.422676  -2.422676  -2.120210  -9.435114 -11.002162
[475]  -4.616826  -2.422676 -12.072413  -2.422676  -1.679460  -2.101388
[481]  -1.679460  -1.679460  -2.422676  -4.616826  -1.679460  -2.120210
[487]  -8.498335  -2.422676  -1.679460  -2.120210  -2.422676  -2.101388
[493]  -2.120210  -2.101388  -1.434265  -1.434265  -1.679460  -2.101388
```

```
[499]  -1.679460  -1.679460  -4.616826  -2.120210  -2.422676  -2.422676
[505]  -1.679460  -2.101388  -2.101388  -5.051128  -2.101388  -2.101388
[511]  -2.422676  -2.422676  -1.679460  -1.679460  -1.679460  -2.120210
[517]  -4.397679  -2.120210  -2.422676  -4.616826  -1.679460  -1.679460
[523]  -2.101388  -1.679460  -1.679460  -2.422676  -2.101388  -1.679460
[529]  -4.616826  -2.120210  -2.101388  -2.101388  -1.679460  -1.679460
[535]  -9.002402  -5.870634  -4.616826  -2.120210  -2.422676  -1.679460
[541]  -1.679460  -2.422676  -4.666673  -2.120210  -2.120210  -1.679460
[547]  -2.422676  -2.101388  -1.679460  -2.422676  -2.422676  -1.679460
[553]  -1.679460  -2.422676  -2.120210  -1.679460  -2.101388  -1.679460
[559]  -6.868237  -1.679460  -2.422676  -1.679460  -1.679460  -2.120210
[565]  -4.616826  -8.967725  -2.120210  -1.679460  -2.422676  -2.101388
[571]  -2.422676  -4.919359 -17.443579  -2.120210  -2.120210  -4.616826
[577]  -2.120210  -1.679460  -1.679460  -1.679460  -1.679460  -2.422676
[583]  -1.679460 -17.851460  -8.914019  -1.679460  -7.277152  -2.120210
[589]  -2.422676  -2.422676  -2.120210  -7.291883  -8.967725  -9.119159
[595]  -1.679460  -2.120210  -2.101388 -11.418015  -2.101388  -1.679460
[601]  -2.422676  -2.101388  -2.120210  -2.120210  -1.679460  -4.397679
[607]  -1.679460  -1.679460  -4.397679  -2.101388 -17.947893  -2.101388
[613]  -2.120210  -2.120210  -1.679460  -2.422676  -2.422676  -2.101388
[619]  -1.679460  -1.679460  -1.679460  -1.679460  -2.422676  -2.422676
[625]  -2.422676  -2.120210  -2.120210

$method
[1] "ML"

$model
[1] "HKY85+G4"

$parameters
    FreqA     FreqC     FreqG     FreqT     FreqI       A/G       C/T       A/C
0.1813192 0.2341452 0.3458013        NA        NA 3.6906117        NA        NA
      A/T       C/G    Indels     alpha
       NA        NA        NA 0.1788084

$score
[1] 4368.291

$midpoint
[1] 10.07617
> 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)
```

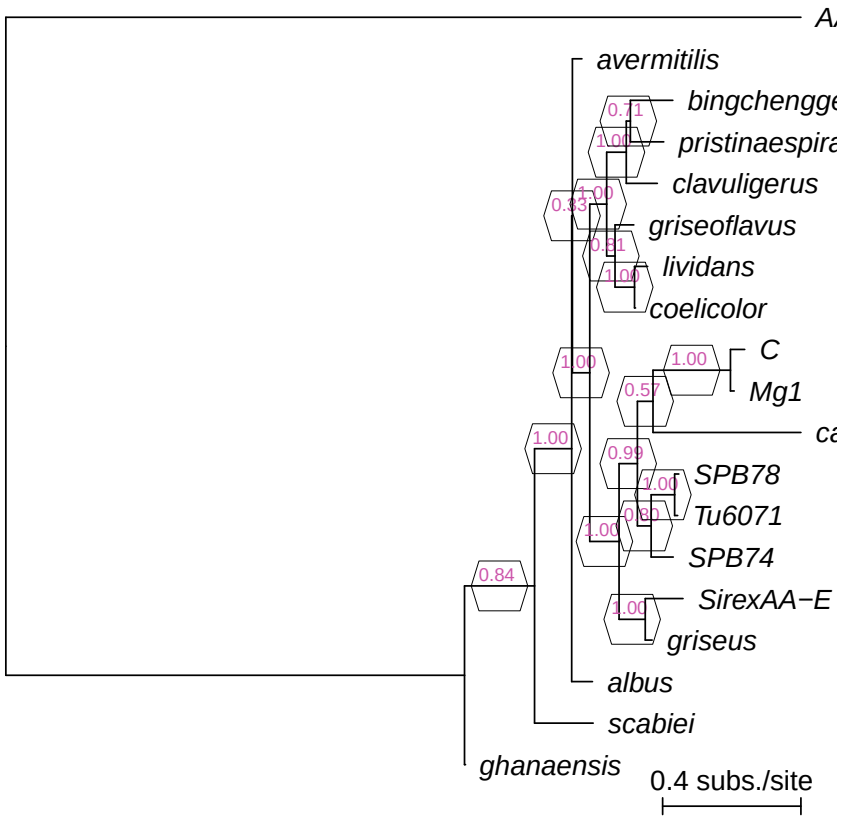

Figure 2: Tree with (aBayes) support probabilities at each internal node.

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)
```

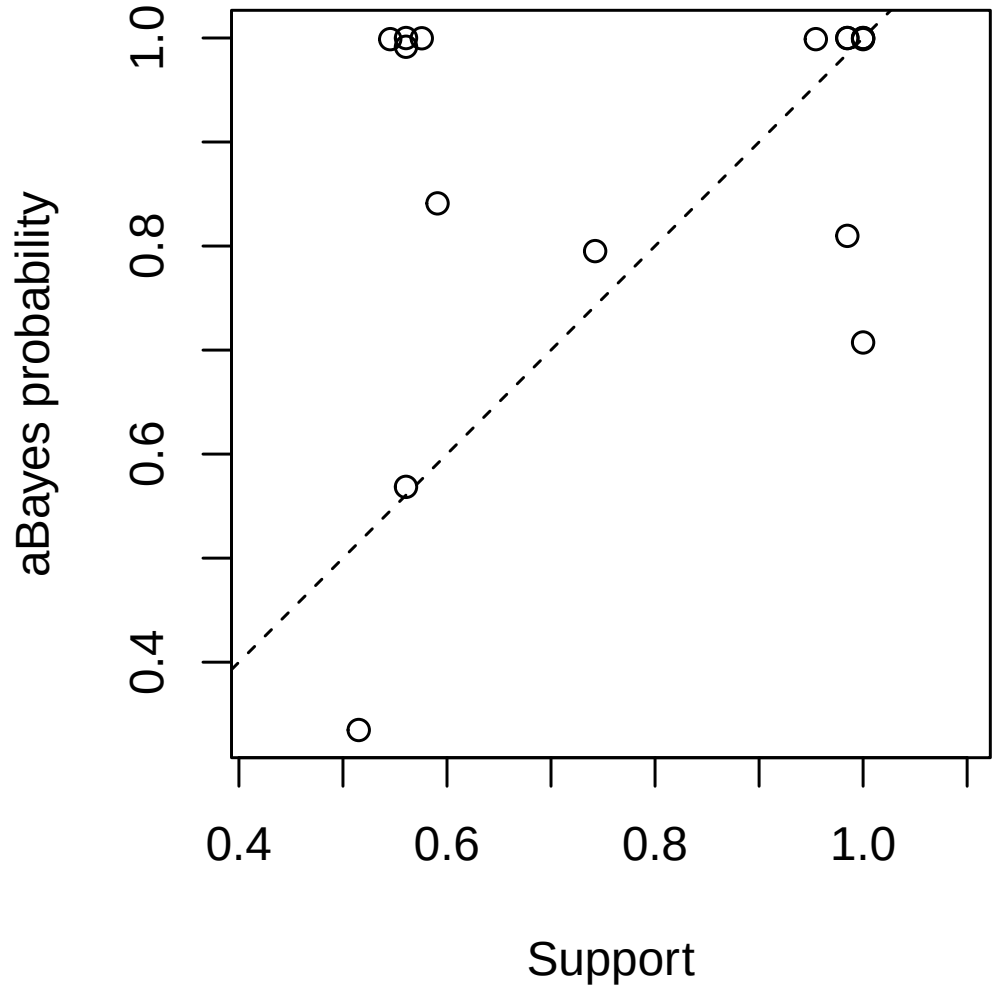

Figure 3: Comparison of aBayes probabilities and branch support values.

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"
```

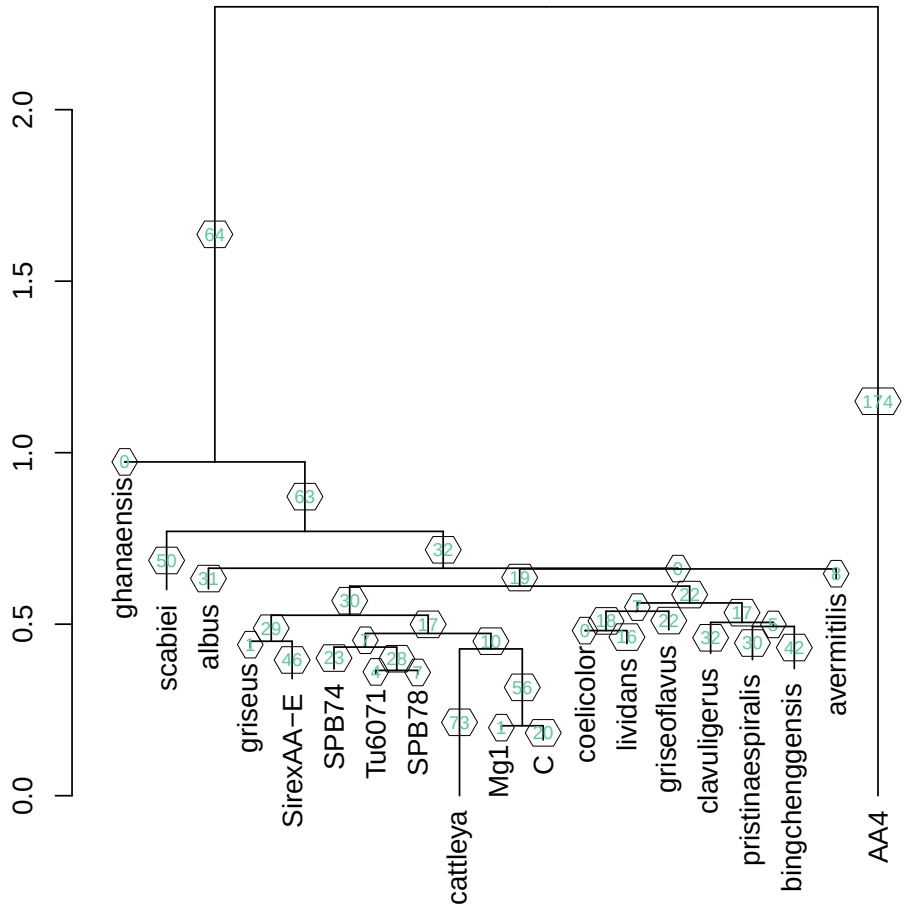

Figure 4: Edges labeled with the number of state transitions.

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), `x86_64-apple-darwin17.0`
- Running under: `macOS Mojave 10.14.6`
- Matrix products: default
- BLAS: `/Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib`
- LAPACK: `/Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib`
- 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.