Bayesian Phylogenetic Analysis and Trait Evolution in Cucurbita Species

Samantha A. Taylor

01 November 2021

Overview

For this exercise, you will be looking at the evolutionary relationships among 8 different species of pumpkins, squashes, and gourds. Your goals are to 1) set up and run a phylogenetic analysis with MrBayes with appropriate parameter settings, and 2) use your resulting tree to try and determine how many different times domestication occurred in this genus (Cucurbita).

The samples in the dataset include:

- 3 wild species: Cu. foetidissima, Cu. ecuadorensis, Cu. martinez
- 5 domesticated varieties: Cu. ficifolia, Cu. maxima, Cu. pepo ssp. pepo, Cu. argyrosperma ssp. sororia, Cu. moschata.
- 1 outgroup: Citrullus lanatus

Download the Data Set

You will have a copy of the NADH gene sequence taken from each of the taxa. You can download the fasta file with these sequences. You can also download a text file with the table of information about each species.

Part One: Align the Sequences and Determine the Best Nucleotide Substitution Model

Read the fasta file into R and perform an alignment with the msa() function. For this data set, please use the Muscle alignment method instead of Clustal-Omega. Use modelTest() to determine the best nucleotide substitution model to use in your analysis. Once you have your alignment, you can save it as a nexus file.

```r
dnaSeqs = readDNAStringSet("pumpkin-nadh.fa",format="fasta")
aligns = msa(dnaSeqs, method="Muscle")

forPhang = msaConvert(aligns, type="phangorn::phyDat")
mt = modelTest(forPhang)
```

```
## Model         df  logLik   AIC      BIC
##            JC 15 -3729.52 7489.041 7571.179
##          JC+I 16 -3689.949 7411.897 7499.512
##       JC+G(4) 16 -3692.646 7417.291 7504.906
##     JC+G(4)+I 17 -3690.467 7414.933 7508.024
##           F81 18 -3717.503 7471.005 7569.572
##         F81+I 19 -3677.774 7393.549 7497.591
##      F81+G(4) 19 -3680.522 7399.044 7503.086
##    F81+G(4)+I 20 -3678.322 7396.644 7506.162
##           K80 16 -3729.31 7490.62 7578.235
##         K80+I 17 -3689.754 7413.507 7506.598
##      K80+G(4) 17 -3692.445 7418.89 7511.98
##    K80+G(4)+I 18 -3690.266 7416.533 7515.099
##           HKY 19 -3717.327 7472.653 7576.695
##         HKY+I 20 -3677.627 7395.254 7504.773
##      HKY+G(4) 20 -3680.365 7400.73 7510.248
##    HKY+G(4)+I 21 -3678.17 7398.34 7513.334
##          TrNe 17 -3725.327 7484.654 7577.745
##        TrNe+I 18 -3685.892 7407.785 7506.351
##     TrNe+G(4) 18 -3688.462 7412.923 7511.49
##   TrNe+G(4)+I 19 -3686.181 7410.362 7514.404
##           TrN 20 -3714.97 7469.939 7579.457
##         TrN+I 21 -3675.548 7393.096 7508.09
##      TrN+G(4) 21 -3678.164 7398.327 7513.321
##    TrN+G(4)+I 22 -3675.946 7395.892 7516.361
##          TPM1 17 -3728.015 7490.029 7583.12
##        TPM1+I 18 -3688.254 7412.507 7511.074
##     TPM1+G(4) 18 -3691.015 7418.029 7516.596
##   TPM1+G(4)+I 19 -3688.769 7415.537 7519.579
##           K81 17 -3728.015 7490.029 7583.12
##         K81+I 18 -3688.254 7412.507 7511.074
##      K81+G(4) 18 -3691.015 7418.029 7516.596
##    K81+G(4)+I 19 -3688.769 7415.537 7519.579
##         TPM1u 20 -3715.981 7471.963 7581.481
##       TPM1u+I 21 -3676.037 7394.073 7509.067
##    TPM1u+G(4) 21 -3678.859 7399.717 7514.711
## TPM1u+G(4)+I 22 -3676.583 7397.165 7517.635
##          TPM2 17 -3729.177 7492.355 7585.445
##        TPM2+I 18 -3689.709 7415.417 7513.984
##     TPM2+G(4) 18 -3692.371 7420.741 7519.308
##   TPM2+G(4)+I 19 -3690.209 7418.419 7522.461
##         TPM2u 20 -3717.263 7474.525 7584.044
##       TPM2u+I 21 -3677.618 7397.237 7512.231
##    TPM2u+G(4) 21 -3680.342 7402.683 7517.677
## TPM2u+G(4)+I 22 -3678.156 7400.312 7520.782
##          TPM3 17 -3729.309 7492.619 7585.709
##        TPM3+I 18 -3689.754 7415.507 7514.073
##     TPM3+G(4) 18 -3692.445 7420.89 7519.456
##   TPM3+G(4)+I 19 -3690.266 7418.533 7522.575
##         TPM3u 20 -3717.196 7474.392 7583.911
##       TPM3u+I 21 -3677.47 7396.939 7511.933
##    TPM3u+G(4) 21 -3680.212 7402.423 7517.417
## TPM3u+G(4)+I 22 -3678.014 7400.027 7520.497
##         TIM1e 18 -3724.043 7484.086 7582.652
##      TIM1e+I 19 -3684.412 7406.824 7510.866
##   TIM1e+G(4) 19 -3687.046 7412.092 7516.134
## TIM1e+G(4)+I 20 -3684.702 7409.404 7518.922
##         TIM1 21 -3713.633 7469.267 7584.261
##       TIM1+I 22 -3673.974 7391.948 7512.418
##    TIM1+G(4) 22 -3676.669 7397.338 7517.808
##  TIM1+G(4)+I 23 -3674.375 7394.749 7520.695
##        TIM2e 18 -3725.201 7486.403 7584.969
##      TIM2e+I 19 -3685.849 7409.699 7513.741
##   TIM2e+G(4) 19 -3688.391 7414.781 7518.824
## TIM2e+G(4)+I 20 -3686.123 7412.246 7521.764
##         TIM2 21 -3714.909 7471.819 7586.813
##       TIM2+I 22 -3675.541 7395.082 7515.552
##    TIM2+G(4) 22 -3678.143 7400.286 7520.756
##  TIM2+G(4)+I 23 -3675.933 7397.866 7523.812
##        TIM3e 18 -3725.326 7486.653 7585.219
##      TIM3e+I 19 -3685.892 7409.784 7513.826
##   TIM3e+G(4) 19 -3688.461 7414.922 7518.965
## TIM3e+G(4)+I 20 -3686.181 7412.361 7521.879
##         TIM3 21 -3714.838 7471.676 7586.67
##       TIM3+I 22 -3675.396 7394.793 7515.263
##    TIM3+G(4) 22 -3678.015 7400.03 7520.5
##  TIM3+G(4)+I 23 -3675.797 7397.593 7523.539
##         TVMe 19 -3727.837 7493.674 7597.716
##       TVMe+I 20 -3688.176 7416.353 7525.871
##    TVMe+G(4) 20 -3690.902 7421.805 7531.323
##  TVMe+G(4)+I 21 -3688.676 7419.351 7534.346
##          TVM 22 -3715.788 7475.577 7596.046
##        TVM+I 23 -3675.913 7397.826 7523.772
##     TVM+G(4) 23 -3678.71 7403.419 7529.365
##   TVM+G(4)+I 24 -3676.45 7400.9 7532.321
##          SYM 20 -3723.874 7487.748 7597.266
##        SYM+I 21 -3684.339 7410.678 7525.672
##     SYM+G(4) 21 -3686.939 7415.878 7530.872
##   SYM+G(4)+I 22 -3684.61 7413.219 7533.689
##          GTR 23 -3713.443 7472.887 7598.833
##        GTR+I 24 -3673.857 7395.715 7527.136
##     GTR+G(4) 24 -3676.528 7401.056 7532.478
##   GTR+G(4)+I 25 -3674.249 7398.497 7535.395
```

```
mt[which(mt$logLik==max(mt$logLik)),]
```

```
##    Model df    logLik      AIC       AICw     AICc      AICcw      BIC
## 90 GTR+I 24 -3673.857 7395.715 0.03475327 7396.404 0.0324798 7527.136
```

```
mt[which(mt$BIC==min(mt$BIC)),]
```

```
##   Model df    logLik      AIC     AICw     AICc     AICcw      BIC
## 6 F81+I 19 -3677.774 7393.549 0.102644 7393.984 0.1089266 7497.591
```

```
dna = as.DNAbin(forPhang)
D = dist.dna(dna, model="F81", gamma=FALSE)
```

```
write.nexus.data(as.DNAbin(forPhang), file="pumpkin-aln.nex")
```

Question 1 (3 pts)

Which nucleotide substitution model is the best fit for your data? What are the parameters of this model?

Answer: Based on the results of the R modelTest() function, the model with the highest log likelihood score was the general time reversible model with gamma and inverse gamma (GTR+G+I). Also, based on the results of the R modelTest() function, the model with the lowest BIC score was the Felsenstein 1981 model (also known as F81 + Inverse gamma model or F81 + I), so I used the F81 model with gamma = FALSE (since gamma is not equal to inverse gamma) in my distance calculation. This model is a four-parameter model that incorporates different equilibrium frequency distributions for each nucleotide.

Part Two: Run MrBayes

Upload your nexus files to the cluster, and perform the Bayesian analysis with MrBayes. You may perform the run in interactive mode OR with a slurm script, but either way make sure to save a copy of your parameter settings in order to answer the next question in this assignment. If you run MrBayes in interactive mode, simply type "showmodel" after editing your settings, then you may either copy and past the results of this into a text file, or you may take a screenshot of your settings. If you run MrBayes with a slurm script, all you need is a saved copy of your input parameters file.

Edit your model settings in MrBayes according the the best fit model you obtained in part one. If you need to adjust your priors as well in order to achieve the right model, then do so. Add a command to also set the outgroup species: 'Outgroup lanatus'.

In interactive mode, just enter the above command as it is shown; in the slurm script, just add a line with this command some time before the "mcmc" command starts.

Run MrBayes with a minimum of 20,000 iterations (for those of you using the job script, note that you do not need to go above 100,000 iterations, and would probably be fine with 30,000).

Question 2 (3 pts)

Provide a copy of the final model settings/parameters you used in your MrBayes run. You can either copy and paste the text from showmodel, copy the code block from your slurm script, or even provide a screenshot showing one of those things.

```
## Running MrBayes in interactive mode
module load mrbayes
mb

  MrBayes > execute pumpkin-aln.nex
  MrBayes > showmodel

  MrBayes > lset nst=1 rates=propinv
  MrBayes > Outgroup lanatus

  MrBayes > showmodel
  MrBayes > mcmc ngen=20000 samplefreq=100

  MrBayes > sump relburnin=yes burninfrac=0.25
  MrBayes > sumt relburnin=yes burninfrac=0.25 conformat=simple
```

Answer: I used the setting “Nst = 1” because we want to specify that we are only allowing 1 substitution rate. I set Nst equal to 1 because, since I used the F81 model, we are only allowing one possible substitution rate, because all possible mutations must happen at the same rate. Because I am using the model F81 + I, I need to tell MrBayes to set the rates to “propinv”, which is the same as using just (+I) without the G. I used ‘Outgroup lanatus’ to run MrBayes specifying that the outgroup of the dataset is watermelon.

Question 3 (3 pts)

Do you think that you ran your model long enough to achieve convergence and appropriate levels of mixing? Why or why not? Be sure to provide clear evidence, either in the form of screenshots, text that you have copied and pasted, or plots that you have made in R.

```
Average standard deviation of split frequencies: 0.002341

Continue with analysis? (yes/no): no

Analysis completed in 3 seconds
   Analysis used 3.20 seconds of CPU time on processor 0
   Likelihood of best state for "cold" chain of run 1 was -3678.83
   Likelihood of best state for "cold" chain of run 2 was -3678.83

   Acceptance rates for the moves in the "cold" chain of run 1:
      With prob.   (last 100)   chain accepted proposals by move
         11.3 %     ( 13 %)     Dirichlet(Pi)
         28.9 %     ( 23 %)     Slider(Pi)
         61.6 %     ( 66 %)     Slider(Pinvar)
          0.6 %     (  0 %)     ExtSPR(Tau,V)
          0.2 %     (  0 %)     ExtTBR(Tau,V)
          0.7 %     (  1 %)     NNI(Tau,V)
          0.8 %     (  0 %)     ParsSPR(Tau,V)
         49.7 %     ( 36 %)     Multiplier(V)
         34.0 %     ( 36 %)     Nodeslider(V)
         20.9 %     ( 24 %)     TLMultiplier(V)

   Acceptance rates for the moves in the "cold" chain of run 2:
      With prob.   (last 100)   chain accepted proposals by move
         14.1 %     ( 17 %)     Dirichlet(Pi)
         25.4 %     ( 23 %)     Slider(Pi)
         56.6 %     ( 54 %)     Slider(Pinvar)
          0.4 %     (  1 %)     ExtSPR(Tau,V)
          0.1 %     (  0 %)     ExtTBR(Tau,V)
          0.3 %     (  0 %)     NNI(Tau,V)
          0.7 %     (  1 %)     ParsSPR(Tau,V)
         51.6 %     ( 47 %)     Multiplier(V)
         33.0 %     ( 32 %)     Nodeslider(V)
         20.4 %     ( 17 %)     TLMultiplier(V)
   Chain swap information for run 1:

              1     2     3     4
        --------------------------
      1 |        0.76  0.58  0.40
      2 |  3314        0.78  0.55
      3 |  3270  3365        0.75
      4 |  3402  3368  3281

   Chain swap information for run 2:

              1     2     3     4
        --------------------------
      1 |        0.79  0.64  0.47
      2 |  3354        0.81  0.64
      3 |  3365  3271        0.82
      4 |  3369  3333  3308

   Upper diagonal: Proportion of successful state exchanges between chains
   Lower diagonal: Number of attempted state exchanges between chains

   Chain information:

     ID -- Heat
    -----------
      1 -- 1.00  (cold chain)
      2 -- 0.91
      3 -- 0.83
      4 -- 0.77

   Heat = 1 / (1 + T * (ID - 1))
      (where T = 0.10 is the temperature and ID is the chain number)
```

Answer: I believe I ran the model long enough to achieve convergence and appropriate levels of mixing due to the standard deviation of split frequencies that were reported. The lower the value is, the more the chains have converged, with 0.01 being the value that determines whether or not to continue iterating the MCMC calculations. My value was '0.002341' after 20000 iterations, which is much lower than 0.01. **Therefore, because $0.002341 < 0.01$ (the cut-off value used), I conclude that I ran my model long enough to achieve convergence and appropriate levels of mixing.**

Part Three: Plot the Results with Node Support Values

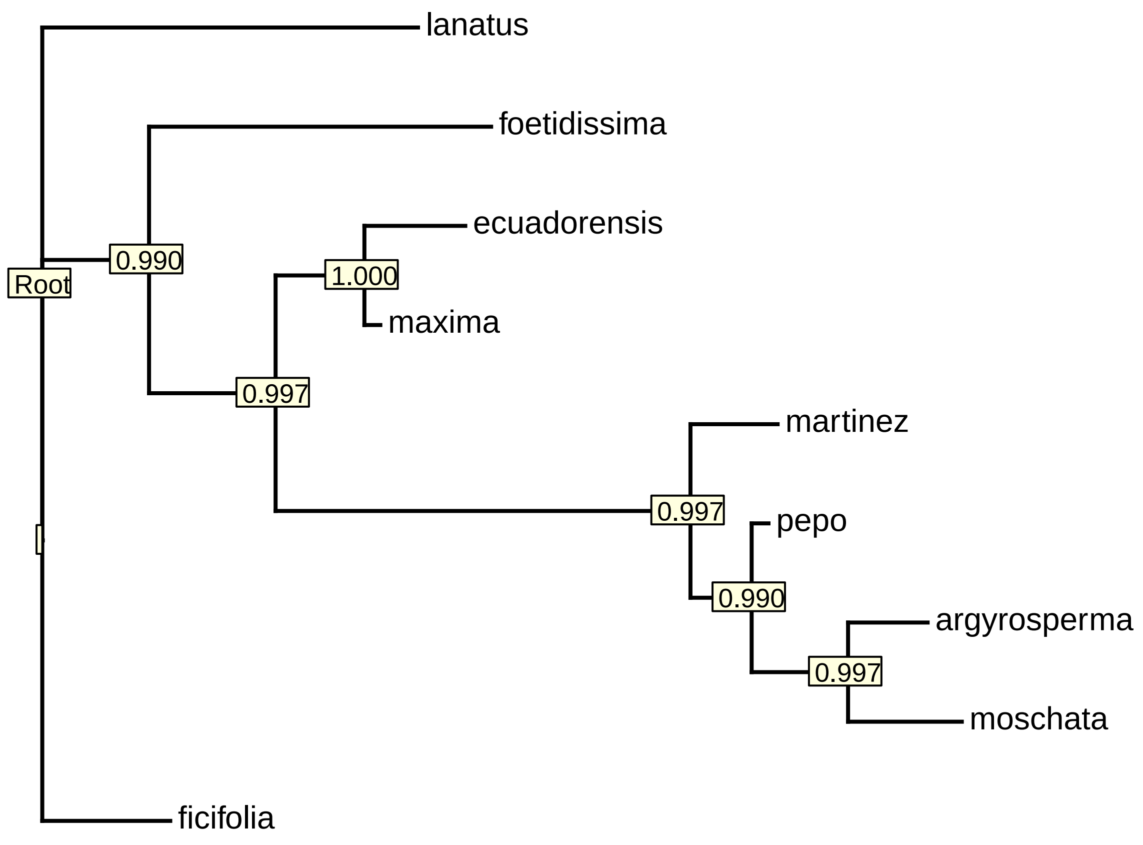
When MrBayes has finished running, download and plot your final consensus tree with posterior probability values at each node.

```
bayesTree = read.nexus("pumpkin-aln.nex.con.tre")

bayesTree1 = bayesTree[[1]]

bayesTree2 = root(bayesTree1, "lanatus", resolve.root=T)
```

Question 4 (2 pts)

Provide the plot of your tree with support values.

```
plotTree(bayesTree2, edge.width=2, font=1)
nodelabels(bayesTree2$node.label, cex=0.8, bg = "lightyellow")
```

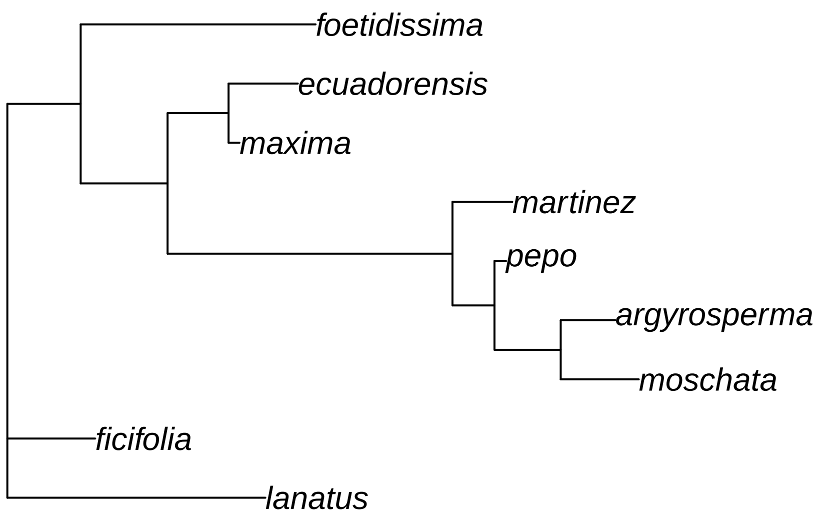


Part Four: Ancestral State Reconstruction of Domestication

Using the Maximum Parsimony method, perform an ancestral state reconstruction of the domestication trait. Because it is essentially impossible to go back to a wild state after being domesticated, you should set up a custom transition cost matrix where we set the cost of going from wild to domesticated as 1, but the cost of going in the opposite direction as Infinity. Here is my code for setting up a matrix like this (where trait 1 is domesticated, and trait 2 is wild): 'Q = matrix(data=c(0,1,Inf,0), nrow=2, dimnames=list(c(1,2), c(1,2)))'.

```
options(stringsAsFactors=FALSE)
plot(bayesTree1, main="Starting Ancestral State Tree")
```

Starting Ancestral State Tree

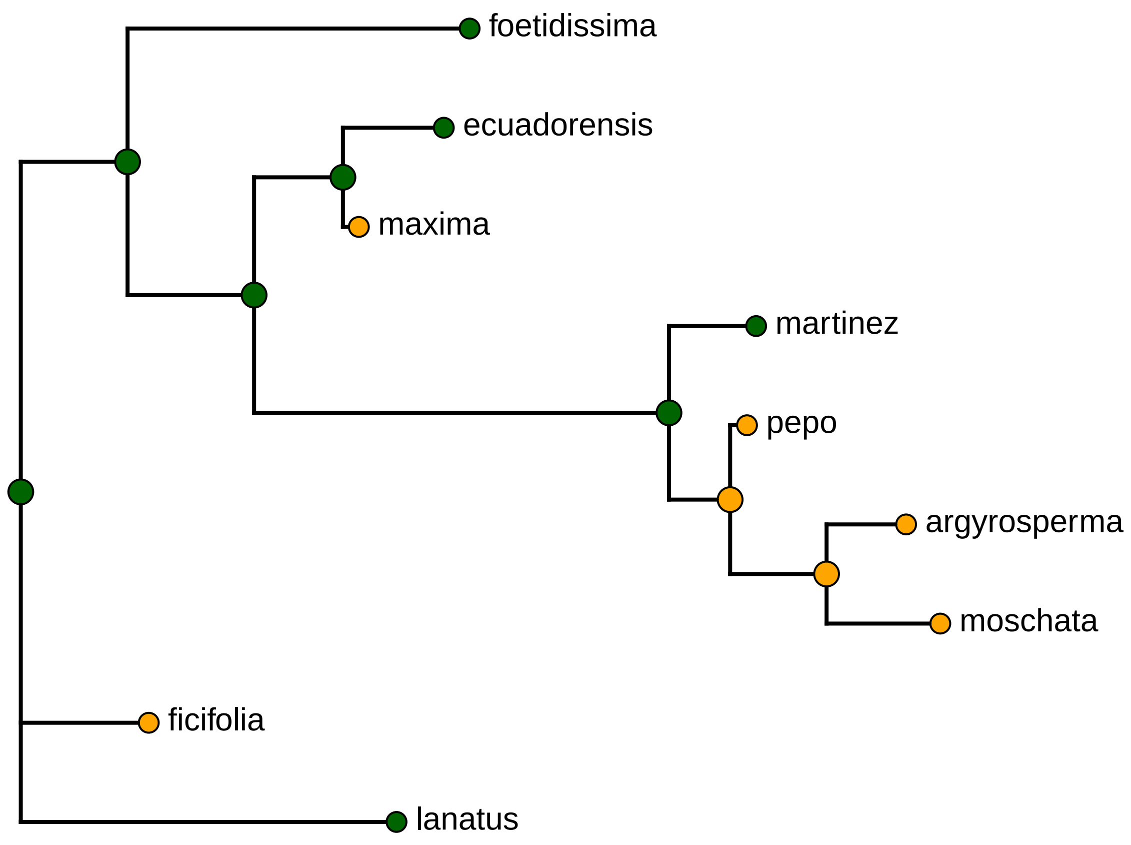


```
my.states = c('domesticated', 'domesticated', 'domesticated', 'wild', 'wild', 'wild', 'domesticated', '
names(my.states) = bayesTree1$tip.label

numeric.states = as.numeric(as.factor(my.states))

Q = matrix(data=c(0,1,Inf,0), nrow=2, dimnames=list(c(1,2), c(1,2)))

mp_custom = asr_max_parsimony(tree=bayesTree1, tip_states=numeric.states, Nstates=2, transition_costs=Q

pie.matrix = matrix(mp_custom$ancestral_likelihoods, ncol=2)
rownames(pie.matrix) = seq(10,16)
colnames(pie.matrix) = c("wild", "domesticated")
```

Question 5 (2 pts)

Provide the plot of your tree with ancestral states.

```
plotTree(bayesTree1, offset=0.5)
tiplabels(pie = to.matrix(my.states, sort(unique(my.states))), piecol = c("orange", "darkgreen"), cex =
nodelabels(node = as.numeric(rownames(pie.matrix)), pie = pie.matrix, piecol = c("orange", "darkgreen")
```



```
#add.simmap.legend(prompt=T, leg=c("wild", "domesticated"), colors=c("darkgreen", "orange"))
```

Question 6 (2pts)

How many separate domestication events do you think occurred based on your tree? Be sure to describe which lineages or nodes each domestication event happened on.

Answer: I believe 3 total domestication events occurred based on the tree: 1 when maxima evolved into domesticated from its common ancestor with ecuadorensis, 1 when pepo (+ argyrosperma + moschata) evolved into domesticated from its common ancestor with martinez, and one during the polytomy of ficifolia, lanatus, and everyone else, because lanatus stayed wild, as did the common ancestor of foetidissima and the ecuadorensis clade (+) martinez clade) (all start as wild).

Bonus Round: Continuous State Evolution of Size

One of the traits that often undergoes strong selection during domestication is size, with domesticated varieties typically being selected to be much larger than their wild ancestors. To look at size evolution in pumpkins, and see how it correlates with domestication, perform a continuous model of ancestral state reconstruction.

First, define your set of continuous scores based on average size:

```
info = read.table("pumpkin-info.txt", header=TRUE, sep="\t")
cont.scores = info$AvgWeight_lbs
names(cont.scores) = info$Species
m = match(bayesTree1$tip.label, names(cont.scores))
```

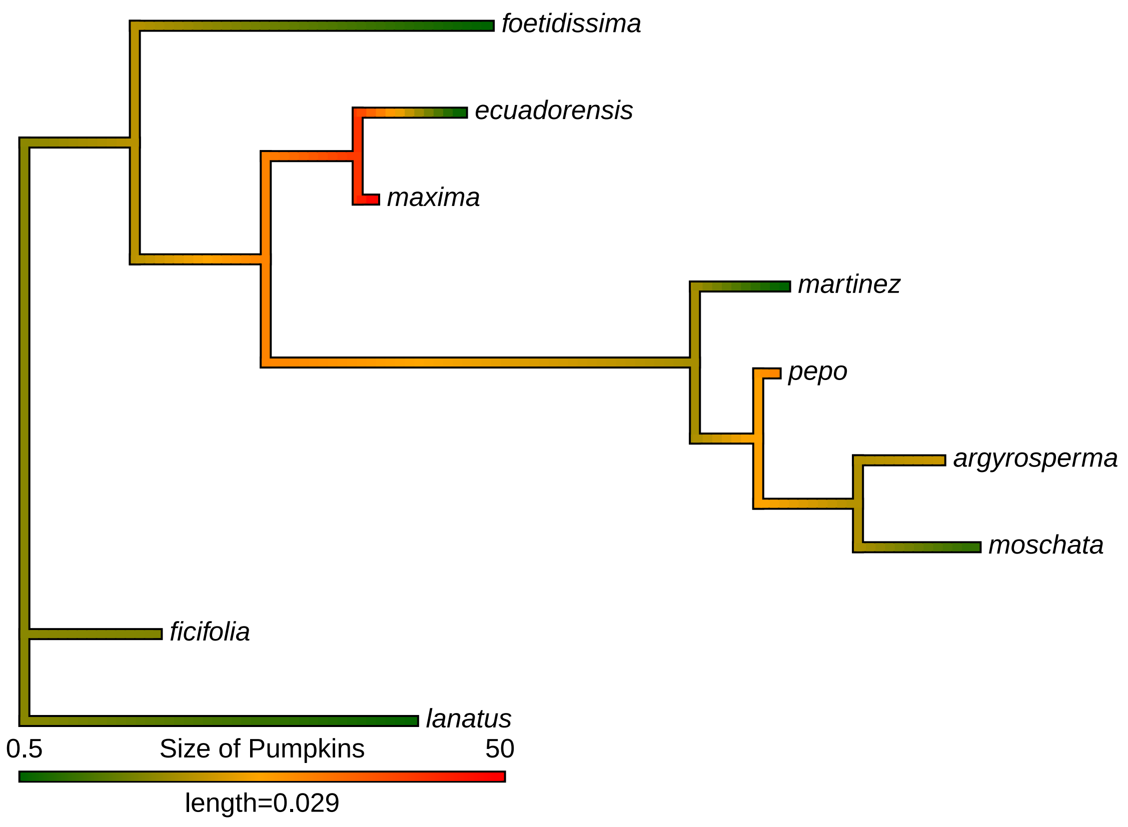
Then use the contMap() function to simultaneously perform the reconstruction and plot the results. As a bonus, we'll customize the colors to be more fall-appropriate!

```
obj = contMap(bayesTree1, cont.scores, plot = FALSE)
obj = setMap(obj, colors=c("darkgreen","orange", "red"))
```

Bonus Question (+1 point)

Show the plot of your continuous trait evolution tree. Does it look to you like size correlates with evolution, or not really? Be sure to explain your answer.

```
plot(obj, fsize=c(0.8,0.8), leg.txt="Size of Pumpkins")
```



Answer: It kind of looks like size correlates with evolution, but only for the wild type. This is because the 4 wild species are green (meaning low length/size), whereas the domesticated species range from large size/length (maxima) to a middle range (pepo + argyrosperma) to small range (ficifolia + moschata). If we domesticate pumpkins, we can select for sizing, however, in the wild, species come with the size they are.