Longitudinal Model Evaluation and Distance-Based Analysis of Influenza (H3N2)

Samantha A. Taylor

20 September 2021

Install and Load Packages

We'll be using four different packages for the exercise today: msa to align sequences with Clustal-O, phangorn to perform distance calculations, phytools to perform bootstrapping, and adegenet to plot our resulting phylogenies with colors.

If the adegenet installation is problematic, try installing without re-compiling from source code. If it continues to be a problem, then do not worry too much about it; you can still make your plots and complete the assignment without this package-you just will not get the nice colors. I'll provide alternative plotting instructions where relevant.

Download the Fasta sequences

For this exercise, you will be using a dataset that consists of 48 DNA sequences from the seasonal influenza virus (H3N2). These 48 samples were collected over 16 years (from 1993 - 2008), with 3 samples from each year. The samples are labeled by the year they were collected.

Click here Download here to download the .fasta file for these sequences.

Align the sequences with Clustal-O using the MSA package

Once you have downloaded the sequences, you need to align them with Clustal-O, like we did in the first week of class.

Except this time, instead of going through the web, we'll use the msa package from R:

```
dnaSeqs = readDNAStringSet("flu_seqs.fasta",format="fasta")
aligns = msa(dnaSeqs, method="ClustalOmega")
```

```
## using Gonnet
```

Use the Phangorn package to compare substitution models

Next, we'll use the phangorn package to test for the best nucleotide substitution model.

The first thing we have to do is convert the msa output to a phyDat object that phangorn can work with:

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

Next, we use the modelTest function to compare how well the different substitution models fit our data. Write the results of modelTest to a text file, and save this file somewhere that you can access it again later.

```
mt = modelTest(forPhang)
```

```
## Model        df  logLik   AIC      BIC
##           JC 93 -4898.331 9982.662 10488.49
##         JC+I 94 -4868.024 9924.048 10435.31
##      JC+G(4) 94 -4867.283 9922.566 10433.83
##    JC+G(4)+I 95 -4867.175 9924.35 10441.05
##          F81 96 -4858.893 9909.786 10431.93
##        F81+I 97 -4828.603 9851.206 10378.79
##     F81+G(4) 97 -4827.907 9849.814 10377.39
##   F81+G(4)+I 98 -4827.816 9851.632 10384.65
##          K80 94 -4754.472 9696.944 10208.21
##        K80+I 95 -4723.603 9637.207 10153.91
##     K80+G(4) 95 -4722.816 9635.632 10152.33
##   K80+G(4)+I 96 -4722.685 9637.37 10159.51
##          HKY 97 -4715.342 9624.684 10152.26
##        HKY+I 98 -4684.759 9565.518 10098.54
##     HKY+G(4) 98 -4683.832 9563.665 10096.68
##   HKY+G(4)+I 99 -4683.686 9565.371 10103.83
##         TrNe 95 -4754.467 9698.934 10215.64
##       TrNe+I 96 -4723.602 9639.205 10161.35
##    TrNe+G(4) 96 -4722.816 9637.631 10159.77
##  TrNe+G(4)+I 97 -4722.684 9639.369 10166.95
##          TrN 98 -4714.059 9624.117 10157.14
##        TrN+I 99 -4683.352 9564.703 10103.16
##     TrN+G(4) 99 -4682.564 9563.127 10101.59
##   TrN+G(4)+I 100 -4682.447 9564.895 10108.79
##         TPM1 95 -4746.849 9683.699 10200.4
##       TPM1+I 96 -4716.049 9624.097 10146.24
##    TPM1+G(4) 96 -4715.284 9622.567 10144.71
##  TPM1+G(4)+I 97 -4715.163 9624.325 10151.91
##          K81 95 -4746.849 9683.699 10200.4
##        K81+I 96 -4716.049 9624.097 10146.24
##     K81+G(4) 96 -4715.284 9622.567 10144.71
##   K81+G(4)+I 97 -4715.163 9624.325 10151.91
##        TPM1u 98 -4707.129 9610.257 10143.28
##      TPM1u+I 99 -4676.591 9551.181 10089.64
##   TPM1u+G(4) 99 -4675.681 9549.363 10087.82
## TPM1u+G(4)+I 100 -4675.544 9551.089 10094.99
##         TPM2 95 -4745.383 9680.767 10197.47
##       TPM2+I 96 -4713.696 9619.393 10141.53
##    TPM2+G(4) 96 -4712.927 9617.854 10140
```

```
##  TPM2+G(4)+I 97 -4712.788 9619.576 10147.16
##        TPM2u 98 -4709.069 9614.139 10147.16
##      TPM2u+I 99 -4678.465 9554.93 10093.39
##   TPM2u+G(4) 99 -4677.59 9553.181 10091.64
## TPM2u+G(4)+I 100 -4677.452 9554.903 10098.8
##         TPM3 95 -4753.651 9697.302 10214
##       TPM3+I 96 -4722.831 9637.663 10159.8
##    TPM3+G(4) 96 -4722.057 9636.114 10158.26
##  TPM3+G(4)+I 97 -4721.933 9637.867 10165.45
##        TPM3u 98 -4713.871 9623.743 10156.76
##      TPM3u+I 99 -4683.232 9564.463 10102.92
##   TPM3u+G(4) 99 -4682.326 9562.652 10101.11
## TPM3u+G(4)+I 100 -4682.188 9564.375 10108.27
##        TIM1e 96 -4746.844 9685.688 10207.83
##      TIM1e+I 97 -4716.048 9626.095 10153.68
##   TIM1e+G(4) 97 -4715.283 9624.566 10152.15
## TIM1e+G(4)+I 98 -4715.162 9626.324 10159.34
##         TIM1 99 -4705.849 9609.698 10148.16
##       TIM1+I 100 -4675.189 9550.377 10094.27
##    TIM1+G(4) 100 -4674.418 9548.837 10092.73
##  TIM1+G(4)+I 101 -4674.312 9550.624 10099.96
##        TIM2e 96 -4745.378 9682.756 10204.9
##      TIM2e+I 97 -4713.696 9621.393 10148.97
##   TIM2e+G(4) 97 -4712.924 9619.849 10147.43
## TIM2e+G(4)+I 98 -4712.784 9621.568 10154.59
##         TIM2 99 -4707.792 9613.583 10152.04
##       TIM2+I 100 -4677.025 9554.051 10097.95
##    TIM2+G(4) 100 -4676.284 9552.568 10096.47
##  TIM2+G(4)+I 101 -4676.175 9554.351 10103.69
##        TIM3e 96 -4753.646 9699.292 10221.43
##      TIM3e+I 97 -4722.831 9639.661 10167.24
##   TIM3e+G(4) 97 -4722.057 9638.113 10165.69
## TIM3e+G(4)+I 98 -4721.932 9639.864 10172.88
##         TIM3 99 -4712.586 9623.172 10161.63
##       TIM3+I 100 -4681.811 9563.622 10107.52
##    TIM3+G(4) 100 -4681.045 9562.091 10105.99
##  TIM3+G(4)+I 101 -4680.938 9563.877 10113.21
##         TVMe 97 -4734.965 9663.93 10191.51
##       TVMe+I 98 -4703.335 9602.669 10135.69
##    TVMe+G(4) 98 -4702.548 9601.096 10134.11
##  TVMe+G(4)+I 99 -4702.409 9602.818 10141.28
##          TVM 100 -4697.901 9595.803 10139.7
##        TVM+I 101 -4667.304 9536.609 10085.94
##     TVM+G(4) 101 -4666.418 9534.835 10084.17
##   TVM+G(4)+I 102 -4666.282 9536.564 10091.34
##          SYM 98 -4734.959 9665.918 10198.94
##        SYM+I 99 -4703.334 9604.669 10143.13
##     SYM+G(4) 99 -4702.544 9603.089 10141.55
##   SYM+G(4)+I 100 -4702.404 9604.808 10148.71
##          GTR 101 -4696.628 9595.257 10144.59
##        GTR+I 102 -4665.858 9535.716 10090.49
##     GTR+G(4) 102 -4665.108 9534.216 10088.99
##   GTR+G(4)+I 103 -4665.004 9536.009 10096.22
```

```
write.table(mt, "ModelTestResults.txt", quote=FALSE, sep="\t",row.names=FALSE, col.names=TRUE)
```

Questions for Part I:

1. Which model had the HIGHEST log likelihood value? Briefly describe this model, its assumptions in terms of base frequencies and equal or unequal substitution rates, and any other parameters it may include. (5 pts)

```
max(mt$logLik)
```

```
## [1] -4665.004
```

```
mt[24, ]
```

```
##          Model  df    logLik      AIC        AICw    AICc        AICcw      BIC
## 24 TrN+G(4)+I 100 -4682.447 9564.895 6.74951e-08 9577.52 8.518027e-08 10108.79
```

- The highest log likelihood value is -4675.717, which corresponds to model GTR+G+I. GTR models are generalised time-reversible models, and have unequal base frequencies and six substitution rates. Our model has:
  - 103 degrees of freedom
  - A log likelihood value (measure of goodness of fit) of -4675.717. This is the highest log likelihood value, implying it is the best model if based on logLik alone.
  - AIC (estimator of prediction error) of 9557.435. This is the second lowest AIC value, meaning it is the second most parsimonious model in the data set.
  - BIC (model scoring method). This is the fourth lowest BIC value, meaning it is not the strongest, but far from the weakest, model.

2. The Bayesian Information Criterion (BIC) is a method for comparing models that includes penalties for more complex models. With this method, the best one is the model with the LOWEST BIC score. In your test, which model is this? Is it different from the model with the highest likelihood? Describe this model in terms of its assumptions. (5 pts)

```
min(mt$BIC)
```

```
## [1] 10084.17
```

```
mt[23, ]
```

```
##       Model df    logLik      AIC         AICw     AICc        AICcw      BIC
## 23 TrN+G(4) 99 -4682.564 9563.127 1.633217e-07 9575.494 2.344639e-07 10101.59
```

- The highest log likelihood value is 10110.66, which corresponds to model GTR+G.This model has:

- 102 degrees of freedom
- A log likelihood value (measure of goodness of fit) of -4675.945. This is the second highest log likelihood value, implying it is the second best model if based on logLik alone.
- AIC (estimator of prediction error) of 9555.889. This is the lowest AIC value, meaning it is the most parsimonious model in the data set.
- BIC (model scoring method). This is the lowest BIC value, meaning it is the strongest model.

Calculate Genetic Distance Under the Best Model

For this next step, we want to convert our phyDat object, which is the variable named forPhang to a DNAbin object:

```
dna = as.DNAbin(forPhang)
```

Now, calculate the genetic distance using the dist.dna function. This function allows you to specify the nucleotide substitution model you think best fits your data. Ideally, we want to use the model that scored the best in terms of BIC score in the modelTest from Part I. To see all of the model options for dna.dist, go to the help page:

```
?dist.dna()
```

As you read through the model options, you will notice that your best scoring model is not listed, although there are several models that seem like they have similar assumptions. I recommend using the TN93 model, with the gamma parameter set to TRUE. Read the description of this model in the dist.dna() help page to see why I made this selection.

```
D = dist.dna(dna, model="TN93", gamma=TRUE)
```

Create both UPGMA and Neighbor-Joining Trees

To create the two types of trees, the commands are very straightforward:

```
t.upgma = upgma(D)
t.nj = nj(D)
```

After creating the trees, you can also manually define an outgroup, to make "rooted" trees. Here is how I do that with the UPGMA tree:

```
t.upgma.root = root(phy=t.upgma, outgroup='1993a')
```

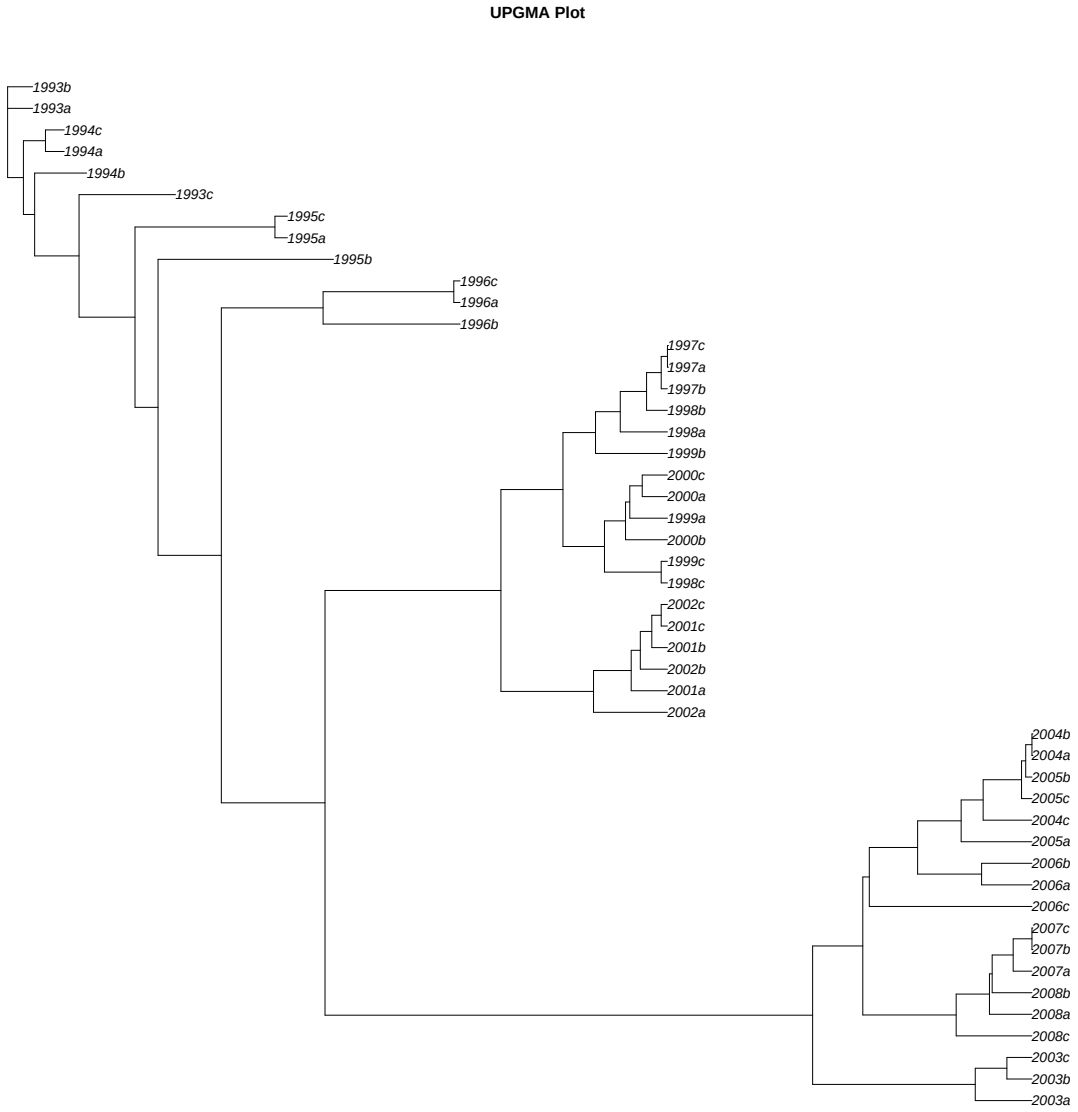
Use the same function to get a rooted Neighbor-Joining tree.

```
t.nj.root = root(phy=t.nj, outgroup='1993a')
```

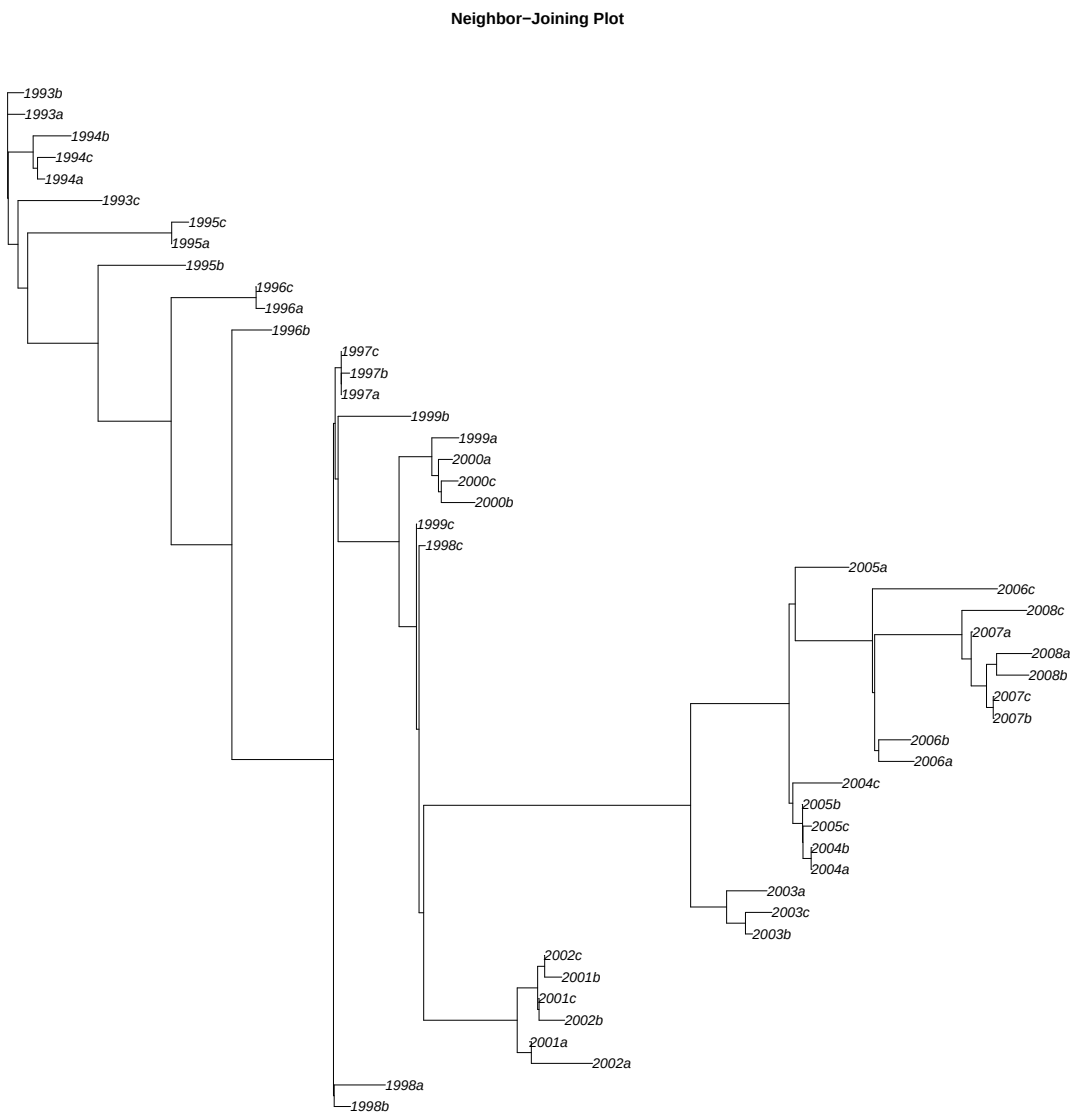
Plot the Rooted Trees to Compare each Method

You can use a simple plot function to plot each tree individually, and then save each plot as a pdf manually. Note that this is NOT the plot command which will require adegenet, so if you could not get that package installed your plot will still work.

```
UPGMA_TREE = plot(t.upgma.root, main="UPGMA Plot")
```



```
NJ_TREE = plot(t.nj.root, main="Neighbor-Joining Plot")
```



Note that the main option lets us define a plot title. You will also see that I am using the option cex=0.75, which shrinks the tip label font size, making them more readable with the number of sequences we have. Feel free to adjust this parameter to your liking.

Questions for Part II:

3. Describe any differences in clades that you observe between the UPGMA and NJ trees (be sure to include a plot of each tree in your answer!). In particular, are viruses from the same year always in the same clade in each tree? Do viruses collected in later years branch off from the viruses of previous years (like we would expect)? 5 pts

- One of the most obvious difference between the trees is the grouping of clades. The UPGMA tree has a very severe grouping of 1997-2002 and another severe grouping of 2003-2008. The neighbor-joining tree is more sporadic. 2001-2002 is closely knit, 2003 is alone, and then 2004-2005 then 2007-2008 for the most part are closely knit groups.
- Viruses from the same year are not always in the same clade in each tree - for example, 1993c is more closely related to 1994b/c/a than 1992b/a.
- Outside of the pattern noted above, for the most part, viruses collected in later years branch off from the viruses of previous years. On the left of the trees is the earliest year, 1993, and the furthest on the right of the tree is the most recent year, 2008.

Get a Parsimony Score for each Tree

To figure out which of our trees might be a better representation of influenza evolution, we can calculate a parsimony score for each of our trees. To do this, you can use the parsimony() function:

```
parsimony(t.upgma.root, forPhang)
```

```
## [1] 380
```

```
parsimony(t.nj.root, forPhang)
```

```
## [1] 340
```

Questions for Part III:

4. What are your parsimony scores for your UPGMA and Neighbor-Joining trees? Given that a lower score is better, which of your trees is the most parsimonious? 2 pts

- The parsimony score for the UPGMA tree is 380, while the NJ tree has a parsimony score of 340. because $340 < 380$, the Neighbor-Joining tree is the most parsimonious.

5. Based on what you know about the definition of parsimony, what does it mean when we say that one tree is more parsimonious than another? i.e. How should we interpret this result? 3 pts

- When one tree is more parsimonious than another, it means that one tree has a smaller sum of the smallest number of substitutions needed for each site. Basically, the most parsimonious tree requires the fewest nucleotide substitutions to explain the data.

Perform Bootstrapping on the Most Parsimonious Tree

To see how much confidence we should have in the clades shown in our most parsimonious tree, let's perform 100 bootstrapping replicates of the same tree construction method. The function for this is boot.phylo().

This command will look different depending on which of your trees is the most parsimonious. If it is the UPGMA tree you want to test, you'll have a command that looks like this:

```
myBoots = boot.phylo(t.upgma.root, dna, function(x) root(upgma(dist.dna(x, model="TN93", gamma=TRUE)),1
```

```
## Running bootstraps:       100 / 100
## Calculating bootstrap values... done.
```

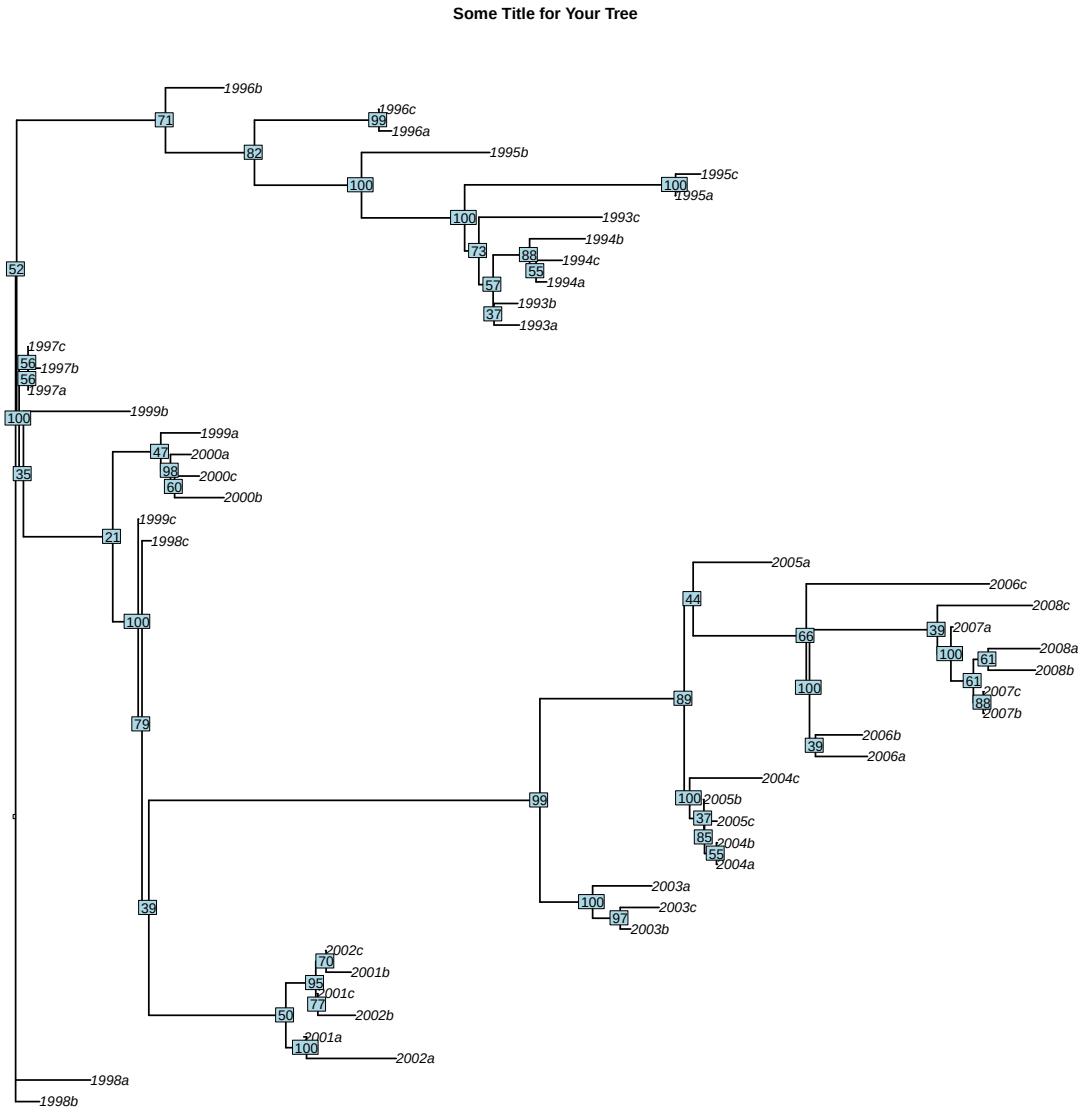
If your most parsimonious tree is the neighbor-joining tree, then your command will look like this:

```
myBoots = boot.phylo(t.nj.root, dna, function(x) root(nj(dist.dna(x, model="TN93", gamma=TRUE)),1))
```

```
## Running bootstraps:       100 / 100
## Calculating bootstrap values... done.
```

Once you get your bootstrap values, you can plot them on the tree with the functions below:

```
plot(t.nj, cex=1, edge.width=2, main="Some Title for Your Tree")
nodelabels(myBoots, cex=1)
```



Questions for Part IV:

6. Are there any nodes in your tree that seem to have very weak bootstrap support (i.e. less than 50%)? How many of these weakly supported nodes do you see? Which node has the weakest support, and what clade is this node at the base of? If it is hard to see exactly where the nodes are with the bootstrap labels on them, you might also want to refer back to your original plot without the labels to help you. 5 pts

- I see 9 total weakly supported nodes. The node with the weakest support has a value of 26, with clades 1993b and 1993a on its base.

Remove Uncertain Nodes and Optimize the Tree for Parsimony

To make a tree where we're only showing nodes that are more well-supported, we can find the nodes with <50% bootstrap support and collapse them. Even though this removes some information from our tree, it leaves us with a tree where we are more confident in all of the shown results.

```
N <- length(t.nj$tip.label)
toCollapse <- match(which(myBoots<50)+N, t.nj$edge[,2])
t.nj$edge.length[toCollapse] = 0
new.tree = di2multi(t.nj, tol=0.00001)
```

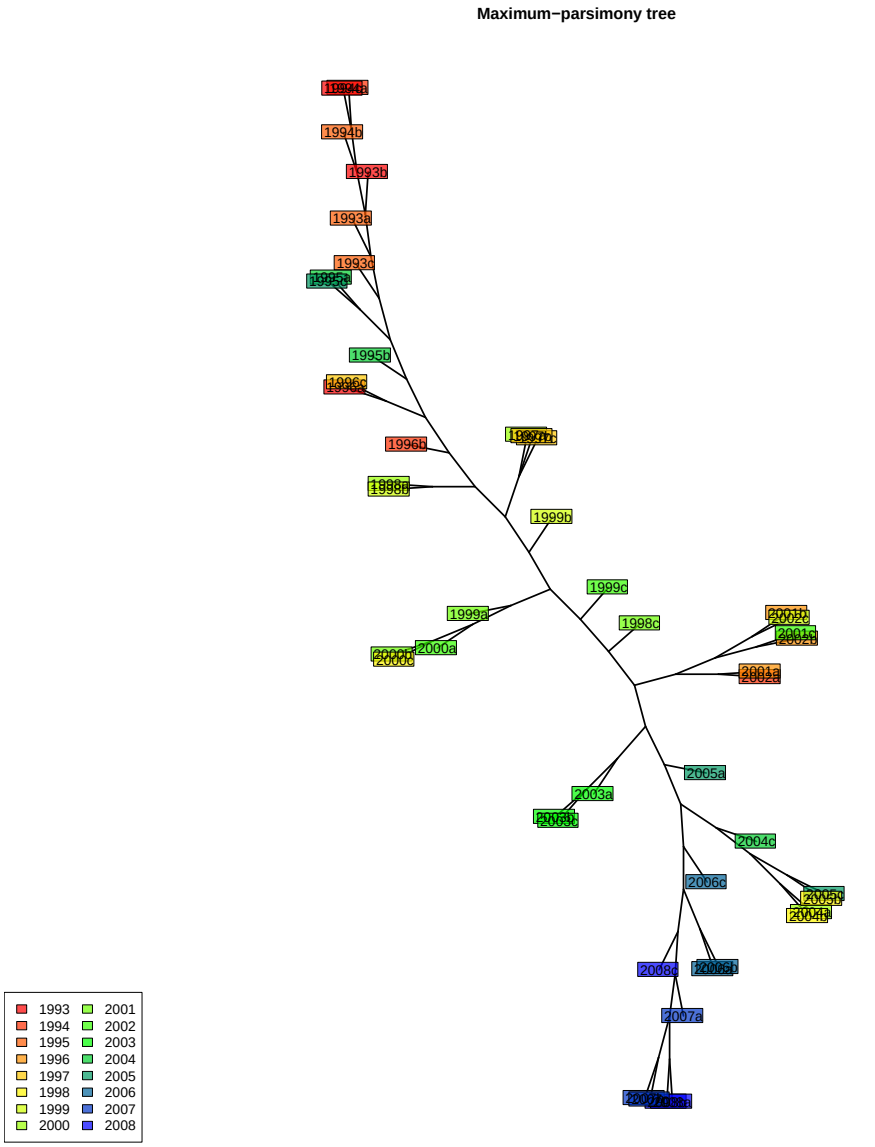
Now, let's make sure our reconstructed tree is the most parsimonious tree possible, with the function optim.parsimony():

```
tre.pars = optim.parsimony(new.tree, forPhang)
```

```
## Final p-score 340 after  5 nni operations
```

Finally, let's plot our most parsimonious tree. For this plot, we're going to 1) plot an unrooted tree, since this version actually seems to make the trends more clear, and 2) add colored labels to our plots to make it easier to see where different years group together.

```
plot(tre.pars, type="unr", show.tip=FALSE, edge.width=2)
title("Maximum-parsimony tree")
year = as.numeric(gsub("[a-c]", "", names(forPhang)))
myPal <- colorRampPalette(c("red","yellow","green","blue"))
tiplabels(tre.pars$tip.label, bg=transp(num2col(year, col.pal=myPal),.7), cex=1, fg="transparent")
temp <- pretty(1993:2008, 16)
legend("bottomleft", fill=transp(num2col(temp, col.pal=myPal),.7),leg=temp, ncol=2)
```



Questions for Part V:

7. Show the plot for your final tree. Even though there are still some uncertain relationships, can you now see any kind of trend with respect to the years the viruses were sampled and where they appear on the tree? Describe your new tree and the trends that you see. 5 pts

- I definitely see a trend of red on the far left (earliest years) to the blue on the far right (latest years). While this is similar to the trend I saw earlier, it is more evident now. In the middle are the green, where the years in the middle lie.