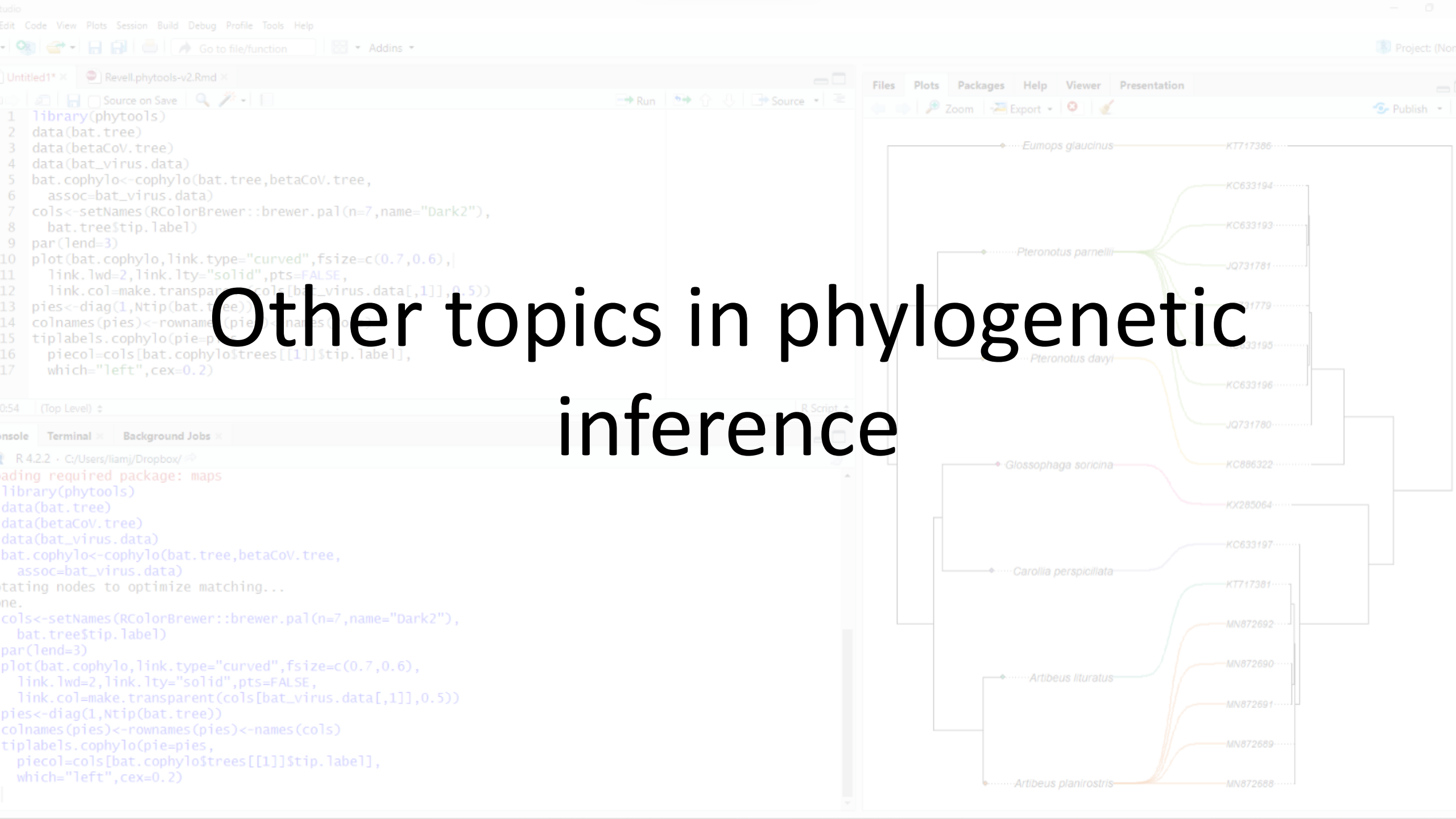
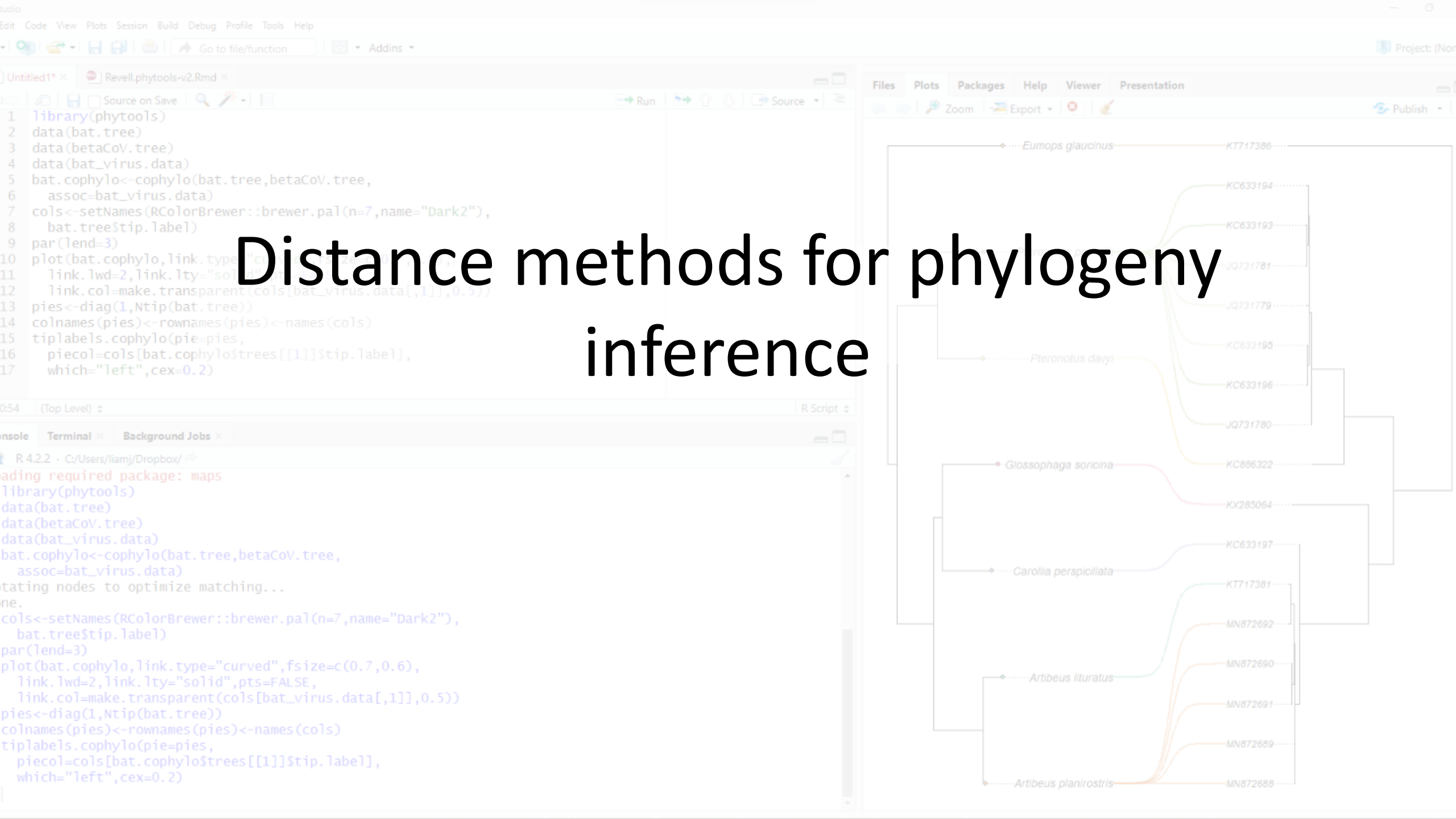
Other topics in phylogenetic inference



Topics

1. Distance matrix methods
2. Bayesian phylogenetic inference
3. The gene-tree species-tree problem

Distance methods for phylogeny inference



Agenda

- Distance vs. character methods in phylogeny inference.
- How do we compute the distance between sequences?
- Optimality criteria methods for estimating a phylogeny from distances:
  - Least squares.
  - The minimum evolution method.
- Algorithm-based methods for estimating a phylogeny from distances:
  - UPGMA.
  - Neighbor-joining.

Character methods vs. distance methods

- So far we have focused on parsimony & maximum likelihood, “character methods.”
- A major family of methods for phylogenetic inference are called “distance methods” or “distance matrix methods.”
- These methods first involve the calculation of an *n* x *n* matrix (for *n* species) containing the *distance* between each pair of species in our study.
- One would think that distilling all the information about the similarity and dissimilarity of two species down to a number would substantially reduce our power for phylogenetic inference.
- Remarkably – studies have shown that this is not the case.

Calculating the distances between sequences

To take into account superimposed substitutions, we need a model by which substitutions accrue.

The simplest such model is called the *Jukes-Cantor model* (Jukes & Cantor 1969).

Under Jukes-Cantor, nucleotide substitution occurs according to a *Poisson process*.

Under this process, the probability of given site changing state over the interval of time *t* given substitution rate *u* is given by:

$$D_S = \frac{3}{4}\left(1 - e^{-\frac{4}{3}ut}\right)$$

Calculating the distances between sequences

We could just count the number of differences between two sequences.

What's wrong with this?

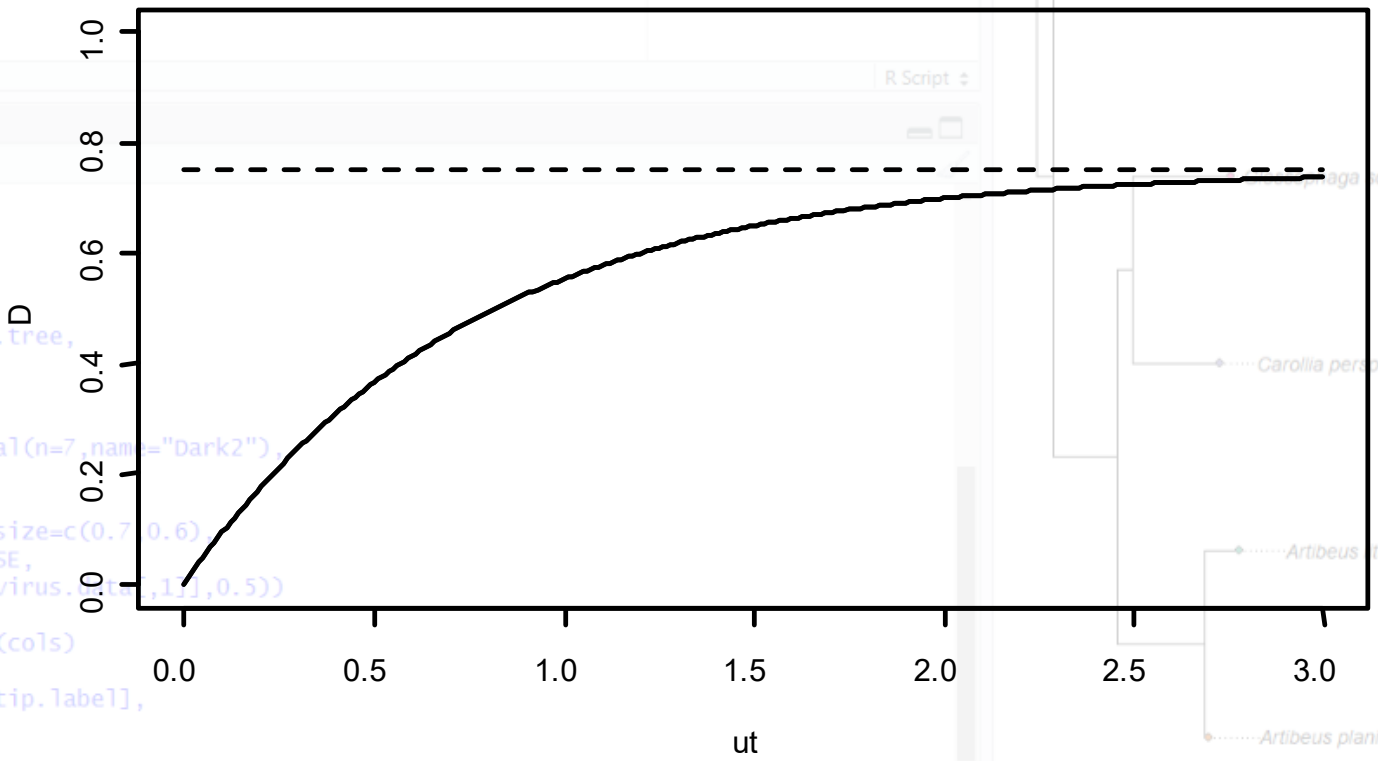
ACGCCCTATG	0
ACTCCCTATG	1
TCTCCCTATG	2
TCTCGCTATG	3
TCTCGCTTTG	4
TCACGCTTTG	4

We have to take into account the possibility of *superimposed substitution*, i.e., substitutions that take place in the same location as prior changes.

Calculating the distances between sequences

We can map the function given by the Jukes-Cantor equation

$$D_S = \frac{3}{4}\left(1 - e^{-\frac{4}{3}ut}\right)$$



Calculating the distances between sequences

Similarly, by solving for *ut* we can estimate the *actual divergence* given an amount of *observed divergence*.

$$D = -\frac{3}{4}\ln\left(1 - \frac{4}{3}D_S\right)$$

There are many other models of DNA or AA substitution that we can use (this is merely the simplest) that take into account realistic attributes of the substitution process.

Digression on optimality criteria vs. clustering algorithm phylogeny inference methods

- Something we have not yet discussed is the distinction between phylogeny methods that *minimize* (or maximize) a criterion (called the **optimality criterion**) and methods that compute a phylogeny from an algorithm.
- *[Note that if we had an analytic solution for the criterion as a function of the tree we might be able to use an algorithm to minimize (or maximize) the criterion – but this is generally not the case for phylogenetic topology (although it can be for branch lengths, as we will see).]
- **Distance matrix methods are of both types** (although the most popular methods are algorithm-based).

Least squares method

The *least squares method* is an optimality criterion distance method that seeks to minimize the (weighted or unweighted) sum of squared differences between the *observed distances* and the *patristic distances* on our hypothesized tree.

We can write this as:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (D_{ij} - d_{ij})^2$$

Here, Q is the quantity we're trying to minimize; the D_{ij} are our *observed distances* (i.e., our computed differences between sequences under the model); and the d_{ij} are our *predicted distances* (i.e., the patristic distances on our hypothesized tree.

The weights, w_{ij} come into play only for *weighted least squares* – so we'll ignore them for now.

Unweighted least squares method

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - d_{ij})^2$$

Given **D** and **d** we can compute *Q* easily. *Q* is then the *residual sum of squares* – which is the quantity that we want to minimize.

Even better, given **D** and a phylogeny, we can find **d** that minimizes *Q* using standard least squares estimation procedures.

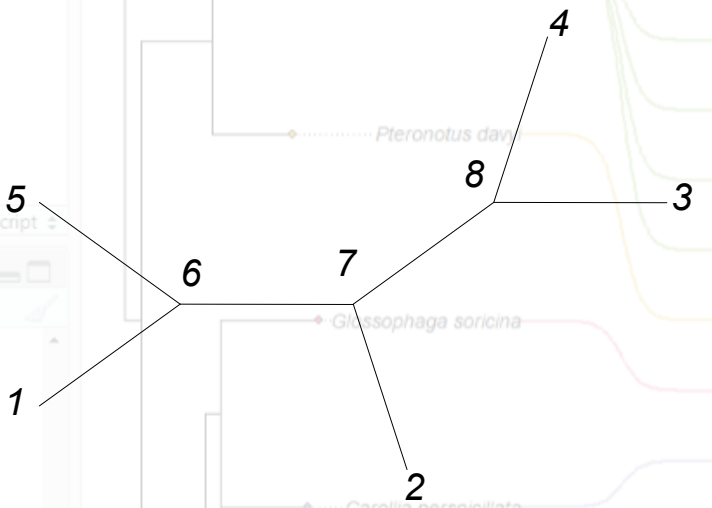
How do we do this?

Unweighted least squares method

First, we need to construct a *design matrix*.

The design matrix contains $n(n\text{-}1)/2$ rows [for the $n(n\text{-}1)/2$ distances among species] and m columns for m edges, where $m=2n\text{-}3$ for a fully bifurcating tree.

We then populate the matrix with 1s and 0s depending on whether each edge is on the path separating each pair of species.

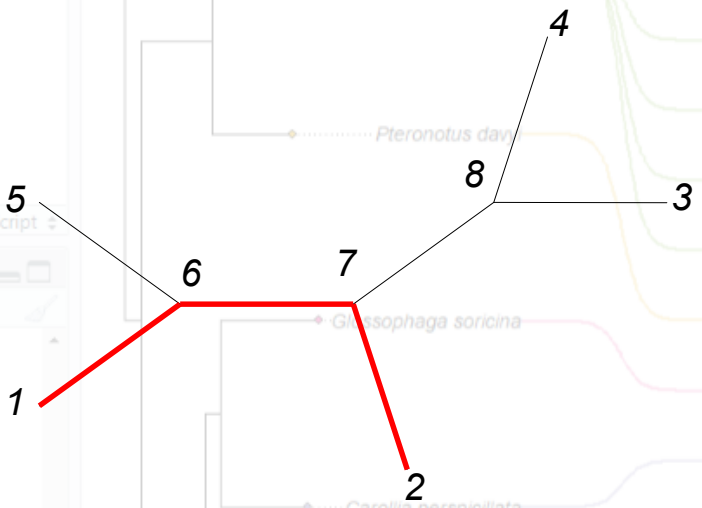


Unweighted least squares method

Let's do this for the tree at right:

$\mathbf{X} =$

	6,1	6,7	7,2	7,8	8,3	8,4	6,5
1,2							
1,3							
1,4							
1,5							
2,3							
2,4							
2,5							
3,4							
3,5							
4,5							

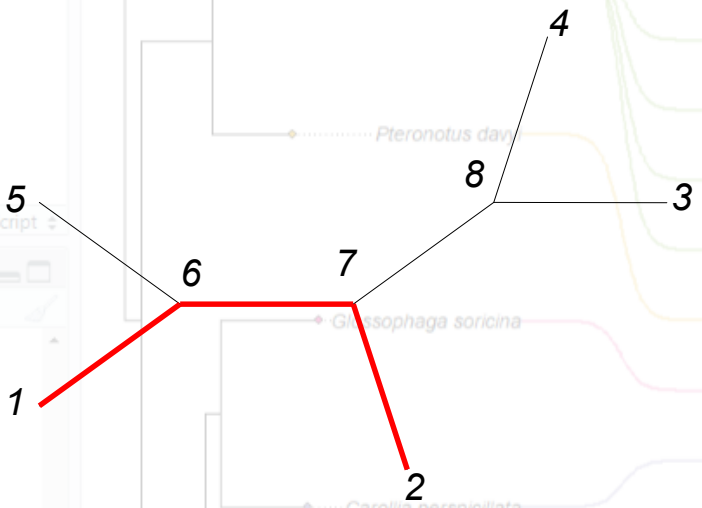


Unweighted least squares method

Let's do this for the tree at right:

$\mathbf{X} =$

	6,1	6,7	7,2	7,8	8,3	8,4	6,5
1,2	1	1	1	0	0	0	0
1,3							
1,4							
1,5							
2,3							
2,4							
2,5							
3,4							
3,5							
4,5							

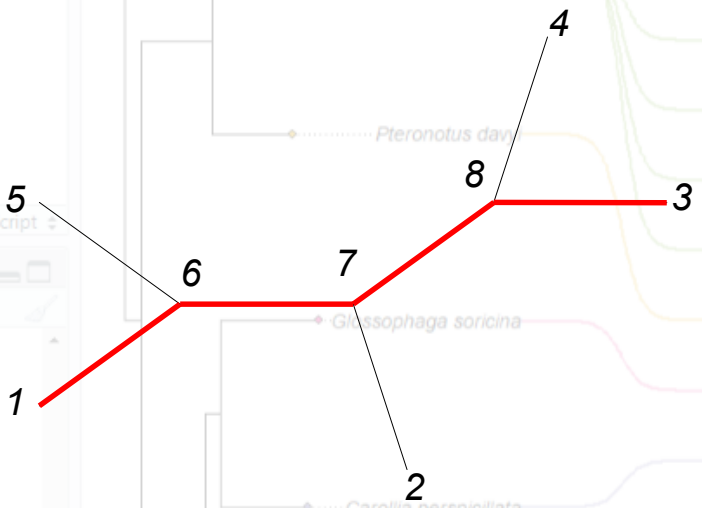


Unweighted least squares method

Let's do this for the tree at right:

$\mathbf{X} =$

	6,1	6,7	7,2	7,8	8,3	8,4	6,5
1,2	1	1	1	0	0	0	0
1,3							
1,4							
1,5							
2,3							
2,4							
2,5							
3,4							
3,5							
4,5							

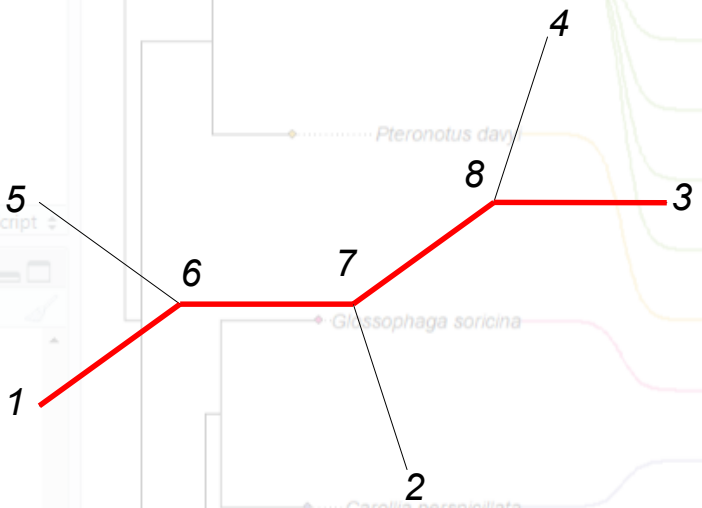


Unweighted least squares method

Let's do this for the tree at right:

$\mathbf{X} =$

	6,1	6,7	7,2	7,8	8,3	8,4	6,5
1,2	1	1	1	0	0	0	0
1,3	1	1	0	1	1	0	0
1,4							
1,5							
2,3							
2,4							
2,5							
3,4							
3,5							
4,5							

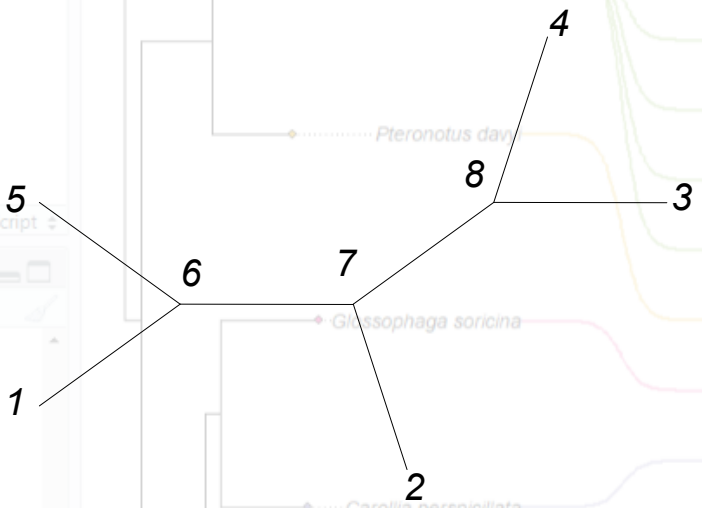


Unweighted least squares method

Let's do this for the tree at right:

$\mathbf{X} =$

	6,1	6,7	7,2	7,8	8,3	8,4	6,5
1,2	1	1	1	0	0	0	0
1,3	1	1	0	1	1	0	0
1,4	1	1	0	1	0	1	0
1,5	1	0	0	0	0	0	1
2,3	0	0	1	1	1	0	0
2,4	0	0	1	1	0	1	0
2,5	0	1	1	0	0	0	1
3,4	0	0	0	0	1	1	0
3,5	0	1	0	1	1	0	1
4,5	0	1	0	1	0	1	1



Unweighted least squares method

Our binary matrix is a *design matrix* for the model: $D = Xd + e$ which we can easily solve using standard least-squares estimating equations.

Now, we can solve for the branch lengths using standard estimating procedures.

Take the following matrix:

	1	2	3	4	5
1	0.000	2.142	2.616	2.634	1.499
2	2.142	0.000	2.393	2.411	2.146
3	2.616	2.393	0.000	1.363	2.619
4	2.634	2.411	1.363	0.000	2.637
5	1.499	2.146	2.619	2.637	0.000

Unweighted least squares method

We first calculate the lower triangle of this (to correspond with the rows of our design matrix):

```
D=[2.142 2.616 2.634 1.499 2.393 2.411 2.146
1.363 2.619 2.637]
```

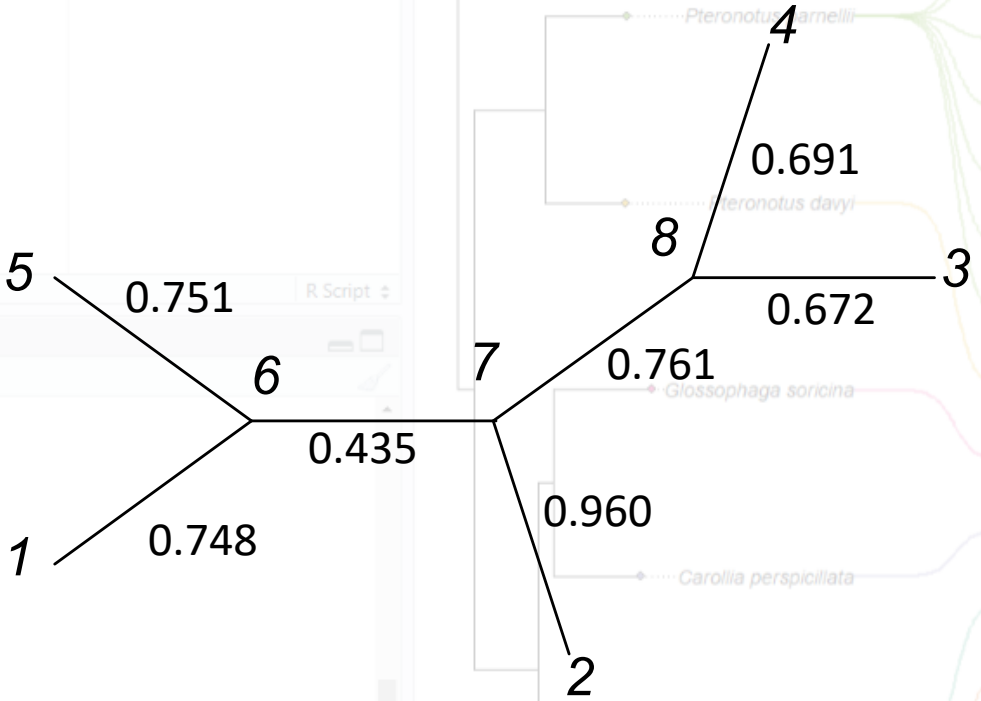
Then, we use the least squares estimating equation to get the least squares branch lengths **(v):**

$$\mathbf{v} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{D}$$

Unweighted least squares method

Next, we can map these distances on the tree:

```
6,1 0.748
6,7 0.435
7,2 0.960
7,8 0.761
8,3 0.672
8,4 0.691
6,5 0.751
```



Unweighted least squares method

Finally, we calculate *Q* or residual sum of squares:

This is done as:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - d_{ij})^2$$

or in matrix notation, as:

$$Q = (\mathbf{D} - \mathbf{d})'(\mathbf{D} - \mathbf{d})$$

Unweighted least squares method

To find the tree that minimizes *Q* (our optimality criterion) we can use our typical procedures for searching treespace & each time recompute the least-squares branch lengths and the value of our criterion.

Least squares method

The following is an alternative, more compact explanation of LS.

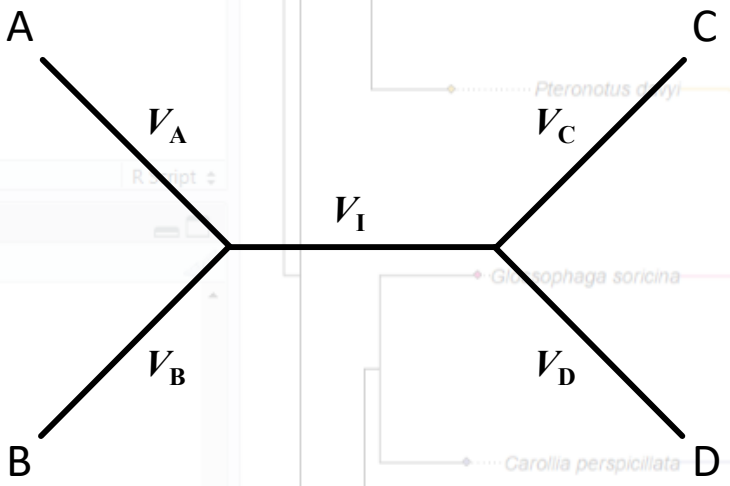
The Least Squares (LS) method seeks to minimize the sum of squared differences between the *observed distances* (that is the distances we have calculated between our DNA sequences) and the *hypothesized distances* on our tree (that is the sum of the branch lengths separating species.

Least squares method

So, for instance, taking the following set of distances between four species:

$$D_{AB}$$
$$D_{AC}$$
$$D_{AD}$$
$$D_{BC}$$
$$D_{BD}$$
$$D_{CD}$$

And a hypothesis for the phylogeny:



With implied distances:

$$d_{AB} = v_A + v_B$$
$$d_{AC} = v_A + v_I + v_C$$
$$d_{AD} = v_A + v_I + v_D$$
$$d_{BC} = v_B + v_I + v_C$$
$$d_{BD} = v_B + v_I + v_D$$
$$d_{CD} = v_C + v_D$$

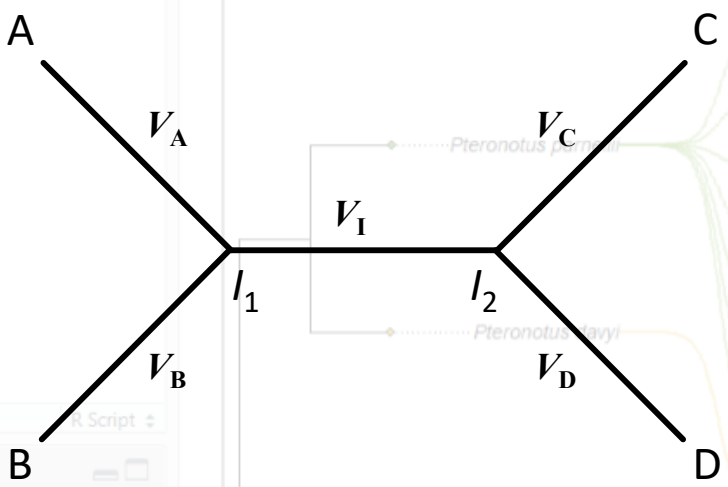
We need to find the tree and set of v_i that minimizes:

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} (D_{ij} - d_{ij})^2$$

Least squares method

This is the same as writing:

$$D_{AB} = 1v_A + 1v_B + 0v_I + 0v_C + 0v_D + e_{AB}$$
$$D_{AC} = 1v_A + 0v_B + 1v_I + 1v_C + 0v_D + e_{AC}$$
$$D_{AD} = 1v_A + 0v_B + 1v_I + 0v_C + 1v_D + e_{AD}$$
$$D_{BC} = 0v_A + 1v_B + 1v_I + 1v_C + 0v_D + e_{BC}$$
$$D_{BD} = 0v_A + 1v_B + 1v_I + 0v_C + 1v_D + e_{BD}$$
$$D_{CD} = 0v_A + 0v_B + 0v_I + 1v_C + 1v_D + e_{CD}$$



Which is equivalent to:

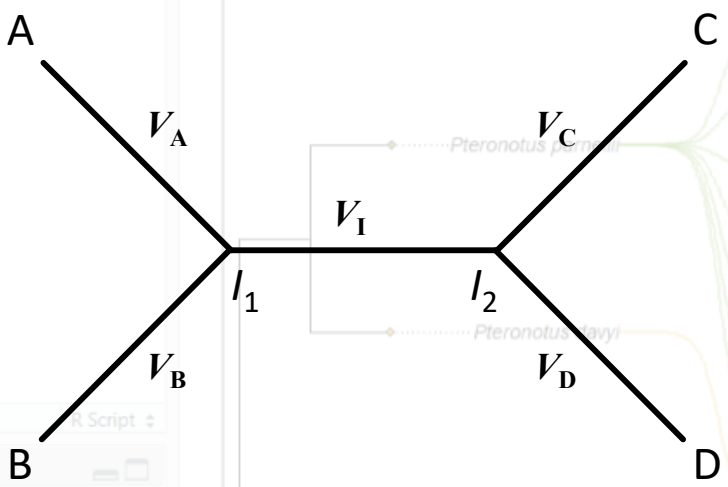
$$\begin{bmatrix} D_{AB} \\ D_{AC} \\ D_{AD} \\ D_{BC} \\ D_{BD} \\ D_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_I \\ v_C \\ v_D \end{bmatrix} + \begin{bmatrix} e_{AB} \\ e_{AC} \\ e_{AD} \\ e_{BC} \\ e_{BD} \\ e_{CD} \end{bmatrix}$$

	I_1,A	I_1,B	I_1, I_2	I_1,C	I_1,D
A,B					
A,C					
A,D					
B,D					
B,C					
C,D					

Least squares method

This is the same as writing:

$$D_{AB} = 1v_A + 1v_B + 0v_I + 0v_C + 0v_D + e_{AB}$$
$$D_{AC} = 1v_A + 0v_B + 1v_I + 1v_C + 0v_D + e_{AC}$$
$$D_{AD} = 1v_A + 0v_B + 1v_I + 0v_C + 1v_D + e_{AD}$$
$$D_{BC} = 0v_A + 1v_B + 1v_I + 1v_C + 0v_D + e_{BC}$$
$$D_{BD} = 0v_A + 1v_B + 1v_I + 0v_C + 1v_D + e_{BD}$$
$$D_{CD} = 0v_A + 0v_B + 0v_I + 1v_C + 1v_D + e_{CD}$$



Which is equivalent to:

$$\begin{bmatrix} D_{AB} \\ D_{AC} \\ D_{AD} \\ D_{BC} \\ D_{BD} \\ D_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_I \\ v_C \\ v_D \end{bmatrix} + \begin{bmatrix} e_{AB} \\ e_{AC} \\ e_{AD} \\ e_{BC} \\ e_{BD} \\ e_{CD} \end{bmatrix}$$

	I_1,A	I_1,B	I_1, I_2	I_1,C	I_1,D
A,B	1	1	0	0	0
A,C	1	0	1	1	0
A,D	1	0	1	0	1
B,D	0	1	1	1	0
B,C	0	1	1	0	1
C,D	0	0	0	1	1

Least squares method

So when we have this:

$$\begin{bmatrix} D_{AB} \\ D_{AC} \\ D_{AD} \\ D_{BC} \\ D_{BD} \\ D_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_I \\ v_C \\ v_D \end{bmatrix} + \begin{bmatrix} e_{AB} \\ e_{AC} \\ e_{AD} \\ e_{BC} \\ e_{BD} \\ e_{CD} \end{bmatrix}$$

Make some substitutions:

Least squares method

So when we have this:

$$\begin{bmatrix} D_{AB} \\ D_{AC} \\ D_{AD} \\ D_{BC} \\ D_{BD} \\ D_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} v_A \\ v_B \\ v_I \\ v_C \\ v_D \end{bmatrix} + \begin{bmatrix} e_{AB} \\ e_{AC} \\ e_{AD} \\ e_{BC} \\ e_{BD} \\ e_{CD} \end{bmatrix}$$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

Least squares method

Now, to minimize the sum of squares of **e**:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

We can use the least squares estimating equation:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

(for a proof if desired, see Lynch & Walsh 1998; pp. 200-202)

Least squares method

Back in our original terms:

$$\hat{\boldsymbol{\beta}} = \begin{bmatrix} \hat{\beta}_A \\ \hat{\beta}_B \\ \hat{\beta}_I \\ \hat{\beta}_C \\ \hat{\beta}_D \end{bmatrix}$$

Least squares method

Back in our original terms:

$$\hat{\mathbf{v}} = \begin{bmatrix} \hat{v}_A \\ \hat{v}_B \\ \hat{v}_I \\ \hat{v}_C \\ \hat{v}_D \end{bmatrix}$$

then: $\hat{\mathbf{d}} = \mathbf{X}\hat{\mathbf{v}}$

or, equivalently:

$$\begin{bmatrix} \hat{d}_{AB} \\ \hat{d}_{AC} \\ \hat{d}_{AD} \\ \hat{d}_{BC} \\ \hat{d}_{BD} \\ \hat{d}_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{v}_A \\ \hat{v}_B \\ \hat{v}_I \\ \hat{v}_C \\ \hat{v}_D \end{bmatrix}$$

Least squares method

Now let's consider this equation:

$$\begin{bmatrix} \hat{d}_{AB} \\ \hat{d}_{AC} \\ \hat{d}_{AD} \\ \hat{d}_{BC} \\ \hat{d}_{BD} \\ \hat{d}_{CD} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{v}_A \\ \hat{v}_B \\ \hat{v}_I \\ \hat{v}_C \\ \hat{v}_D \end{bmatrix}$$

Which, remember, is the same as:

$$\hat{d}_{AB} = \hat{v}_A + \hat{v}_B$$

$$\hat{d}_{AC} = \hat{v}_A + \hat{v}_I + \hat{v}_C$$

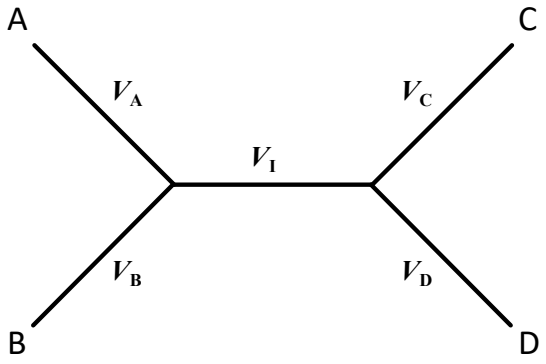
$$\hat{d}_{AD} = \hat{v}_A + \hat{v}_I + \hat{v}_D$$

$$\hat{d}_{BC} = \hat{v}_B + \hat{v}_I + \hat{v}_C$$

$$\hat{d}_{BD} = \hat{v}_B + \hat{v}_I + \hat{v}_D$$

$$\hat{d}_{CD} = \hat{v}_C + \hat{v}_D$$

Compare to the original tree:



Least squares method

Finally, we can calculating the residual sum of squares as follows:

$$Q = (\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})'(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})$$

Equivalently:

$$Q = (\mathbf{D} - \mathbf{X}\hat{\mathbf{v}})'(\mathbf{D} - \mathbf{X}\hat{\mathbf{v}})$$

Or, equivalently:

$$Q = (\mathbf{D} - \hat{\mathbf{d}})'(\mathbf{D} - \hat{\mathbf{d}})$$

Or, equivalently:

$$Q = \sum_{i=1}^{n}\sum_{j=1}^{n}(D_{ij} - d_{ij})^2$$

(remember, this is the quantity we said we wanted to minimize in the beginning!)

Minimum evolution method

Minimum evolution is an optimality criterion distance method.

According to the minimum evolution method, we *minimize the summed branch lengths of the tree*.

Why not just set all the branch lengths to zero?

For any tree, we compute the branch lengths using weighted or unweighted least squares, and we choose the tree with the minimum summed length.

UPGMA and Neighbor-Joining (NJ)

UPGMA and NJ differ from the previous distance methods we have been covering in the class today.

This is because they represent *clustering algorithms* rather than *optimality criterion methods*. Thus, we will learn two different classes of distance methods today.

Clustering algorithms apply a set of rules to a distance matrix to obtain a tree that groups like members. This differs fundamentally from the procedures we have discussed so far.

Clustering algorithms are generally much faster, because they do not require that calculations be conducted across many trees (as is typical of optimality criterion methods). However, they come with disadvantages (which depend on the method).

UPGMA

UPGMA, which stands for *Unweighted Pair Group Method with Arithmetic mean*, is the simplest of all clustering methods.

Rules:

1. Find species *i* and *j* with the smallest distance.
2. Create a new group (*ij*). Join *i* and *j* on the tree, and assign branch lengths $D_{ij}/2$.
3. Compute the distance from (*ij*) to all other "species" as follows:

$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

**this is just a weighted mean where the weights are proportional to the diversity of each subtree.

4. Delete the columns and rows for *i* and *j* and create *new* columns and rows for the group (*ij*).
5. If there is only one item left in the matrix, stop – otherwise return to step 1.

UPGMA

Consider the following matrix:

	A	B	C	D	E
A	0.000	1.282	1.282	1.282	0.840
B	1.282	0.000	0.576	0.576	1.282
C	1.282	0.576	0.000	0.256	1.282
D	1.282	0.576	0.256	0.000	1.282
E	0.840	1.282	1.282	1.282	0.000

UPGMA

Rules:

1. Find species *i* and *j* with the smallest distance.
2. Create a new group (*ij*). Join *i* and *j* on the tree, and assign branch lengths $D_{ij}/2$.
3. Compute the distance from (*ij*) to all other "species" as follows:

$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

4. Delete the columns and rows for *i* and *j* and create *new* columns and rows for the group (*ij*).
5. If there is only one item left in the matrix, stop – otherwise return to step 1.

UPGMA

Consider the following matrix:

	A	B	C	D	E
A	0.000	1.282	1.282	1.282	0.840
B	1.282	0.000	0.576	0.576	1.282
C	1.282	0.576	0.000	0.256	1.282
D	1.282	0.576	0.256	0.000	1.282
E	0.840	1.282	1.282	1.282	0.000

UPGMA

Rules:

1. Find species *i* and *j* with the smallest distance.
2. Create a new group (*ij*). Join *i* and *j* on the tree, and assign branch lengths $D_{ij}/2$.
3. Compute the distance from (*ij*) to all other "species" as follows:

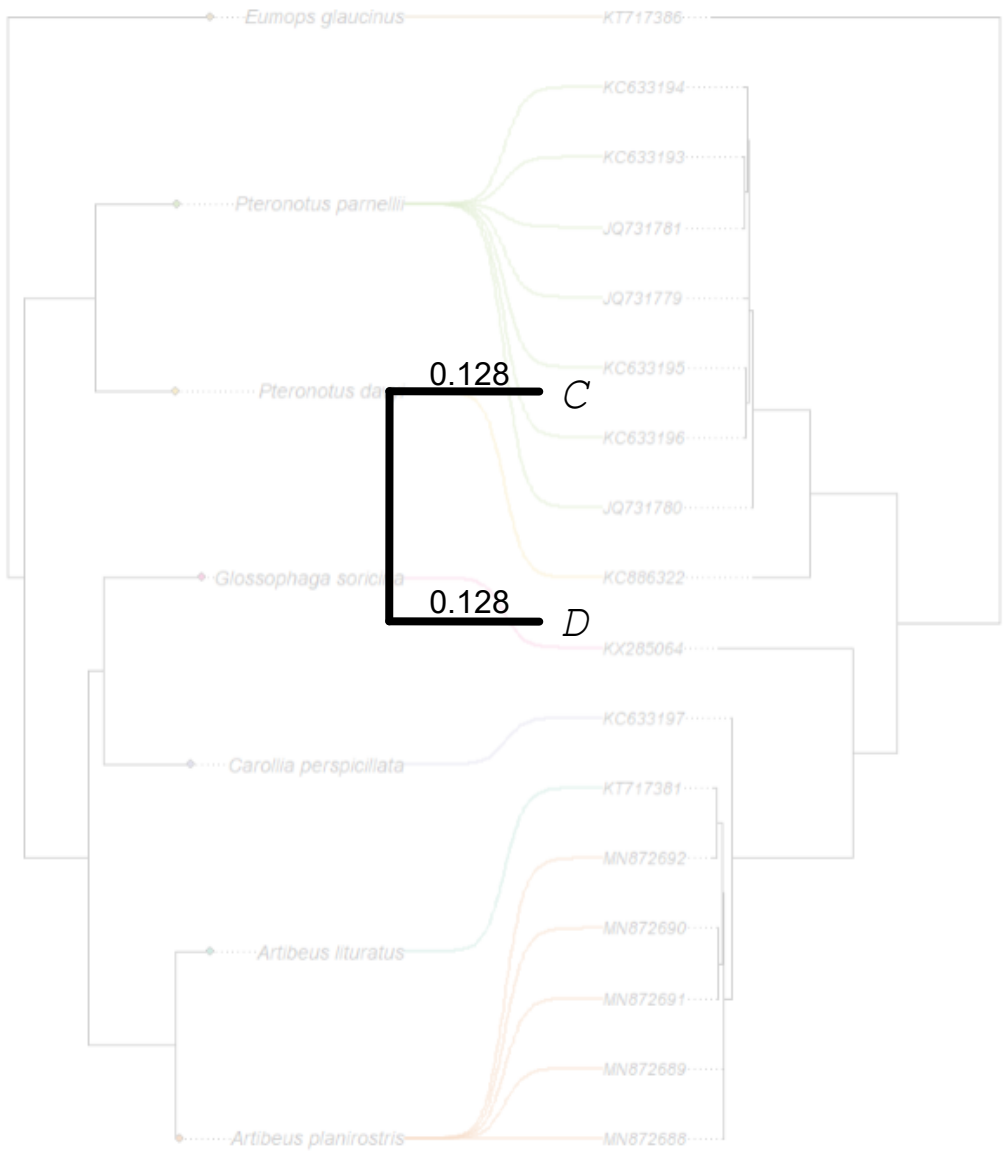
$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

4. Delete the columns and rows for *i* and *j* and create *new* columns and rows for the group (*ij*).
5. If there is only one item left in the matrix, stop – otherwise return to step 1.

UPGMA

Consider the following matrix:

	A	B	C	D	E
A	0.000	1.282	1.282	1.282	0.840
B	1.282	0.000	0.576	0.576	1.282
C	1.282	0.576	0.000	0.256	1.282
D	1.282	0.576	0.256	0.000	1.282
E	0.840	1.282	1.282	1.282	0.000



UPGMA

Rules:

1. Find species *i* and *j* with the smallest distance.
2. Create a new group (*ij*). Join *i* and *j* on the tree, and assign branch lengths $D_{ij}/2$.
3. Compute the distance from (*ij*) to all other "species" as follows:

$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

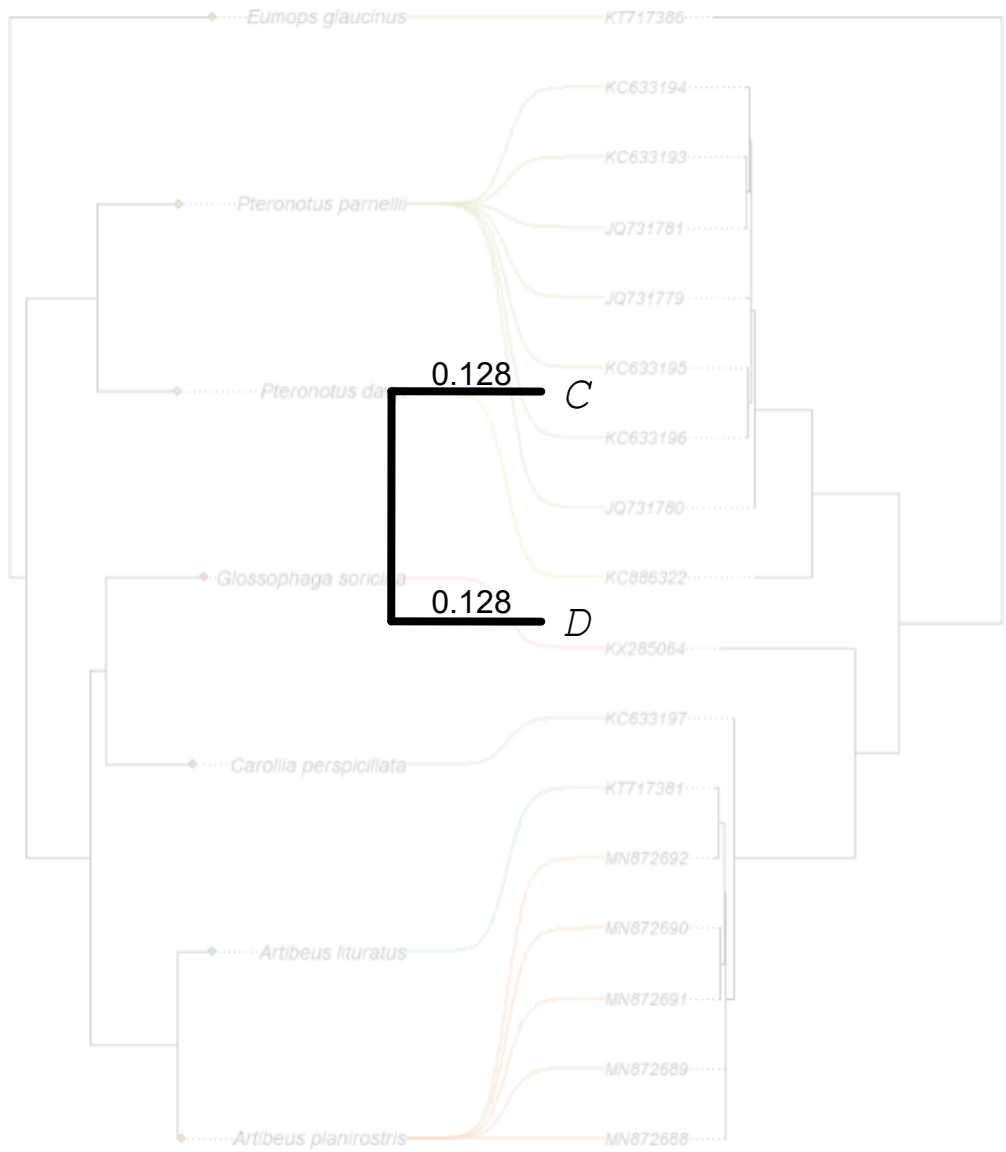
4. Delete the columns and rows for *i* and *j* and create *new* columns and rows for the group (*ij*).
5. If there is only one item left in the matrix, stop – otherwise return to step 1.

UPGMA

Consider the following matrix:

	A	B	C	D	E
A	0.000	1.282	1.282	1.282	0.840
B	1.282	0.000	0.576	0.576	1.282
C	1.282	0.576	0.000	0.256	1.282
D	1.282	0.576	0.256	0.000	1.282
E	0.840	1.282	1.282	1.282	0.000

	A	B	CD	E
A	0.000	1.282	**1.282**	0.840
B	1.282	0.000	**0.576**	1.282
CD	**1.282**	**0.576**	**0.000**	**1.282**
E	0.840	1.282	**1.282**	0.000



UPGMA

Rules:

1. Find species *i* and *j* with the smallest distance.
2. Create a new group (*ij*). Join *i* and *j* on the tree, and assign branch lengths $D_{ij}/2$.
3. Compute the distance from (*ij*) to all other "species" as follows:

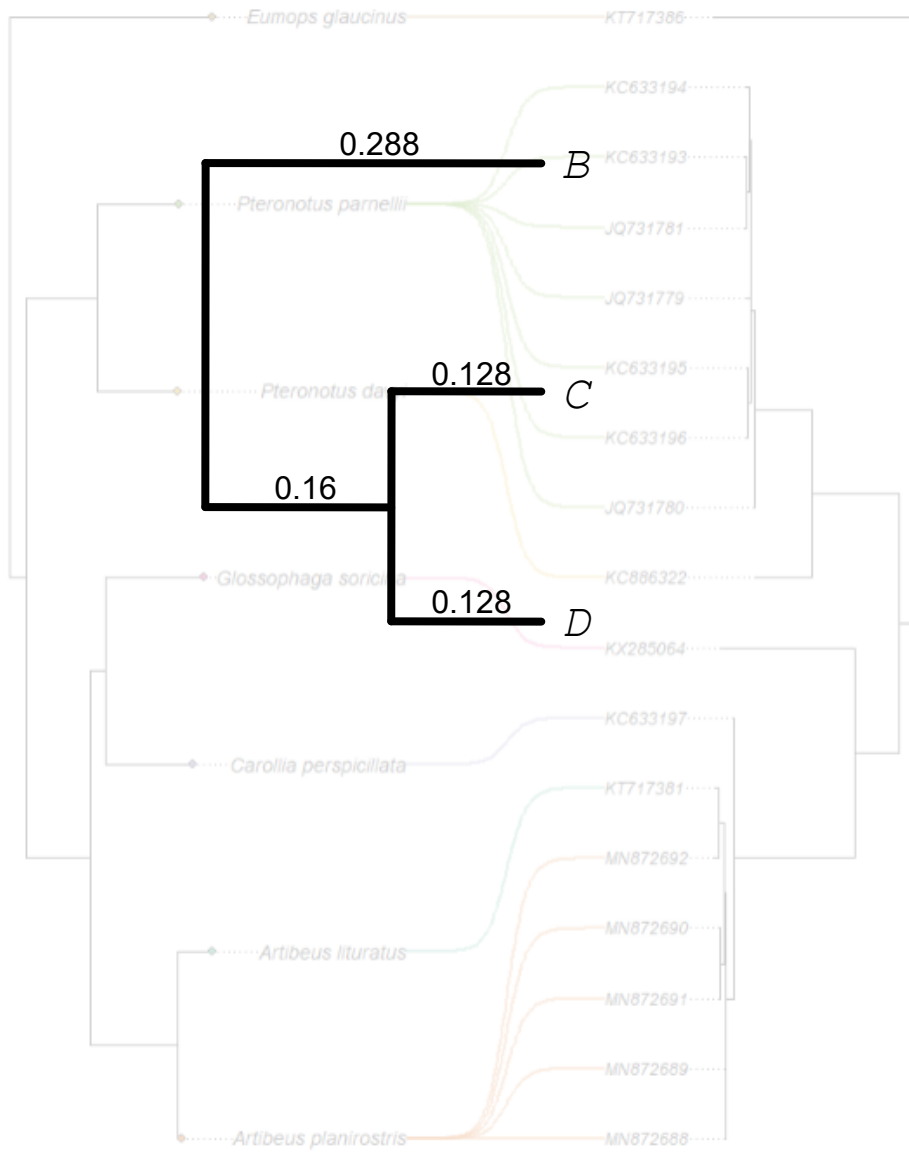
$$D_{(ij),k} = \left(\frac{n_i}{n_i + n_j}\right) D_{ik} + \left(\frac{n_j}{n_i + n_j}\right) D_{jk}$$

4. Delete the columns and rows for *i* and *j* and create *new* columns and rows for the group (*ij*).
5. If there is only one item left in the matrix, stop – otherwise return to step 1.

UPGMA

Consider the following matrix:

	A	B	CD	E
A	0.000	1.282	1.282	0.840
B	1.282	0.000	0.576	1.282
CD	1.282	0.576	0.000	1.282
E	0.840	1.282	1.282	0.000

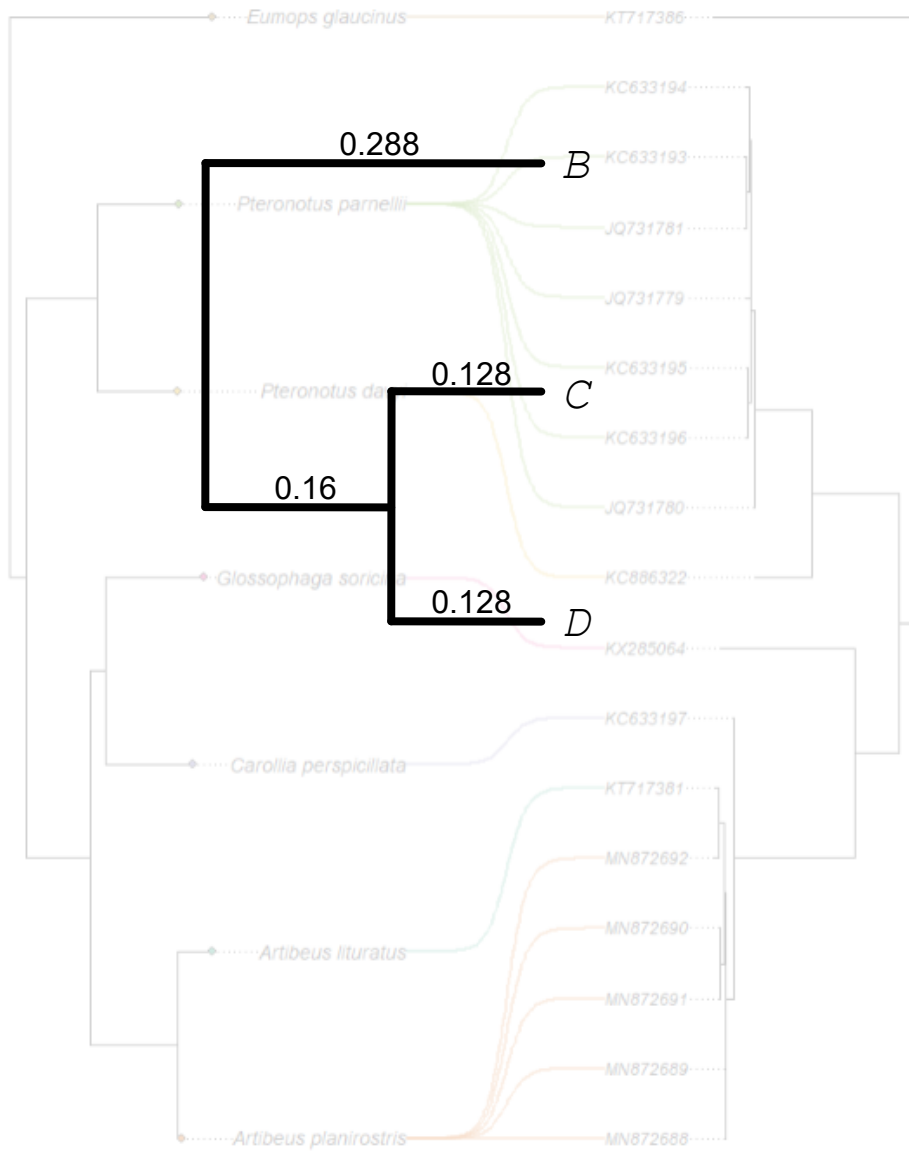


UPGMA

Consider the following matrix:

	A	B	CD	E
A	0.000	**1.282**	**1.282**	0.840
B	**1.282**	**0.000**	**0.576**	**1.282**
CD	**1.282**	**0.576**	**0.000**	**1.282**
E	0.840	**1.282**	**1.282**	0.000

	A	BCD	E
A	0.000	1.282	0.840
BCD	1.282	0.000	1.282
E	0.840	1.282	0.000

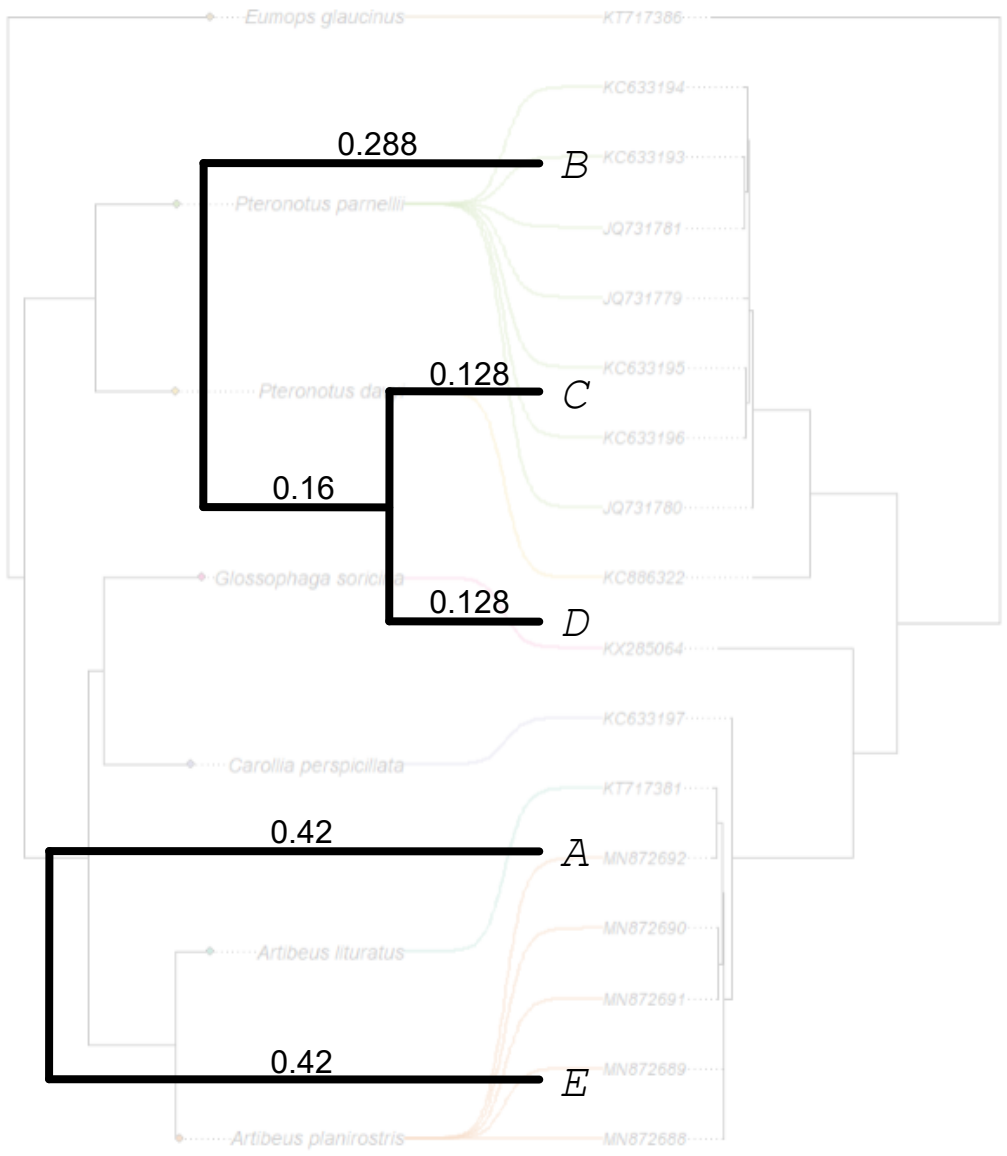


UPGMA

Consider the following matrix:

	A	BCD	E
A	0.000	1.282	0.840
BCD	1.282	0.000	1.282
E	0.840	1.282	0.000

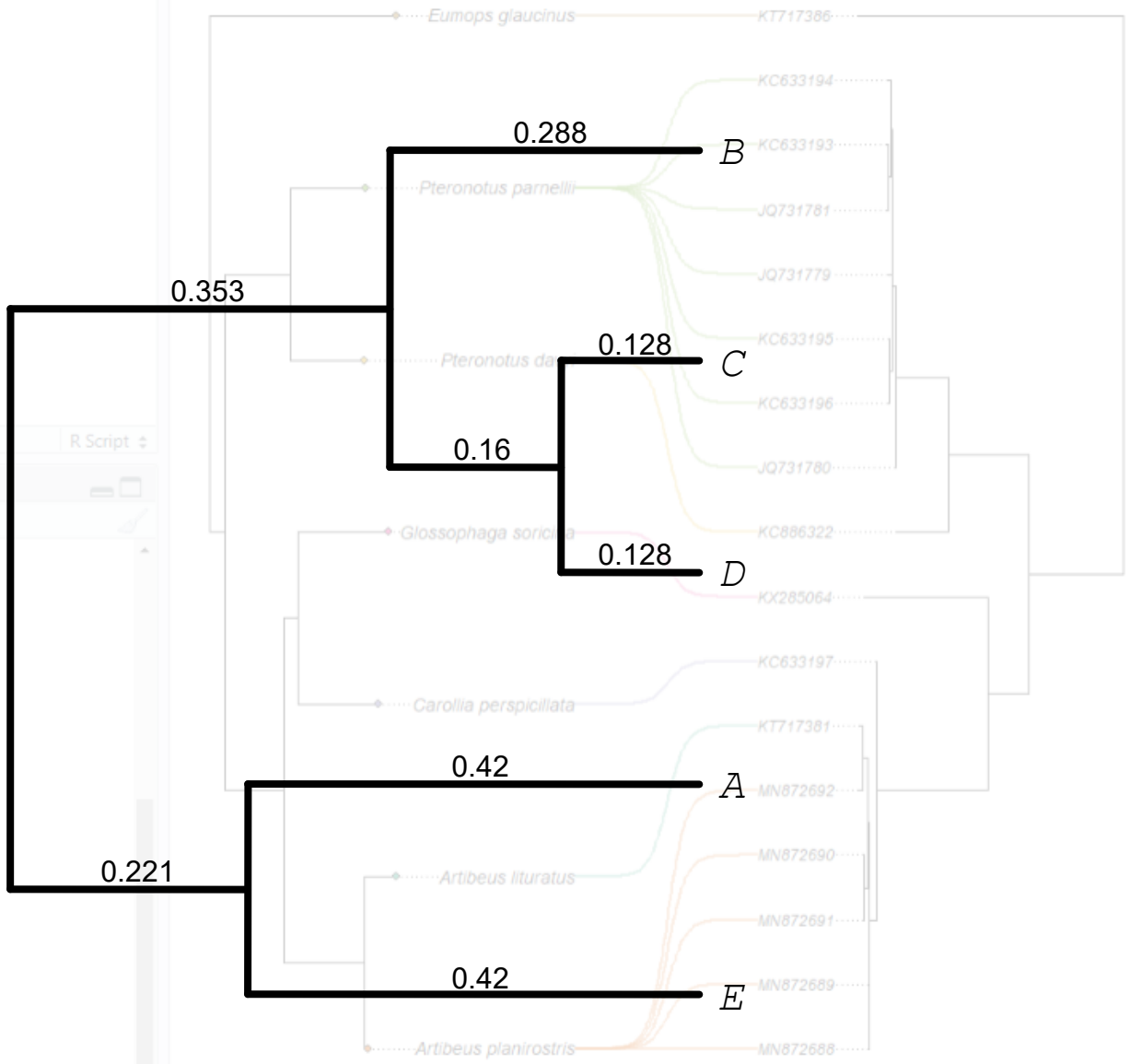
	A	BCD
AE	0.000	1.282
BCD	1.282	0.000



UPGMA

Consider the following matrix:

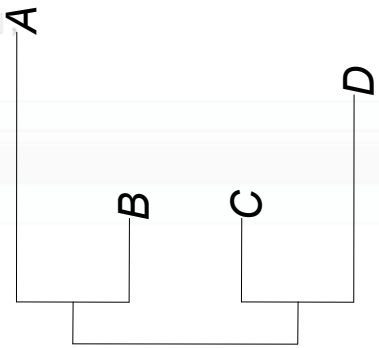
	A	BCD
AE	0.000	1.282
BCD	1.282	0.000



UPGMA

UPGMA is very fast, but it is extremely poorly behaved on non-clocklike trees.

For example, consider the following tree and the distances (known exactly):



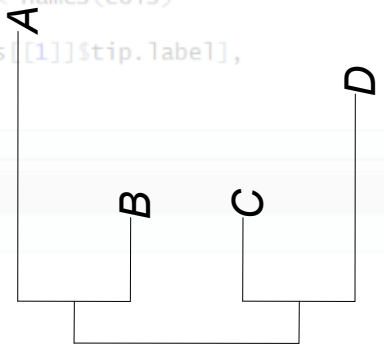
	A	B	C	D
A	0	17	21	27
B	17	0	12	18
C	21	12	0	14
D	27	18	14	0

We can see immediately *that we will get the wrong tree* with UPGMA in this case.

UPGMA

Non clocklike trees:

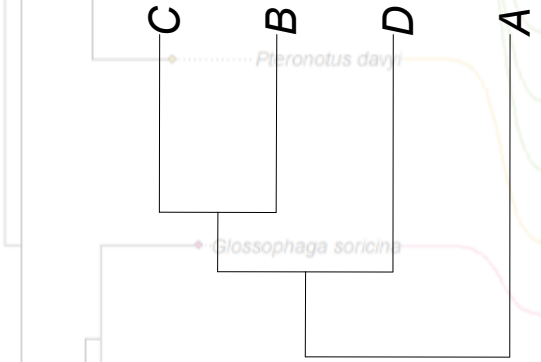
True tree:



True matrix:

	A	B	C	D
A	0	17	21	27
B	17	0	12	18
C	21	12	0	14
D	27	18	14	0

Inferred tree:



Neighbor-Joining (NJ)

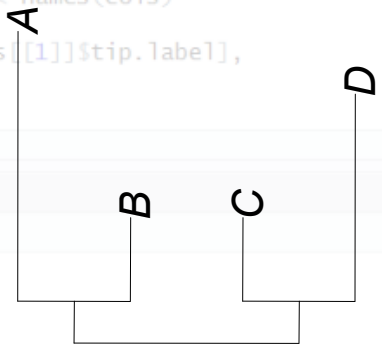
Neighbor joining is a second popular clustering algorithm.

It is just as fast as UPGMA, but is not susceptible to the bad behavior when the true distances are not clocklike.

Neighbor Joining

Non clocklike trees:

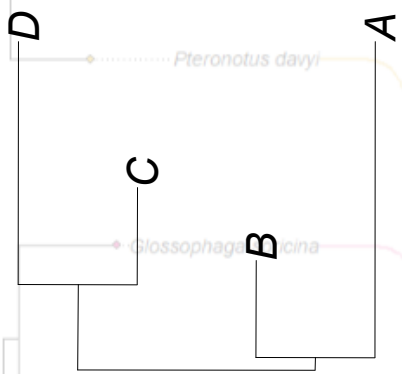
True tree:



True matrix:

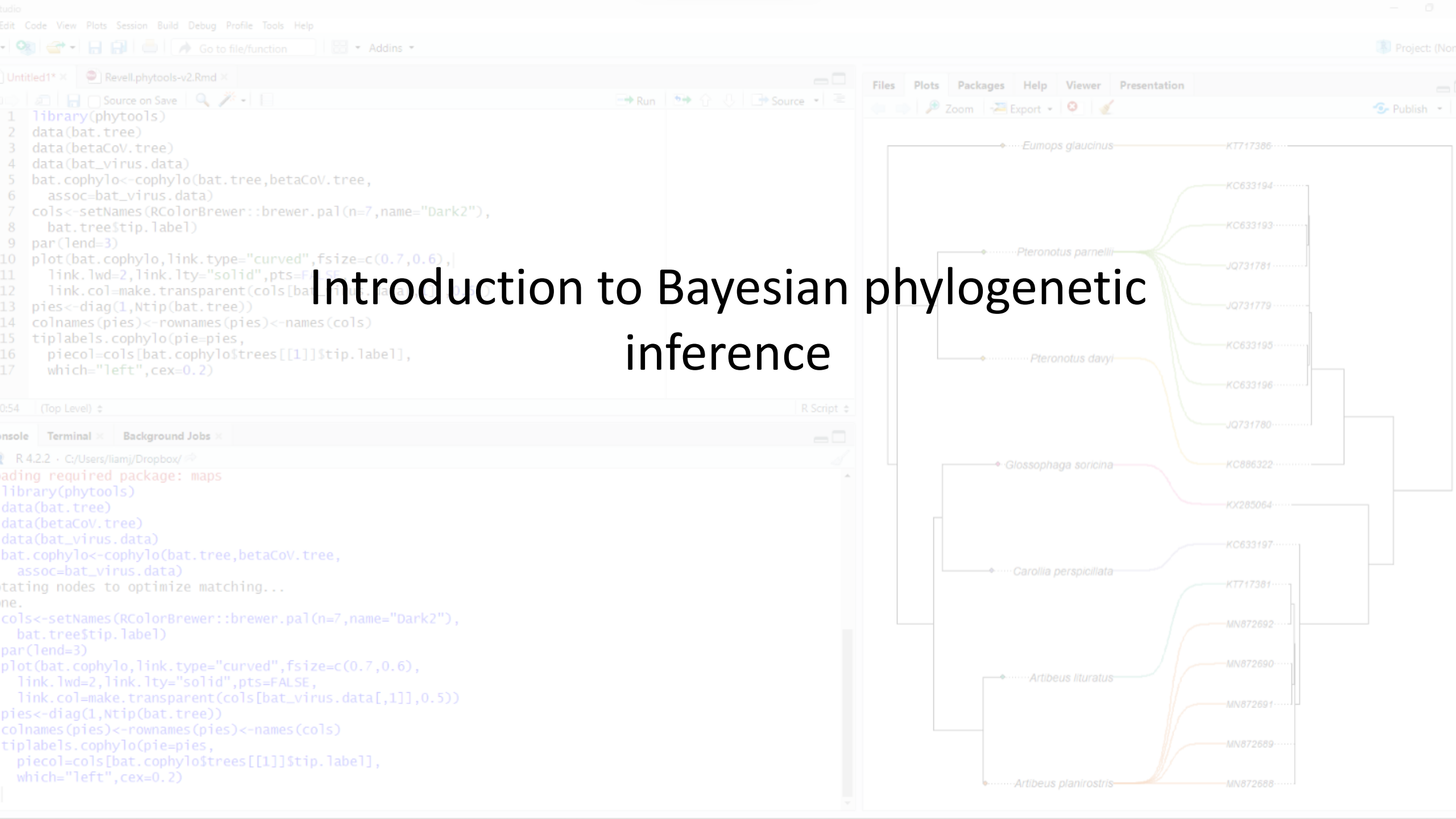
	A	B	C	D
A	0	17	21	27
B	17	0	12	18
C	21	12	0	14
D	27	18	14	0

Inferred tree:

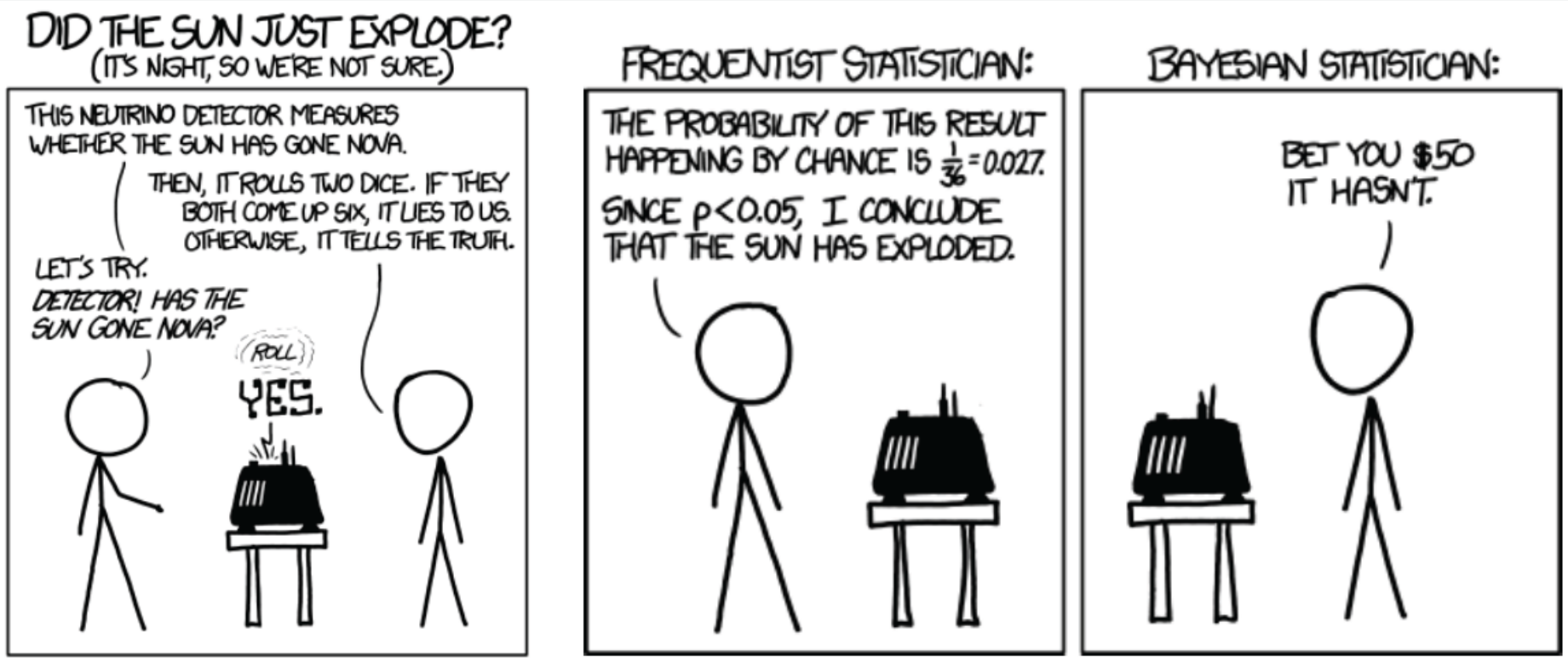


In fact, if we treat the inferred tree as unrooted, we see that it is *exactly* the same as the true tree – and this will always be the case with NJ when the true distances are known.

Introduction to Bayesian phylogenetic inference



The Bayesian paradigm



http://xkcd.com/1132/

Bayes' theorem

One of the central concepts in Bayesian statistics is Bayes' theorem, which gives the conditional probability of A given B.

$$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$

P(A) and P(B) are here the *prior probabilities* of A & B; P(B|A) is the probability of B given A; and P(A|B) is our *posterior probability*.

P(B), the total probability of B, can be written as:

$$P(B) = P(A) \times P(B|A) + P(\text{not } A) \times P(B|\text{not } A)$$

Bayes' theorem: example

<u>False positives in a clinical test:</u>

A new test (B) has been developed for disease A with incidence 0.1%; if a subject has disease A, the test is positive 99% of the time; if a subject does not have disease A, the test is positive only 2% of the time. What is the probability that a person who has tested positive has the disease?

Bayes' theorem: example

<u>False positives in a clinical test:</u>

A new test (B) has been developed for disease A with incidence 0.1%; if a subject has disease A, the test is positive 99% of the time; if a subject does not have disease A, the test is positive only 2% of the time. What is the probability that a person who has tested positive has the disease?

From the problem, we know P(A) = 0.001 & P(B|A) = 0.99. Let's compute the remaining quantities.

$$P(B) = P(A) \times P(B \mid A) + P(\text{not } A) \times P(B \mid \text{not } A)$$

$$P(B) = 0.001 \times 0.99 + 0.999 \times 0.02 = 0.01998$$

Now we have enough to evaluate Bayes' theorem:

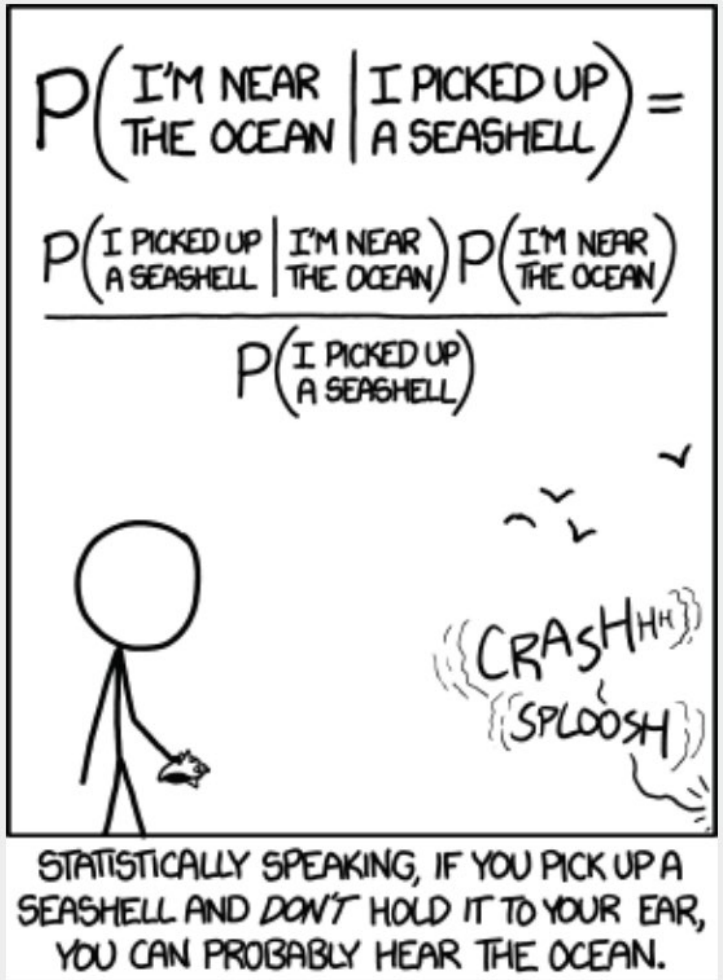
$$P(A \mid B) = \frac{P(A) \times P(B \mid A)}{P(B)}$$

$$P(A \mid B) = \frac{0.001 \times 0.99}{0.01998} = 0.0495$$

This means that only ~5% of positives are true positives and the remainder (95%) are false!

Bayesian statistics

Conditional probability



http://xkcd.com/1236/

Bayesian statistics

Conditional probability

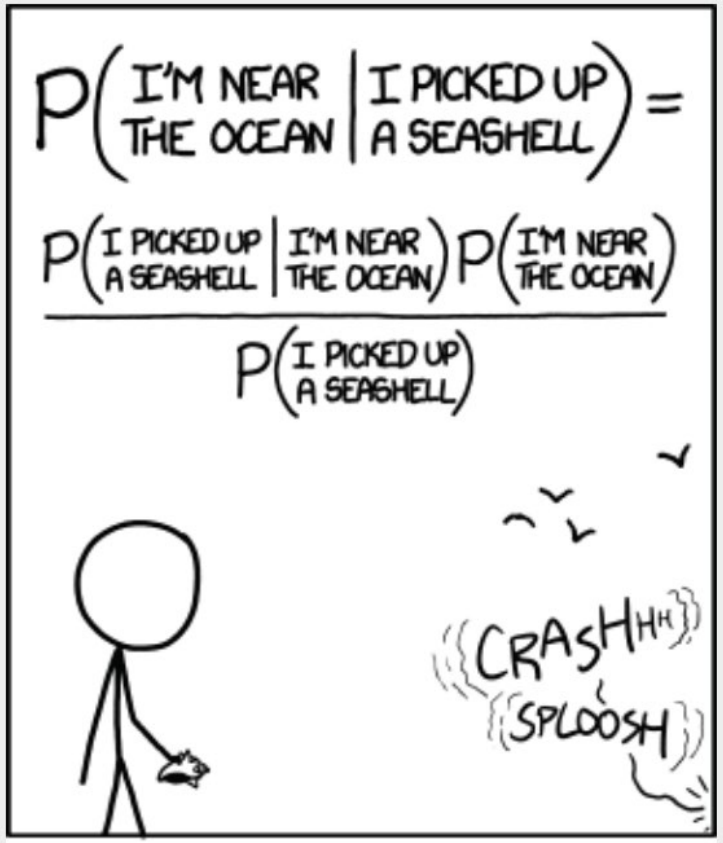
$$P(\text{Hyp.} \mid \text{Data}) = \frac{P(\text{Data} \mid \text{Hyp.})\,P(\text{Hyp.})}{P(\text{Data})}$$

STATISTICALLY SPEAKING, IF YOU PICK UP A SEASHELL AND DON'T HOLD IT TO YOUR EAR, YOU CAN PROBABLY HEAR THE OCEAN.

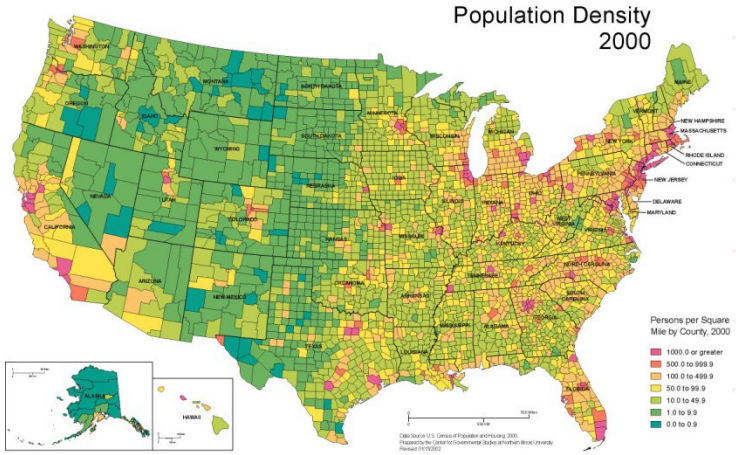
http://xkcd.com/1236/

Bayesian statistics

Conditional probability



http://xkcd.com/1236/



Bayesian statistics

The clinical trial example, given previously, is uncontroversial. What is controversial in Bayesian analysis is assigning prior probabilities to hypotheses for which all the available data is in X.

$$P(H_1 \mid X) = \frac{P(H_1) \times P(X \mid H_1)}{P(H_1) \times P(X \mid H_1) + P(H_2) \times P(X \mid H_2)}$$

Here H_1 and H_2 are two mutually exclusive and comprehensive hypotheses for X.

For continuous parameters, rather than discrete alternative hypotheses, Bayes' theorem has the following form:

$$f(\theta \mid X) = \frac{f(\theta) f(X \mid \theta)}{\int f(\theta) f(X \mid \theta) \mathrm{d}\theta}$$

There are two inherent difficulties in evaluating this expression: 1) It requires *a priori* specification of the prior density; and 2) the integral in the numerator is often difficult or impossible to compute.

The Bayesian paradigm

Two basic philosophies exist in statistics: *classical* or *frequentist*; and *Bayesian*.

Frequentist statistics is the statistics that is covered in a typical introductory or biostats course. In frequentist statistics, the parameters of our data-generating model are treated as fixed but unknown constants. We assess the performance of our statistical method by virtue of the long-run convergence of our parameter estimates to the data-generating values.

The methods that we have covered so far (e.g., ML estimation), as well as many familiar methods in statistics (e.g., t-test, ANOVA, correlation, etc.) fall into the frequentist category.

The Bayesian paradigm

Two basic philosophies exist in statistics: *classical* or *frequentist*; and *Bayesian*.

Bayesian statistics, by contrast, treats the parameters of the model as uncertain; and defines a probability distribution to describe their possible values.

There are two important probability distributions: the *prior distribution*, this is the probability distribution for our parameters before we have analyzed the data; and the *posterior distribution*, this is the probability distribution for our parameters after we have analyzed our data. All inferences about our parameters are performed on the *posterior distribution*.

How can we define a prior probability for our parameters?

This is one of the most significant challenges in Bayesian analysis. In some cases, we just seek the least informative prior. However, in other cases, an informative prior is essential.

Bayesian MCMC

Although the difficulty of the prior still exists, the difficulty of computing the integral of the marginal probability of the data (X) has largely been solved due to the development and application of *Markov chain Monte Carlo* (MCMC) algorithms.

In a MCMC algorithm, new values for θ are proposed and accepted with probability equal to the posterior probability ratio.*

That is to say, given current values for θ of θ_1, a newly proposed value for θ, θ_2, will be accepted with probability:

$$\alpha = \min\left(1, \frac{f(\theta_2) f(X \mid \theta_2)}{f(\theta_1) f(X \mid \theta_1)}\right)$$

This is just:

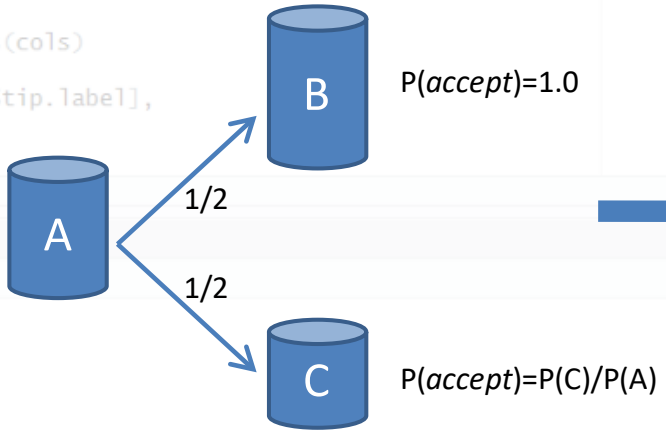
$$\alpha = \min\big(1, (\text{prior ratio}) \times (\text{likelihood ratio})\big)$$

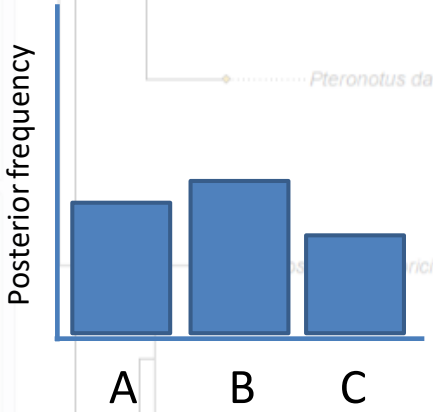
*times the proposal ratio

Bayesian MCMC

Bayesian MCMC

In a MCMC algorithm, new values for θ are proposed and accepted with probability equal to the posterior probability ratio.





Accepted proposals are retained to create a *posterior sample*.

In a well designed MCMC algorithm, the posterior sample will converge to the *posterior density*.

Bayesian Phylogenetic Inference Using DNA Sequences: A Markov Chain Monte Carlo Method

Ziheng Yang and Bruce Rannala
Department of Integrative Biology, University of California, Berkeley

Yang, Z., Rannala, B. (1997)

Markov Chain Monte Carlo Algorithms for the Bayesian Analysis of Phylogenetic Trees

Bret Larget and Donald L. Simon
Department of Mathematics and Computer Science, Duquesne University

Larget, B., Simon, D. L. (1999)

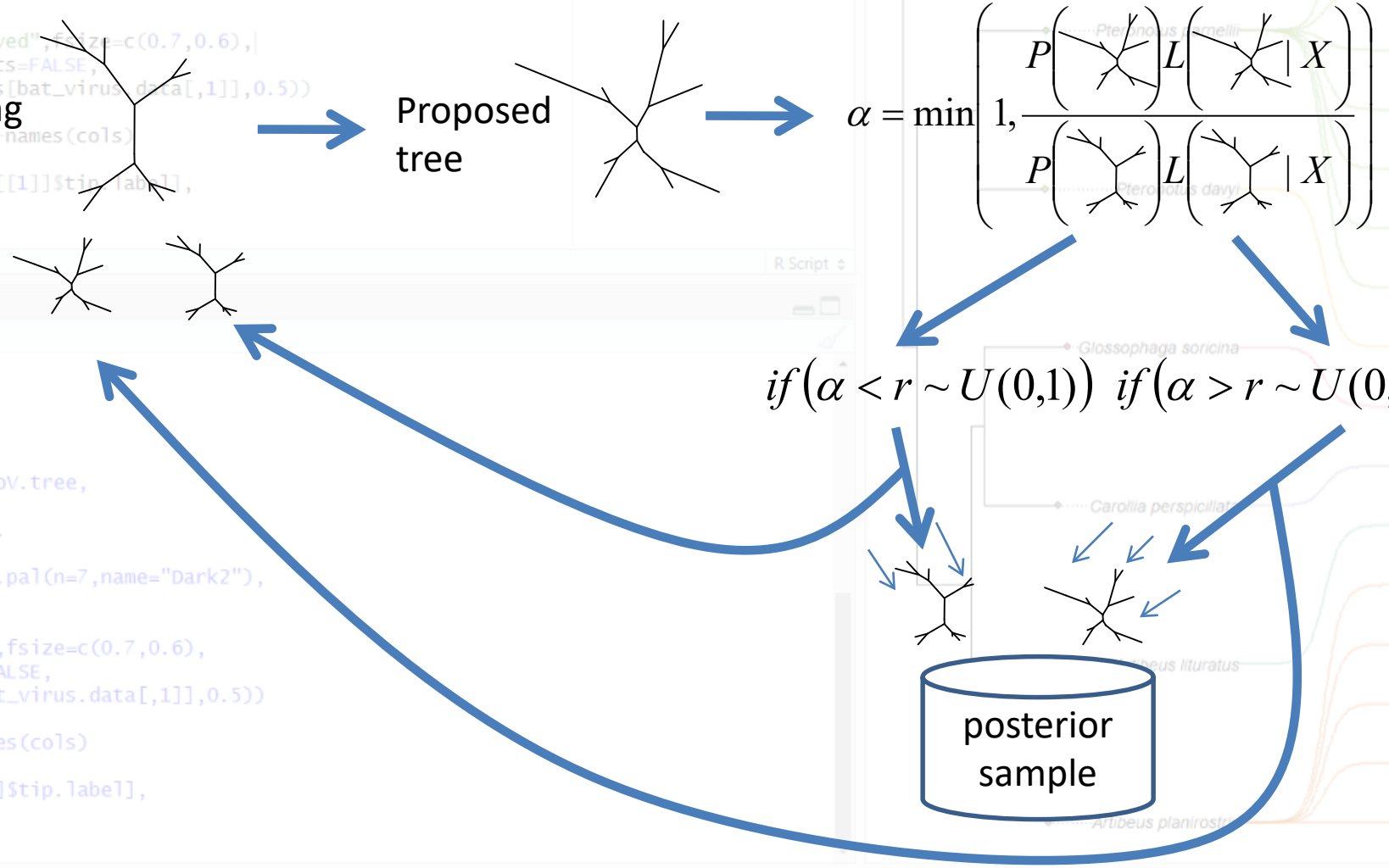
Bayesian Phylogenetic Inference via Markov Chain Monte Carlo Methods

Bob Mau,[1,*] Michael A. Newton,[1,2] and Bret Larget[3]

Mau, B., Newton, M. A., Larget, B. (1999)

Bayesian phylogenetics

Bayesian phylogeny inference works as follows



Bayesian phylogenetics

How do we set our priors?

This depends largely on our *philosophy*.

Normally, our philosophy as scientists causes us to choose to be *objective Bayesians*. That means we should choose a prior that is as uninformative as possible: this will allow the likelihood of our tree given the data to dominate the posterior odds ratio.

For example, we can choose to set the prior probability of any topology, τ, equal to $P(\tau)=1/N_\tau$. Since these quantities will cancel in the posterior odds ratio, this means that the probability of accepting a proposal converges to the likelihood ratio.

In addition to having priors on the tree topologies, we also need to set priors for branch lengths (uniform and exponential are common) and substitution model parameters.

Bayesian phylogenetics

What do we do with the posterior sample?

This is a pervasive problem for Bayesian MCMC as “averaging” a sample of trees is non-trivial.

It is statistically valid to simply choose the tree with the *maximum posterior probability*. This tree is called the *MAP tree* and it tends to be similar to the ML tree for an uninformative prior.

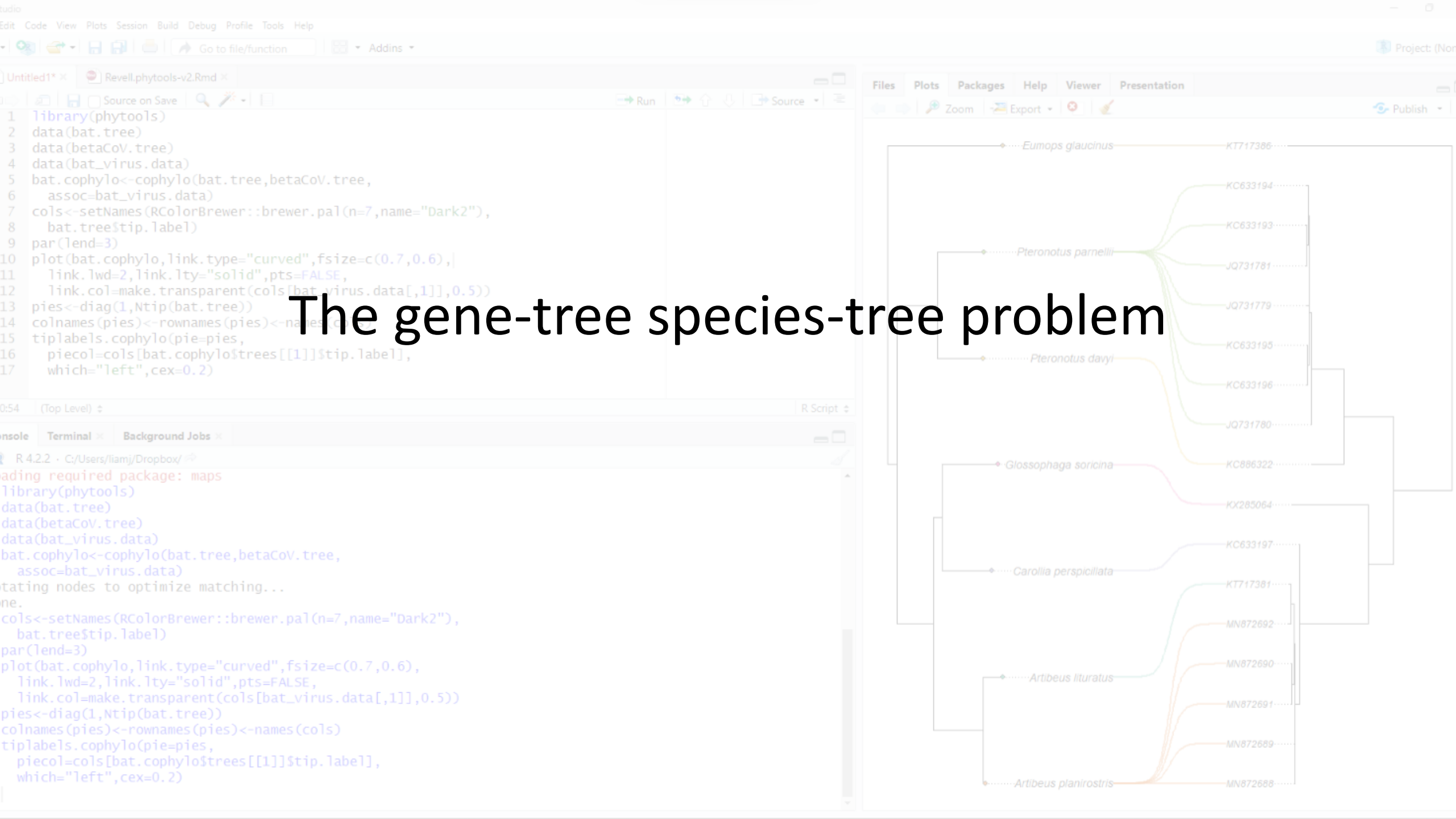
However, it is much more common to compute a *majority-rule consensus topology* from our posterior sample of trees, retaining the relative frequencies of each clade in the consensus analysis. These relative frequencies can be interpreted as the *posterior probabilities* of each clade.

Finally, branch lengths on the consensus topology are typically obtained by *averaging across the sample of trees in which that branch is found*. However, this approach has been criticisms, and alternatives are certainly possible (for instance running another MCMC holding the topology constant).

Advantages & disadvantages

- **Advantages:**
  - Can be used for larger datasets where ML is not possible.
  - The posterior probabilities can be interpreted as clade probabilities (given the model & assumptions).
  - Allows us to incorporate uncertainty by providing a sample of trees, rather than the single ‘best’ tree.
- **Disadvantages:**
  - The issue of the prior is still not totally resolved. The prior can have unexpected consequences on inferences.
  - The posterior probabilities can be interpreted as clade probabilities (*given the model & assumptions*). When model assumptions are violated, these can give us false confidence in our hypothesis.

The gene-tree species-tree problem

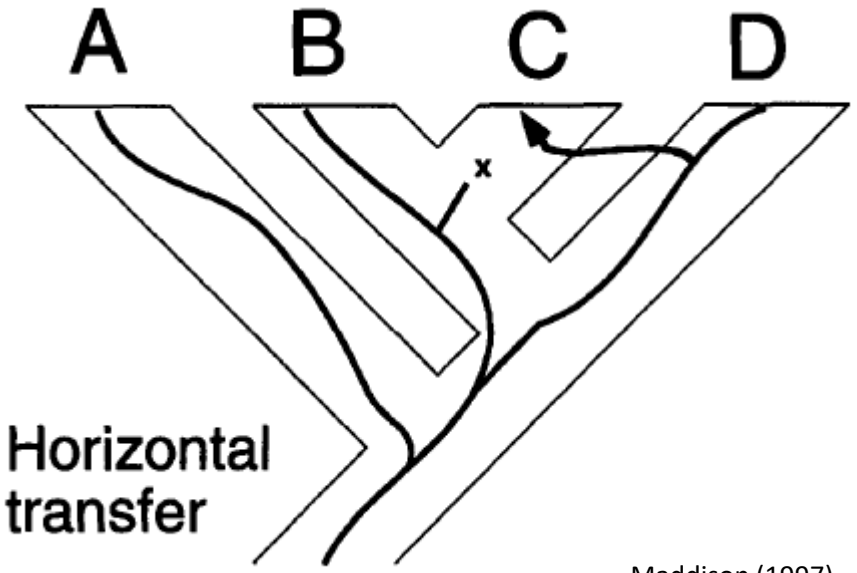


The gene-tree, species-tree problem

- So far in the course we have been estimating the evolutionary relationships between gene-sequences, and sometimes treating these as synonymous with the underlying species tree.
- We have discussed the possibility that our tree might be wrong, because it is poorly estimated or the method assumptions are unsatisfactory.
- However, we have not considered the possibility that our gene-tree might be *genuinely discordant* with our species tree.
- Genuine discordance between gene-trees and the species tree occurs primarily for <u>three</u> reasons.

The gene-tree, species-tree problem

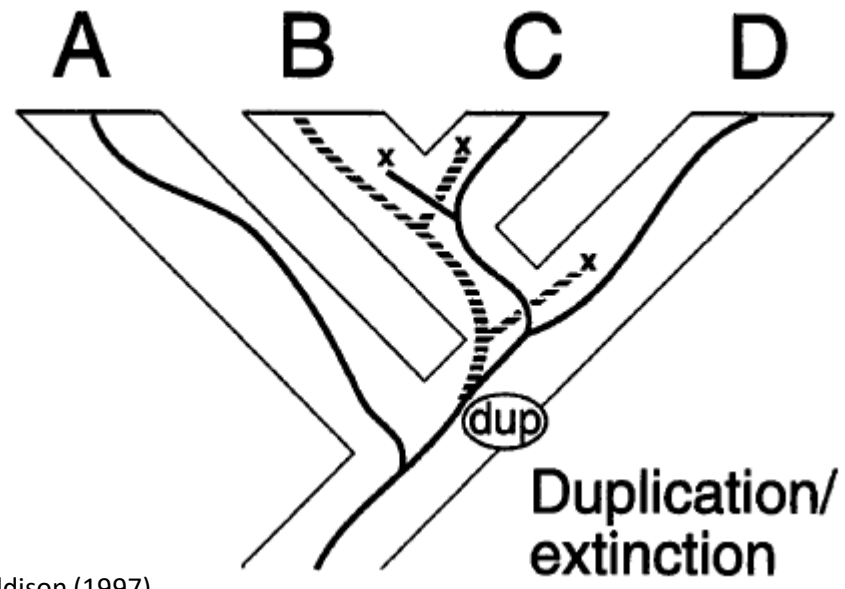
- Horizontal gene transfer:
  - This is when two closely or distantly related species exchange genes & the transferred gene copy is fixed.



Maddison (1997)

The gene-tree, species-tree problem

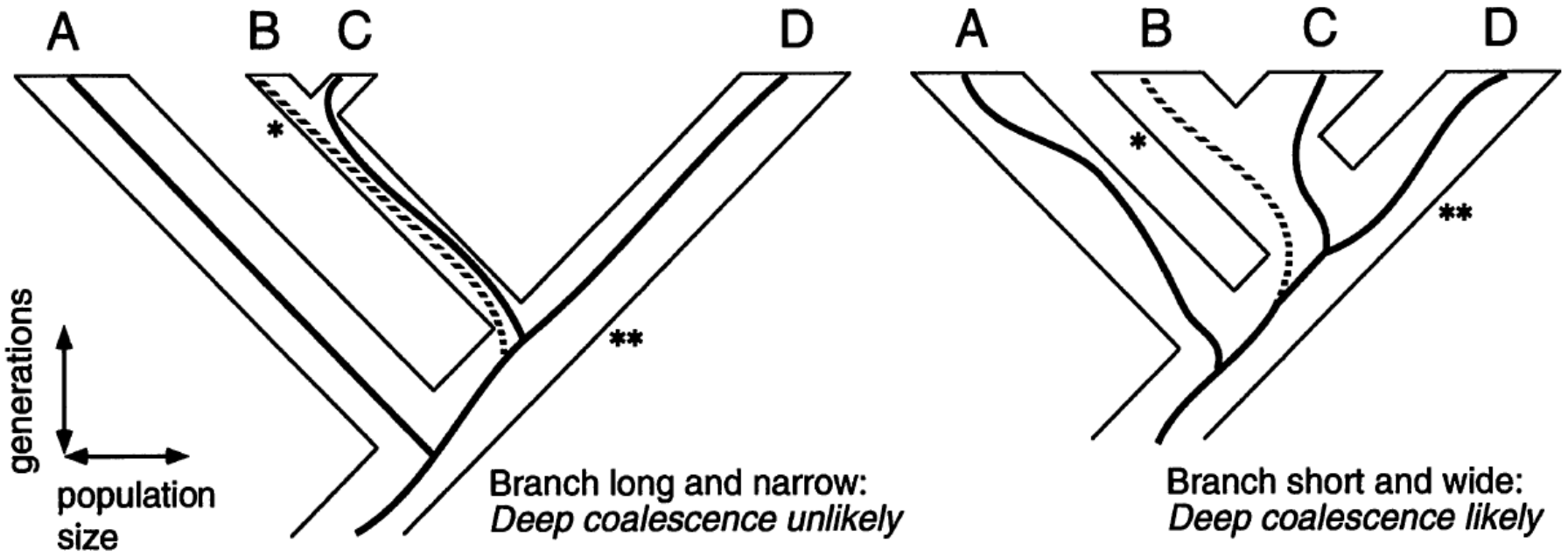
- Gene duplication:
  - This is when gene duplication/extinction causes us to mistake paralogs for orthologs. (This could be viewed as a *homology* problem.)



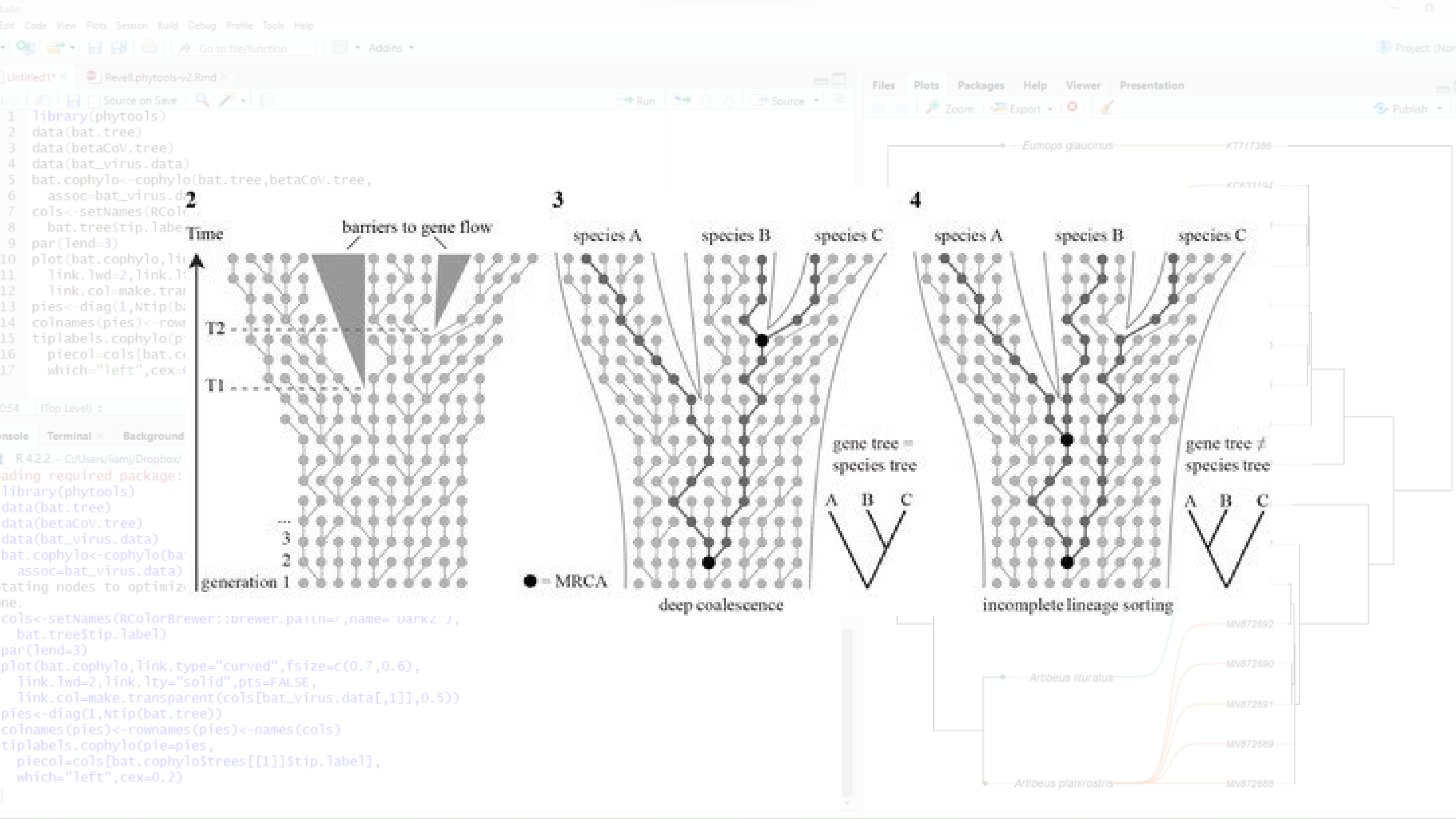
Maddison (1997)

The gene-tree, species-tree problem

- Incomplete lineage sorting:
  - This is when within-species polymorphism *fails* to fix along one or multiple internal edges of the tree.
  - Incomplete lineage sorting is also called *deep coalescence.*



Maddison (1997)


2
Time
barriers to gene flow
T2
T1
...
3
2
generation 1
3
species A
species B
species C
● = MRCA
gene tree =
species tree
A B C
deep coalescence
4
species A
species B
species C
gene tree ≠
species tree
A B C
incomplete lineage sorting
Eumops glaucinus
KT717386
KC633194
Artibeus lituratus
Artibeus planirostris
MN872692
MN872690
MN872691
MN872689
MN872688

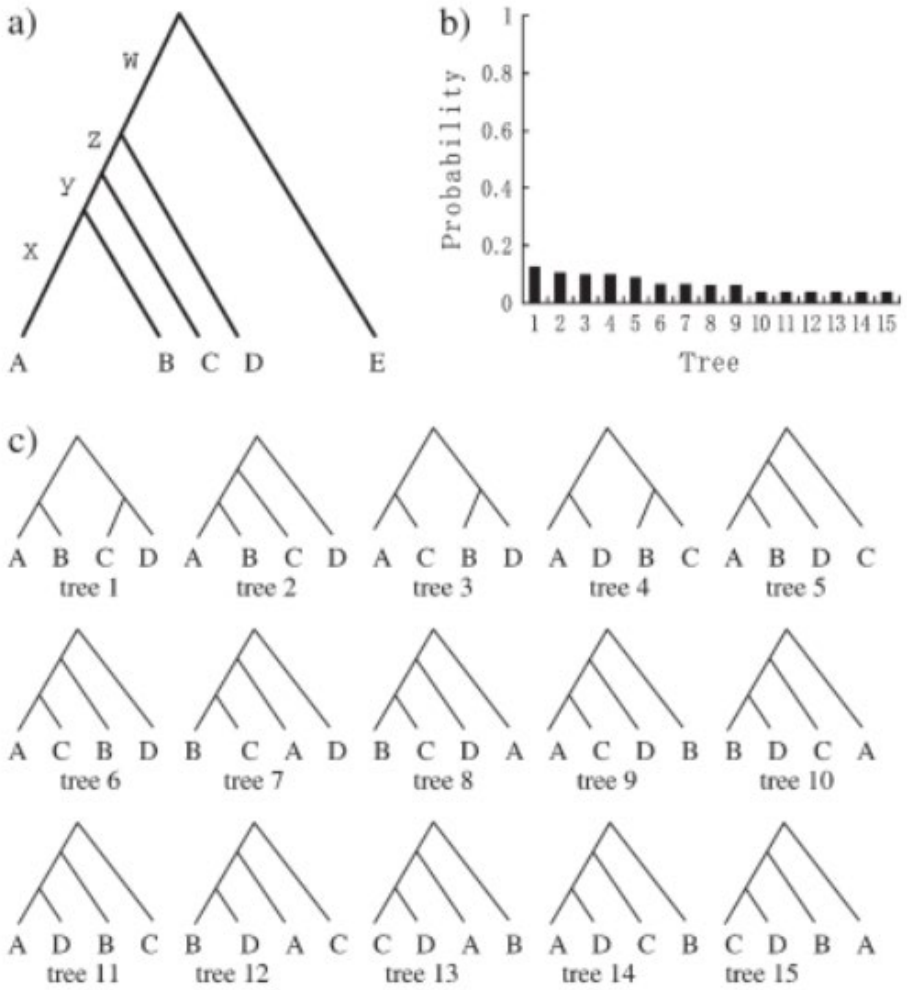
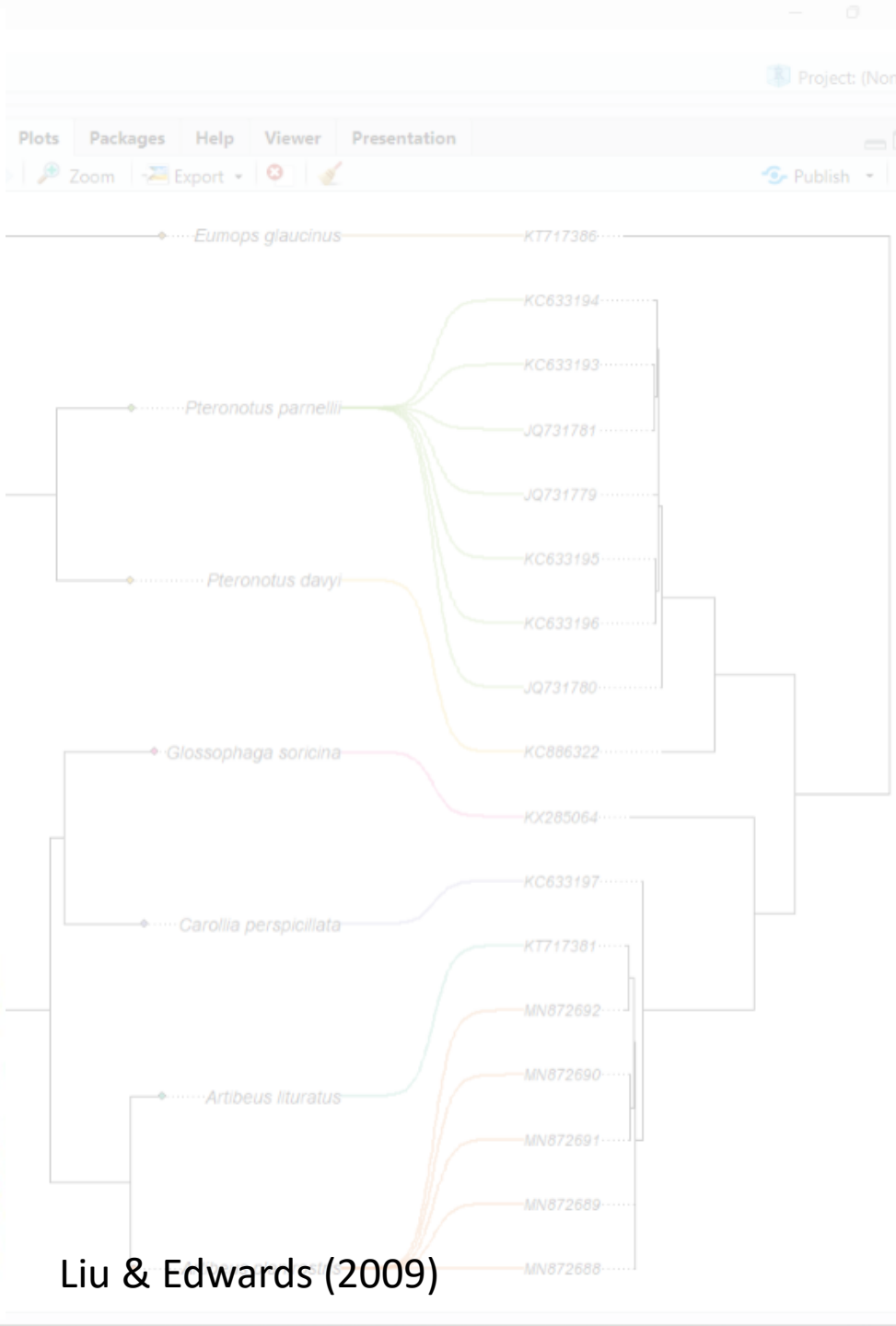


FIGURE 2. The 5-taxon species tree and the probability distribution of the gene trees generated from the species tree. a) The 5-taxon anomalous species tree used in simulations. The 5-taxon trees is ultrametric with population size of 0.1 for all populations. The lengths of branches are $x = 0.05$, $y = 0.005$, $z = 0.005$, and $w = 1.0$ in mutation units. The out-group is species E; b) the probability distribution of the 15 gene tree topologies generated from the species tree in (a); c) the list of the 15 gene tree topologies on the x-axis in (b). To simplify the notation, the out-group (species E) is not included in the trees. Tree 1 is the most probable gene tree, whereas Tree 2 is the gene tree that matches the species tree.



Liu & Edwards (2009)

The gene-tree, species-tree problem

- Various methods have been developed to *estimate* species trees from both multi-locus data (taking into account the possibility of gene tree incongruence), and from estimated gene trees for multiple loci.
- The majority of methods focus on the problem of incomplete lineage sorting.
- These methods can be divided in two categories: heuristic methods and joint estimation.
- Heuristic methods generally use the individual gene trees to try to find a species tree, whereas joint estimation computes the probability that the data observed for genes arose on the species tree, taking into account the possibility of incongruent histories.

How does estimation from gene trees work?

- Imagine various gene trees evolving on a single species tree by the coalescent.
- Coalescent events have to precede speciation events, so we should be able to use minimum distances between taxa to identify the maximum times to speciation events.

```r
library(phytools)
data(bat.tree)
data(betaCoV.tree)
data(bat_virus.data)
bat.cophylo<-cophylo(bat.tree,betaCoV.tree,
  assoc=bat_virus.data)
cols<-setNames(RColorBrewer::brewer.pal(n=7,name="Dark2"),
  bat.tree$tip.label)
par(lend=3)
plot(bat.cophylo,link.type="curved",fsize=c(0.7,0.6),
  link.lwd=2,link.lty="solid",pts=FALSE,
  link.col=make.transparent(cols[bat_virus.data[,1]],0.5))
pies<-diag(1,Ntip(bat.tree))
colnames(pies)<-rownames(pies)<-names(cols)
tiplabels.cophylo(pie=pies,
  piecol=cols[bat.cophylo$trees[[1]]$tip.label],
  which="left",cex=0.2)
```

```
ading required package: maps
library(phytools)
data(bat.tree)
data(betaCoV.tree)
data(bat_virus.data)
bat.cophylo<-cophylo(bat.tree,betaCoV.tree,
  assoc=bat_virus.data)
tating nodes to optimize matching...
ne.
cols<-setNames(RColorBrewer::brewer.pal(n=7,name="Dark2"),
  bat.tree$tip.label)
par(lend=3)
plot(bat.cophylo,link.type="curved",fsize=c(0.7,0.6),
  link.lwd=2,link.lty="solid",pts=FALSE,
  link.col=make.transparent(cols[bat_virus.data[,1]],0.5))
pies<-diag(1,Ntip(bat.tree))
colnames(pies)<-rownames(pies)<-names(cols)
tiplabels.cophylo(pie=pies,
  piecol=cols[bat.cophylo$trees[[1]]$tip.label],
  which="left",cex=0.2)
```