Phylogenetics & Bioinformatics

Colby College BI164

25 March 2024

Objectives

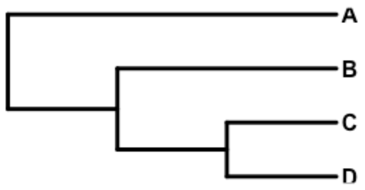
1. Interpret phylogenetic trees based on morphological and molecular characteristics.
2. Connect the information in phylogenetic trees to the evolutionary and natural history of organisms.
3. Increase your familiarity with data manipulation and analysis in R.

Introduction

Biological **systematics** is the study of the relationships among living things. Systematists often seek to produce a classification or **taxonomy**, which reflects the evolutionary history of organisms. Because evolution has occurred over a long time with no witnesses, that history is not always obvious. It must be inferred using phylogenetic methods. Taxonomic relationships are often visualized as phylogenetic trees in which organisms are grouped together based on relatedness to their common ancestors. These trees are essentially hypotheses for the relationships of organisms, and like any hypothesis they can be revised and updated with new data.

Understanding a Tree

Let's consider the simple tree below, where the letters at the tips represent different species.



- The branching pattern of the tree represents the evolutionary history of populations of organisms over time. The moments in time when a lineage splits into two are the branch points, or **nodes**, in the tree. The tips of the tree are taxa in the present time. The taxa at the tips are what have been sampled to create the tree. (More on that later!)

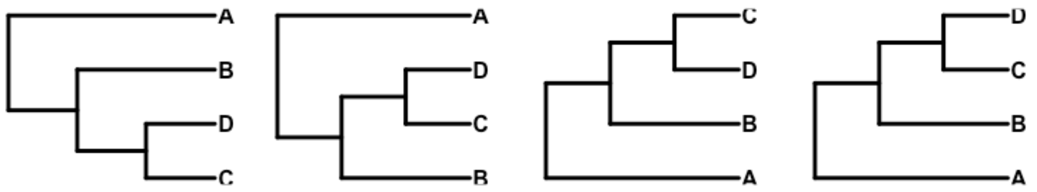
- Time moves from the base to the tips of the tree (left to right in this example). This tree tells us that the first event was the split into two different lineages, one leading to species A and one to the other tips of the tree. The next event was the splitting of B from the lineage that leads to C and D. The most recent event is the splitting of C and D into two separate species. We know this is the most recent split because it's the closest node to the tips.

- You can also read a tree from the present (the tips) backwards in time to find the **most recent common ancestor (MRCA)** of existing groups at the tips. For example, the most recent common ancestor of Species C and D is represented by the right-most node. Deeper in the tree (closer to the base) is the MRCA of Species B, C and D. All tips of the tree share a MRCA at the base.

- You can identify three different monophyletic groups (ABCD, BCD, and CD). A **monophyletic group** is a clade that consists of all species descended from a common ancestor. (So, as a counter-example, a group containing only species A and B is *not* monophyletic because species C and D also share the same common ancestor.)

- Species B is more closely related to Species D than it is to Species A. That's because Species B and D share a MRCA more recently (closer to the tips) than do Species B and A. Another way of thinking of this is if you trace the distance along the branches from A to B it's greater than the distance from D to B. (The width of the nodes doesn't matter. Only distance in the base-to-tip direction represents evolutionary time.)

- All nodes can be flipped (rotated 180°) without changing relationships in the tree, because the order of branching is preserved. So, Species B is equally related to Species C and D. In fact, all of the trees below show the *same* relationships as the figure above.



What do you need to develop a phylogeny?

Characters and character states: To build a phylogeny a researcher must identify traits or characters that are relevant for distinguish taxa within the group. Characters can include morphological, life history or molecular information. The more characters that are used, the more accurate the tree may be. Once the characters have been identified, each organism is scored or graded on which character state it possesses. A character state is the value of a character. For example if the number of cervical vertebrae is a character in the analysis, then the character state of different taxa might be 7 (for humans) or 14 (for chickens). If the dataset includes DNA sequences, then each position in a sequence is a character and the character state for a taxon can be one of the four nucleotides (A, C, G or T).

Outgroups: We cannot know the real character states for ancestral species at the base of our tree. Instead, we must choose an existing species that can stand in for an ancestor. We call these taxa the outgroup.

- The outgroup should represent a lineage that diverges before the group under consideration (the "ingroup").
- Therefore, the outgroup species cannot be a member of the group for which you are creating a tree. The outgroup must be *out of the group*!
- With that said, the outgroup should have as many similarities as possible while not being considered a member of the ingroup. Importantly the outgroup must have all the characters being used to create the tree. So, if you were building a phylogeny of fish based on their number of cervical vertebrae, you could not use invertebrates as your outgroup, because they do not have vertebrae! There would no meaningful value for their character state.

Shared Derived Characters: Not all characters are equally helpful in creating a phylogeny. A character state that is shared by two or more taxa may suggest that those organisms are closely related. From there, you have to distinguish between two types of character states: ancestral and derived.

- Ancestral (or "primitive") character states are those that have been inherited unchanged from a distant ancestor.
- Derived character states are those that have undergone evolutionary change to a different value.

- Characters for which some taxa share derived character states are the most helpful in creating a tree, because it suggests that those taxa share a recent common ancestor since the base of the tree. In contrast, an ancestral character states shared by all taxa doesn't help to distinguish anything. So they are not particularly useful in creating a tree.

Parsimony: The oldest and simplest methods to construct a phylogeny assume that the evolution of a trait will be the most parsimonious one, meaning that it reflects the simplest and most straight-forward evolutionary history. The total number of evolutionary changes is often referred to as the tree length. A tree that reflects a single evolutionary change for a trait is more parsimonious than one with an alternate arrangement that reflects multiple evolutionary changes. Why do you think we can often assume that a more parsimonious evolutionary history is likely to have occurred?

Convergence and Reversal: Sometimes, the same character state can evolve separately in distantly related organisms. In such cases, we say the character state shows evolutionary convergence. Additionally, a derived character state can revert back to the ancestral condition. When this happens it is known as an evolutionary reversal.

Molecular and morphological characters

The characters used to generate a phylogeny can come from multiple sources, including morphological characters and biomolecular sequences, as well as information on ecology, behavior or the life cycles of organisms. In the early days of systematics, taxonomy was based solely on morphological traits. However choosing characters based on morphology can be subjective. Molecular sequence data is often less prone to bias, and it has become the dominant basis for systematics in recent decades. However it's important to remember that molecular sequences can be influenced by convergent evolution. In the best circumstances, a combination of molecular and morphological data are used to produce the most accurate evolutionary trees.

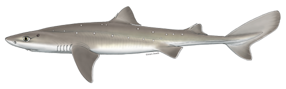
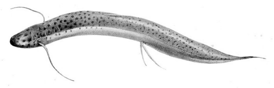
Bioinformatics is a field of science that develops and applies methods to analyze and interpret biological data, especially when the data sets are large and complex. Bioinformatics and computational biology have become critical in all areas of biology. Therefore developing familiarity with common data types and computational tools of bioinformatics is essential to a modern education in biology.

Exercises to practice phylogenetics

Case Study: The Origin of Tetrapods

The superclass Tetrapoda includes all the the vertebrates with four limbs, including amphibians, reptiles, birds, and mammals, as well as animals like snakes and whales whose ancestors also had four limbs. The earliest tetrapods evolved approximately 390 million years ago from an ancient group of fish. However many details of their adaptation to land remain unclear, and this subject is an active area of research among paleontologists and evolutionary biologists.

Today you will explore which type of fish may have given rise to the first tetrapods. Starting with those that have a true jaw, fish are either classified as having cartilaginous (Chondrichtyes) or bony (Osteichthyes) skeletons. Within Osteichthyes, fish are further classified based on the structure of their fins, either ray-finned (Actinopterygii) or lobe-finned (Sacropterygii). Using living examples of each of these four types of fish and two tetrapods, you will create a phylogeny using amino acid sequences for a mitochondrial protein called Cytochrome C Oxidase subunit II (abbreviated as COX2 or COII, pronounced "C O 2").

superclass	class	example species	common name	image
	Chondrichthyes	*Squalus acanthias*	spiny dogfish	
Osteichthyes	Sacropterygii	*Latimeria chalumnae*	coelacanth	
Osteichthyes	Sarcopterygii	*Protopterus dolloi*	lungfish	
Osteichthyes	Actinopterygii	*Danio rerio*	zebrafish	
Tetrapoda	Amphibia	*Bombina variegata*	yellow-bellied toad	

Methods

An orientation to Rstudio

Start by logging into Colby's Rstudio server at https://rstudio.colby.edu/ (https://rstudio.colby.edu/) The Rstudio window is divided into four panels that each provide different tools.

- The top left panel displays **scripts** and other files that you can edit and save.
- The R command line or "**R console**" is in the bottom left panel. When a command is run here, it can change the value of an object, but the command itself is not saved. So it's good practice to write your commands first in the script panel.
- The top right panel shows either a list of objects currently in memory (the environment), or the command history. I find this to be the least useful panel, and you can minimize it by double-clicking on the tabs in the bottom right panel.
- The bottom right panel displays lots of useful stuff, including plots generated in the command line. It also has tabs for navigating folders and files, looking at installed packages, or consulting the help pages.

In this tutorial, R code will appear in fixed-width font. (That is, a blocky computer font, `like this`.) Things you should enter into the R console will start with an arrow or greater-than sign (>). Example output of the command will be shown below that, also in fixed-width font.

Start by ensuring that R is working in your main user folder. You can set R's working directory (folder) using the command `setwd` as shown below. Type what's shown below into the R console panel and hit the enter key.

```
> setwd("~")
```

Throughout this tutorial, you are meant to type the code shown next the prompt (>) into the R console window. (You don't need to type the > character. It will already be there to start the line.)

When you run `setwd("~")`, you won't see any obvious change. In general, if things work correctly in R you won't see anything except a new line prompt (>). If you get an error message, that's usually a sign something hasn't gone as planned. If you do get an error, such as `Error: object '~' not found`, be sure you have quotes around the tilda ("~"). On an America English keyboard, the tilda can be found to the left of the number row, using the shift key. The tilda represents your user home directory in the linux file system. Remember, you are doing this work on a remote server run by Colby, not on your local computer. (R (https://cran.r-project.org/) and Rstudio (https://posit.co/download/rstudio-desktop/) are free, and once you're comfortable with it, you might consider using it on your own laptop.)

Next, load the package of tools we'll use for this exercise, called `sativum`.

```
> library(sativum)
```

Okay, you should be ready to get started!

Procedures

First you need to obtain the COII amino acid sequences from the fish and tetrapods. As a starting point, most of these sequences have been provided for you already as part of the R package `sativum`. You can preview these sequences by entering the name of the data object, `fish.COII`, in the R console. You should see the output shown below.

```
> print(fish.COII)
```

You should see output in the R console that looks like what's below. Your console output may be color-coded to highlight specific amino acids.

```
AAStringSet object of length 5:
    width seq                                               names
[1]   230 MAHPSQLGFQDAASPVMEELLHF...PIVVEAVPLEHFESWSSLMLEEA shark NP_008526.1...
[2]   230 MAHPSQLGLQDAASPVMEELLHF...PIVLEAIPLDPFEDWSSSMLEEA coelacanth NP_008...
[3]   230 MAHPSQLGLQDAASPVMEELIHF...PIVVEAAPLQHFENWSSLMLEKA lungfish NP_00826...
[4]   230 MAHPAQLGFQDAASPVMEELLCF...PIVVEAVPLEFFENWSSAMLEDA zebrafish NP_0593...
[5]   229 MAHPAQLGFQDAASPIMEELLHF...MPIVVEAVPLKTFENWSSSMLET toad YP_001122775...
```

In addition to these COII sequences, you will add the amino acid sequence for an additional amphibian, the African clawed frog (*Xenopus laevis*).

The National Center for Biotechnology Information (NCBI (https://www.ncbi.nlm.nih.gov)) hosts a series of databases important to bioinformatics, including GenBank (https://www.ncbi.nlm.nih.gov/genbank/), one of the oldest, most extensive, and most well-annotated databases of biomolecular sequences. Follow the directions below to download protein sequence data for *Xenopus laevis* COII.

1. **Go to NCBI** at http://www.ncbi.nlm.nih.gov (http://www.ncbi.nlm.nih.gov)

2. **Search** "All Databases" for *Xenopus laevis*. Since we are in search of COII, which is a mitochondrial protein, find the "Proteins" panel and **click on "Protein"**.

This search returns more than 200,000 results! So you will need to narrow the search.

3. **Refine the search** text to read `Xenopus laevis [orgn] cytochrome c oxidase ii`. This text specifies that the search for COII should be limited to the organism *Xenopus laevis*. It's often safer to perform searches using the full name of a gene or protein as opposed to an abbreviation.

4. The results should now include 10 proteins, a manageable number. **Choose one** sequence whose title includes “cytochrome c oxidase subunit II” (or “subunit 2”) but not the word “partial”. (We want the complete protein sequence.) Make your selection by checking the box to the left of the entry. In the “Summary” drop-down menu, select “FASTA (text)”. This will bring you to the amino acid sequence of the *Xenopus laevis* cytochrome c oxidase subunit II protein in FASTA format.

```
>sp|P00407.1|COX2_XENLA RecName: Full=Cytochrome c oxidase subunit 2
MAHPSQLGFQDAASPIMEELLHFHDHTLMAVFLISTLVLYIITIMMTTKLTNTNLMDAQEIEMVWTIMPA
ISLIMIALPSLRILYLMDEVNDPHLTIKAIGHQWYWSYEYTNYEDLSFDSYMIPTNDLTPGQFRLLEVDN
RMVVPMESPTRLLVTAEDVLHSWAVPSLGVKTDAIPGRLHQTSFIATRPGVFYGQCSEICGANHSFMPIV
VEAVPLTDFENWSSSMLEA
```

The FASTA format (https://en.wikipedia.org/wiki/FASTA_format) is the most common way DNA and protein sequences are stored and used by bioinformatics software. The first line contains information about the sequence (the metadata), including the NCBI accession number (`P00407.1`), sequence name (`COX2_XENLA`) and description (`RecName: Full=Cytochrome c oxidase...`). The next lines represent the amino acid sequence of the protein using the standard one-letter amino acid abbreviations (https://en.wikipedia.org/wiki/Proteinogenic_amino_acid#General_chemical_properties). This is the sequence that comprises the COII protein in this species.

5. **Download the new sequence** for *Xenopus laevis* COII. There are several ways to do this, but the R package `sativum` provides convenient tools. We will need the sequence’s GenBank ID or “GI” number. (Sorry, this is different from the accession number!) On the NCBI search results, notice that below the description of each sequence, in smaller print, are two numbers labeled “Accession” and “GI”. The first is the NCBI accession number and the second is the GenBank GI number we need. You can copy the GI number and paste it into an R command using the function `fetch.sequence` which will download the sequence matching that GI number into your instance of R. The code below will store that sequence information in a new data object that we’ll call `frog.sequence`.

```
> frog.sequence <- fetch.sequence(ids = "117049", database = "protein", format = "AA")
```

The sequence will appear in the console, as shown below.

```
AAStringSet object of length 1:
    width seq                                               names
[1]   229 MAHPSQLGFQDAASPIMEELLHF...MPIVVEAVPLTDFENWSSSMLEA Xenopus_laevis_P0...
```

To better understand how we use R, let’s dissect what’s happening in this command. The `fetch.sequence` function requires several “arguments”. The first one, `ids`, take the GI number that uniquely identifies the sequence we want. We got that from the NCBI website. (We could have also used another function, `search.for.ncbi.ids`, to search for the GI number using the GenBank accession number or a description of the sequence we want.) The next argument, `database`, is set to `"protein"` to indicate that we’d like to get data from the NCBI protein database. The default for this function is to use the nucleotide database. The last argument, `format`, is set to `"AA"` indicating that we want to the results returned to us in amino acid format. Each function in R takes different arguments, and some functions don’t need any arguments at all. (For example `ls()` will list all the objects currently in memory.) About now, you might be wondering how you’re meant to *know* all this stuff! The good news is that you don’t have to. The more you use R, the more you’ll just become familiar with common functions. But for functions that are new to you, you can always get a help page that describes how the function works, what arguments it takes and what it’s output will be. You can find a function’s help page by searching for it in the Help tab in Rstudio’s lower right panel. You can also enter `?` before the function name in the R console, as shown below.

```
> ?fetch.sequence
```

Optional: If you're not comfortable with scientific names, like *Xenopus laevis* or if you simply want this sequence's name to match the style of the others, you can change it as shown below.

```
> names(frog.sequence) <- "frog P00407.1 COII [Xenopus laevis]"
```

6. **Combine** the old and new COII sequences into one dataset using the `combine.sequences` function.

```
> combined.coii <- combine.sequences(fish.COII, frog.sequence)
```

This command will provide a short summary of what it did, as shown below.

```
Combined 5 and 1 sequences: 6
```

7. **Save** this dataset to a file in your R working directory. We shouldn't need to do anything with this file. However, it's best practice to save your dataset to a file at the stage just before beginning your analysis.

```
> write.fasta(combined.coii, filename = "combined.coii.fasta")
```

If you click on the File tab in the lower right panel of Rstudio, a new file should appear called `combined.coii.fasta`

8. **Align the sequences**. Inspect the combined sequences by calling up the object `combined.coii`.

```
> combined.coii
```

The sequence data will appear as output in the console.

```
AAStringSet object of length 6:
    width seq                                               names
[1]   230 MAHPSQLGFQDAASPVMEELLHF...PIVVEAVPLEHFESWSSLMLEEA shark NP_008526.1...
[2]   230 MAHPSQLGLQDAASPVMEELLHF...PIVLEAIPLDPFEDWSSSMLEEA coelacanth NP_008...
[3]   230 MAHPSQLGLQDAASPVMEELIHF...PIVVEAAPLQHFENWSSLMLEKA lungfish NP_00826...
[4]   230 MAHPAQLGFQDAASPVMEELLCF...PIVVEAVPLEFFENWSSAMLEDA zebrafish NP_0593...
[5]   229 MAHPAQLGFQDAASPIMEELLHF...MPIVVEAVPLKTFENWSSSMLET toad YP_001122775...
[6]   229 MAHPSQLGFQDAASPIMEELLHF...MPIVVEAVPLTDFENWSSSMLEA frog P00407.1 COI...
```

Notice that the amino acids line up pretty well, but not perfectly. The amphibian sequences are shifted on the right. We need to "align" the sequences so that the amino acids that are homologous (the equivalent characters in our dataset) appear in the same positions. Sequence alignment is a routine step in any phylogenetic analysis. While you can do this manually, it's more reproducible (and faster) to use software. Several algorithms have been developed for alignment of multiple sequences of amino acids or nucleotides. We will use a common algorithm for multiple sequence alignment (MSA) called ClustalOmega (http://www.clustal.org/omega/). The code below will create a new object called `coii.alignment` using the sequences in the `combined.coii` object as input to the function `align.sequences`.

```
> coii.alignment <- align.sequences(
>   combined.coii,
>   method = "ClustalOmega",
>   order = "input"
> )
```

Notice that this command breaks across multiple lines. That's optional. You could also enter the same command into the console as `coii.alignment <- align.sequences(combined.coii, method = "ClustalOmega", order = "input")`

Either way, the command will provide a short summary of what it did, as shown below.

```
using Gonnet
Aligned 6 sequences using method ClustalOmega
```

To see the full alignment we can enter the following command into the console.

```
> print(coii.alignment, show="complete")
```

It will produce the following output in the console.

```
MsaAAMultipleAlignment with 6 rows and 230 columns
    aln (1..54)                                            names
[1] MAHPSQLGFQDAASPVMEELLHFHDHTLMIVFLISTLVLYIIMAMVSTKLTNKY shark NP_008526.1...
[2] MAHPSQLGLQDAASPVMEELLHFHDHALMIVFLISTLVFYIILAMMTTKMTDKY coelacanth NP_008...
[3] MAHPSQLGLQDAASPVMEELIHFHDHALMIVFLISTLVLYIIVAMVSTKFTNKF lungfish NP_00826...
[4] MAHPAQLGFQDAASPVMEELLCFHDHALMIVFLISTLVLYIIIAMVSTKLTNKF zebrafish NP_0593...
[5] MAHPAQLGFQDAASPIMEELLHFHDHALMAVFLISTLVLYIITTMMTTKLTNTN toad YP_001122775...
[6] MAHPSQLGFQDAASPIMEELLHFHDHTLMAVFLISTLVLYIITIMMTTKLTNTN frog P00407.1 COI...
Con MAHPSQLGFQDAASPVMEELLHFHDHALMIVFLISTLVLYII?AM??TKLTNK? Consensus

    aln (55..108)                                          names
[1] ILDSQEIEIVWTILPAVILIMIALPSLRILYLMDEINDPHLTIKAMGHQWYWSY shark NP_008526.1...
[2] ILDAQEIEIVWTLLPAIVLILVALPSLRILYLIDEVENPHLTIKAMGHQWYWSY coelacanth NP_008...
[3] ILDSQEIEIVWTILPAVILIMIALPSLRILYLMDEINDPHLTVKAVGHQWYWSY lungfish NP_00826...
[4] ILDSQEIEIVWTVLPAIILILIALPSLRILYLMDEINDPHVTIKAVGHQWYWSY zebrafish NP_0593...
[5] AMDAQEIEMVWTIMPAIILIVIALPSLRILYLMDEISDPHLTVKAIGHQWYWSY toad YP_001122775...
[6] LMDAQEIEMVWTIMPAISLIMIALPSLRILYLMDEVNDPHLTIKAIGHQWYWSY frog P00407.1 COI...
Con ILD?QEIEIVWTILPAIILIMIALPSLRILYLMDEINDPHLTIKA?GHQWYWSY Consensus

    aln (109..162)                                         names
[1] EYTDYEDLGFDSYMIQTQDLTPGQFRLLETDHRMVVPMESPIRVLVSAEDVLHS shark NP_008526.1...
[2] EYTDYEELSFDSYMTPLQDLNPGQFRLLETDHRMVIPMESLIRVLISAEDVLHS coelacanth NP_008...
[3] EYSDYETLNFDSYMTPTQDLTPGQFRLLETDYRMVVPMESPIRVLITADDVIHS lungfish NP_00826...
[4] EYTDYENLEFDSYMVPTQDLTPGGFRLLETDHRMVVPKESPIRILVSAEDVLHS zebrafish NP_0593...
[5] EFTNYEDLAFDSYMIPTQDLSPGQFRLLEVDNRMVVPMESPTRMLITAEDVLHS toad YP_001122775...
[6] EYTNYEDLSFDSYMIPTNDLTPGQFRLLEVDNRMVVPMESPTRLLVTAEDVLHS frog P00407.1 COI...
Con EYTDYEDL?FDSYMIPTQDLTPGQFRLLETDHRMVVPMESPIRVL??AEDVLHS Consensus

    aln (163..216)                                         names
[1] WTVPALGVKMDAVPGRLNQTAFIISRPGVYYGQCSEICGANHSFMPIVVEAVPL shark NP_008526.1...
[2] WAVPALGVKMDAVPGRLNQITFMISRPGLYYGQCSEICGANHSFMPIVLEAIPL coelacanth NP_008...
[3] WAVPALGIKMDAVPGRLNQASFITARPGMFYGQCSEIWGANHSFMPIVVEAAPL lungfish NP_00826...
[4] WAVPSLGIKMDAVPGRLNQTAFIVSRPGVFYGQCSEICGANHSFMPIVVEAVPL zebrafish NP_0593...
[5] WAVPALGIKTDAIPGRLNQTSFIATRPGVFYGQCSEICGANHSFMPIVVEAVPL toad YP_001122775...
[6] WAVPSLGVKTDAIPGRLHQTSFIATRPGVFYGQCSEICGANHSFMPIVVEAVPL frog P00407.1 COI...
Con WAVPALG?KMDAVPGRLNQTSFI?SRPGVFYGQCSEICGANHSFMPIVVEAVPL Consensus

    aln (217..230)  names
[1] EHFESWSSLMLEEA shark NP_008526.1...
[2] DPFEDWSSSMLEEA coelacanth NP_008...
[3] QHFENWSSLMLEKA lungfish NP_00826...
[4] EFFENWSSAMLEDA zebrafish NP_0593...
[5] KTFENWSSSMLET- toad YP_001122775...
[6] TDFENWSSSMLEA- frog P00407.1 COI...
Con ??FENWSSSMLE?A Consensus
```

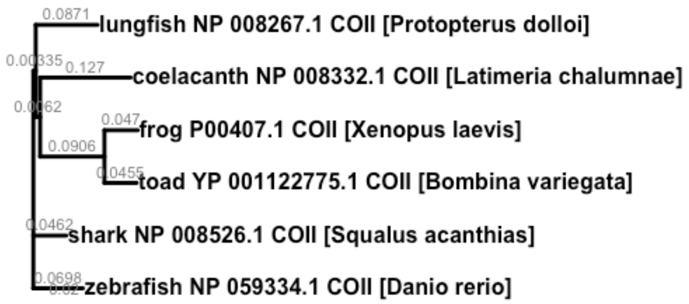
At this point the sequences are aligned, meaning that each column represents a homologous character. Notice that the character states (in this case the specific amino acids) are all identical at some positions (e.g. position 1), mostly identical at others (e.g. position 55) and very different at some (e.g. position 217). This dataset is what we need to build a phylogenetic tree that will describe the relationships between these species.

9. **Build a phylogeny.** There are many methods to infer or build phylogenetic trees from aligned biological sequence data. The neighbor-joining (NJ) method of creating a tree is based on the similarities and differences (genetic distance) that existed between the sequences in the alignment. We will use the NJ method to create a new data object that we'll call, `coii.tree`, by calling the `infer.phylogeny` function.

```
> coii.tree <- infer.phylogeny(coii.alignment)
```

This command will produce a short summary output in the console and plot a phylogeny in the graphics panel.

```
Inferred phylogeny for 6 aligned sequences using NJ with model Blosum62
```



0.0871
lungfish NP 008267.1 COII [Protopterus dolloi]
0.00335
0.127
coelacanth NP 008332.1 COII [Latimeria chalumnae]
0.0062
0.047
frog P00407.1 COII [Xenopus laevis]
0.0906
0.0455
toad YP 001122775.1 COII [Bombina variegata]
0.0462
shark NP 008526.1 COII [Squalus acanthias]
0.0698
zebrafish NP 059334.1 COII [Danio rerio]

We have a tree! Next we'll want to improve the way it is displayed so we can better understand and interpret this result. Let's start by examining the structure of the `coii.tree` object that's been created. We can examine its components using the `names` command.

```
> names(coii.tree)
```

This will list the "names" of the different components of the the `coii.tree` data object.

```
[1] "edge"        "edge.length" "tip.label"   "Nnode"
```

Each of these components can be individually displayed. For example:

```
> coii.tree$edge.length
```

This will print out the numerical values below.

```
[1] 0.045489724 0.047007562 0.090636429 0.127461773 0.006200215 0.087122735
[7] 0.069773605 0.046206996 0.003349739
```

These values are the genetic distances along each branch (or "edge") of the tree that was just plotted above. **Genetic distance** is a measure of percent change in the sequence data from the most recent ancestral node to the taxa at the tips of the tree. Therefore it can range from 0 to 1. Genetic distance is used in the NJ method to calculate the lengths of the tree branches.

10. Let's do some things to clean up this tree and make it easier to interpret. The labels on the tree are long and have more information than we need. So, let's **make tip labels simpler**. We can examine them first by calling the `tip.label` component of the `coii.tree` data object

```
> coii.tree$tip.label
```

The current name of each taxon (or tip) will be appear as below.

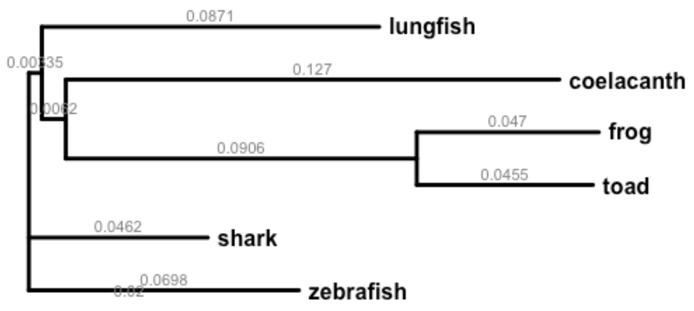
```
[1] "shark NP_008526.1 COII [Squalus acanthias]"
[2] "coelacanth NP_008332.1 COII [Latimeria chalumnae]"
[3] "lungfish NP_008267.1 COII [Protopterus dolloi]"
[4] "zebrafish NP_059334.1 COII [Danio rerio]"
[5] "toad YP_001122775.1 COII [Bombina variegata]"
[6] "frog P00407.1 COII [Xenopus laevis]"
```

We don't need the NCBI accession numbers anymore, and we know these are all COII sequences. So let's just use the short common names for each taxon. We can assign new values to `coii.tree$tip.label` that we choose. (We just need to be sure to maintain the same order. Otherwise we will change the relationships represented in the tree.)

```
> coii.tree$tip.label <- c("shark","coelacanth","lungfish","zebrafish","toad","frog")
```

Let's draw the tree again to see the effect that's had.

```
> draw.tree(coii.tree)
```



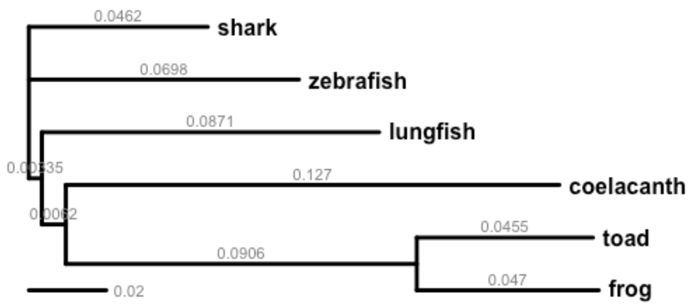
11. Next, notice that the shark, which is our outgroup for the bony fish, appears in the middle of the tree. Let's **rotate nodes** in the tree to place the outgroup at the top. We'll do this by modifying the `coii.tree` object using the `rotateConstr` function. (Some people prefer the outgroup to be at the bottom. This is a subjective preference, but it's helpful to put the outgroup to one extreme or the other.) This rotation won't change any relationships shown in the tree, but it should make it easier to interpret relationships. The order of the species listed will be the order in which the nodes appear.

The indentation in the code below is optional, but it helps keep the code organized. For example, this indent pattern reminds a reader that the most indented lines are surrounded by the parentheses of the `constraint` argument.

```
> coii.tree <- rotateConstr(
>   coii.tree,
>   constraint = c(
>     "frog", "toad", "coelacanth",
>     "lungfish", "zebrafish", "shark"
>   )
> )
```

If the `rotateConstr` command works properly, you won't see any messages or output. We can view the effect to plotting the tree again using the function `draw.tree`.

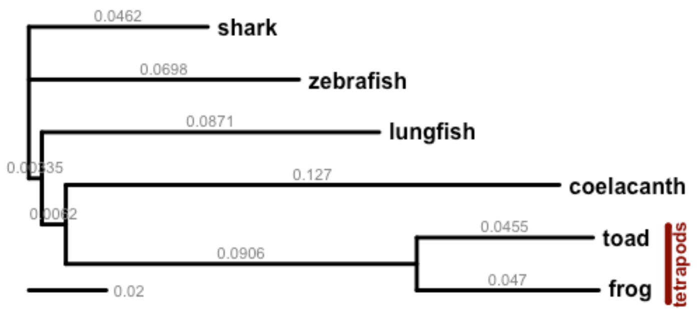
```
> draw.tree(coii.tree)
```



Now we have the outgroup, “shark”, at the top of the tree.

12. Finally, we can label the clade that contains our tetrapod species, the frog and toad, using the `label.clade` function. This function takes several arguments: `tree` is the tree object. The monophyletic group we want to label is defined by `clade`. (If we wanted to use this on a large group, we’d only need to name two taxa in it that define the outer-most members of the group.) The label itself is given by the `text` argument. The last argument `offset` is optional; ot just moves the label a bit to the right so it doesn’t run into the tip labels.

```
> label.clade(
>   tree = coii.tree,
>   clade = c("frog","toad"),
>   text = "tetrapods",
>   offset = 2
> )
```



Questions

Use this tree to **answer the questions below** regarding relationships among fish and tetrapods. Check your conclusions with your instructor.

- According to this tree, which is the group of fish most closely related to the tetrapods?
- Explain what the genetic distance for the Lungfish represents.
- Based on this tree, are the bony fish a monophyletic group? Explain your answer.