10. Phylogenetic Diversity - Traits

Z620: Quantitative Biodiversity, Indiana University

OVERVIEW

Up to this point, Quantitative Biodiversity has focused on aspects of taxonomic diversity, a form of classification that ignores traits and the evolutionary relatedness of individuals in a community.

In this Handout, we introduce concepts and tools that will allow you to integrate phylogenetic information to test and better understand how evolutionary history influences patterns of biodiversity while considering traits of those organisms.

After completing this exercise you will be able to:

1. create phylogenetic trees to view evolutionary relationships from sequence data
2. map the traits of taxa onto phylogenetic trees
3. test for phylogenetic signal within trait distributions and trait-based patterns of biodiversity
4. control for the relatedness of taxa to identify evolutionary unbiased correlations among traits

BACKGROUND

Functional traits are morphological, behavioral, or physiological characteristics of an individual that affect its performance or fitness under a set of environmental conditions. We can think of functional traits as falling into two general categories. First, "response traits" influence how an individual contends with features of its environment (Lavorel and Garnier 2002). For example, waxy leaves are generally thought to aid in plant tolerance to desiccation. Second, "effect traits" are characteristics of an individual that can alter the environment (Lavorel and Garnier 2002). For example, the ability for a plant and its associated microorganisms to fix atmospheric nitrogen can influence soil fertility. It is often assumed that more closely related (i.e., more recently diverged) taxa have a greater degree of trait similarity than more distantly related taxa that diverged further back in time. Therefore, it is possible to detect trait-environment and trait-function relationships simply because an assemblage is comprised of closely related taxa. We can not only test for this so-called **phylogenetic signal**, but also correct for it. In order to explore the relationship between the evolutionary relatedness of taxa and the distribution of functional traits among those taxa, we must introduce another primary data structure: the **phylogeny**. A phylogeny is an evolutionary tree constructed by comparing the divergence in gene or protein sequences among a group of organisms. Constructing a phylogeny relies on the assumption that more distantly related organisms have accumulated more differences in their sequences (i.e., substitutions) since the time they last shared a common ancestor, while the sequences of more closely related taxa would be more similar. In the following sections, we will provide an introduction to constructing phylogenetic trees. From this, you will gain evolutionary insight into patterns and processes of biodiversity (e.g., diversification, adaptation, etc.).

1) SETUP

A. Retrieve and Set Your Working Directory

```
rm(list = ls())
getwd()
setwd("~/GitHub/QB-2023/1.HandOuts/10.PhyloTraits")
```

B. Load Packages

Several R packages have been developed for conducting phylogenetic analyses (http://goo.gl/DtU16j). We will describe the packages in greater details in the sections below. The `require()` function in `R` returns `TRUE` if the package was successfully loaded or `FALSE` if the package failed to load. This `for loop` loads each package and installs the package when `require()` returns `FALSE`.

```
package.list <- c("ape", "seqinr", "phylobase", "adephylo", "geiger",
    "picante", "stats", "RColorBrewer", "caper", "phylolm", "pmc",
    "ggplot2", "tidyr", "dplyr", "phangorn", "pander", "phytools",
    "vegan", "cluster", "dendextend", "phylogram", "bios2mds")
for (package in package.list) {
    if (!require(package, character.only = TRUE, quietly = TRUE)) {
        install.packages(package)
        library(package, character.only = TRUE)
    }
}

# Some bioinformatics packages come through BioConductor
# Requires own installation method
if (!require("BiocManager", quietly = TRUE)) {
    install.packages("BiocManager")
}
if (!require("msa", quietly = TRUE)) {
    BiocManager::install("msa")
}
library(msa)
```

2) DESCRIPTION OF DATA

We will address question related to functional traits and phylogenetics using a data set that contains information on 39 strains of freshwater bacteria. The bacteria were isolated from two lakes that greatly differ in their biogeochemical properties, especially nutrient content. One lake, Little long (LL), has extremely low concentrations of phosphorus (i.e., oligotrophic) while the other lake, Wintergreen (WG), is adjacent to a bird sanctuary and receives high inputs of phosphorus (i.e., eutrophic). We hypothesize that these two environments contain taxa with different traits that influence their ability to use different types of phosphorus. To test this hypothesis, we extracted DNA from all of bacterial strains and amplified the 16S rRNA gene, which is a commonly used "marker" for delineating microbial taxa and examining phylogenies of bacteria. In addition, we measured the maximum growth rate (μ_{max}) of each strain on 18 different phosphorus resources. With this information, we can measure the **niche breadth** of each strain and consider this a functional trait. In the following sections, we take a phylogenetic approach to mapping phosphorus resource use onto a phylogenetic tree while testing for ecological trade-offs related to resource use. Specifically, we will determine whether organisms exhibit generalist or specialist strategies when it comes to growing on different phosphorus types.

3) SEQUENCE ALIGNMENT

One of the first steps in performing a phylogenetic analysis is to align the sequences that have been obtained from your samples. The sequences can represent nucleotides (DNA or RNA) or proteins (amino acids). Computer algorithms are used to arrange sequences relative to one another based on conserved (i.e., non-variable) regions. The non-conserved or "variable" regions are then used for making phylogenetic inference, including tree construction. There are two different ways to align sequences. **Global alignment** relies on algorithms that attempt to align the entire span of sequences. **Local alignment** is typically a preferable approach and uses algorithms that initiate alignment by identifying regions in a sequence that have a high degree of similarity. In the example below, we will be using a local alignment approach.

A. Examining a FASTA-File

We will read in a FASTA-formatted file into the `muscle` alignment program, a common aligner used for protein and 16S rRNA gene sequences. FASTA is a very common text-based format that is used across platforms and disciplines. The files contain DNA sequences (e.g., CTAAG) and names that are associated with samples. Let us take a second to look at the *p.isolates.fasta* file. We can use the `Biostrings` package to load our sequences into the R environment and do a quick check to make sure they look correct.

```
# Import and view unaligned sequences {Biostrings}
seqs <- readDNAStringSet("data/p.isolates.fasta", format = 'fasta')
seqs # View sequences
```

```
## DNAStringSet object of length 40:
##      width seq                                               names
##  [1]   619 ACACGTGAGCAATCTGCCCTTCT...TTCTCTGGGAATACCTGACGCT LL9
##  [2]   597 CGGCAGCGGGAAGTAGCTTGCTA...AACTGTTCAGCTAGAGTCTTGT WG14
##  [3]   794 CAGCGGCGGACGGGTGAGTAACA...GCTAACGCATTAAGCACTCCGC WG28
##  [4]   716 CTTCAGAGTTAGTGGCGGACGGG...TGCTAGTTGTCGGGATGCATGC LL24
##  [5]   803 ACGAACTCTTCGGAGTTAGTGGC...TAAAACTCAAAGGAATTGACGG LL41A
##  ...   ... ...
## [36]   652 TTCGGGAGTACACGAGCGGCGAA...TTCTCTGGGAATACCTGACGCT LL46
## [37]   661 GCGAACGGGTGAGTAACACGTGG...GAGCGAAAGCGTGGGTAGCGAA WG26
## [38]   694 GGCGAACGGGTGAGTAACACGTG...ACCCTGGTAGTCCACGCCGTAA WG42
## [39]   699 TACAGGTACCAGGCTCCTTCGGG...AAAGCATGGGTAGCGAACAGGA LLX17
## [40]  1426 TTCTGGTTGATCCTGCCAGAGGT...AACCTNAATTTTGCAAGGGGGG Methanosarcina
```

B. Performing an Alignment

In addition to being one of the best languages for data analysis and statistics, R can also be used to conduct bioinformatic analyses. Occasionally, you may find that an R tool has yet to be developed for a certain task that exists in a different platform (e.g., Bash or Python). Fortunately, there are ways that allow R to interact with other software and languages. Here, we will tap into `Bioconductor`, an open-source software project designed for reproducible genomics workflows in R. Specifically, we will use the the `msa` package to perform a multiple sequence aliment using MUltiple Sequence Comparison by Log- Expectation (MUSCLE).

Let us align the sequences we loaded into the R Studio environment.

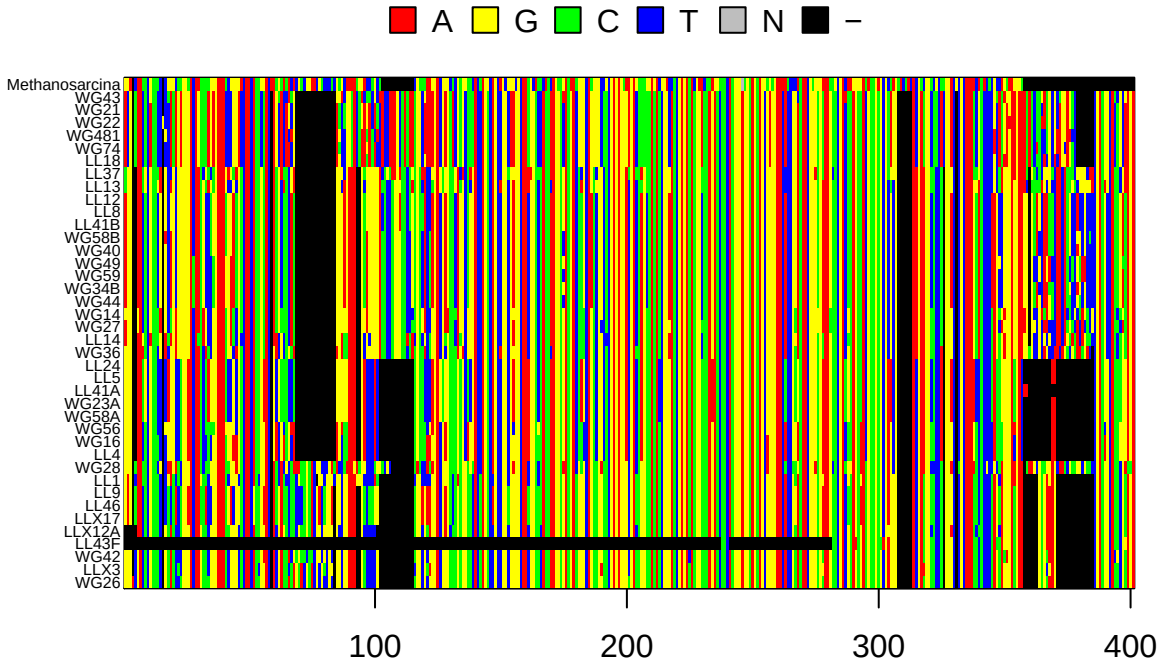
```
# Align sequences using default MUSCLE parameters {msa}
read.aln <- msaMuscle(seqs)

# Save and export the alignment to use later
```

```
save.aln <- msaConvert(read.aln,type = "bios2mds::align")
export.fasta(save.aln,"./data/p.isolates.afa")
```

C. Visualizing the Alignment

After performing the alignment, it is good practice to view the aligned sequence file. From this, one can visualize conserved and variable regions among sequences. This may not be practical for databases with a large number of long sequences, such as more complex phylogenomic reconstructions. In our dataset, however, the alignment only contains sequences for 39 bacterial taxa plus a distantly related reference sequence that we will use as an **outgroup**. To visualize our alignment, we will first, using the `ape` package, convert the alignment file into a DNAbin object, which is a bit-level coding scheme that allows R to store and manipulate sequence data. Then, we will visualize the alignment by color-coding nucleotides different colors, where red = adenine (A), yellow = guanine (G), green = cytosine (C), blue = thymidine (T), grey = ambiguous call, and black = alignment gaps. (Alignment gaps are sites that are introduced during an alignment to increase matching and can reflect insertion and deletions.)

```
# Convert Alignment to DNAbin Object {ape}
p.DNAbin <- as.DNAbin(read.aln)

# Identify Base Pair Region of 16S rRNA Gene to Visualize
window <- p.DNAbin[, 100:500]

# Command to Visualize Sequence Alignment {ape}
image.DNAbin(window, cex.lab = 0.50)
```



An inspection of the figure suggests that our alignment is good because many vertical bars are aligned with the same color, indicating they share the same nucleotide at that site. If this visualization showed a jumbled mess, we may want to realign our sequences using different parameters or a different programs. If that does not improve the alignment, you might want to consider the quality of the data. Sometimes we may also find that some areas poorly align and need to be removed from our final alignment to make sure we only retain phylogenetically informative sites. Tools like GBlocks or TrimAI can be very useful. GBlocks can be implemented with the `ips` package.

4) MAKING A PHYLOGENETIC TREE

Once you have aligned your sequences, the next step is to construct a phylogenetic tree. Phylogenetic trees are an effective way to visualize the evolutionary relationship among taxa. In addition, phylogenetic trees are often required for making quantitative and statistical evolutionary inferences. There are many algorithms, theories, and tools that are used for making phylogenetic trees. It is not possible to cover all of these topics in Quantitative Biodiversity, but we try to highlight some of the major concepts in this handout. The following table provides a summary:

Evolutionary Model	**Properties**
Jukes-Cantor model (JC69)	Simplest model. Assumes all nucleotides occur at equal frequencies and that these nucleotides can mutate from one to another with equal probability.
Felsenstein model (F81)	Builds from JC69 by allowing nucleotide frequencies to vary.
Kimura model (K80)	Assumes equal frequencies of nucleotides, but recognizes that transition mutations (e.g., purine to a purine [A -> G] or pyrimidine to pyrimidine [C -> T]) occur with higher probability than transversion mutations (e.g., purine to pyrimidine or vice-versa).
Felsenstein model (F84)	Assumes different rates of base transitions and transversions while allowing for differences in base frequencies.
Tamura model (T92)	Similar to K80 but accounts for G + C content.
General Time Reversible model (GTR)	Captures the probabilities associated with nucleotide reversions (e.g., T -> C -> T). All substitution rates are different. Does not assume equal base frequencies.

Fortunately, the `dist.dna` function that we used above for generating the neighbor joining tree can easily create distance matrices for a large number of DNA substitution models. Let's create a distance matrix among our bacterial isolates based on the Felsenstein (F84) substitution model using the `dist.dna` function in `ape`.

A. Neighbor Joining Trees

Nearest neighbor joining takes a distance matrix that specifies the distance between each pair of taxa as input. The algorithm starts with a tree where all taxa are equally related, finds and joins the taxa with the lowest distance into a new node, re-calculates the distance between all the remaining taxa and the new node, and repeats this process until all the taxa are accounted for.

Neighbor joining is commonly used for making a "guide tree" that can incorporate more sophisticated models of evolution. The first step in making a neighbor joining tree is to create a **distance matrix**. We will do this with the `dist.dna()` function in the `ape` package. In the R chunk below, the "model = raw" argument

means that `dist.dna` will estimate distances based on the proportion of sites that differ between pairs of sequences. The "pairwise.deletion = FALSE" argument means that an entire site (i.e., column) is deleted if there is a missing observation (i.e., a gap) for one or more of the sequences (i.e., rows).

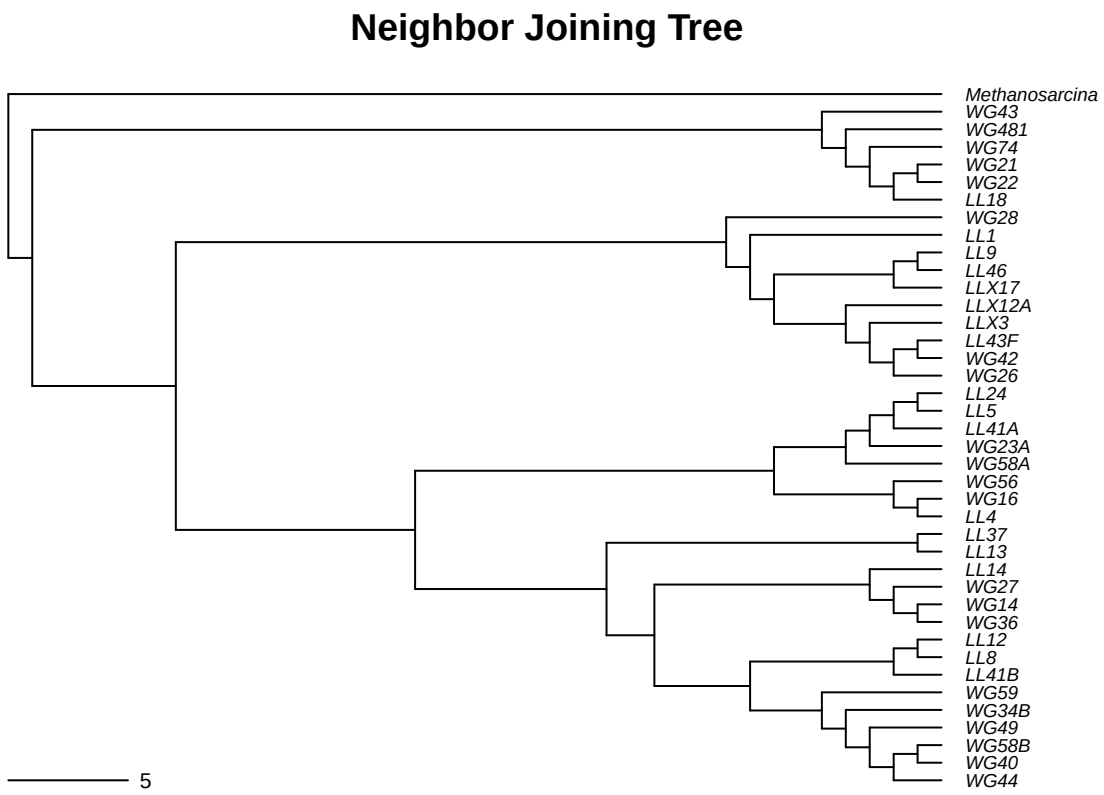
```
# Create Distance Matrix with "raw" Model {ape}
seq.dist.raw <- dist.dna(p.DNAbin, model = "raw", pairwise.deletion = FALSE)
```

After calculating the distances between our sequences, we are now ready to make a neighbor joining tree using the `bionj()` function in `ape`. In the process, we will use *Methanosarcina* as a reference to determine the evolutionary relationship among the members of our tree (i.e., an **outgroup**). By using an outgroup we have a point of comparison for the rest of our taxa (i.e., the **ingroup**), which allows us to **root** the tree. Ideally, the outgroup should be a taxon that you know all of your taxa diverged from farther back in the past than when they diverged from each other. In our case *Methanosarcina* is a microbe belonging to the Archaea, a domain of life that is distinct from the bacteria in our data set.

To store phylogenies we will be using the `phylo` class, the basic phylogenetic data structure used in R. A `phylo` object stores four objects: 1) the tip labels (i.e., taxa) of the tree, 2) the number of nodes (i.e., the inferred most recent common ancestor) on the tree, 3) the names of the edges (i.e., the evolutionary distance between nodes) on the tree, and 4) and the length of the edges.

At this point, we can plot our tree:

```
# Neighbor Joining Algorithm to Construct Tree, a 'phylo'
# Object {ape}
nj.tree <- bionj(seq.dist.raw)

# Identify Outgroup Sequence
outgroup <- match("Methanosarcina", nj.tree$tip.label)

# Root the Tree {ape}
nj.rooted <- root(nj.tree, outgroup, resolve.root = TRUE)

# Plot the Rooted Tree{ape}
par(mar = c(1, 1, 2, 1) + 0.1)
plot.phylo(nj.rooted, main = "Neighbor Joining Tree", "phylogram",
    use.edge.length = FALSE, direction = "right", cex = 0.6,
    label.offset = 1)
add.scale.bar(cex = 0.7)
```



B) SUBSTITUTION MODELS OF DNA EVOLUTION

You now have a distance matrix that represents the shared evolutionary history of your taxa. However, these are raw estimates of phylogenetic distance and they do not account for the fact that multiple substitutions may have occurred at the same site over time (e.g., A -> T -> G -> C would be counted as one change of base), something that is likely common if you are dealing with taxa that diverged hundreds of millions of years ago. In addition, raw distance measures do not take into account substitution biases for one particular nucleotide over the other (e.g., A-T biased mutation). Various models have been developed to account for multiple substitutions and estimate the rates that nucleotides change over time. In the following table, we describe a few of the more commonly used models that correct for multiple substitutions as well as the probability of one nucleotide transitioning to a different nucleotide. We will then correct the raw distances of your matrix so that more evolutionary informed conclusions can be drawn.

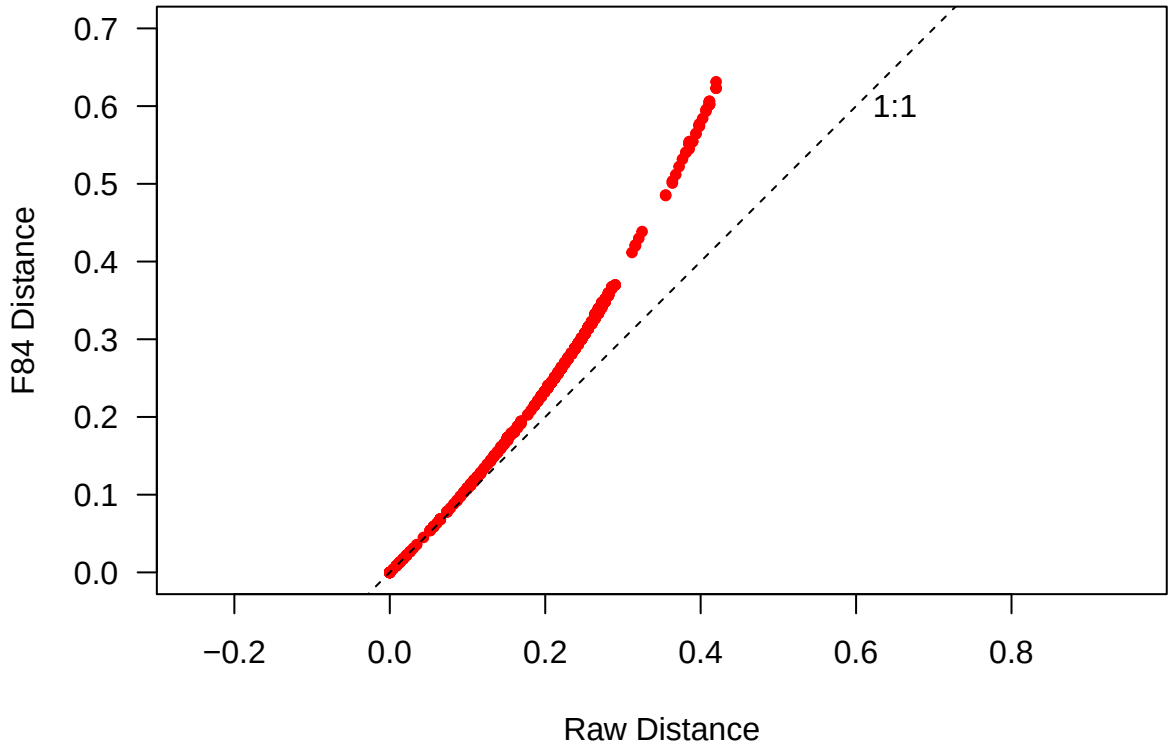
Evolutionary Model	Properties
Jukes-Cantor model (JC69)	Simplest model. Assumes all nucleotides occur at equal frequencies and that these nucleotides can mutate from one to another with equal probability.
Felsenstein model (F81)	Builds from JC69 by allowing nucleotide frequencies to vary.
Kimura model (K80)	Assumes equal frequencies of nucleotides, but recognizes that transition mutations (e.g., purine to a purine [A -> G] or pyrimidine to pyrimidine [C -> T]) occur with higher probability than transversion mutations (e.g., purine to pyrimidine or vice-versa).
Felsenstein model (F84)	Assumes different rates of base transitions and transversions while allowing for differences in base frequencies.
Tamura model (T92)	Similar to K80 but accounts for G + C content.
General Time Reversible model (GTR)	Captures the probabilities associated with nucleotide reversions (e.g., T -> C -> T). All substitution rates are different. Does not assume equal base frequencies.

Fortunately, the `dist.dna` function that we used above for generating the neighbor joining tree can easily create distance matrices for a large number of DNA substitution models. Let us create a distance matrix among our bacterial isolates based on the Felsenstein (F84) substitution model using the `dist.dna()` function in `ape`.

```
# Create distance matrix with "F84" model {ape}
seq.dist.F84 <- dist.dna(p.DNAbin, model = "F84", pairwise.deletion = FALSE)
```

Now, let us compare the "raw" and "F84" distance matrices. First, we will make a **saturation plot**, which allow us to visualize the effect of our DNA substitution model compared to a distance matrix that does not include this feature.

```
# Plot Distances from Different DNA Substitution Models
par(mar = c(5, 5, 2, 1) + 0.1)
plot(seq.dist.raw, seq.dist.F84,
     pch = 20, col = "red", las = 1, asp = 1, xlim = c(0, 0.7),
     ylim = c(0, 0.7), xlab = "Raw Distance", ylab = "F84 Distance")
abline(b = 1, a = 0, lty = 2)
text(0.65, 0.6, "1:1")
```

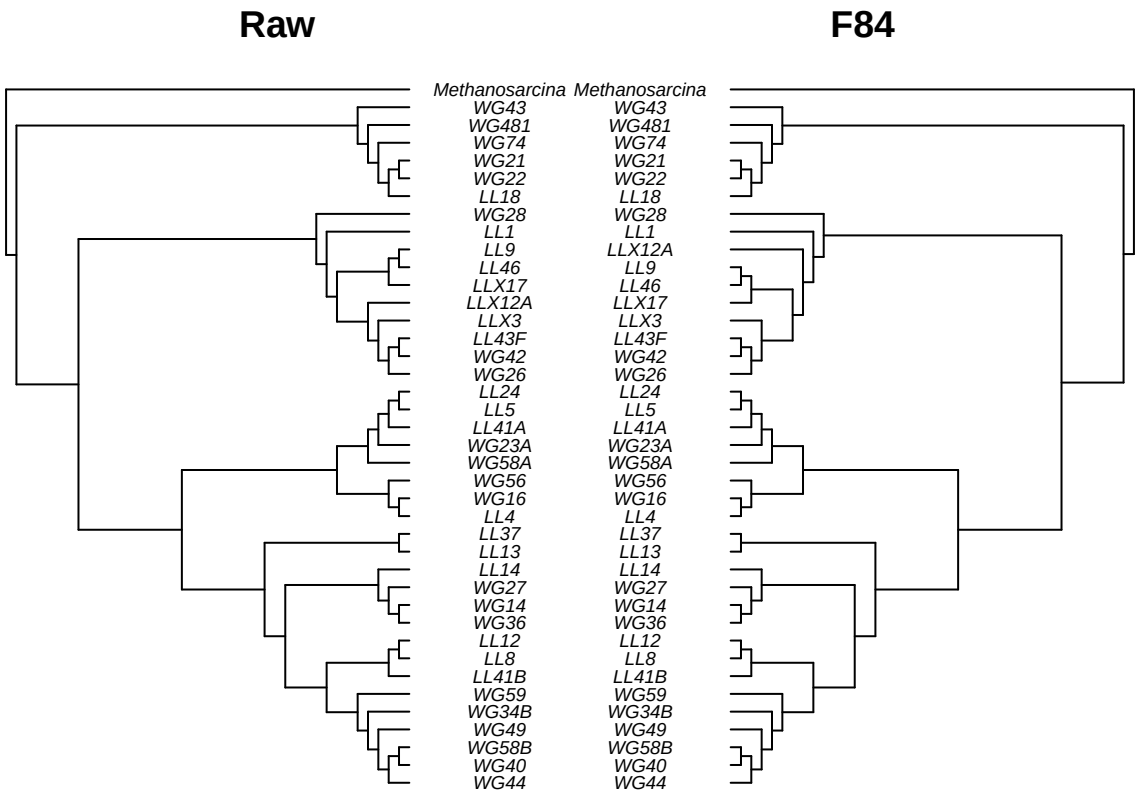


We can see from this figure that the F84 distance tends to be greater than the raw distance, so it is **correcting** for multiple substitutions

Second, we will assess how the "raw" and "F84" distance matrices affect tree topology by plotting each tree next to each other (also known as a **cophylogenetic plot**). This will give us a visual sense of whether or not the two trees are in agreement with one another.

```
# Make Neighbor Joining Trees Using Different DNA Substitution Models {ape}
raw.tree <- bionj(seq.dist.raw)
F84.tree <- bionj(seq.dist.F84)

# Define Outgroups
raw.outgroup <- match("Methanosarcina", raw.tree$tip.label)
F84.outgroup <- match("Methanosarcina", F84.tree$tip.label)

# Root the Trees {ape}
raw.rooted <- root(raw.tree, raw.outgroup, resolve.root = TRUE)
F84.rooted <- root(F84.tree, F84.outgroup, resolve.root = TRUE)

# Make Cophylogenetic Plot {ape}
layout(matrix(c(1, 2), 1, 2), width = c(1, 1))
par(mar = c(1, 1, 2, 0))
plot.phylo(raw.rooted, type = "phylogram", direction = "right",
      show.tip.label = TRUE, use.edge.length = FALSE, adj = 0.5,
      cex = 0.6, label.offset = 2, main = "Raw")
par(mar = c(1, 0, 2, 1))
plot.phylo(F84.rooted, type = "phylogram", direction = "left",
```

```
      show.tip.label = TRUE, use.edge.length = FALSE, adj = 0.5,
      cex = 0.6, label.offset = 2, main = "F84")
```



C) QUANTIFYING PHYLOGENETIC SIMILARITY

Sometimes we will want to quantify the difference between two phylogenies rather than simply plotting them side-by-side. The package **ape** includes two ways to calculate this difference: 1) the branch length score (BLS) (Kuhner and Felsenstein, 1994) and the symmetric difference (Robinson and Foulds 1981). The symmetric difference is simply the number of partitions (i.e., the number of ways to group the taxa in our tree) that are in one tree and not the other. This is a very simple calculation, but it is extremely sensitive to small differences between the trees. The branch length score between trees T and T' can be described as $BLS(T, T') = \sum_{i=1}^{N}(t_i - t_i')^2$, where t_i is the branch length if the ith partition is in both trees and is zero if it is not. This function should look familiar, since it is just the squared Euclidean distance between two trees and it ranges from values of 0 to 1. The history, appropriateness, and limitations of these methods are nicely discussed in chapter 30 of Felsenstein (2004).

Here we will calculate branch length score between the raw tree and the tree calculated with F84 distances.

```
# Set method = 'PH85' for the symmetric difference Set
# method = 'score' for the symmetric difference This
# function automatically checks for a root and unroots
# rooted trees Can then pass it either the rooted or
# unrooted tree and get same answer
dist.topo(raw.rooted, F84.rooted, method = "score")
```

```
##            tree1
## tree2 0.04219896
```

D) MAXIMUM LIKELIHOOD TREE

Often a nearest neighbor joined tree does not accurately reconstruct the real phylogeny. While neighbor joining is very fast and has the convenient property of returning the correct tree for a correct distance matrix, we are rarely in the position where we know the correct distance matrix. Neighbor joining has some other shortcomings. First, it only uses a distance matrix, which does not take into account specific nucleotide states. Second, it only gives one tree, meaning that it is not a statistical method. Last, it can be sensitive to the underlying substitution model.

Instead, a maximum likelihood (ML) tree is often a superior choice, as it is built on the robust statistical procedure of ML (i.e., the process of finding the parameter value that maximizes the likelihood of the data). In addition, ML has the advantage of being the estimation method that is least affected by sampling error, is fairly robust to the assumptions of a particularly model, allows you to compare different trees, and takes into account nucleotide states (as opposed to just distance).

A downside of ML estimation is that it is a computationally intensive process. ML estimation needs to find the tree that gives the data the highest probability, a difficult task that requires the use of optimized algorithms. Because this can take a while, we will not ask you to generate a maximum likelihood tree for your exercises. However, below we have supplied code using functions from the package `phangorn` to generate a ML tree.

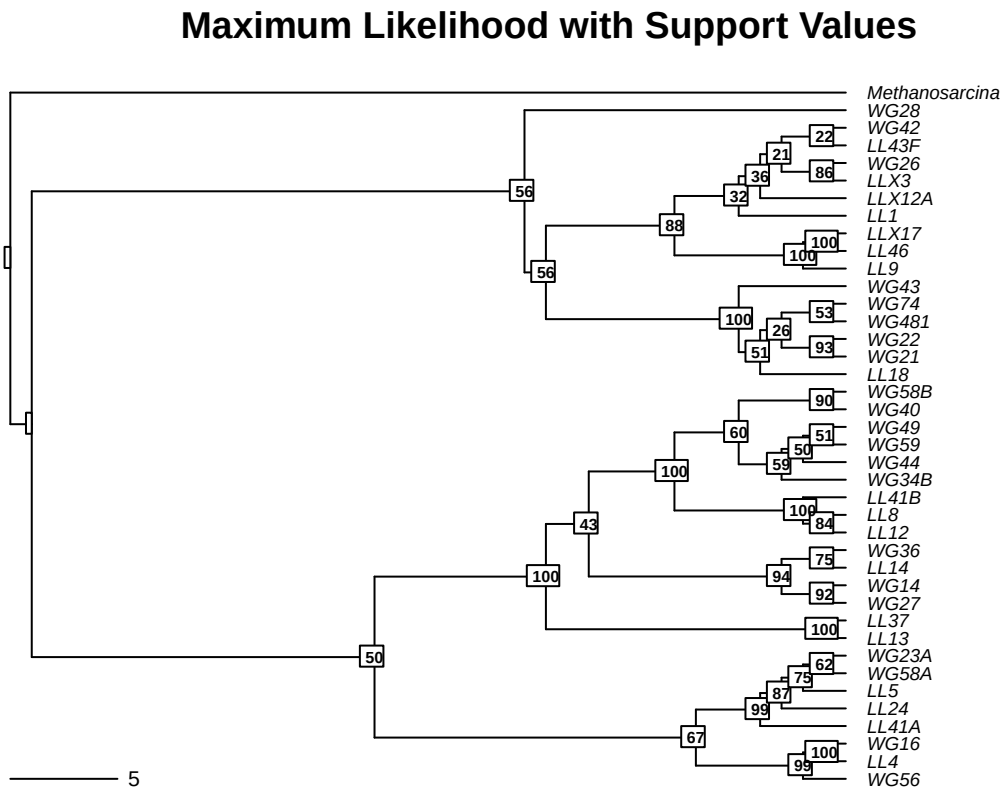
```
# Requires alignment to be read in with as phyDat object
phyDat.aln <- msaConvert(read.aln, type = "phangorn::phyDat")

# Make the NJ tree for the maximum likelihood method.
# {Phangorn} requires a specific attribute (attr) class.
# So we need to remake our trees with the following code:
aln.dist <- dist.ml(phyDat.aln)
aln.NJ <- NJ(aln.dist)

fit <- pml(tree = aln.NJ, data = phyDat.aln)

# Fit tree using a JC69 substitution model
fitJC <- optim.pml(fit, TRUE)

# Fit tree using a GTR model with gamma distributed rates.
fitGTR <- optim.pml(fit, model = "GTR", optInv = TRUE, optGamma = TRUE,
          rearrangement = "NNI", control = pml.control(trace = 0))

# Perform model selection with either an ANOVA test or with AIC
anova(fitJC, fitGTR)
AIC(fitJC)
AIC(fitGTR)
```

E) BOOTSTRAP SUPPORT

Because building a phylogenetic tree is very much a statistical exercise we often want to know how confident we are in the placement of each branch in our phylogeny. For example, say that we make a tree today and tomorrow we run the same code, but get a tree with a node in a different place. How do we know which tree is correct? Since we do not know the real tree, we will use the next best thing. We will re-sample our own data to determine how reliable the tree is. This process is known as **bootstrapping**. To perform bootstrapping,

we will select a subset of columns from our alignment randomly with replacement. Next we will use that sample to construct a new tree and compare its topology to the original tree. Each interior branch of the original tree that is different from the bootstrap tree gets a score of 0 while all other interior branches get a score of 1. This procedure is done a large number of times (typically 100 - 10,000, depending on the size of the tree) and it gives us the **bootstrap value** for each interior branch. A general rule of thumb is to treat bootstrap values 95% or higher as operationally correct. Nodes with >70% have moderate support and nodes of 50% or less are insufficiently resolved. Again, because this procedure is computationally intensive you will plot bootstrap support values for a tree that has been previously generated.

```
ml.bootstrap <- read.tree("./data/ml_tree/RAxML_bipartitions.T1")
par(mar = c(1, 1, 2, 1) + 0.1)
plot.phylo(ml.bootstrap, type = "phylogram", direction = "right",
    show.tip.label = TRUE, use.edge.length = FALSE, cex = 0.6,
    label.offset = 1, main = "Maximum Likelihood with Support Values")
add.scale.bar(cex = 0.7)
nodelabels(ml.bootstrap$node.label, font = 2, bg = "white", frame = "r",
    cex = 0.5)
```



F) A NOTE ON MAKING YOUR OWN TREES

R is useful for a lot of things, but it is not optimized for phylogenetic reconstruction and therefore can be extremely slow. If you need to make a phylogeny with more than a dozen or so taxa, then you may need to use packages that do not easily run in an R environment. Two of the packages that you will want to check out is the (1) Randomized Axelerated Maximum Likelihood (RAxML) package (http://sco.h-its.org/exelixis/web/software/raxml/index.html) and (2) IQ-Tree package (http://www.iqtree.org/) for generating

maximum likelihood trees with bootstrap support. IQ-Tree is particular has the added benefit of performing your model testing as part of your reconstruction pipeline, tends to be more efficient, and has really useful tutorials. If you are dealing with communities with a large number of taxa (e.g., 16S rRNA amplicon data from microbial communities), programs like RAxML may be too slow. Here you can use the program FastTree to generate approximately-maximum-likelihood phylogenetic trees, sacrificing exactness for speed (http://www.microbesonline.org/fasttree/).

If you are interested in learning more about how to get started, Young and Gillung (2019) is a nice primer for phylogenetics and phylogenomics (https://tinyurl.com/3us9b9uu).

To help you with your future tree-building efforts we have included a short script under the folder `Week10-PhyloTraits/bash/` that will allow you to generate a ML tree on the IU computer cluster Carbonate using the program RAxML. If you are an IU student, postdoc, or faculty member you can register for an account on Carbonate. See the IU Carbonate webpage for more information (https://kb.iu.edu/d/aolp).

5) INTEGRATING TRAITS AND PHYLOGENY

In the context of biodiversity, many researchers are interested in being able to map traits onto phylogenetic trees. This can be important for addressing questions related to diversification, trait evolution, and ecological interactions. In this section, we will import functional trait data that was collected on the 39 freshwater isolates for which we have 16S rRNA sequences. Specifically, we measured the growth rates of each strain on 18 different types of phosphorus (e.g., phosphate, DNA, ATP, etc.). With this information we will visualize the functional traits (i.e., growth on different forms of phosphorus) in a phylogenetic context by mapping them onto the neighbor joining tree constructed under the F84 DNA substitution model that we created.

A. Loading Trait Database

```
# Import Growth Rate Data
p.growth <- read.table("./data/p.isolates.raw.growth.txt", sep = "\t",
            header = TRUE, row.names = 1)

# Standardize Growth Rates Across Strains
p.growth.std <- p.growth / (apply(p.growth, 1, sum))
```

B. Trait Manipulations

From the section above where we described the data, you will recall that we were interested in answering questions related to trait evolution and ecological interactions. Specifically, we will be testing for a well-known trade-off related to resource use. This trade-off contends that there are different advantages to growing exceptionally well on one or a few resources, versus being able to grow moderately well on many resources, i.e., a **generalist-specialist trade-off** (MacArthur and MacArthur, 1961).

To explore the generalist-specialist trade-off using phylogenetic data, we need to create two new variables. First, we need to calculate the maximum growth rate (μ_{max}) of each isolate across all phosphorus types. This will help us test the expectation that generalists will have lower maximum growth rates because there is a cost associated with being able to use a lot of different chemical types of phosphorus. In contrast, we expect that specialists will have a high maximum growth rate on their preferred phosphorus resource.

```
# Calculate Max Growth Rate
umax <- (apply(p.growth, 1, max))
```

Second, we need to come up with a way of quantifying whether a strain is a generalist or a specialist. We will use the niche breadth (nb) index from Levins (1968) which is defined as $\frac{1}{(n \cdot \Sigma p_{xi}^2)}$, where n is the total number of resources and p_{xi} is the proportion of observed growth for isolate i on each resource (x).

```
levins <- function(p_xi = ""){
  p = 0
  for (i in p_xi){
    p = p + i^2
    }
  nb = 1 / (length(p_xi) * p)
  return(nb)
  }
```

Let us apply our `nb()` function to the isolate growth rate data on the different phosphorus resources.

```
# Calculate Niche Breadth for Each Isolate
nb <- as.matrix(levins(p.growth.std))

# Add Row Names to Niche Breadth Matrix
nb <- setNames(as.vector(nb), as.matrix(row.names(p.growth)))
```

C. Visualizing Traits on Trees

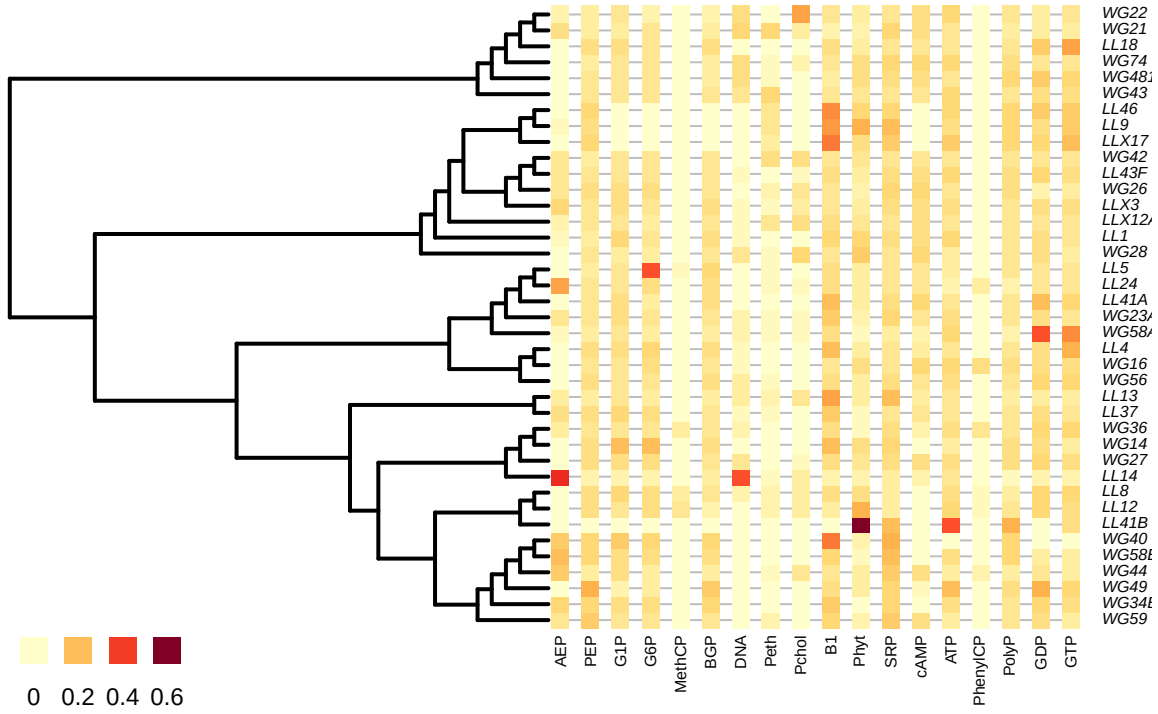
Now that we have identified and calculated our traits of interest, we can map those traits onto our phylogenetic tree in order to observe any major patterns. Before we start, there are a few things that need to be done. First, we need to be sure that we are dealing with the correct version of the tree, so we will recreate the tree. Second, we need to make sure that this tree is rooted. Last, we need to remove the root, because this is not part of our trait analysis.

```
# Generate Neighbor Joining Tree Using F84 DNA Substitution Model {ape}
nj.tree <- bionj(seq.dist.F84)

# Define the Outgroup
outgroup <- match("Methanosarcina", nj.tree$tip.label)

# Create a Rooted Tree {ape}
nj.rooted <- root(nj.tree, outgroup, resolve.root = TRUE)

# Keep Rooted but Drop Outgroup Branch
nj.rooted <- drop.tip(nj.rooted, "Methanosarcina")

# Plot to look at our tree
# plot(nj.rooted)
```

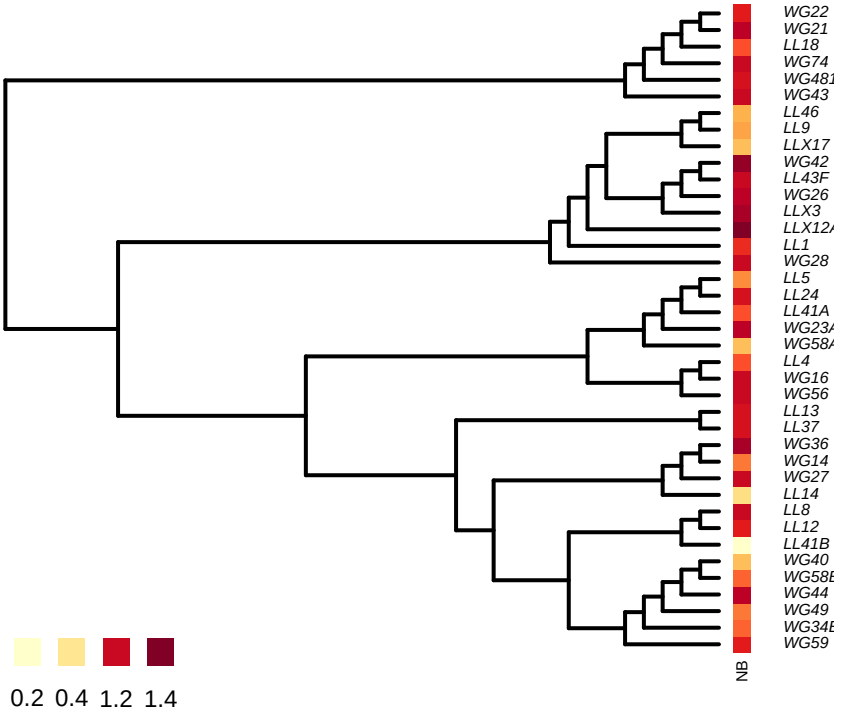
Now we can plot our traits onto the tree. This can be done with either a group of traits, such as the growth rates on different phosphorus sources, or it can be done on a single trait, such as niche breadth. Here, we will do both.

```
# Define Color Palette
mypalette <- colorRampPalette(brewer.pal(9, "YlOrRd"))

# First, Correct for Zero Branch-Lengths on Our Tree
```

```
nj.plot <- nj.rooted
nj.plot$edge.length <- nj.plot$edge.length + 10^-1

# Map Phosphorus Traits {adephylo}
par(mar = c(1, 1, 1, 1) + 0.1)
x <- phylo4d(nj.plot, p.growth.std)
table.phylo4d(x, treetype = "phylo", symbol = "colors", show.node = TRUE,
              cex.label = 0.5, scale = FALSE, use.edge.length = FALSE,
              edge.color = "black", edge.width = 2, box = FALSE,
              col=mypalette(25), pch = 15, cex.symbol = 1.25,
              ratio.tree = 0.5, cex.legend = 1.5, center = FALSE)
```




```
# Niche Breadth
par(mar = c(1, 5, 1, 5) + 0.1)
x.nb <- phylo4d(nj.plot, nb)
table.phylo4d(x.nb, treetype = "phylo", symbol = "colors", show.node = TRUE,
      cex.label = 0.5, scale = FALSE, use.edge.length = FALSE,
      edge.color = "black", edge.width = 2, box = FALSE, col = mypalette(25),
      pch = 15, cex.symbol = 1.25, var.label = ("NB"),ratio.tree = 0.90,
      cex.legend = 1.5, center = FALSE)
```



6) HYPOTHESIS TESTING

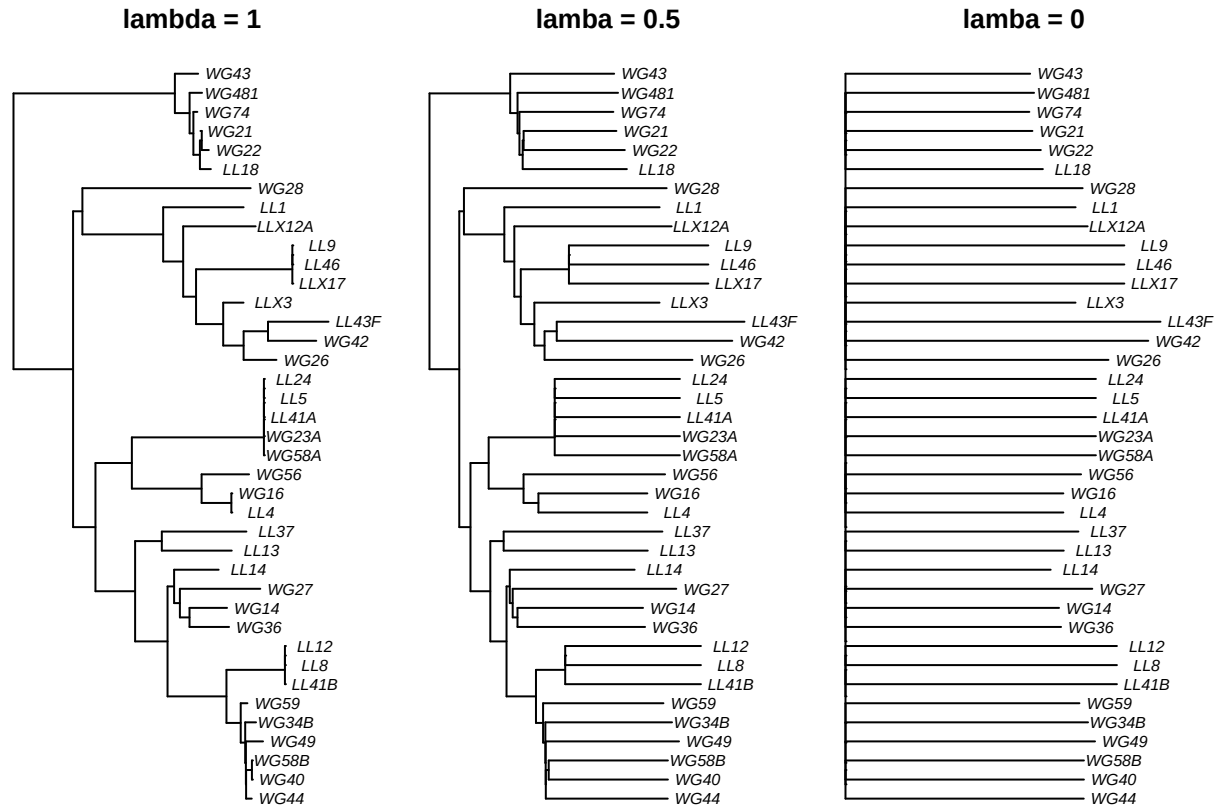
An emerging goal of biodiversity research is to understand the effect traits have on the performance of species under different environmental conditions, but also how traits can affect ecosystem functioning. Before attempting to establish relationships among traits, however, it is important to understand the role evolution plays in the distribution of traits. Owing to shared ancestry, we expect species traits to be non-independent with respect to evolutionary history. Therefore, it is important that we test for the presence of this **phylogenetic signal** before drawing conclusions about patterns among traits over a phylogeny. In addition, these patterns of phylogenetic signal provide insight into ecological and evolutionary processes giving rise to the **clustering** or **overdispersion** of traits, which may be associated with environmental filtering and competitive repulsion, respectively. In the following sections, we introduce some methods for testing hypotheses about the distribution of traits with respect to phylogeny.

A) Phylogenetic Signal: Pagel's Lambda

Mark Pagel developed a fairly simple technique to quantify phylogenetic signal, which involves a tree transformation (http://goo.gl/zSSxoB). The transformation is accomplished by scaling the off-diagonal elements of the variance-covariance matrix, which describe the tree topology and branch lengths, by the value "lambda". When lambda equals 1, you have your original, non-transformed branch lengths. When lambda equals 0, you have removed all phylogenetic signal from your tree. Let us look at what happens when we transform our tree with lambda values of 1, 0.5, and 0.

```
# Visualize Trees With Different Levels of  Phylogenetic Signal {geiger}
nj.lambda.5 <- geiger::rescale(nj.rooted, "lambda", 0.5)
```

```
nj.lambda.0 <- geiger::rescale(nj.rooted, "lambda", 0)
layout(matrix(c(1, 2, 3), 1, 3), width = c(1, 1, 1))
par(mar=c(1, 0.5, 2, 0.5) + 0.1)
plot(nj.rooted, main = "lambda = 1", cex = 0.7, adj = 0.5)
plot(nj.lambda.5, main = "lamba = 0.5", cex = 0.7, adj = 0.5)
plot(nj.lambda.0, main = "lamba = 0",  cex = 0.7, adj = 0.5)
```



Now, let us generate quantitative output to assess how much phylogenetic signal is in our data set. For this test, we will compare the phylogenetic signal of niche breadth using trees with lambda values of 1 (untransformed) and 0 (transformed).

```
# Generate Test Statistics for Comparing Phylogenetic Signal {geiger}
fitContinuous(nj.rooted, nb, model = "lambda")
fitContinuous(nj.lambda.0, nb, model = "lambda")
```

From here we are able to get AIC scores. Differences in AIC scores provide maximum likelihood inference regarding the quality of model fit to a given data set. Models with lower AIC values are better. However, if the difference in AIC values between two models (Δ AIC) is not greater than at least 2, then the models are considered equivalent.

Next, we can compare the AIC results between the two model fits to a likelihood ratio test measuring whether lambda is significantly different from 0.

```
# Compare Pagel's lambda score with likelihood ratio test
# Lambda = 0, no phylogenetic signal
phylosig(nj.rooted, nb, method = "lambda", test = TRUE)
```

```
##
## Phylogenetic signal lambda : 0.00698413
## logL(lambda) : 21.5034
## LR(lambda=0) : 0.00181764
## P-value (based on LR test) : 0.965993
```

B) Phylogenetic Signal: Blomberg's K

Blomberg et al. (2003) derived a statistic that quantifies phylogenetic signal by comparing observed trait distributions on a tree to what would evolve under Brownian (i.e., random) motion (http://goo.gl/0xU9Et). Blomberg's K is calculated as the mean squared error of the tip data (i.e., trait) measured from the phylogenetic-corrected mean and the mean squared error based on the variance–covariance matrix derived from the given phylogeny under the assumption of Brownian motion (Münkemüller et al 2012, http://goo.gl/c08XrA). A K-value of 1 means that a trait is distributed on the tree according to null expectation of Brownian motion. A K-value > 1 means that traits are **clustered** with closely related species being *more similar* than expected by chance. A K-value of < 1 means that traits are **overdispersed** with closely related species *less similar* than expected by chance.

Let us test for phylogenetic signal using Blomberg's K for the standardized growth of our bacterial isolates on *each phosphorus source* based on our neighbor joining tree.

Note: In this particular analysis, we are generating 18 test statistics with associated *P*-values (one for each phosphorus resource). This means that we are increasing our probability of rejecting the null hypothesis (i.e., "no phylogenetic signal") when it is in fact true. This is called **false discovery rate** (FDR). We are going to correct for FDR using the Benjamini-Hochberg method ("PIC.P.BH").

```
# First, Correct for Zero Branch-Lengths on Our Tree
nj.rooted$edge.length <- nj.rooted$edge.length + 10^-7

# Calculate Phylogenetic Signal for Growth on All Phosphorus Resources
# First, Create a Blank Output Matrix
p.phylosignal <- matrix(NA, 6, 18)
colnames(p.phylosignal) <- colnames(p.growth.std)
rownames(p.phylosignal) <- c("K", "PIC.var.obs", "PIC.var.mean",
                             "PIC.var.P", "PIC.var.z", "PIC.P.BH")

# Use a For Loop to Calculate Blomberg's K for Each Resource
for (i in 1:18){
  x <- setNames(as.vector(p.growth.std[,i]),row.names(p.growth))
  out <- phylosignal(x, nj.rooted)
  p.phylosignal[1:5, i] <- round(t(out), 3)
}

# Use the BH Correction on P-values:
p.phylosignal[6, ] <- round(p.adjust(p.phylosignal[4, ], method = "BH"), 3)

# Check out the results
print(p.phylosignal)
```

```
##                   AEP      PEP      G1P      G6P  MethCP      BGP      DNA
## K               0.000    0.000    0.000    0.000   0.000    0.000    0.000
## PIC.var.obs  4050.685  659.175  926.262 5887.270 350.859  510.561  237.150
## PIC.var.mean 7208.477 1345.316 1666.065 3220.579 446.829 1542.519 4509.431
## PIC.var.P       0.302    0.121    0.161    0.814   0.403    0.058    0.001
```

```
## PIC.var.z      -0.722   -1.077   -0.996    1.104  -0.296   -1.511   -1.176
## PIC.P.BH        0.604    0.363    0.414    0.862   0.659    0.209    0.018
##                  Peth    Pchol       B1     Phyt      SRP     cAMP      ATP
## K               0.000    0.000    0.000    0.000    0.000    0.000    0.000
## PIC.var.obs   192.520  397.370 3357.131 9230.269 1166.316  678.817 3942.591
## PIC.var.mean 1675.942 2931.278 4730.035 8135.555 1383.710 2736.402 2670.913
## PIC.var.P       0.003    0.010    0.282    0.616    0.381    0.009    0.670
## PIC.var.z      -1.878   -1.475   -0.632    0.143   -0.395   -2.353    0.576
## PIC.P.BH        0.027    0.045    0.604    0.792    0.659    0.045    0.804
##               PhenylCP    PolyP      GDP      GTP
## K               0.000    0.000    0.000    0.000
## PIC.var.obs  1224.017 1081.902 4469.581 2714.560
## PIC.var.mean  690.412 1100.643 3136.675 2618.172
## PIC.var.P       0.862    0.555    0.724    0.576
## PIC.var.z       1.201   -0.033    0.646    0.074
## PIC.P.BH        0.862    0.792    0.815    0.792
```

And from the output of `p.phylosignal` we can see what traits have statistically significant phylogenetic signal with a q-value threshold < 0.05 for the BH adjusted P values.

In addition, let us test for phylogenetic signal using Blomberg's K on *niche breadth*:

```
# Calcualate Phylogenetic Signal for Niche Breadth
signal.nb <- phylosignal(nb, nj.rooted)
signal.nb
```

```
##              K PIC.variance.obs PIC.variance.rnd.mean PIC.variance.P
## 1 3.608803e-06         48546.48              44520.38          0.603
##   PIC.variance.Z
## 1      0.2107415
```

And we can see here that phylogenetic signal is particularly low ($\sim 10^{-6}$) and that niche breadth does not have a statistically significant phylogenetic signal under Blomberg's K ($P \approx 0.5$).

C. Calculate Dispersion of a Trait

Another way to calculate the phylogenetic signal of a trait is to use a formal test of dispersion. In ecology, dispersion is commonly used to measure how things (e.g., traits or species) are distributed in space. However, we can use similar approaches to determine how well traits are distributed across a phylogenetic tree. Fritz and Purvis (2010) derived a measure of dispersion (D) for categorical traits (http://goo.gl/SUrm5j). In our case study, we can use D to determine if the ability of a bacterial isolate to grow on a specific phosphorus resource is overdispersed or clustered. Here, we will calculate D using the `phylo.d()` function in the `caper` package.

There are a few things we need to do: 1) turn our continuous growth data into categorical data (i.e., growth vs. no growth) 2) add a column to our data set for our isolate names, and 3) combine our tree and trait data using the `comparative.data()` command in the `caper` package.

The output of `phylo.d()` will return the estimated value of D. Estimates of D can be negative (clustered) or positive (overdispersed). When D is close 0, traits are randomly clumped. When D is close to 1, traits are dispersed in a way that is consistent with Brownian motion.

The `phylo.d()` function uses a permutation test to calculate the probability of D being different from 1 (no phylogenetic structure; random) or 0 (Brownian phylogenetic structure). Let us use this function to

calculate dispersion (D) on a few of our phosphorus traits. If we look at the statistical differences from 0 and 1, make sure to adjust the P value threshold to correct for multiple tests. In this case, we can use Bonferroni correction ($0.05/2 = 0.025$).

```
# Turn Continuous Data into Categorical Data
p.growth.pa <- as.data.frame((p.growth > 0.01) * 1)

# Look at Phosphorus Use for Each Resource
apply(p.growth.pa, 2, sum)

# Add Names Column to Data
p.growth.pa$name <- rownames(p.growth.pa)

# Merge Trait and Phylogenetic Data; Run `phylo.d`
p.traits <- comparative.data(nj.rooted, p.growth.pa, "name")
phylo.d(p.traits, binvar = AEP, permut = 10000)
phylo.d(p.traits, binvar = PhenylCP, permut = 10000)
phylo.d(p.traits, binvar = DNA, permut = 10000)
phylo.d(p.traits, binvar = cAMP, permut = 10000)
```

D. Correspondence between evolutionary history and ecology

So far we have quantified the distribution of traits along the phylogeny. We can also draw a dendrogram based on our traits to visualize similarities in traits among species independent of phylogeny. This will allow us to visualize the degree that, for example, phylogeny, resource use, and environment influence the species we are studying. For the case of our phosphorus use data, we can see the degree that phylogenetic relationships and lake origin have on our resource use data.

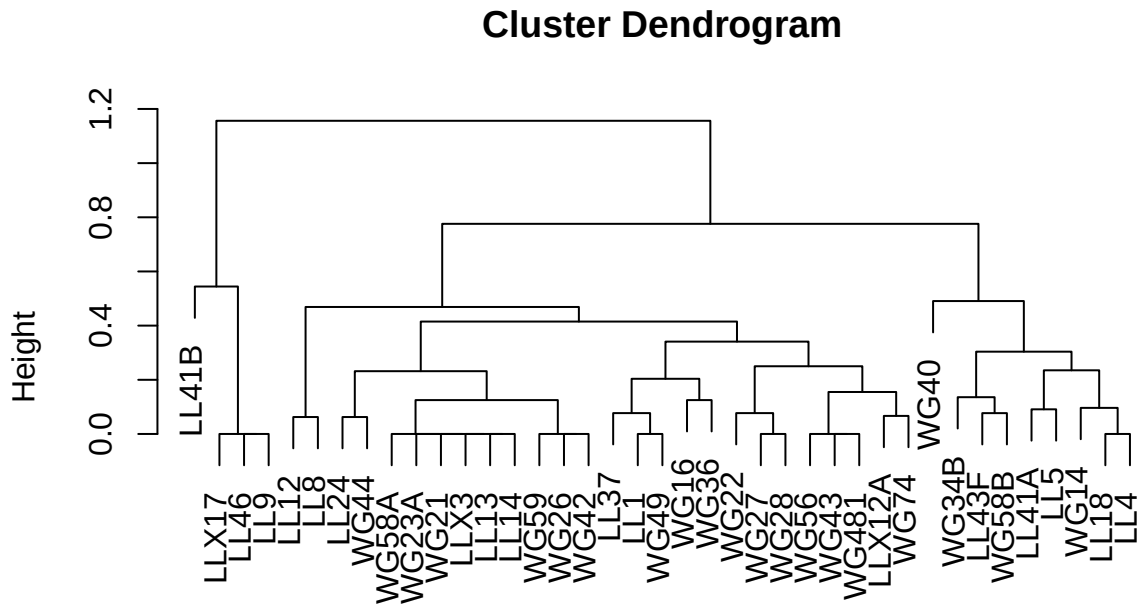
To create a dendrogram of resource use, we will make a hierarchical agglomerative cluster of the binary trait data (i.e., growth vs. no growth). A hierarchical cluster allows us to visualize the pairwise similarity among bacterial taxa for resource use, which we can use to infer **niche overlap**. Niche overlap indicates the potential for exploitative competition among taxa at each site because two taxa may compete for the same resource. Here we will map our trait data onto the phylogeny to compare the hierarchical structures in each dataset. We can compare our two dendrograms using a **tanglegram** (a plot that shows the alignment between two dendrograms), the Robinson-Fould metric, and a Mantel correlation test.

```
# Generate Jaccard's index (dissimilarity)
no = vegdist(p.growth.pa[,1:18], method = 'jaccard', binary = TRUE)

# Test the clustering method that best fits the data
# Define linkage methods
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# Function to compute agglomerative coefficient
ac <- function(x) {
  agnes(no, method = x)$ac
}

# Calculate agglomerative coefficient for each clustering linkage method
# Use the method with the highest coefficient, indicates better fit
sapply(m, ac)
```

```
##   average    single  complete      ward
```

```
## 0.9064731 0.8881997 0.9207206 0.9470011
```

```
# Generate hierarchical cluster
no.tree <- hclust(no,method = "ward.D2")
plot(no.tree)
```

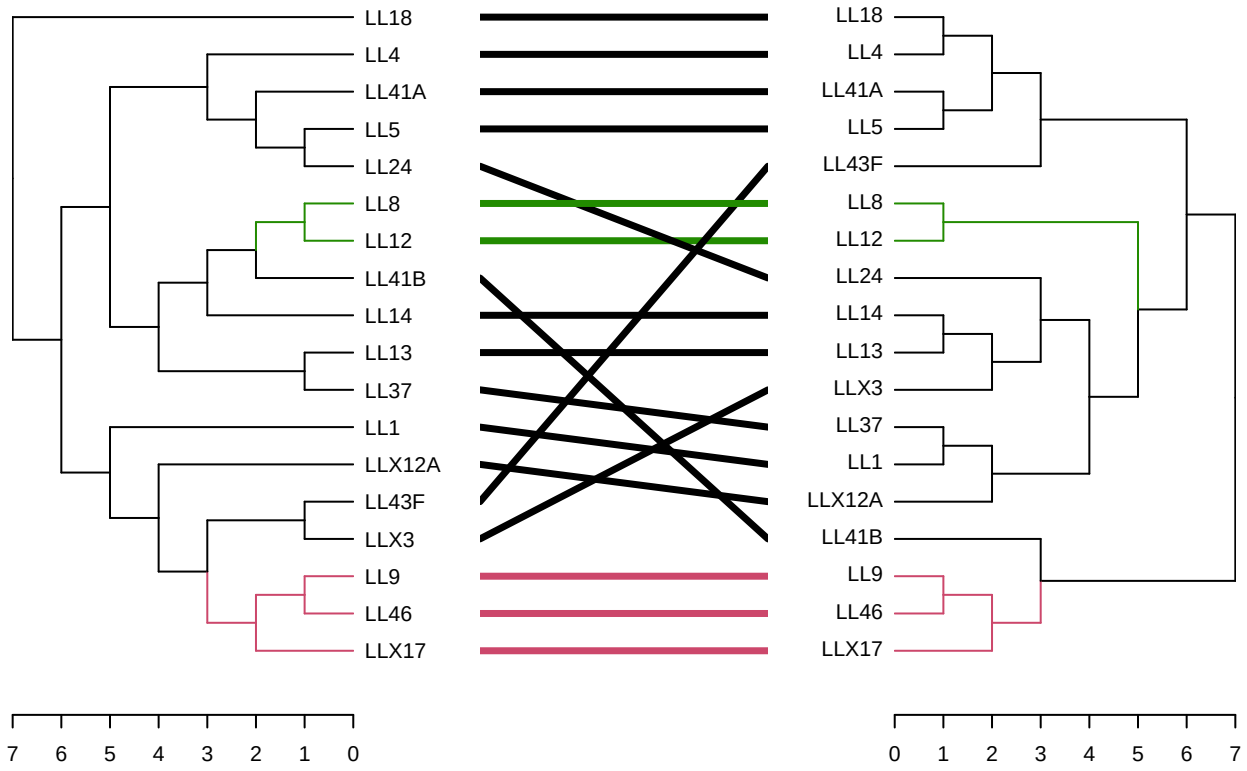



```
# Compare topology between phylogeny and niche overlap with a tanglegram
# Requires ultrametric tree (or a dendrogram)
# Branch lengths are equidistant from the root node
is.ultrametric(nj.rooted) #false
```

```
## [1] FALSE
```

```
#is.ultrametric(no.tree) #false

# Visualize differences between each lake
# Drop tips from each dendrogram to plot tips that come from a single lake
LL.tree <- drop.tip(nj.rooted,c(nj.rooted$tip.label[grepl("WG",
           nj.rooted$tip.label)]))
LL.function <- drop.tip(as.phylo(no.tree),
          c(no.tree$labels[grepl("WG", no.tree$labels)]))

WG.tree <- drop.tip(nj.rooted, c(nj.rooted$tip.label[grepl("LL",
           nj.rooted$tip.label)]))
```

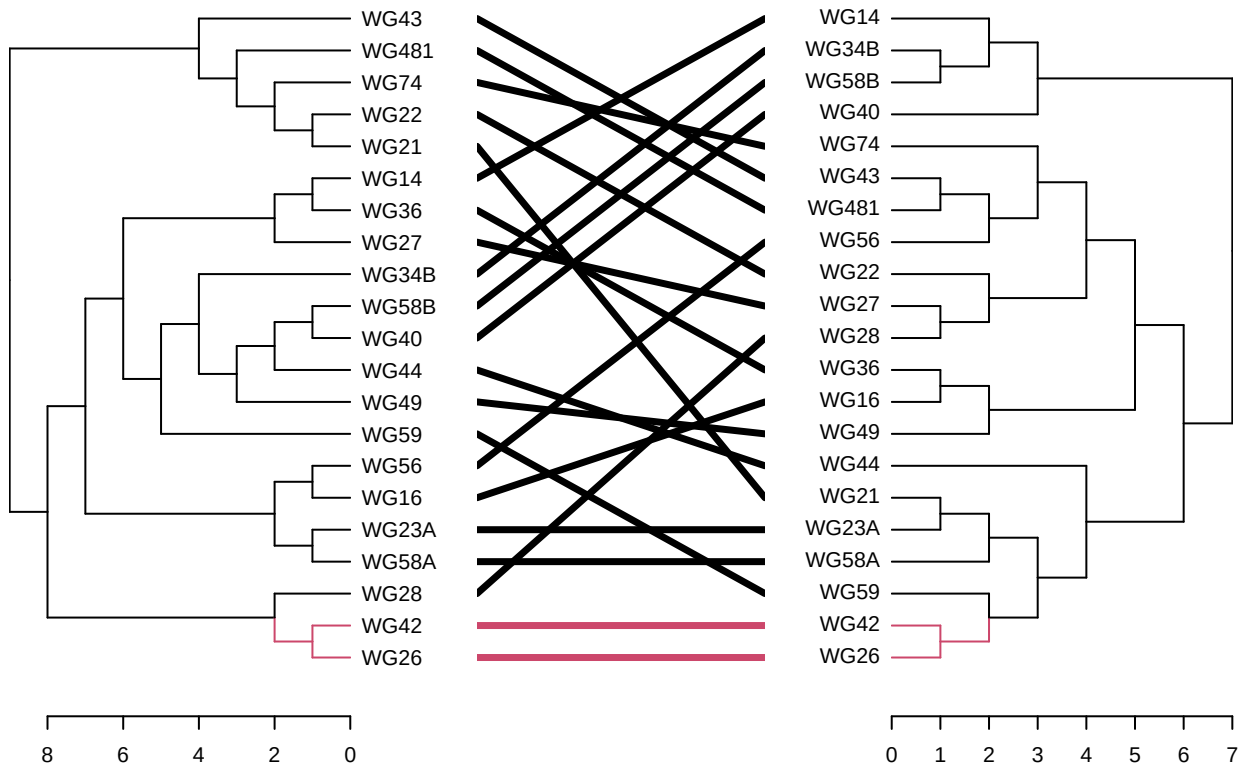
```
WG.function <- drop.tip(as.phylo(no.tree),
           c(no.tree$labels[grepl("LL",no.tree$labels)]))

# Plot each dendrogram and link their tips.
# An untangling algorithm is used to maximize alignment.
# For tanglegram visualization, highlight matches between the two dendrograms.
# Resource use similarity on right; phylogenetic relatedness on the left

par(mar = c(1, 5, 1, 5) + 0.1)
dendlist(as.cladogram(as.dendrogram.phylo(LL.tree)),
  as.cladogram(as.dendrogram(LL.function))) %>%
  untangle(method = "step2side") %>% # Find the best alignment layout
  tanglegram(common_subtrees_color_branches = TRUE,
  highlight_distinct_edges = FALSE, highlight_branches_lwd = FALSE,
  margin_inner = 5) %>%# Draw the two dendrograms
  entanglement() #score between 0 and 1, closer to 0 is a better alignment
```




```
## [1] 0.1124409
```

```
par(mar = c(1, 5, 1, 5) + 0.1)
dendlist(as.cladogram(as.dendrogram.phylo(WG.tree)),
  as.cladogram(as.dendrogram(WG.function))) %>%
  untangle(method = "step2side") %>% # Find the best alignment layout
  tanglegram(common_subtrees_color_branches = TRUE,
  highlight_distinct_edges = FALSE, highlight_branches_lwd = FALSE,
```

```
  margin_inner = 5) %>%# Draw the two dendrograms
  entanglement() #score between 0 and 1, closer to 0 is a better alignment
```




```
## [1] 0.2692713
```

```
# Measure the degree of correspondence between each dendrogram
# 0 = complete congruence, 1 = complete incongruence
RF.dist(LL.tree,as.phylo(as.dendrogram(LL.function)), normalize = TRUE,
  check.labels = TRUE, rooted = FALSE)
```

```
## [1] 0.8
```

```
RF.dist(WG.tree, as.phylo(as.dendrogram(WG.function)), normalize = TRUE,
  check.labels = TRUE, rooted = FALSE)
```

```
## [1] 0.9444444
```

```
# Mantel test to correlate patristic distance (pairwise sum of branch lengths)
# in phylogeny and Jaccard index in the resource use hierarchical cluster

mantel(cophenetic.phylo(LL.tree), cophenetic.phylo(LL.function),
  method = "spearman", permutations = 999)
```

```
##
## Mantel statistic based on Spearman's rank correlation rho
##
## Call:
## mantel(xdis = cophenetic.phylo(LL.tree), ydis = cophenetic.phylo(LL.function),      method = "spearma
##
## Mantel statistic r: 0.0784
##       Significance: 0.209
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.129 0.171 0.204 0.266
## Permutation: free
## Number of permutations: 999
```

```
mantel(cophenetic.phylo(WG.tree), cophenetic.phylo(WG.function),
  method = "spearman", permutations = 999)
```

```
##
## Mantel statistic based on Spearman's rank correlation rho
##
## Call:
## mantel(xdis = cophenetic.phylo(WG.tree), ydis = cophenetic.phylo(WG.function),      method = "spearma
##
## Mantel statistic r: -0.08174
##       Significance: 0.764
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.134 0.196 0.229 0.292
## Permutation: free
## Number of permutations: 999
```

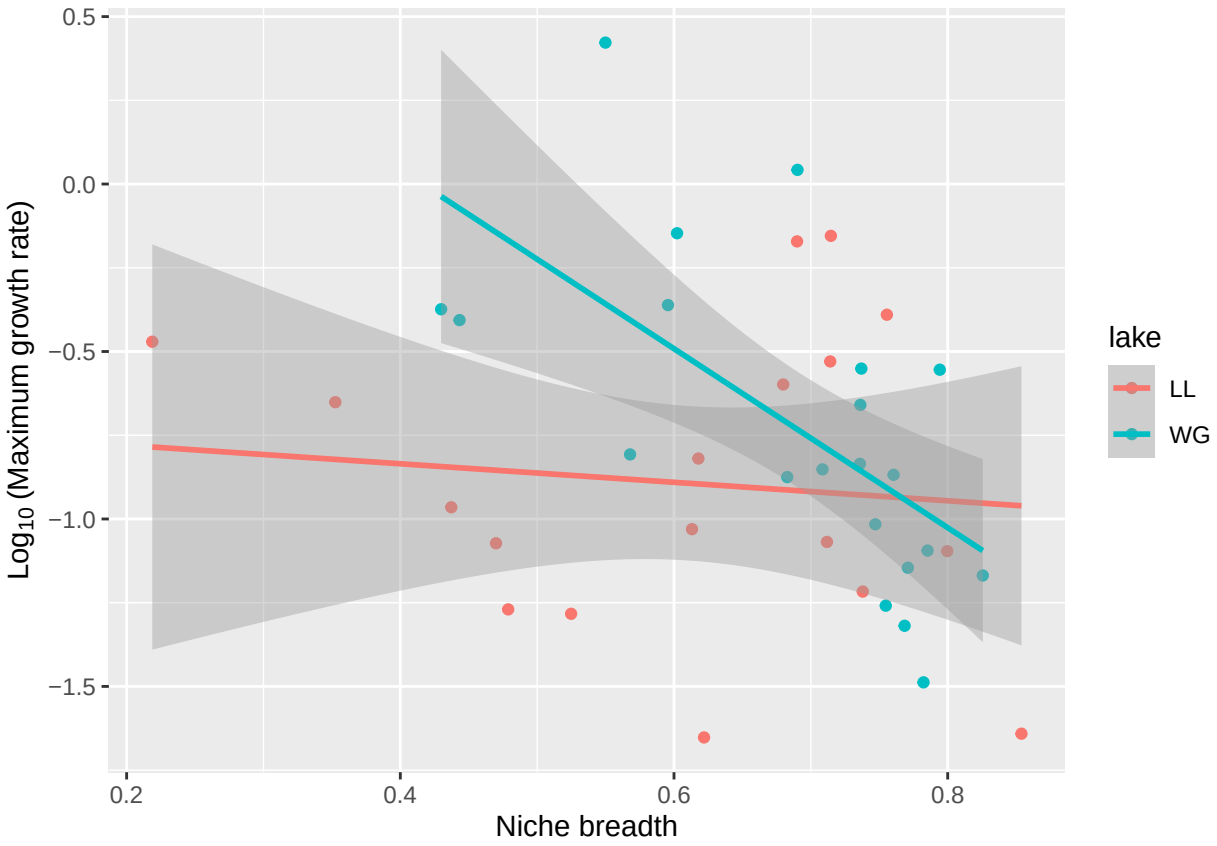
E. PHYLOGENETIC REGRESSION ANALYSIS

You will often want to examine the relationship between two variables. The simplest way to do this is by performing a simple linear regression, a method that you learned in Week 1. However, as outlined above, your samples violate the assumption of independence for regression analysis due to their shared evolutionary history. The way around this issue is to perform a phylogenetic regression. The main difference between the two is that in a simple linear regression the residual errors ε, given an independent variable (X), are assumed to be independent and identically distributed random variables that follow a normal distribution with a mean of 0 and a variance of σ^2 : $\varepsilon \mid X \sim N(0, \sigma^2 I_n)$. However, in a phylogenetic regression the variance of the residual errors are described by a covariance matrix (V) that takes into account the branch lengths of the underlying phylogeny: $\varepsilon \mid X \sim N(0, V)$. It is also important to take into account the phylogenetic signal in the residual error when V is estimated, not just the evolutionary history described by the phylogeny. If phylogenetic signal is not taken into account your phylogenetic regression can have poor statistical performance (Revell, 2010).

For the phylogenetic regression, we will associate the maximum growth rate across all resources with niche breadth and compare whether these trends differ between lakes. This will give use insight into specialist-generalist trade-off between each lake. From our analysis we will see how correcting for phylogeny influences our ability to test this relationship.

```
# Using the niche breadth data, create a column that indicates the lake origin of each strain
nb.lake = as.data.frame(as.matrix(nb))
nb.lake$lake = rep('A')

for(i in 1:nrow(nb.lake)) {
    ifelse(grepl("WG",row.names(nb.lake)[i]), nb.lake[i,2] <- "WG",
        nb.lake[i,2] <- "LL")
}

#Add a meaningful column name to the niche breadth values
colnames(nb.lake)[1] <- "NB"

#Calculate the max growth rate
umax <- as.matrix((apply(p.growth, 1, max)))
nb.lake = cbind(nb.lake,umax)

# Plot maximum growth rate by niche breadth
ggplot(data = nb.lake,aes(x = NB, y = log10(umax), color = lake)) +
  geom_point() +
  geom_smooth(method = "lm") +
  xlab("Niche breadth") +
  ylab(expression(Log[10]~"(Maximum growth rate)"))
```

```
## `geom_smooth()` using formula 'y ~ x'
```



First let us run a simple linear regression.

```
# Simple linear regression
fit.lm = lm(log10(umax) ~ NB*lake,data = nb.lake)
summary(fit.lm)
```

```
##
## Call:
## lm(formula = log10(umax) ~ NB * lake, data = nb.lake)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7557 -0.3108 -0.1077  0.3102  0.7800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.7247     0.3852  -1.882   0.0682 .
## NB           -0.2763     0.6097  -0.453   0.6533
## lakeWG        1.8364     0.6909   2.658   0.0118 *
## NB:lakeWG    -2.3958     1.0234  -2.341   0.0251 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.418 on 35 degrees of freedom
## Multiple R-squared:  0.2595, Adjusted R-squared:  0.196
## F-statistic: 4.089 on 3 and 35 DF,  p-value: 0.01371
```

```
AIC(fit.lm)
```

```
## [1] 48.413
```

Now let us correct for phylogeny using the `phylolm` function from the package `phylolm`.

```
# Run a phylogeny-corrected regression with no bootstrap
# replicates
fit.plm = phylolm(log10(umax) ~ NB * lake, data = nb.lake, nj.rooted,
    model = "lambda", boot = 0)
summary(fit.plm)
```

```
##
## Call:
## phylolm(formula = log10(umax) ~ NB * lake, data = nb.lake, phy = nj.rooted,
##     model = "lambda", boot = 0)
##
##    AIC logLik
##  41.08 -14.54
##
## Raw residuals:
##      Min       1Q   Median       3Q      Max
## -0.75804 -0.18999 -0.07425  0.32496  0.95857
##
## Mean tip height: 0.1814508
## Parameter estimate(s) using ML:
## lambda : 0.4861386
```

```
## sigma2: 0.9184409
##
## Coefficients:
##               Estimate     StdErr t.value p.value
## (Intercept) -0.8912676  0.3700360 -2.4086 0.02142 *
## NB          -0.0048049  0.5213029 -0.0092 0.99270
## lakeWG       1.4389308  0.5772311  2.4928 0.01755 *
## NB:lakeWG   -1.9663889  0.8487018 -2.3169 0.02648 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared: 0.1935    Adjusted R-squared: 0.1243
##
## Note: p-values and R-squared are conditional on lambda=0.4861386.
```

```
AIC(fit.plm)
```

```
## [1] 41.07572
```

By accounting for how shared evolutionary history impacts the relationship between growth rate and niche breadth (i.e. the non-independence of our data) we actually find that phylogeny was impacting our results, and shows that we had a better fit model because we have moderate phylogenetic signal in the residuals.

REFERENCES

Boettiger, C., G. Coop, and P. Ralph. 2012. Is your phylogeny informative? Measuring the power of comparative methods. Evolution 66:2240–2251.

Bartoszek, K., J. Pienaar, P. Mostad, S. Andersson, and T.F. Hansen. 2012. A phylogenetic comparative method for studying multivariate adaptation. Journal of Theoretical Biology 314:204–215.

Butler, M., and A. A. King. 2004. Phylogenetic comparative analysis: a modeling approach for adaptive evolution. American Naturalist 164:683–695.

Capellini, I., C. Venditti, and R. Barton. 2010. Phylogeny and metabolic scaling in mammals. Ecology 91:2783–2793

Cadotte, M. W., and T.J. Davies. 2016. Phylogenies in Ecology: A Guide to Concepts and Methods. Princeton.

Felsenstein, J. 1985. Phylogenies and the comparative method. The American Naturalist 125:1-15.

Felsenstein, J. 2004. Inferring Phylogenies. Sinauer.

Fritz, S. A. and A. Purvis. 2010. Selectivity in mammalian extinction risk and threat types: a new measure of phylogenetic signal strength in binary traits. Conserv. Biol. 24:1042–1051

Goldman, N. 1993. Statistical tests of models of DNA substitution. Journal of Molecular Evolution 36:182–198.

Grafen, A. 1989. The Phylogenetic Regression. Phil. Trans. R. Soc. Lond. B. 326:119-157

Hall, P. and S. Wilson. 1991. Two guidelines for bootstrap hypothesis testing. Biometrics. 47:757–762.

Hansen, T. F., and K. Bartoszek. 2012. Interpreting the evolutionary regression: The interplay between observational and biological errors in phylogenetic comparative studies. Systematic Biology 61:413–425.

Hansen T. F., and E. P. Martins. 1996. Translating between microevolutionary process and macroevolutionary patterns: the correlation structure of interspecific data. Evolution 50:1404–1417.

Kuhner, M. K. and J. Felsenstein. 1994. Simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Molecular Biology and Evolution 11:459–468.

Ho, L. T. and C. Ané. 2014. A Linear-Time Algorithm for Gaussian and Non-Gaussian Trait Evolution Models. Syst Biol 63:397-408.

Lavorel, S., and Garnier, E. (2002) Predicting changes in community composition and ecosystem functioning from plant traits: revisiting the Holy Grail. Functional Ecolology 16:545–556

Losos, J. B. 2008. Phylogenetic niche conservatism, phylogenetic signal and the relationship between phylogenetic relatedness and ecological similarity among species. Ecology Letters 11:995–1003.

Lynch, M. 1991. Methods for the analysis of comparative data in evolutionary biology. Evolution 45: 1065-1080.

MacArthur, R. H., and J. W. MacArthur. 1961. On Bird Species Diversity. Ecology 42:594-598

Münkemüller, T., S. Lavergne, B. Bzeznik, S. Dray, T. Jombart, K. Schiffers, and W. Thuiller. 2012. How to measure and test phylogenetic signal. Methods in Ecology and Evolution 3:743–756.

Pagel, M. 1999. Inferring the historical patterns of biological evolution. Nature 401:877–884.

Paradis, E. 2012. Analysis of Phylogenetics and Evolution with R. Springer.

Revell, L., L. Harmon, and D. Collar. 2008. Phylogenetic Signal, Evolutionary Process, and Rate. Systematic Biology 57:591–601.

Revell, L.J. and D. C. Collar. 2009. Phylogenetic analysis of the evolutionary correlation using likelihood. Evolution 63:1090–1100.

Revell, L. J. 2010. Phylogenetic signal and linear regression on species data. Methods in Ecology and Evolution 1:319–329.

Robinson, D. R. and L. R. Foulds. 1981. Comparison of phylogenetic trees. Mathematical Biosciences 53: 131–147.