Phylogenetic Diversity - Traits

Z620: Quantitative Biodiversity, Indiana University

February 17, 2017

OVERVIEW

Up to this point, we have been focusing on patterns taxonomic diversity in Quantitative Biodiversity. Although taxonomic diversity is an important dimension of biodiversity, it does not consider the evolutionary history or relatedness of species. The goal of this exercise is to introduce basic concepts of phylogenetic diversity.

After completing this exercise you will know how to:

1. create phylogenetic trees
2. map traits onto phylogenetic trees
3. test for phylogenetic signal of traits on a phylogenetic tree

1) SETUP

A. Retrieve and Set Your Working Directory

```
rm(list = ls())
getwd()
setwd("~/GitHub/QB-2017/Week6-PhyloTraits")
```

B. Load Packages

There are many contributed R packages that have been developed for conducting phylogenetic analyses (http://goo.gl/DtU16j). We will rely heavily on the `ape` (http://goo.gl/nExs0), which performs a wide range of phylogenetic analyses. We will supplement this with other packages for some specialized tasks such as reading of aligned sequences (`seqinr`), and testing for phylogenetic signal (`geiger`). We will also align sequences using the program `muscle` through the terminal. You may also want to check out the `phangorn` package (http://goo.gl/iWU6Vt), which is good for conducting maximum likelihood analyses in R. It is also worth looking into the Randomized Axelerated Maximum Likelihood (RAxML) package for generating maximum likelihood values with bootstrap support (http://sco.h-its.org/exelixis/web/software/raxml/index.html). For large numbers of taxa 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. However, both RAxML and FastTree will have to be run in the terminal.

After the initial installation of these packages using the `install.packages()` function, let's load the packages and their dependencies with the `require()` function:

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

2) DESCRIPTION OF DATA

The maintenance of biodiversity is thought to be influenced by **trade-offs** among species in certain functional traits. One such trade-off involves the ability of a highly specialized species to perform exceptionally well on a particular resource. This strategy may be advantageous if the specialist lives in an environment where its preferred resource is found in a relatively high and constant supply. However, the specialist strategy may not be optimal if the species finds itself in an environment where resource conditions are more variable. Under these conditions, it may be beneficial if a species is capable of using a wider range of resources, even though there may be a fitness cost associated with being a generalist. We set out to test this hypothesized trade-off in aquatic bacteria by assessing generalist and specialist strategies on different forms of phosphorus, a resource which commonly limits growth and reproduction of organisms inhabiting freshwater ecosystems. We isolated 39 strains of bacteria from two lakes. After identifying these bacteria based on their 16S rRNA gene sequences, we measured their growth on 18 different forms of phosphorus. In this exercise, we will take a phylogenetic approach to mapping phosphorus resource use onto a phylogenetic tree while testing for specialist-generalist trade-offs.

3) SEQUENCE ALIGNMENT

After generating or retrieving sequence data, the first step of a phylogenetic analysis is to perform an alignment. This can be done with nucleotides (DNA or RNA sequences) or proteins (amino acid sequences). There are various methods for aligning sequences, but the basic premise is to use an algorithm to arrange sequences relative to one another based on conserved regions so that downstream phylogenetic analyses can be performed. In R, there are at least two packages that can be used for aligning sequences: `ape` and `muscle`. In `ape`, alignments can be done with the functions `clustal`, `muscle`, and `tcoffee`, all which have a range of options. However, these functions require that other programs be installed on your computer. Therefore, for this exercise, we are going to use the `muscle` package in R because it will align sequences in a "stand alone" fashion without the need for installing additional software.

A. Examining a FASTA-File

We will read in a FASTA-formatted file into the `muscle` alignment function. This is a very common text-based format that is used across platforms and disciplines. Let's take a second to look at the *p.isolates.fasta* file. Open the terminal and change directory to the data folder. You can then type "nano p.isolates.fasta" at the command line. This will open the FASTA file in nano, which is a text editing program. You can use [Ctrl+V] and [Ctrl+Y] to scroll towards the bottom and top of the file, respectively. When you are done, you can close the file by hitting [Ctrl+X]. Do not save any inadvertent changes to *p.isolates.fasta*.

B. Performing an Alignment

Now, let's align our sequences. You'll often find some function that you want to use that you either can't find in R or code up yourself. However, you can still run all your code in R, you just have to specify what language you'll be using. Here we will be using the `muscle` program in the command language interpreter Bash. Bash is the language that you're using on the lab Macs. Using Bash, we'll call the program `muscle`, read in our FASTA file, and create a FASTA-aligned output file (with an .afa extension).

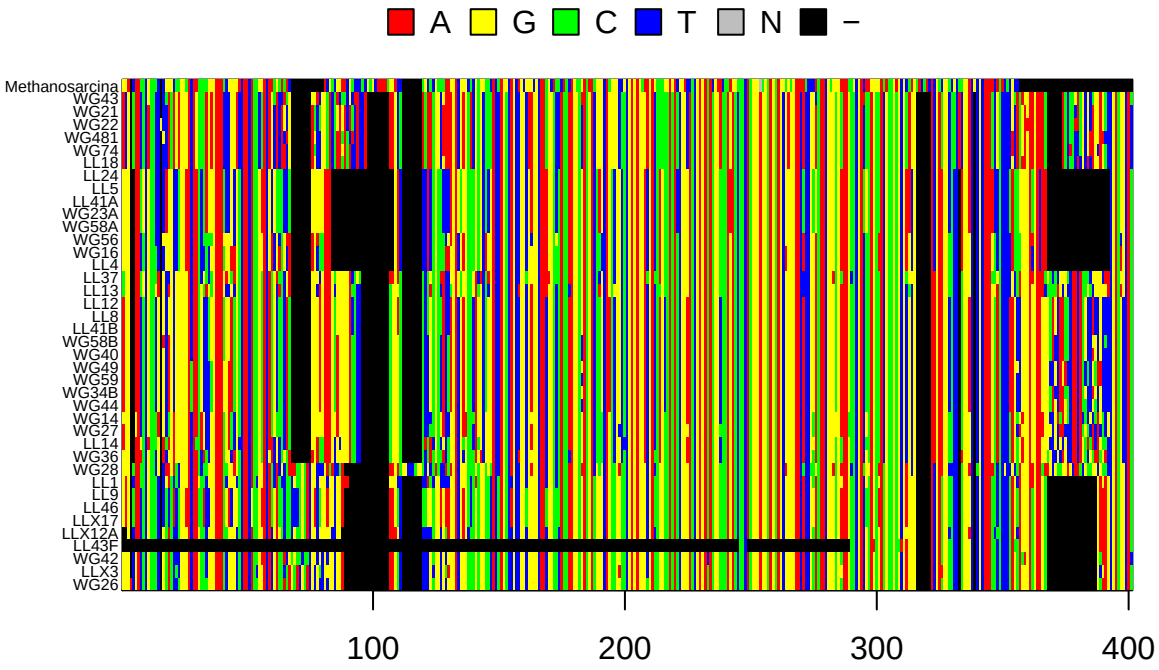
```
muscle -in ./data/p.isolates.fasta -out ./data/p.isolates.afa
```

C. Visualizing the Alignment

It is good practice to view your sequences after alignment. This may not be practical for large databases, but in our example, the alignment only contains sequences for 39 bacterial species plus a distantly related

reference sequence that we will use as an **outgroup**. To visualize our alignment, first, we will read in the FASTA-aligned file using the `read.alignment` function in the `seqinr` package. Then, using the `ape` package, we will convert the alignment file it into a DNAbin object, which is a bit-level coding scheme that allows R to store and manipulate sequence data. Last, we will visualize the alignment by color-coding nucleotides different colors, where red = adenine, yellow = guanine, green = cytosine, blue = thymidine, grey = ambiguous call, and black = alignment gaps.

```
# Read Alignment File {seqinr}
read.aln <- read.alignment(file = "./data/p.isolates.afa", format = "fasta")

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

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

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




```
# Optional Code Adds Grid to Help Visualize Rows of Sequences
#grid(ncol(window), nrow(window), col = "lightgrey")
```

4) MAKING A PHYLOGENETIC TREE

Once you have aligned your sequences, the next step is to construct a phylogenetic tree. Not only is a phylogenetic tree effective for visualizing the evolutionary relationship among taxa, but as you will see later, the information that goes into a phylogenetic tree is needed for downstream analysis.

We could spend an entire semester discussing the theory and practice behind making phylogenetic trees. The objective of this handout is to quickly get you up to speed on some of the basic principles of phylogenetic reconstruction. In addition, we want to convey some of the opportunities and limitations of conducting phylogenetic analyses in R. The following table provides an overview of the methods that are available for evolutionary reconstruction in R.

Tree Method	Properties
UPGMA	Hierarchical clustering technique. Progressively groups the most similar pairs of sequences. Sometimes used as a guide tree for more sophisticated phylogenetic analyses, but prone to distorting tree topology and assumes constant rates of evolution. UPGMA can be implemented in R with the *upgma()* function in the *panghorn* package.
Neighbor Joining	Starts with an unresolved tree in a "star network". Algorithm then identifies the two most similar taxa and creates a node. This process is iteratively applied to the remaining sequences. Can serve as a starting point for assessing evolutionary models with more sophisticated methods like maximum likelihood. Neighbor joining can be implemented in the R package *ape* using the *bionj()* function.
Maximum Parsimony	A character-based, non-parametric approach that involves identifying one of many possible trees based on the minimum number of evolutionary "steps" (or mutations). No underlying evolutionary model. Maximum parsimony can be executed using the *parsimony()* function in the package *panghorn*. The function *optim.parsimony* performs tree rearrangements using nearest-neighbor interchanges (NNI) to optimize parsimony scores.
Maximum Likelihood	Parametric approach to identifying the relative probability associated with data fitting a given model of character evolution. One of the most commonly used approaches for building trees from sequence data. Statistically sound and flexible for accommodating realistic models of sequence evolution. Maximum likelihood can be implemented in R using the *pml()* function in the *panghorn* package. Parameters can be optimized using the *optim.pml()* function and evolutionary models can be compared with the *model.tests()* function
Bayesian	Uses likelihood function and prior probabilities to create a posterior probability of a tree fitting a given model of evolution. Computationally intensive; often uses Markov chains Monte Carlo (MCMC). Unlike other methods, parameters are not constant; rather, they come from probabilistic distributions. Along with maximum likelihood, considered one of the more preferable ways of generating trees. R has some packages (e.g., *ips*) the wrap popular Bayesian software such as MrBayes

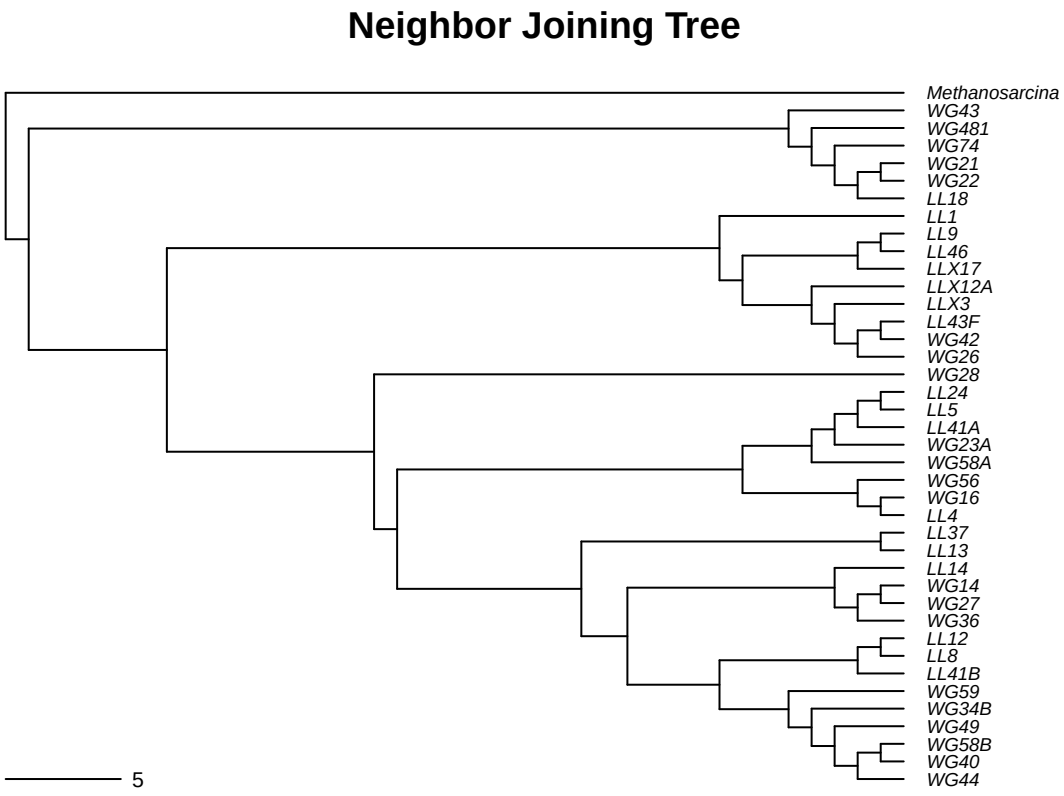
A. Neighbor Joining Trees

For the purposes of this course, we are going to focus primarily on building trees using neighbor joining methods. As you will see in the next section, neighbor joining is commonly used for making “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 identify an outgroup sequence (*Methanosarcina*) that we will use to root the tree. 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

Phylogenetic analyses require assumptions about the rates of DNA evolution. Various models have been developed to estimate the rates that nucleotides change over time. These models are important for inferring

relatedness of genes and taxa. In the following table, we describe a few of the more commonly used models.

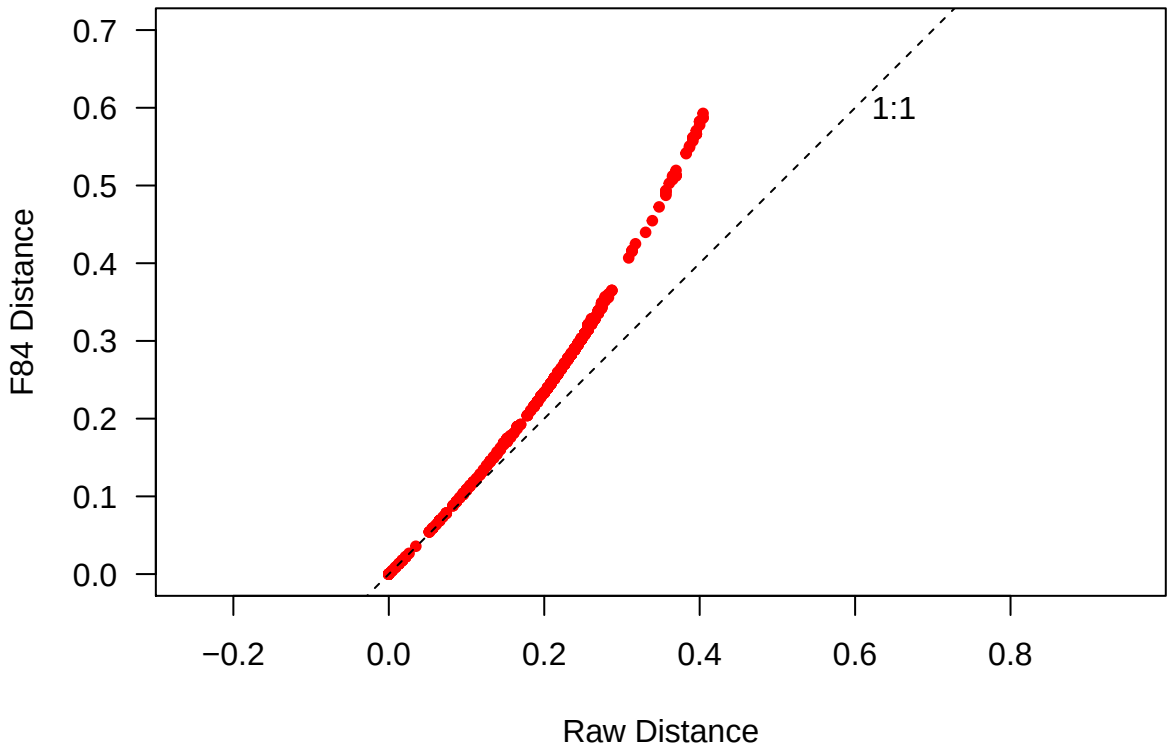
Evolutionary Model	Properties
Jukes-Cantor model (JC69)	Simplest model. Assumes all nucleotides occur at equal frequencies and that these nucleotides can mutate from one to another with equal probability.
Felsenstein model (F81)	Builds from JC69 by allowing nucleotide frequencies to vary.
Kimura model (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`.

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

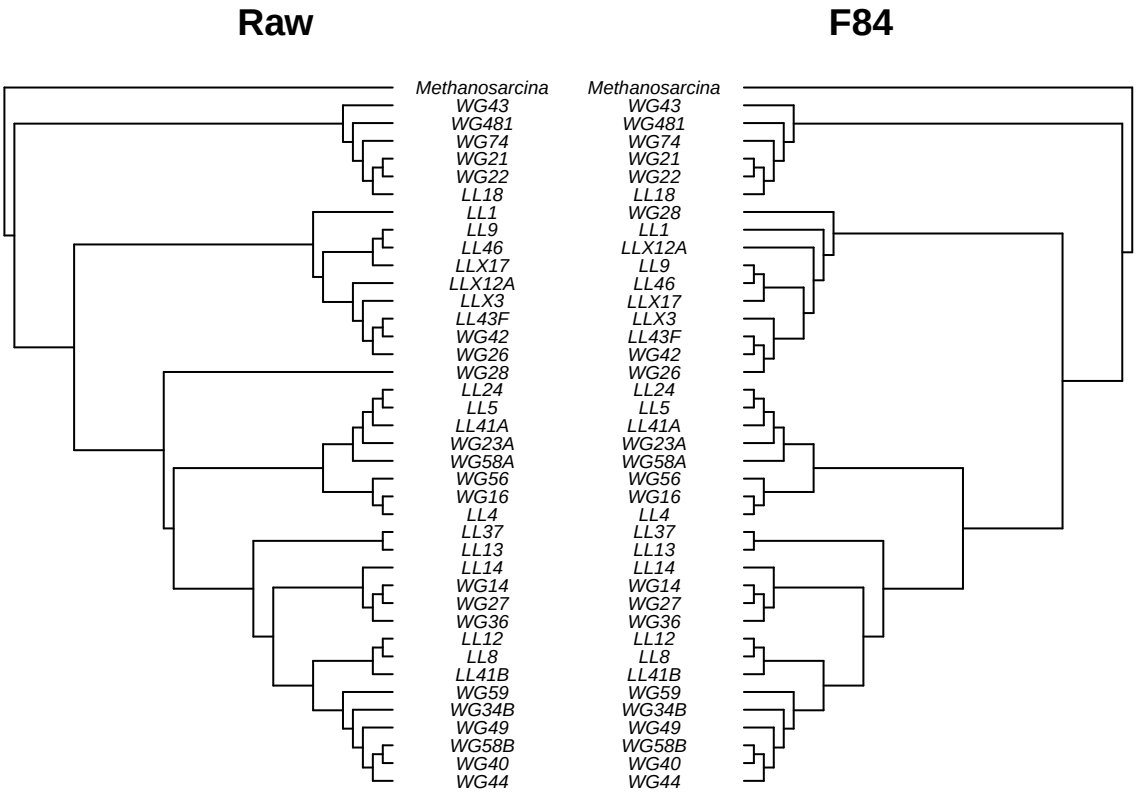
Now, let's compare the "raw" and "F84" distance matrices. First, we will make a **saturation plot**, which allow us to visualize the effect of our DNA substitution model compared to a distance matrix that does not include this feature.

```
# 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")
```



Second, we will assess how the "raw" and "F84" distance matrices affect tree topology by making 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 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 that are in one tree and not the other. This is a very simple calculation, but it is extremely sensitive to 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 isn't. This function should look familiar, since it is just the squared Euclidean distance between two trees. The history, appropriateness, and limitations of these methods are nicely discussed in chapter 30 of Felsenstein (2004).

Here we'll 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,
# so you can pass it either the rooted or unrooted tree and
# get the same answer.
dist.topo(raw.rooted, F84.rooted, method = "score")
```

```
## [1] 0.04387426
```

D) MAXIMUM LIKELIHOOD TREE

Often a nearest neighbor joined tree is not good enough. While neighbor joining if 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. 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, being fairly robust to the assumptions of a particularly model, allows you to compare different tree topologies, and uses all the information in a sequence (as opposed to just distance).

However, ML estimation is a computationally intensive process that can require a lot of time. This is because ML estimation often must consider all topologies of the tree and find the best edge lengths for each topology. Because ML estimation can take a long time, we will not ask you to generate a maximum likelihood tree for your assignment. 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
p.DNAbin.phyDat <- read.phyDat("./data/p.isolates.afa", format="fasta", type="DNA")
fit <- pml(nj.rooted, data=p.DNAbin.phyDat)
# 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(fitGTR, model="GTR", optInv=TRUE, optGamma=TRUE,
#  rearrangement = "NNI", control = pml.control(trace = 0))
# You can perform model selection with either an ANOVA test or by AIC value
anova(fitJC, fitGTR)
AIC(fitJC)
AIC(fitGTR)

# And you can perform a bootstrap test to see how well-supported the edges are
bs = bootstrap.pml(fitJC, bs=100, optNni=TRUE,
  control = pml.control(trace = 0))
```

We have also included a short script in the folder **bash** under Week6 that will allow you to generate a ML tree on the IU computer cluster Mason using the program RAxML. Generating a ML tree is computationally intensive and not all tree-generating software have R packages, so if you need a ML tree it's often easier to run it on a desktop or a computer cluster using a shell script rather than on your laptop. If you are an IU student, postdoc, or faculty member you can register for an account on Mason. See the IU Mason webpage for more information (https://kb.iu.edu/d/bbhh).

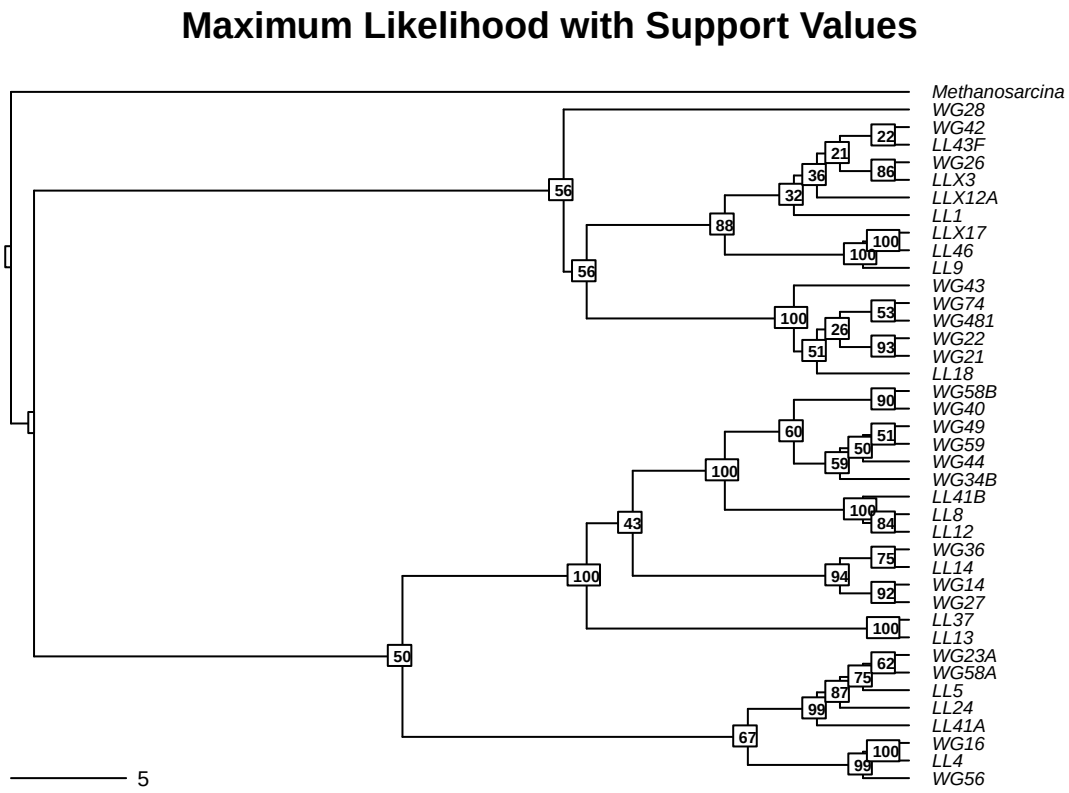
E) BOOTSTRAP SUPPORT

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 don't know the real tree we'll use the next best thing. We'll re-sample our own data to determine how reliable the tree is (i.e. bootstrapping). To do this we'll 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.

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)
```



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 forms of phosphorus (e.g., phosphate, DNA, ATP, etc.). With this information we will visualize the functional traits (i.e., growth on different phosphorus) in a phylogenetic context by mapping them onto the neighbor joining tree (with 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)

# Standadize 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 testing for a **generalist-specialist trade-off**. To do this, we need to create two new variables. First, we need 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 forms 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's 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 & Column Names to Niche Breadth Matrix
rownames(nb) <- row.names(p.growth)
colnames(nb) <- c("NB")
```

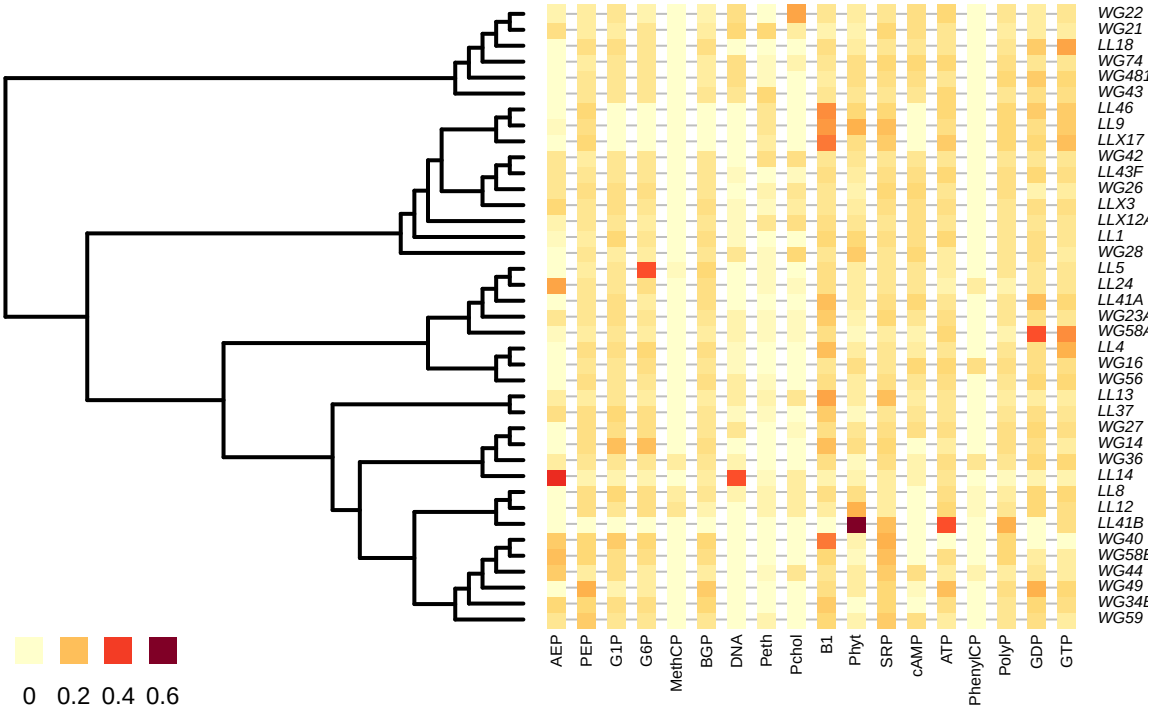
C. Visualizing Traits on Trees

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. This is similar to the hypothesis-generating visualizations we did in other lessons this semester. 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 isn't part of our traits analysis.

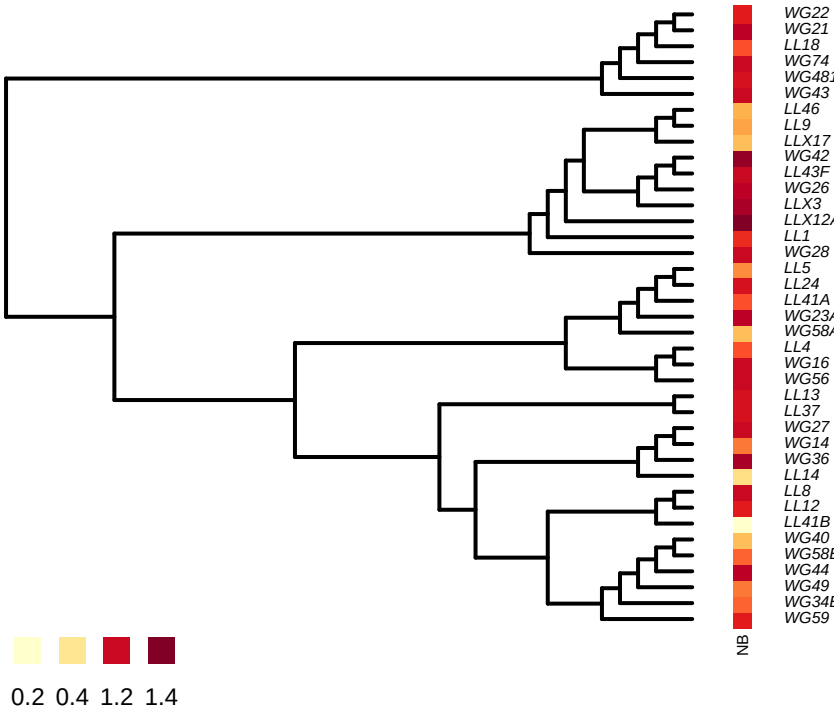
```
# Generate Neighbor Joining Tree Using F84 DNA Substitution Model {ape}
nj.tree <- bionj(seq.dist.F84)

# 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")
```

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"))

# Map Phosphorus Traits {adephylo}
par(mar=c(1,1,1,1) + 0.1)
x <- phylo4d(nj.rooted, 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.rooted, 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

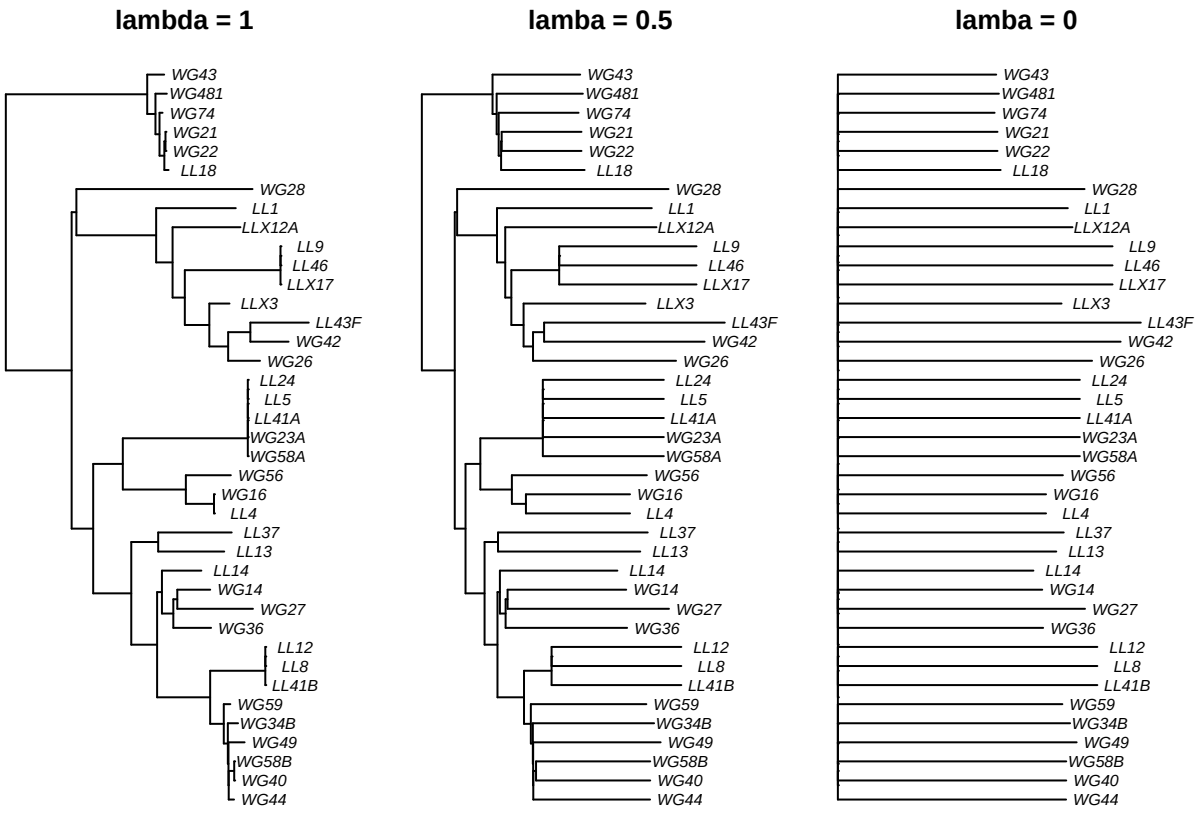
An emerging goal of biodiversity research is to understand the effect traits have on the performance of species under different environmental conditions, but also how traits can affect ecosystem functioning. Before attempting to establish relationships among traits, however, it is important to understand 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. In addition, these patterns of phylogenetic signal provide insight into ecological and evolutionary processes giving rise to the **clustering** or **overdispersion** of traits, which are sometimes 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's 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 <- rescale(nj.rooted, "lambda", 0.5)
nj.lambda.0 <- 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's generate quantitative output that will allow us 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")
```

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 Browning 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's 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 (ie.

"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 <- as.matrix(p.growth.std[ ,i, drop = FALSE])
  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)
```

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

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

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 of 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 & 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.

First, there are a few things we need to do: 1) turn our continuous growth data into categorical data, 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 `caper`.

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 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's use this function to calculate dispersion (D) on a few of our phosphorus traits.

```
# 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)
```

```r
# 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)
phylo.d(p.traits, binvar = PhenylCP)
phylo.d(p.traits, binvar = DNA)
phylo.d(p.traits, binvar = cAMP)
```

D. Compare evolutionary models

Often you will want to determine what stochastic model of evolution best describes your data. To do this you will need to compare the fit of each model to your data and determine which provides the best fit. An appropriate statistic to use is the difference in log-likelihood between our two models: $\delta = -2(log\, L_0 - log\, L_1)$. This is a common statistic and, whether or not you knew it, you used δ in Week2 when you calculated the fit of different species abundance distribution models using the Akaike information criterion (AIC). However, determining which model describes the evolution of trait data along a phylogeny isn't as simple as performing an AIC comparison, as likelihood-based methods such as these have high error rates that can give us the wrong answer if we don't know the uncertainty of our estimates (Boettiger et al., 2012). To get around this we can use Monte Carlo simulations to estimate the distribution of δ and determine which model is appropriate. To do this we'll use the package Phylogenetic Monte Carlo (`pmc`) to determine whether Brownian motion or the Ornstein-Uhlenbeck model describes the evolution of niche breadth. From there we will plot our results using `ggplot2`.

Evolutionary Model	Equation	Properties
Brownian motion	$dY_t = \sigma Y_t dW_t$	Trait evolution proceeds as a random walk through trait space. The rate of trait evolution (dY_t) is set by a standard Brownian motion function (W_t) and a set constant (σ). Analogous to random genetic (or ecological) drift.
Brownian motion with a trend	$dY_t = \mu Y_t dt + \sigma Y_t dW_t$	Trait evolution proceeds as a random walk through trait space biased in a particular direction. Analogous to directional evolution at a constant rate (μ) with random genetic drift.
Ornstein-Uhlenbeck	$dY_t = -\alpha(Y_t - \theta)dt + \sigma Y_t dW_t$	Trait evolution proceeds as a random walk through trait toward an optimal trait value (θ). Analogous to stabilizing selection with random genetic drift, where α describes how the strength of stabilizing selection increases with increasing distance from the optimal trait.

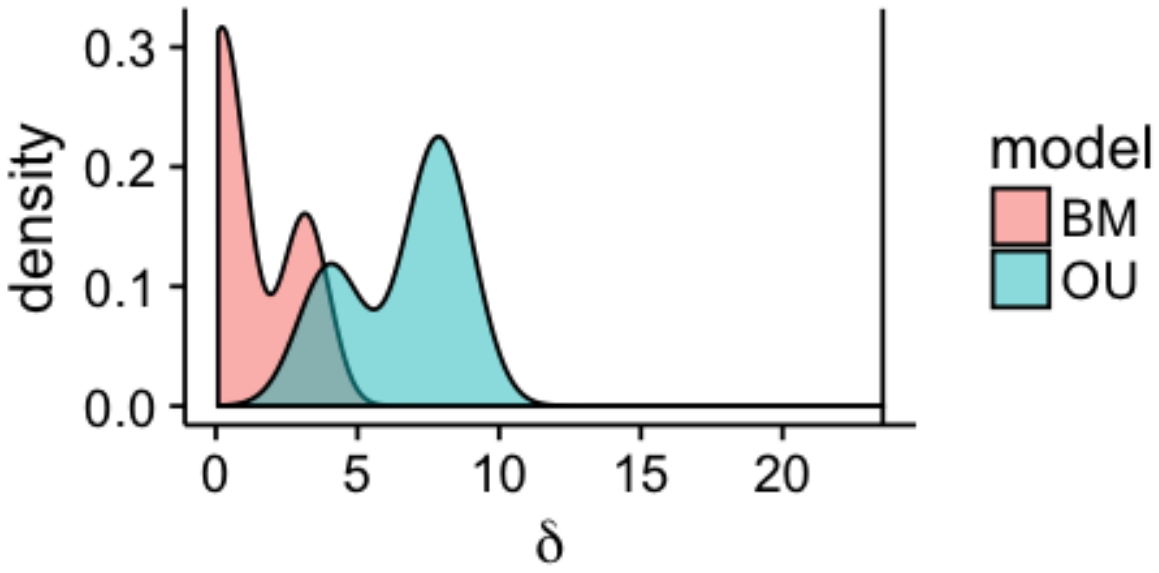


Figure 1: Density plot of the difference in log-likelihood values (δ)

```
# Brownian motion = BM
# Brownian motion + directional selection = trend
# Ornstein-Uhlenbeck = OU
# Pagel's lambda = lambda
is.ultrametric(nj.rooted)
nj.rooted.um <- chronos(nj.rooted)

out <- pmc(nj.rooted.um, nb, "BM", "OU", nboot = 100)
dists <- data.frame(null = out$null, test = out$test)
colnames(dists) <- c("BM", "OU")

#dev.off()

png("./figs/H_test.png", width=480, height=240, res=120)

dists %>%
  gather(model, value) %>%
  ggplot(aes(value, fill = model)) +
  geom_density(alpha = 0.5) +
  geom_vline(xintercept = out$lr) +
  xlab(expression(delta)) +
  theme(panel.background = element_blank())
dev.off()
```

Because the Brownian motion and Ornstein-Uhlenbeck models are nested, the proportion of simulated values larger than our observed δ for the simpler model provides an approximation for the P-value (i.e. the probability that a difference at least as large would be seen under the null model).

```
length(dists$BM[dists$BM > out$lr]) / length(dists$BM)
```

So our P-value is below the traditional alpha of 0.05, implying that we should reject the null hypothesis that Brownian motion (i.e. our simpler model) adequately describes the evolution of niche breadth.

As a side note, often measures of phylogenetic signal build off of these simple stochastic models. For example, Pagel's λ is just the simple Brownian motion model with the internal edges of the phylogeny shortened by λ, reducing the correlations between taxa.

7) 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).

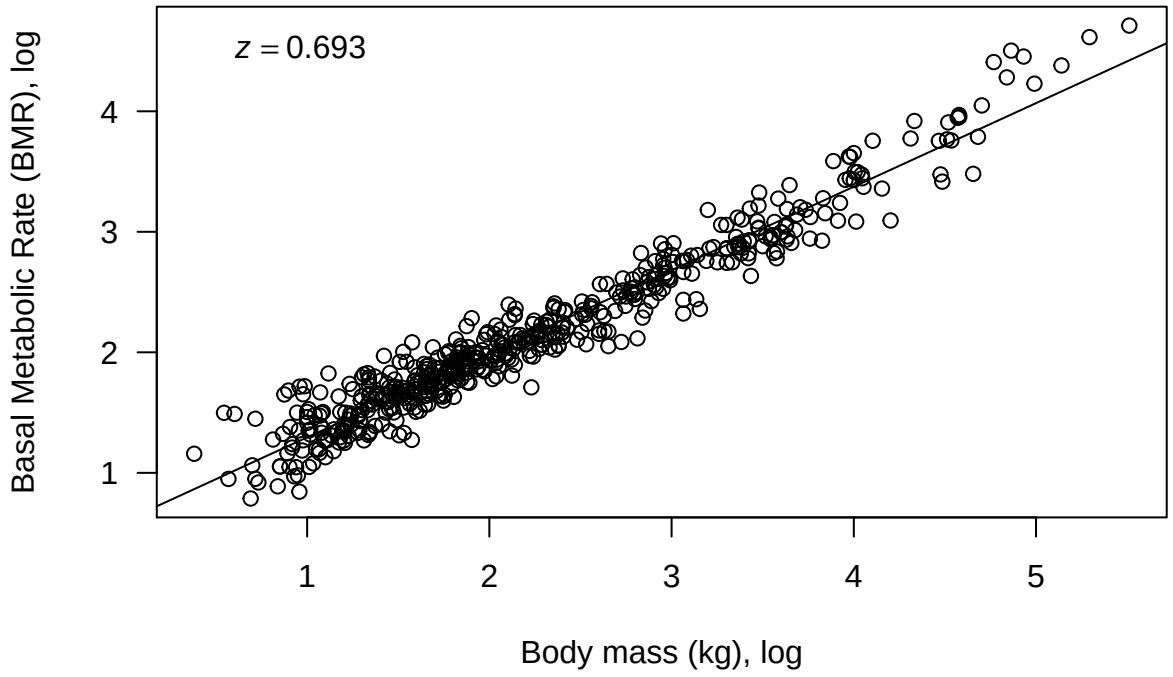
To show how to perform a phylogenetic regression we will examine the scaling relationship (i.e. power law) between body mass (M) and Basal Metabolic Rate (BMR), the minimal rate of energy expenditure per unit time by endothermic animals at rest. This relationship has significant phylogenetic signal and can be described by the power law $BMR = cM^z$, where z is the scaling exponent and c is the scaling factor (Capellini et al., 2010). In regression terms this can be described $log(BMR) = log(c) + zlog(M)$, an equation that you should recognize from Week 4. However, phylogeny is rarely accounted for in foundational power laws such as these. Here we will account for phylogeny in the BMR-mass power law using the super-tree of mammals, a single phylogenetic tree assembled from a combination of smaller phylogenetic trees.

First we will clean the mammal dataset and super-tree.

```
# Input the tree and dataset
mammal.Tree <- read.tree("./data/mammal_best_super_tree_fritz2009.tre")
mammal.data <- read.table("./data/mammal_BMR.txt", sep = "\t",
    header = TRUE)
# Select the variables we want to analyze
mammal.data <- mammal.data[, c("Species", "BMR_.mlO2.hour.",
    "Body_mass_for_BMR_.gr.")]
mammal.species <- array(mammal.data$Species)
# Select the tips in the mammal tree that are also in the
# dataset
pruned.mammal.tree <- drop.tip(mammal.Tree, mammal.Tree$tip.label[-na.omit(match(mammal.species,
    mammal.Tree$tip.label))])
# Select the species from the dataset that are in our prunned
# tree
pruned.mammal.data <- mammal.data[mammal.data$Species %in% pruned.mammal.tree$tip.label,
    ]
# Turn column of Species names into rownames
rownames(pruned.mammal.data) <- pruned.mammal.data$Species
```

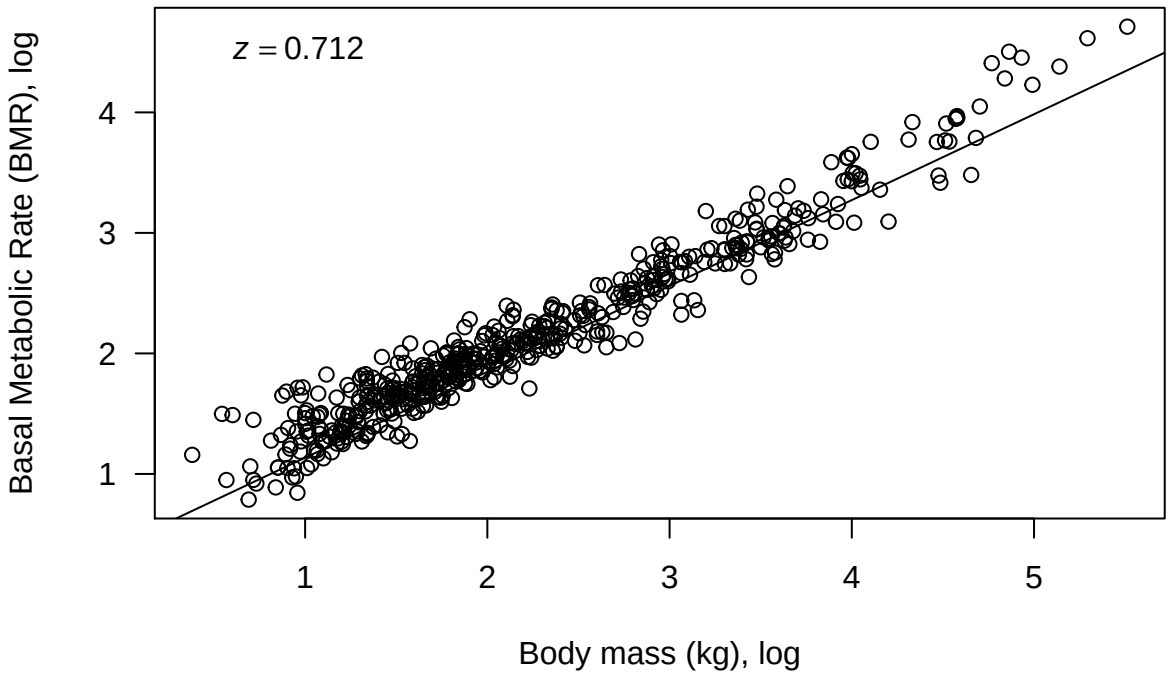
Now let's look at the relationship between mass and BMR:

```
# Run a simple linear regression
fit <- lm(log10(BMR_.mlO2.hour.) ~ log10(Body_mass_for_BMR_.gr.),
    data = pruned.mammal.data)
plot(log10(pruned.mammal.data$Body_mass_for_BMR_.gr.), log10(pruned.mammal.data$BMR_.mlO2.hour.),
    las = 1, xlab = "Body mass (kg), log", ylab = "Basal Metabolic Rate (BMR), log")
abline(a = fit$coefficients[1], b = fit$coefficients[2])
b1 <- round(fit$coefficients[2], 3)
eqn <- bquote(italic(z) == .(b1))
# plot the slope
text(0.5, 4.5, eqn, pos = 4)
```



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

```
# Run a phylogeny-corrected regression with no bootstrap
# replicates
fit.phy <- phylolm(log10(BMR_.mlO2.hour.) ~ log10(Body_mass_for_BMR_.gr.),
    data = pruned.mammal.data, pruned.mammal.tree, model = "lambda",
    boot = 0)
plot(log10(pruned.mammal.data$Body_mass_for_BMR_.gr.), log10(pruned.mammal.data$BMR_.mlO2.hour.),
    las = 1, xlab = "Body mass (kg), log", ylab = "Basal Metabolic Rate (BMR), log")
abline(a = fit.phy$coefficients[1], b = fit.phy$coefficients[2])
b1.phy <- round(fit.phy$coefficients[2], 3)
eqn <- bquote(italic(z) == .(b1.phy))
text(0.5, 4.5, eqn, pos = 4)
```



and we see that we get a power law of approximately $BMR = 0.42M^{0.71}$

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.

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

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.

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.

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.

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., Harmon, L., & Collar, D. (2008). Phylogenetic Signal, Evolutionary Process, and Rate. Systematic Biology, 57(4), 591–601. http://doi.org/10.1080/10635150802302427

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

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

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