Introduction to the phylogenetic (comparative) method

Jamie Winternitz, email: jcwinternitz@gmail.com

Aims

To learn how to use R to apply basic phylogenetic methods. By the end of this practical set you should be able to:

Part I: Introduction to the phylogenetic method

1. Read in your data and phylogeny
2. Match taxa in your phylogeny with those in your dataset
3. View and manipulate your phylogeny
4. Visualize comparative continuous data on phylogenies
5. Visualize comparative discrete data on phylogenies

Part II: Testing evolutionary models accounting for phylogeny

6. Calculate measures of phylogenetic signal (λ and K) using **phytools** and **geiger**
7. Fit different models of trait evolution
8. Perform independent contrasts analyses using **caper**
9. Perform PGLS analyses using **caper**
10. Solve the mystery of the Singing Vole

We will be using the evolution of rodent life-history variables as an example, these data come from the PanTHERIA database (Jones et al. 2009) and the mammal Supertree (Bininda-Emonds et al. 2007). Cytochrome B sequence data was downloaded from Genbank (www.ncbi.nlm.nih.gov/genbank/).

Outline

Part I: Introduction to the phylogenetic method 3
1) Setting the working directory 3
2) Loading packages in R 3
3) Loading the data 4
Species sequence data 5
4) Build phylogenies with sequence data 5
Bootstrap trees for node confidence values 6
5) Loading a phylogeny 10
Tailor existing phylogenies to our dataset 11
6) Graphical methods for visualizing comparative data on phylogenies 12
Continuous character maps 13
Discrete character maps 14
7) Additional plotting information 19
Plotting Symbols 19
Lines 19

Part II: Testing evolutionary models accounting for phylogeny 20
Getting started with data and tree 20
Evolutionary question 20
1) Calculate measures of phylogenetic signal 22
Phylogenetic signal with Pagel's λ (lambda) 22
Phylogenetic signal with Bloomberg's K 25
2) Fit different models of trait evolution 26
3) Perform independent contrasts analyses using caper 26
4) Perform PGLS analyses using caper 28
5) Solve the mystery 30

References

Bininda-Emonds O.R., Cardillo M., Jones K.E., MacPhee R.D., Beck R.M., Grenyer R., Price S.A., Vos R.A., Gittleman J.L., Purvis A. 2007 The delayed rise of present-day mammals. *Nature* 446(7135), 507-512.

Jones K.E., Bielby J., Cardillo M., Fritz S.A., O'Dell J., Orme C.D.L., Safi K., Sechrest W., Boakes E.H., Carbone C. 2009 PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals. *Ecology* 90(9), 2648-2648.

adegenet: Jombart, T. (2008) adegenet: a R package for the multivariate analysis of genetic markers. Bioinformatics 24: 1403-1405. doi: 10.1093/bioinformatics/btn129
ape: Paradis E., Claude J. & Strimmer K. 2004. APE: analyses of phylogenetics and evolution in R language. Bioinformatics 20: 289-290. R package version 3.5.

caper: David Orme, Rob Freckleton, Gavin Thomas, Thomas Petzoldt, Susanne Fritz, Nick Isaac and Will Pearse (2013). caper: Comparative Analyses of Phylogenetics and Evolution in R. R package version 0.5.2. https://CRAN.R-project.org/package=caper

geiger: Harmon Luke J, Jason T Weir, Chad D Brock, Richard E Glor, and Wendell Challenger. 2008. GEIGER: investigating evolutionary radiations. Bioinformatics 24:129-131.

phytools: Revell, L. J. (2012) phytools: An R package for phylogenetic comparative biology (and other things). Methods Ecol. Evol. 3 217-223. doi:10.1111/j.2041-210X.2011.00169.x.

picante: S.W. Kembel, P.D. Cowan, M.R. Helmus, W.K. Cornwell, H. Morlon, D.D. Ackerly, S.P. Blomberg, and C.O. Webb. 2010. Picante: R tools for integrating phylogenies and ecology. Bioinformatics 26:1463-1464.

stringr: Hadley Wickham (2015). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.0.0. https://CRAN.R-project.org/package=stringr.

Part I: Introduction to the phylogenetic method

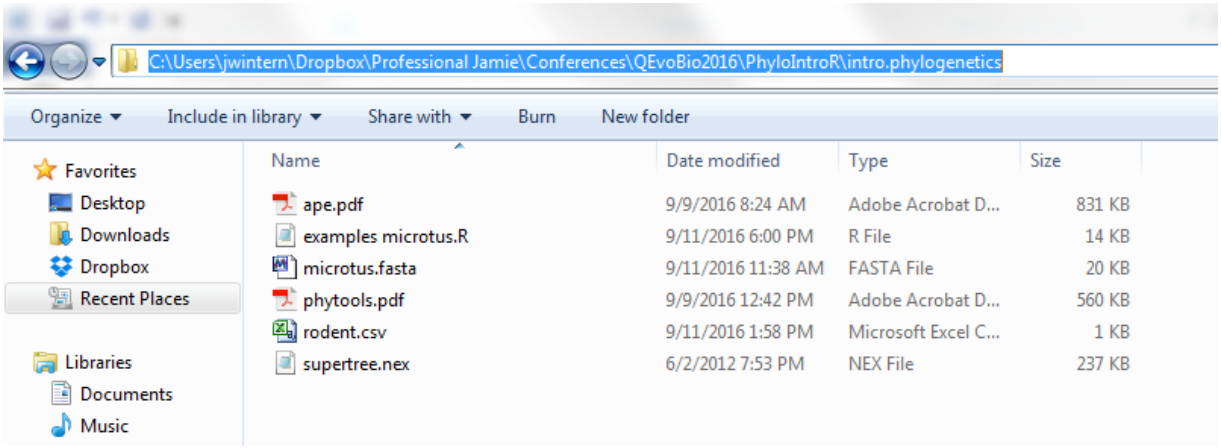
Open Rstudio and start a new R script. You will be adding text in `Lucinda Console` font to the R script and adding your own notes as you go. You can add comments into the R text editor using the hash sign (#) before text. It should turn green and will be ignored. #example

Datasets in 'intro.phylogenetics' folder we will use include:

i) species cytochromeB sequence data in fasta format (microtus.fasta)
ii) supertree nexus file (supertree.nex)
iii) species' trait dataset (rodent.csv)

1) Setting the working directory

Set the working directory to the folder "intro.phylogenetics". To do this, open your folder with the datasets (intro.phylogenetics) and copy the folder path at the top of the screen. For example, my folder path is C:\Users\jwintern\Dropbox\Professional Jamie\Conferences\QEvoBio2016\PhyloIntroR\intro.phylogenetics



You will have to change the backslashes(\) to for-slashes(/) for PCs. So my working directory path is:
setwd("C:/Users/jwintern/Dropbox/Professional Jamie/Conferences/QEvoBio2016/PhyloIntroR/intro.phylogenetics")

Type the text in `Lucida Console` font into the Rstudio script box and then move the cursor to the start of the line and hit 'run' at the top-right hand side of the script.

```
setwd("(fill in your own path here)/intro.phylogenetics")
getwd() #verifies where we are
```

2) Loading packages in R

We need to load packages for each new session of R. We will use the following packages so please load them into the current R session:

adegenet
ape
caper
geiger
phytools
picante
stringr

3) Loading the data

Species trait data

We need to load the species' dataset which includes the scientific names of our species as well as life-history trait data (belly coloration is fake discrete trait data for illustrative purposes). Our data is saved as a comma-delimitated file, but your data could be tab-del (.txt).

```
#read data as comma-delimitated (.csv) file with headers on columns
dataset<-read.csv("rodent.csv",header=TRUE)

#or read data as tab-delimitated (.txt) files
#dataset2<- read.delim("rodent.txt", header=TRUE)
```

If you have your own excel data spreadsheet, you can copy it onto the clipboard and read it in to R

```
# read data copied from clipboard
#dataset3<- read.table(file("clipboard"), header=TRUE)
```

Now let's see this data.

```
#display dataset
dataset
```

The output should look like this:

```
                   Species litter.size body.size_g white.belly
1          Microtus_agrestis        4.65       35.87       white
2           Microtus_arvalis        4.99       26.90       black
3        Microtus_canicaudus        4.62       29.93       brown
4  Microtus_duodecimcostatus        2.51       22.71       brown
5          Microtus_guentheri        6.14       49.99       black
6           Microtus_kikuchii        1.97       46.30       white
7        Microtus_longicaudus        4.73       44.80       brown
8           Microtus_montanus        5.54       42.85       brown
9         Microtus_montebelli        4.30       29.49       white
10       Microtus_ochrogaster        3.87       42.50       brown
11         Microtus_oeconomus        5.62       33.10       white
12           Microtus_oregoni        3.32       20.35       brown
13   Microtus_pennsylvanicus        5.16       42.53       brown
14         Microtus_pinetorum        2.47       25.95       brown
15      Microtus_subterraneus        2.46       17.74       brown
16        Microtus_townsendii        5.06       52.02       brown
```

We can look at the structure of the data using str()

```
str(dataset)
```

This shows that 'dataset' is a data.frame object with 16 observations of 4 variables, 2 factor variables and 2 numeric variables.

We could also create fake random data, if you wish.

```
Species<-seq(1:16) #vector of the numbers 1 to 16
dataset<-data.frame(Species)

#create random life-history data for your species
dataset$litter.size<-sample(1:12, 16, replace=TRUE)
dataset$body.size_g<-sample(15:40, 16, replace=TRUE)
#for random character states
dataset$white.belly<-sample(c("white","brown"), 16, replace=TRUE)
```

Let's see what this variable looks like

```
dataset$white.belly
```

The output:

```
[1] "white" "brown" "white" "brown" "brown" "brown" "brown" "white" "white
" "brown" "white" "brown" "white" "white"
[15] "brown" "white"
```

Species sequence data

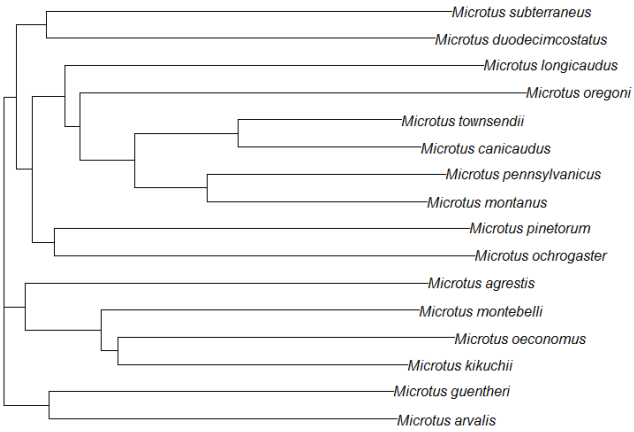
If we have our own sequence data for our species we're studying, we can use this to build a phylogenetic tree. First we need to load the fasta file of our gene, in this case, CytochromeB. We can do this with the **ape** package, which stands for Analysis of Phylogenetics and Evolution in R.

```
xj <-read.dna("microtus.fasta", format="fasta")
lab<-labels(xj)

#replace the labels by Genbank ascension numbers
asc.num <- gsub( " .*$", "", lab) #you are replacing all text
elements after the first whitespace with nothing

library(stringr)
#replace the names by species names (keep only the 2nd and 3rd word)
name.microtus <- word(lab, 2,3)

rownames(xj)<-name.microtus #change the rownames of the DNAbin
matrix to just the species names

image(xj, cex=0.6) #plot the sequences across species
```

4) Build phylogenies with sequence data

Now we can build phylogenies with our sequence data. We first calculate a distance matrix based on a specific distance model. Today we are using nucleotide differences, but there are many more options. You should use model comparison methods (such as likelihood ratios implemented by ModelTest) to select the best model for your data.

```
#calculate distance matrix from sequence data using number of nuc
differences
distxj<-dist.dna(xj, model="raw")

#calculates the tree by neighbor joining
tree <- nj(distxj)

#plot a basic tree
plot.phylo(tree, type="phylogram")
```



We already changed the DNAbin rownames, but here's how to rename the tip labels by species names:

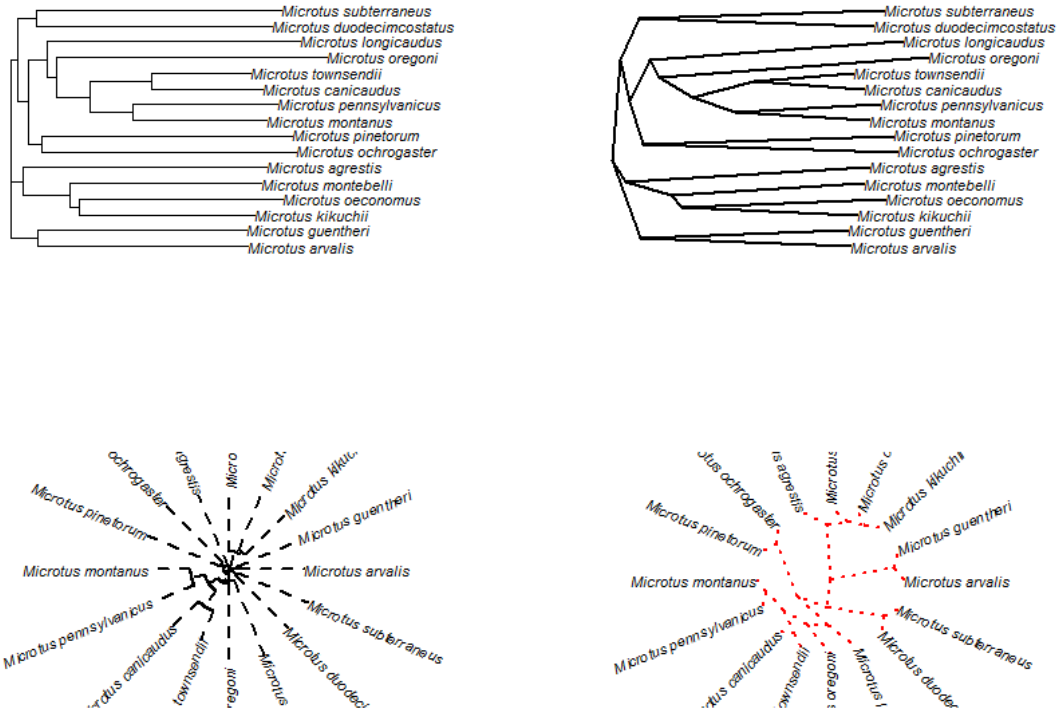
```
tree$tip.label<-name.microtus
```

See the order of the species by the tip numbers:

```
tiplabels()
```

Plot different styles of the tree:

```
par(mfrow=c(2,2)) #plot 2 rows by 2 columns
plot.phylo(tree, type="phylogram")
plot.phylo(x=tree, type="cladogram", edge.width=2)
plot.phylo(x=tree, type="fan", edge.width=2, edge.lty=2, cex=.8)
plot.phylo(x=tree, type="radial", edge.color="red", edge.width=2,
edge.lty=3, cex=.8)
```



To reset the graphic window to just one graph:

```
par(mfrow=c(1,1))
```

Bootstrap trees for node confidence values

To get values of confidence for our nodes we can bootstrap the tree and see how many times the set of species at the nodes are consistent. The tree must be rooted (with an outgroup), and we can choose any. To see the choices, use tiplabel().

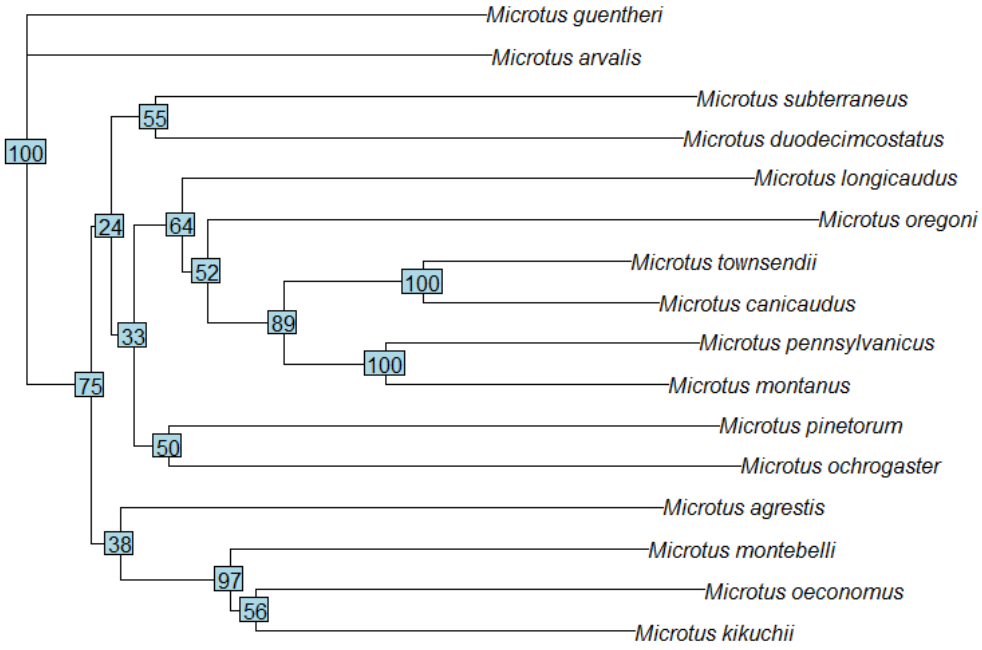
```
outgroup <- 2 #may be several tips, numeric or tip labels
```

Function to root by outgroup:

```
foo <- function(xx) root(nj(dist.dna(xx)), outgroup)
tr <- foo(xj) #xj is the matrix of DNA sequences, tr is the rooted
nj tree
bp <- boot.phylo(tr, xj, foo, B=1000) #1000 bootstraps
```

Plot the tree with bootstrap values at the nodes:

```
plot(tr)
nodelabels(round(bp/10)) # will have "100" at the root
```

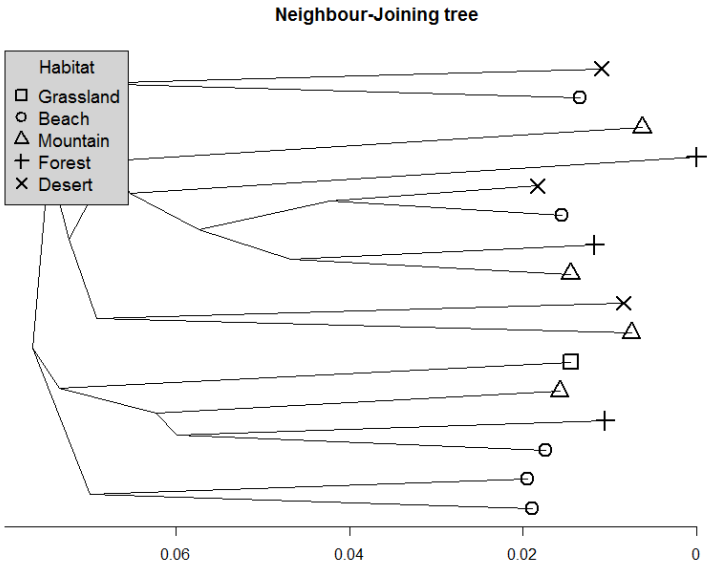


We can plot black and white tree with tip symbols and a legend.

```
plot.phylo(x=tree, type="cladogram", show.tip=FALSE, lwd=3,
main="Neighbour-Joining tree")
#add axis with distances
axisPhylo()

#use symbols (pch) for tip labels
tiplabels(frame="none", pch=rep(x=0:4, times=c(1, 5,  4,  3,  3)),
lwd=2, cex=2)
```

Plot a legend explaining symbols (randomly distributed habitats):

```
legend(x="topleft", legend=c("Grassland", "Beach", "Mountain",
"Forest", "Desert"), border="black", pch=0:4, pt.lwd=2, pt.cex=1.5,
bty="o", bg="lightgrey", box.lwd=1, cex=1.2, title="Habitat")
```



Here are some fancy tree visualization methods using the **adegenet** package.

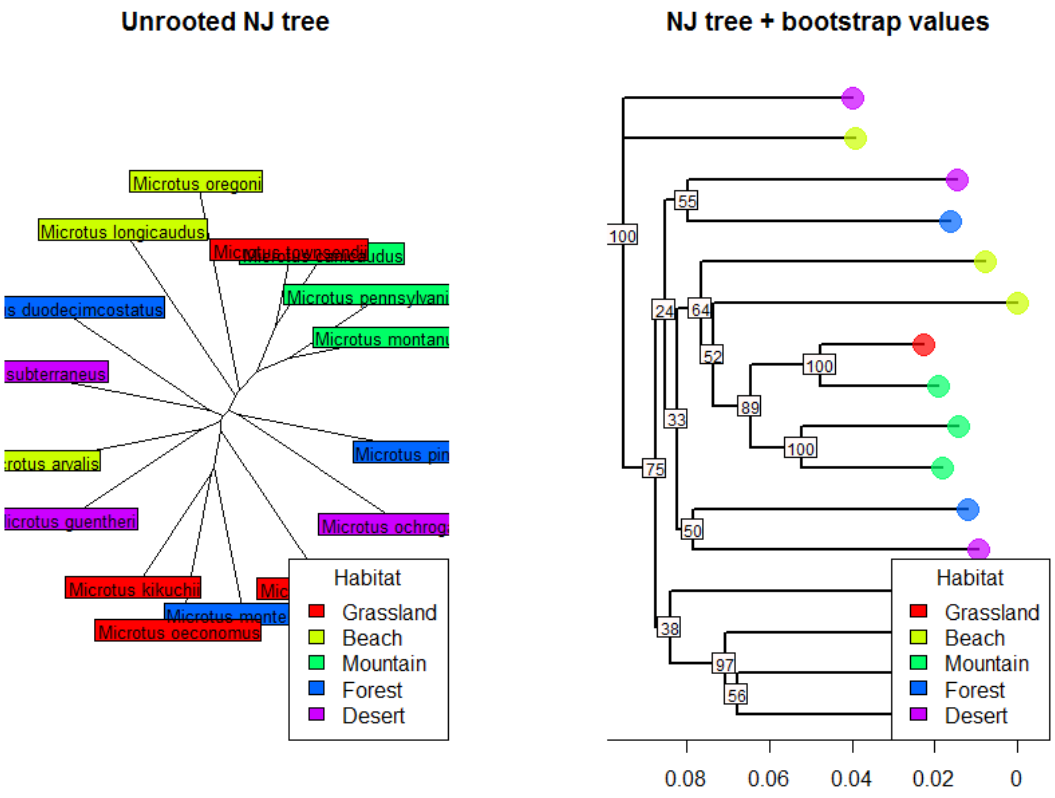
```
par(mfrow=c(1,2)) #1 row, 2 columns
```

Plot unrooted nj tree with tips in colored boxes that designate habitats:

```
plot.phylo(x=tree, type="unrooted", show.tip=FALSE)
title("Unrooted NJ tree")
#coloured tips
rod.pal <- rainbow(5) #will select 5 colors from the rainbow palete
#tip labels
tiplabels(text=tree$tip.label, bg=rod.pal, cex=0.8)
legend(x="bottomright", fill=rod.pal,
       leg=c("Grassland", "Beach", "Mountain", "Forest", "Desert"),
title="Habitat")
```

Plot the tree with colored tips and with bootstrap node values (transform to percent):

```
plot.phylo(x=tr, show.tip=FALSE, edge.width=2)
title("NJ tree + bootstrap values")
tiplabels(frame="none", pch=20, col=transp(rod.pal, alpha=0.7),
cex=3.5, fg="transparent")
axisPhylo()
legend(x="bottomright", fill=rod.pal,leg=c("Grassland", "Beach",
"Mountain", "Forest", "Desert"), title="Habitat")
nodelabels(text=round(bp/10), cex=0.75, bg="snow")
```



We can collapse branches with low bootstrap support (less than 75% consistency).

```
tree2.rod.na.density <- tr
```

Determine branches with low support:

```
rod.tocollapse <- match(x=which(bp < 750)+length(tr$tip.label),
table=tree2.rod.na.density$edge[,2])
```

Set length of bad branches to zero:

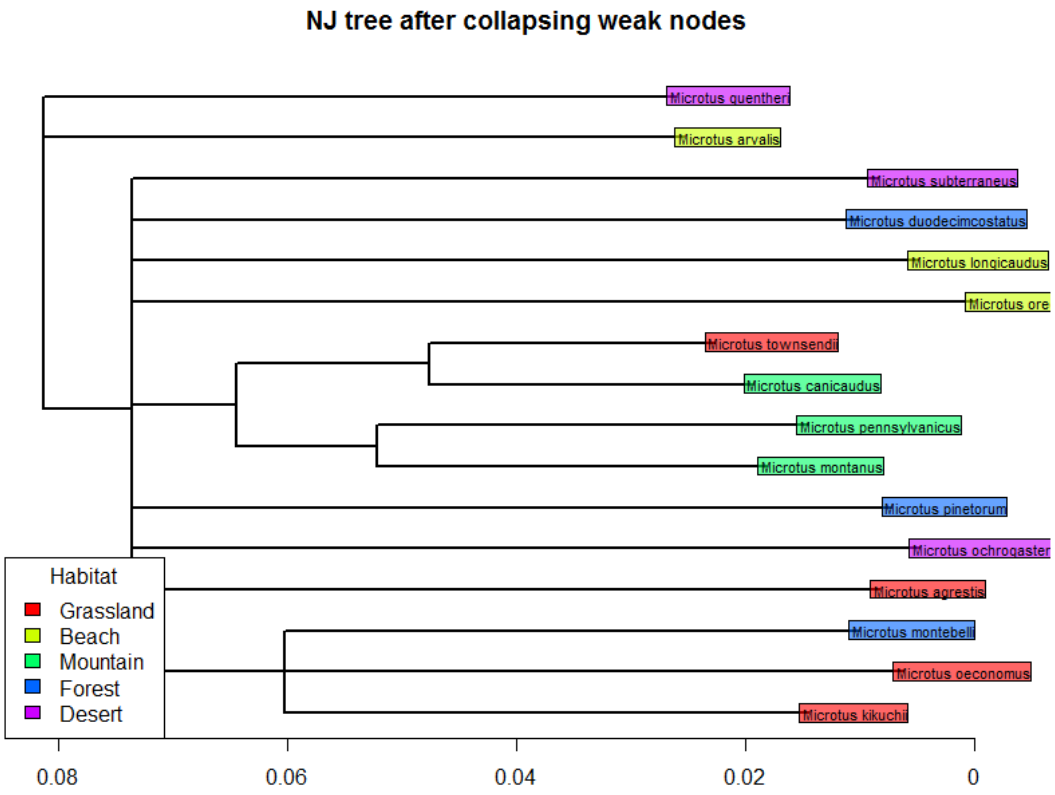
```
tree2.rod.na.density$edge.length[rod.tocollapse] <- 0
```

Create the new tree:

```
tree2.collapsed <- di2multi(phy=tree2.rod.na.density, tol=0.00001)
```

Plot the consensus tree:

```
plot.phylo(x=tree2.collapsed, show.tip=FALSE, edge.width=2)
title("NJ tree after collapsing weak nodes")
tiplabels(text=tree$tip.label, cex=.7, frame="rect", adj=c(.05,0.5),
          bg=transp(rod.pal, alpha=0.6))
axisPhylo()

legend(x="bottomleft", fill=rod.pal,leg=c("Grassland", "Beach",
"Mountain", "Forest", "Desert"), title="Habitat")
```



If we want to add marks to the species tips, and specify the colors, first find out the tip label order.

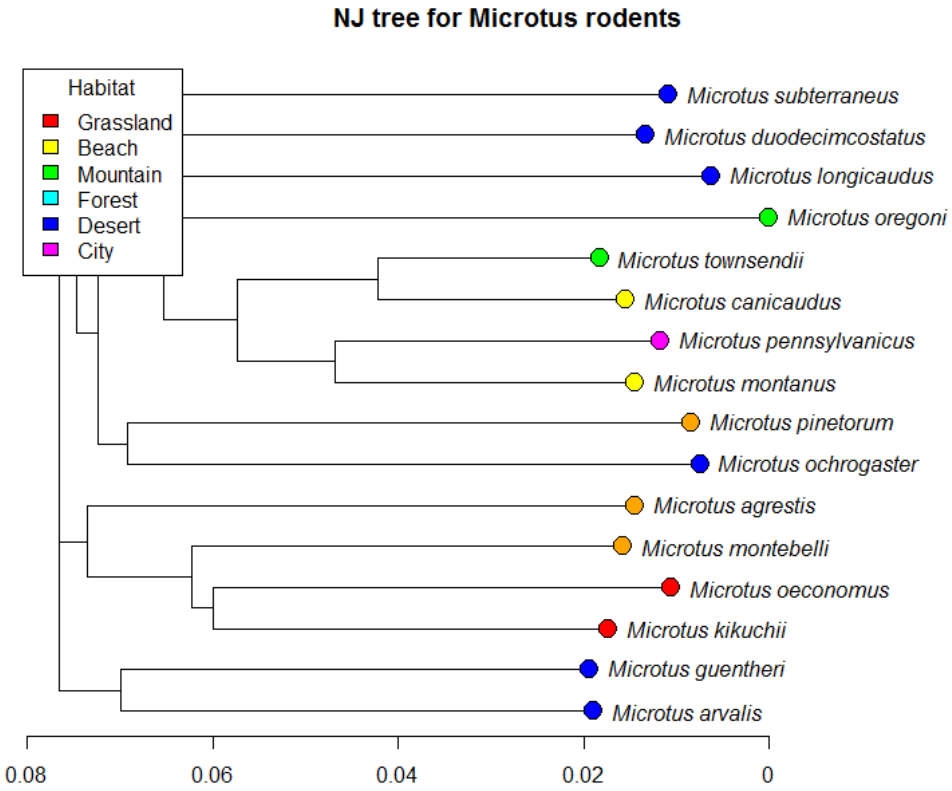
```
tiplabels()#find the order of the tip labels
```

Make a vector of colors with the same number of colors as tips.

```
mycol<-c("orange", "blue", "yellow", "blue", "blue", "red", "blue",
"yellow", "orange", "blue", "red", "green", "#FF00FFFF", "orange",
"blue", "green")
```

We can also adjust the tree plot space by changing the original par(mar= c(5.1 4.1 4.1 2.1)). This stands for c(bottom, left, top, right). The current parameter size can be found using:

```
par()$mar

plot(x=tree, adj=0, lwd=2, label.offset=.002)
tiplabels(pch=21, col="black", adj=0.5, bg=mycol, cex=2)
title("NJ tree for Microtus rodents")
axisPhylo()
legend(x="topleft", fill=rainbow(6),
 leg=c("Grassland", "Beach", "Mountain", "Forest", "Desert",
"City"), title="Habitat")
```



We will export the rooted tree for easy access for Part II, using the write.nexus() command. IMPORTANT: species names must not have spaces between genus and species.

```
write.nexus(tree, file="microtus_tree.nex")
```

5) Loading a phylogeny

We can load an existing tree using the function read.tree or read.nexus. Read.nexus can only deal with NEXUS files. Today, our tree is the mammalian supertree saved in NEXUS format. First load mammal supertree.

```
supertree<-read.nexus("supertree.nex")
```

We can examine the tree structure (str)

```
str(supertree)
supertree
```

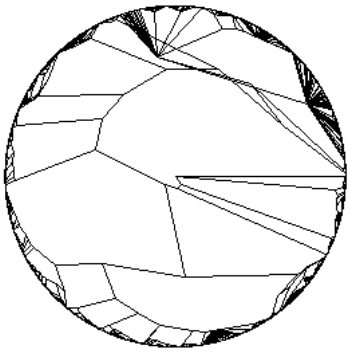
Our supertree has 4510 tips (species) and 2108 internal nodes. It is rooted with branch lengths. Is it fully resolved (i.e., fully binary) with no polytomies?

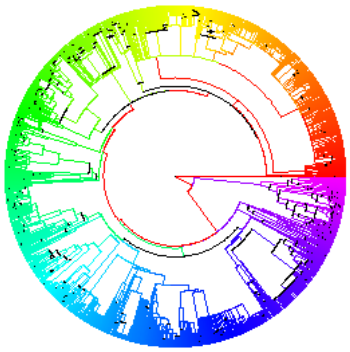
```
is.binary.tree(supertree)
```

Nope. That's ok for now. We can plot the tree by using the plot function of ape.

```
par(mfrow=c(1,2))
plot(supertree, type="radial", show.tip=FALSE)

#plot colorful tree
colpal<-rainbow(8000)
```

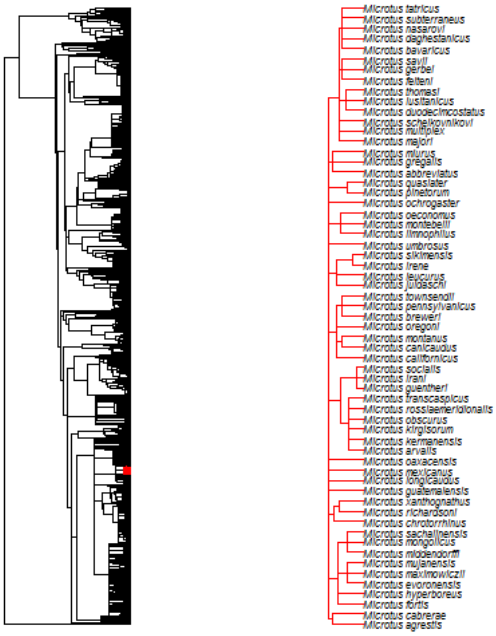
```
plot(supertree, type="fan", show.tip=FALSE, edge.color = colpal)
```





This is a HUGE tree (4510 species). Let's zoom in to the sections we're interested in:

```
#just gives you the tree for Microtus species
zoom(supertree, list(grep("Microtus", supertree$tip.label)), subtree
= FALSE, cex=.5)
```



This code just gives you the tree for Microtus species but also shows you how it fits into the rest of the tree.

```
zoom(supertree, list(grep("Microtus", supertree$tip.label)), subtree
= TRUE, cex=.5)
```

Tailor existing phylogenies to our dataset

We can use the name checking function from **geiger** package to see that dataset and tree species are the same.

```
name.check(supertree, dataset, data.names = dataset$Species)
```

We have 4495 species in the tree not in our dataset, and 1 species (Microtus_kikuchii) in our dataset that is not in the supertree. A common error is when dataset species have spaces between Genus and species. We can fix this by substituting spaces for underscores.

```
#replace any whitespace with underscore
dataset$Species<-gsub(" ", "_", dataset$Species)
```

We want to find species in tree that are in the dataset.

```
matches <- match(dataset$Species, supertree$tip.label, nomatch = 0)
```

Remove species in dataset not in the tree (should be 15 left in the dataset).

```
data <-subset(dataset, matches !=0)
not_in_data<-setdiff(supertree$tip.label, data$Species)
```

Remove species in tree not in the dataset

```
rodent.tree<-drop.tip(supertree, not_in_data)
```

Check that the tree and data match

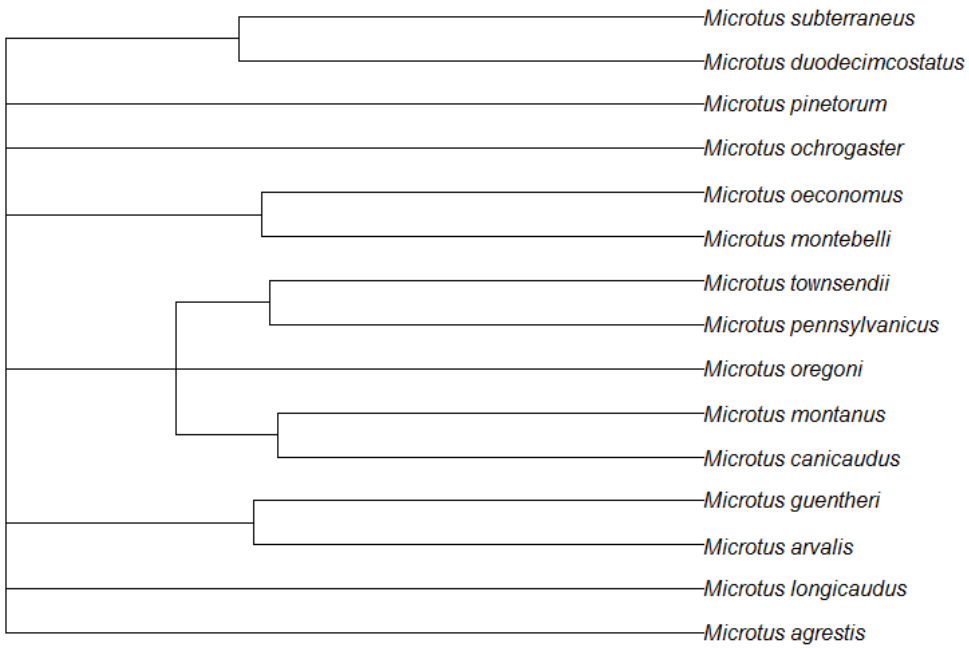
```
name.check(rodent.tree, data, data.names = data$Species)
```

Check original and reduced datasets

```
str(dataset)
str(data)
```

Is the rodent tree fully resolved?

```
is.binary.tree(rodent.tree)
plot(rodent.tree) #see the polytomies
```



The multi2di command will break polytomies in random order.

```
rodent.tree2<-multi2di(rodent.tree, random=TRUE)
```

Now check if it is fully resolved

```
is.binary.tree(rodent.tree2)
```

Output:

```
[1] TRUE
```

6) Graphical methods for visualizing comparative data on phylogenies

To reset the plot window you can touch the broom which will clear all plots and parameters. Now we are going to first visualize continuous data and then discrete data for our phylogenies using the

package **phytools**. Visualizing data is extremely helpful at both the beginning and end stages of data analysis. At the early stages, visualization can alert us to errors in the data or suggest other questions to investigate. At the last stages, visualization allows us to present persuasive and informative figures (Revell 2014, in Garamszegi (ed.) *Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology*, Springer-Verlag Berlin).

Continuous character maps

This maps continuous traits, like body size, to the phylogenetic tree. With these graphics we are able to visualize how the dimensions of body size are distributed among species.

First, make sure the species names for the trait data and the tree tips match exactly.

```
#replace whitespace with underscore
tree$tip.label<-gsub(" ", "_", tree$tip.label)
```

Next, subset the dataset to remove any duplicated species.

```
dat<-dataset
X2<-dat[!duplicated(dat[,'Species']),]
row.names(X2)<-unique(X2[,'Species'])
```

X2 is your unique species dataset with species names as row names. This is important because we need names for the trait data when we extract it and convert it to a matrix.

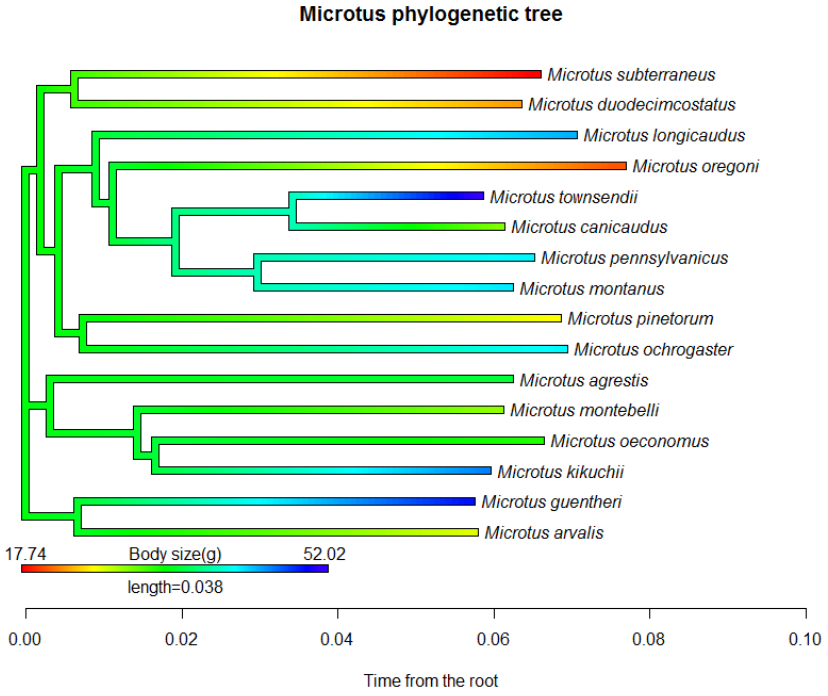
```
body.size<-as.matrix(X2)[,3] #choosing the 3rd column of dataset X2
mode(body.size)<-'numeric' #should be a named numeric matrix
str(X2) #continuous variables should be labeled as 'numeric'
X2$body.size_g<-as.numeric(X2$body.size_g)
```

Assigning body size as a tree tip label:

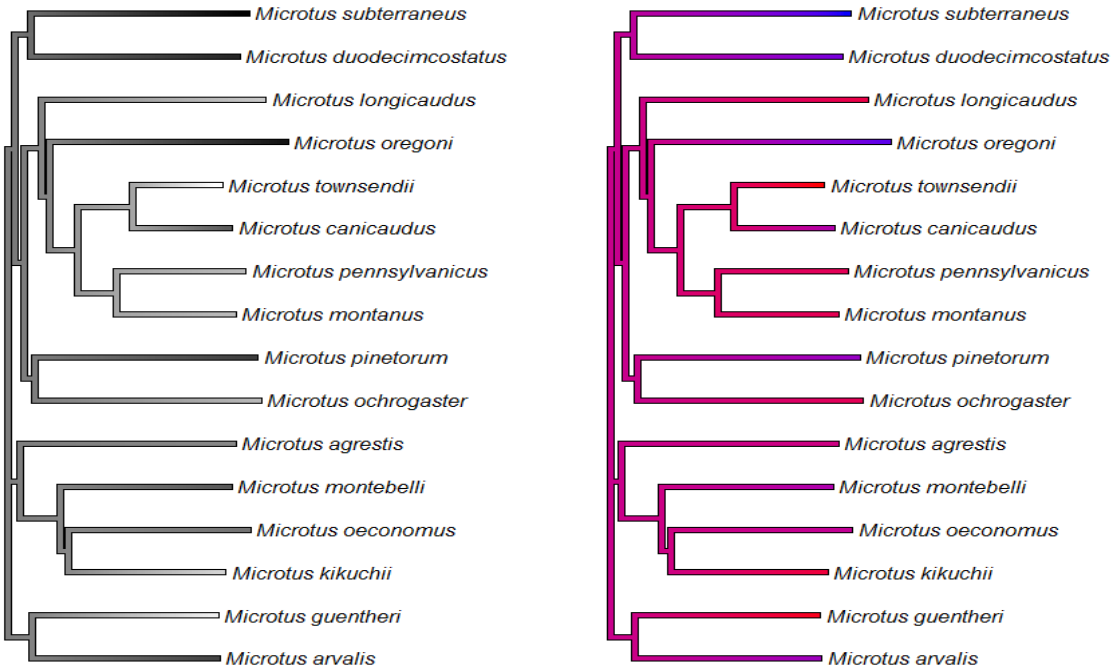
```
body.size<-body.size[tree$tip.label]
```

Now we have the trait data object to map to our tree.

```
obj<-contMap(tree,body.size,plot=FALSE)
plot(obj,type="phylogram",leg.txt="Body size(g)",lwd=6,
mar=c(4,2,4,2))
title(main="Microtus phylogenetic tree")
axis(1)
title(xlab="Time from the root")
```



```
par(mfrow=c(1,2))

#what is the length of the current color ramp?
n<-length(obj$cols)

#change to grey scale
obj$cols[1:n]<-grey(0:(n-1)/(n-1))
plot(obj, legend=FALSE)

#change from blue to red
obj$cols[1:n]<-colorRampPalette(c("blue","red"), space="Lab")(n)
plot(obj, legend=FALSE)
```




```
#plot as a fan
plot(obj, type="fan", lwd=5,legend=FALSE)
```

Discrete character maps

Discrete character maps allow us to visualize the occurrence of discrete traits, like the presence/absence of a certain behavior (e.g., burrowing, paternal care, ect.) or phenotypes (in this case, belly coloration), on a phylogenetic tree. This visualization technique is helpful when trying to reconstruct the evolutionary origins of traits among species.

As before, we need to make a vector of our character data named with the corresponding species names.

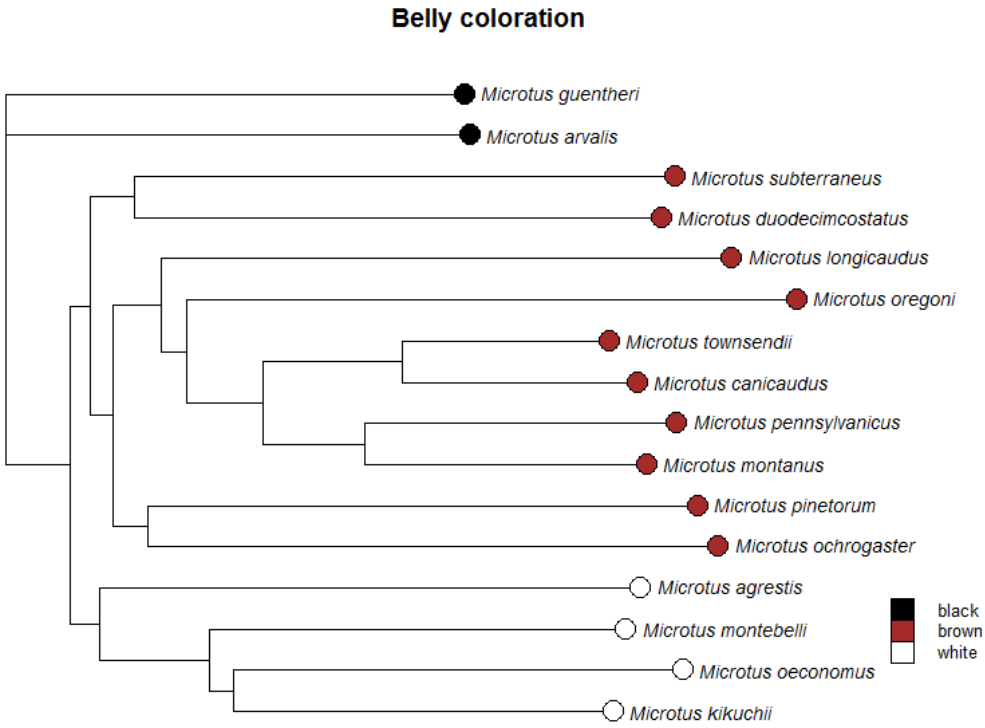
```
mbc<-dataset$white.belly
mbc<-data.frame(mbc)
rownames(mbc)<-tree$tip.label
mbc<-as.matrix(mbc)[,1] #taking the first column from the matrix
x<-mbc
```

Next we will plot the rooted species tree we made with our sequence data.

```
tree<-tr #rooted tree
plot(tree,type="phylogram", cex=0.9, lwd=2, label.offset=.002)
title("Belly coloration")
```

Now we can plot the character traits as colored 'pies' at the tips:

```
cols<-setNames(c("black","brown","white"), sort(unique(x)))
tiplabels(pie=to.matrix(x,sort(unique(x))),piecol=cols,cex=0.5)
#prompt=TRUE means you click to draw the legend
add.simmap.legend(colors=cols,prompt=TRUE,fsize=0.8,
                  vertical=TRUE, shape="square")
```



Nice! But is there a way to reconstruct the ancestral states of belly coloration? Yep, the ace command in **ape** can reconstruct ancestral states for discrete characters using maximum likelihood. We will use the one-parameter equal rates model (ER) to specify the transition probabilities between the states of our discrete character. IMPORTANT: The tree must be rooted and fully dichotomous.

First, double check that the tree is fully dichotomous.

```
tree<-multi2di(tree)
is.binary.tree(tree)
```

And the tree must be rooted.

```
outgroup <- 1 # may be several tips, numeric or tip labels
tree<-root(tree, outgroup, node = NULL, resolve.root = TRUE)
is.rooted(tree) #check if tree is rooted
plot(tree) #to double check
```

Finally, the tree must have greater than zero branch lengths. We can set zero-length branches to be 1/1,000,000 total tree length.

```
dtree<-tree
dtree$edge.length[dtree$edge.length==0]<-max(nodeHeights(tree))*1e-6
```

Ok, now let's run the ace model.

```
ans<-ace(x,dtree, model="ER", type="discrete")
ans
```

Output should be:

```
    Ancestral Character Estimation

Call: ace(x = x, phy = dtree, type = "discrete", model = "ER")

    Log-likelihood: -11.06123

Rate index matrix:
      black brown white
black     .     1     1
brown     1     .     1
white     1     1     .

Parameter estimates:
 rate index estimate std-err
          1   1.5203  0.7928

Scaled likelihoods at the root (type '...$lik.anc' to get them for all nodes):
    black     brown     white
0.0338532 0.1744503 0.7916965
```

Let's see the likelihoods for each state for the nodes of the tree. Higher likelihoods imply more confidence the node is a specific state. For example, node 5 is very likely to be white (0.995).

```
round(ans$lik.anc,3) #round to 3 places
```

Output:

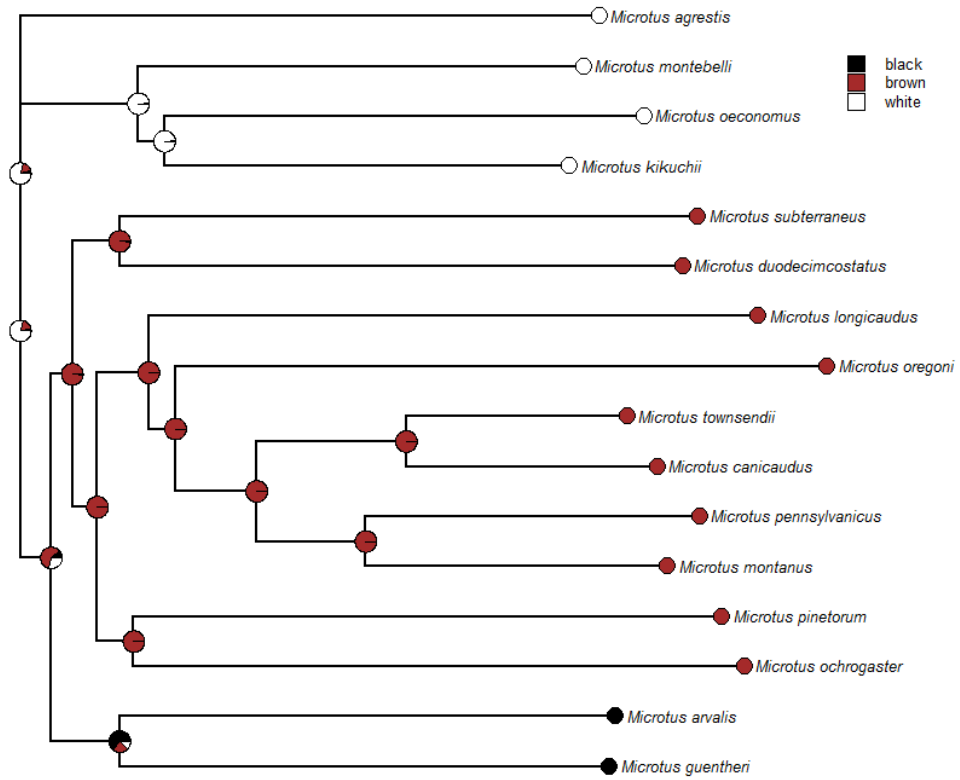
```
     black brown white
 [1,] 0.034 0.174 0.792
 [2,] 0.647 0.235 0.118
 [3,] 0.112 0.595 0.292
 [4,] 0.001 0.007 0.992
 [5,] 0.001 0.004 0.995
 [6,] 0.003 0.991 0.007
 [7,] 0.000 1.000 0.000
 [8,] 0.000 1.000 0.000
 [9,] 0.000 1.000 0.000
[10,] 0.000 1.000 0.000
[11,] 0.000 1.000 0.000
[12,] 0.000 1.000 0.000
[13,] 0.000 1.000 0.000
[14,] 0.002 0.995 0.004
[15,] 0.034 0.174 0.792
```

The element lik.anc gives us the marginal ancestral states, also known as the 'empirical Bayesian posterior probabilities.' We can overlay these posterior probabilities on the tree:

```
plotTree(tree,type="phylogram",fsize=0.8,ftype="i", offset=.5)
```

Now we plot character state probabilities at node labels:

```
nodelabels(node=1:tree$Nnode+Ntip(tree),
           pie=ans$lik.anc,piecol=cols,cex=0.5)
tiplabels(pie=to.matrix(x,sort(unique(x))),piecol=cols,cex=0.4)

#prompt=TRUE means you click to draw the legend
add.simmap.legend(colors=cols,prompt=TRUE,fsize=0.8,
                  vertical=TRUE, shape="square")
```



As an alternative to maximum likelihood we can use an Markov-Chain Monte Carlo approach to sample character histories from their posterior probability distribution. This is called stochastic character mapping (Huelsenbeck et al. 2003). The model is the same but in this case we get a sample of unambiguous histories for our discrete character's evolution **on the tree** - rather than a probability distribution for the character **at nodes**.

First, we simulate 300 stochastic character maps using empirical Bayes method:

```
trees<-make.simmap(tree, x ,model="ER", nsim=300)
#get the states at ancestral nodes
aa<-describe.simmap(trees,plot=FALSE)
aa
```

Output should be:

```
300 trees with a mapped discrete character with states:
 black, brown, white

trees have 2.91 changes between states on average

changes are of the following types:
     black,brown black,white brown,black brown,white white,black white,brown
x->y   0.1533333  0.08666667   0.9066667        0.44   0.5166667   0.8066667

mean total time spent in each state is:
          black     brown     white    total
raw  0.09808591 0.6156469 0.2328992 0.946632
prop 0.10361567 0.6503551 0.2460293 1.000000
```

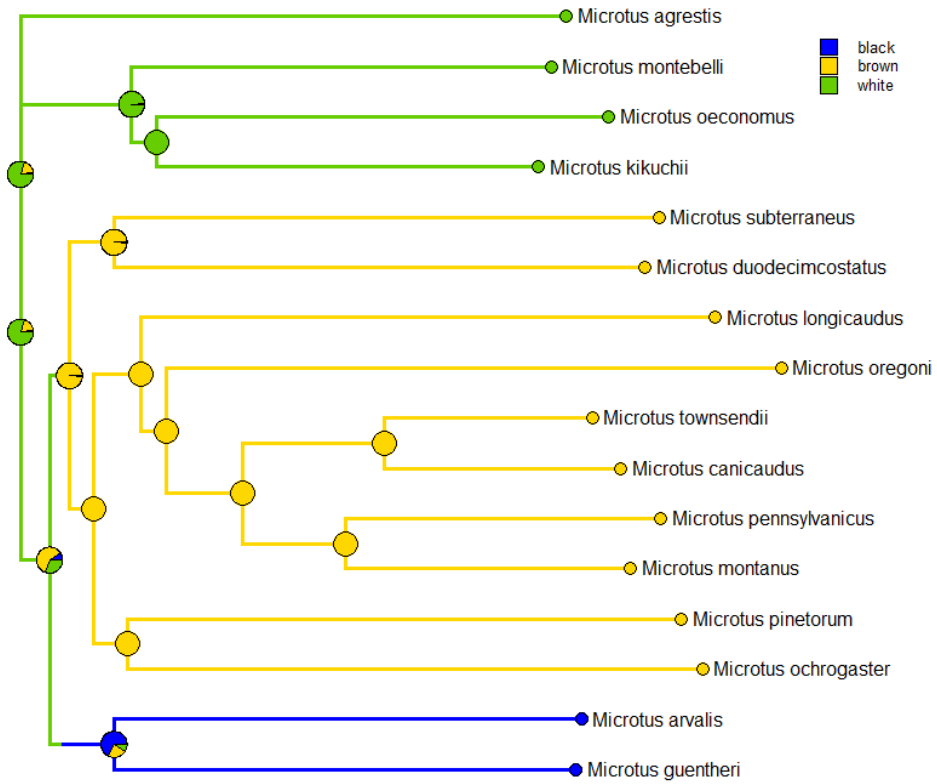
For variety, let's change the colors. You can choose your own colors; just make sure to have as many as you do levels of that variable (3 in this case).

```
cols<-setNames(c("blue","gold1","chartreuse3"),sort(unique(x)))
```

We'll plot a random map (the first), and overlay the posterior probabilities.

```
plotSimmap(trees[[1]],cols,pts=FALSE, lwd=3, setEnv=TRUE, offset=.5)
nodelabels(pie = aa$ace, piecol = cols, cex = 0.6)
tiplabels(pie=to.matrix(x,sort(unique(x))),piecol=cols,cex=0.3)

#prompt=TRUE means you click to draw the legend
add.simmap.legend(colors=cols,prompt=TRUE,fsize=0.8,vertical=TRUE,
shape="square")
```

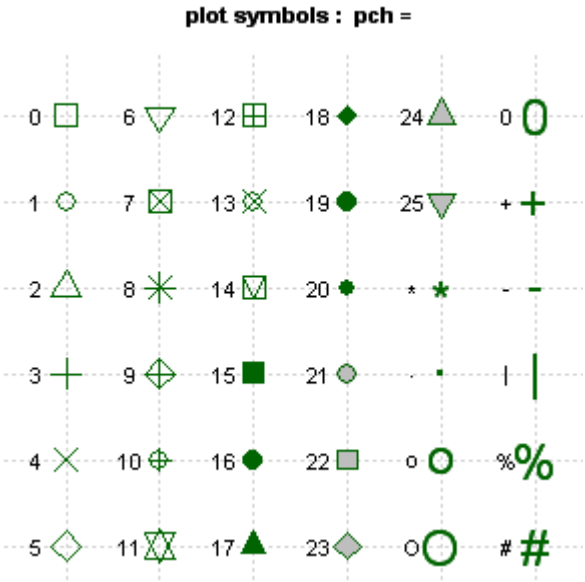
Our plot will now show the probability distributions for the belly color trait along the branches of the tree, with the probabilities plotted as pies at each node. We'll also plot the state of the trait at the tree tips.



7) Additional plotting information
Here are some useful graphing option values for plotting:

Plotting Symbols

Use the **pch=** option to specify symbols to use when plotting points. For symbols 21 through 25, specify border color (col=) and fill color (bg=).



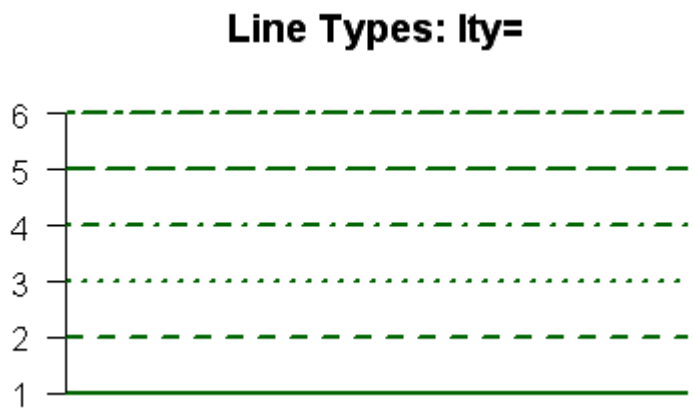
Lines

You can change lines using the following options. This is particularly useful for reference lines, axes, and fit lines.

option description

lty line type. see the chart below.

lwd line width relative to the default (default=1). 2 is twice as wide.



Part II: Testing evolutionary models accounting for phylogeny

By the end of this section, you should be able to:

1. Calculate measures of phylogenetic signal (λ and *K*) using **phytools** and **geiger**
2. Fit different models of trait evolution
3. Perform independent contrasts analyses using **caper**
4. Perform PGLS analyses using **caper**
5. Solve the mystery

Getting started with data and tree

Let's start with the 16 species data from 'rodent.csv' and the tree we made from the sequence data (and exported as "microtus_tree.nex").

```
dataset<-read.csv("rodent.csv",header=TRUE)
tree<-read.nexus("microtus_tree.nex")
```

Checking if our tree is fully resolved (i.e. dichotomous):

```
is.binary.tree(tree)
[1] TRUE
```

Checking if our tree is rooted, and that the root is non-zero length:

```
is.rooted(tree) #check if tree is rooted
[1] FALSE

tree<-root(tree, outgroup, node = NULL, resolve.root = TRUE)
is.rooted(tree) #check if tree is rooted
[1] TRUE
```

Now we'll set any zero-length branches to one-ten-thousandth of the tree size

```
tree$edge.length[tree$edge.length==0]<-max(nodeHeights(tree))*1e-4
```

Making sure tree tip names match dataset names EXACTLY

```
tree$tip.label<-gsub(" ", "_", tree$tip.label) #replace any
whitespace with underscore

rownames(dataset)<-dataset$Species #name the data rows
name.check(tree, dataset)
[1] TRUE
```

Evolutionary question

We are interested in how body mass relates to litter size (i.e., the average number of pups produced during one breeding season). We may speculate that larger rodents should be capable of producing larger litters. Let's run an ordinary least squares (OLS) regression to test this prediction.

But first, we should look at the data:

```
par(mfrow=c(2,2))
hist(dataset$body.size_g)
hist(dataset$litter.size)
```

Body size is not a 'bell-shaped' normal distribution, so let's log10 transform it and see if that improves the assumption of normality

```
par(mfrow=c(1,1))
hist(log10(dataset$body.size_g))
```

That looks better. Now we can make a new log10 transformed variable to work with the more normally-distributed body size data.

```
dataset$log.body.size_g<-log10(dataset$body.size_g)
```

We can perform a non-phylogenetic regression to see what happens. lm=linear model (OLS regression).

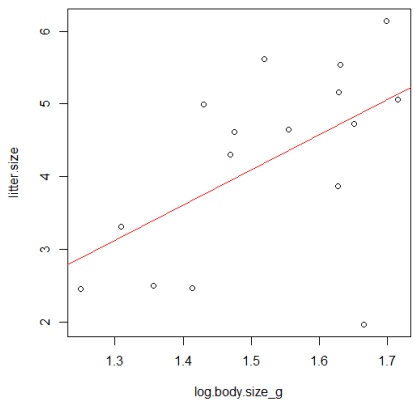
```
model1<-lm(litter.size~log.body.size_g, data=dataset)
summary(model1)
```

Output should be:

```
Call:
lm(formula = litter.size ~ log.body.size_g, data = dataset)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9210 -0.5276  0.2215  0.6844  1.4315

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      -3.137      3.091  -1.015   0.3275
log.body.size_g   4.820      2.019   2.388   0.0316 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.132 on 14 degrees of freedom
Multiple R-squared:  0.2893,      Adjusted R-squared:  0.2386
F-statistic:   5.7 on 1 and 14 DF,  p-value: 0.03162
```

The slope is positive (4.820 ± 2.019) and significant (p-value is 0.0316). We can plot the relationship on a scatterplot, with the regression line in red:

```
plot(litter.size~log.body.size_g, data=dataset)
abline(model1, col="Red")
```

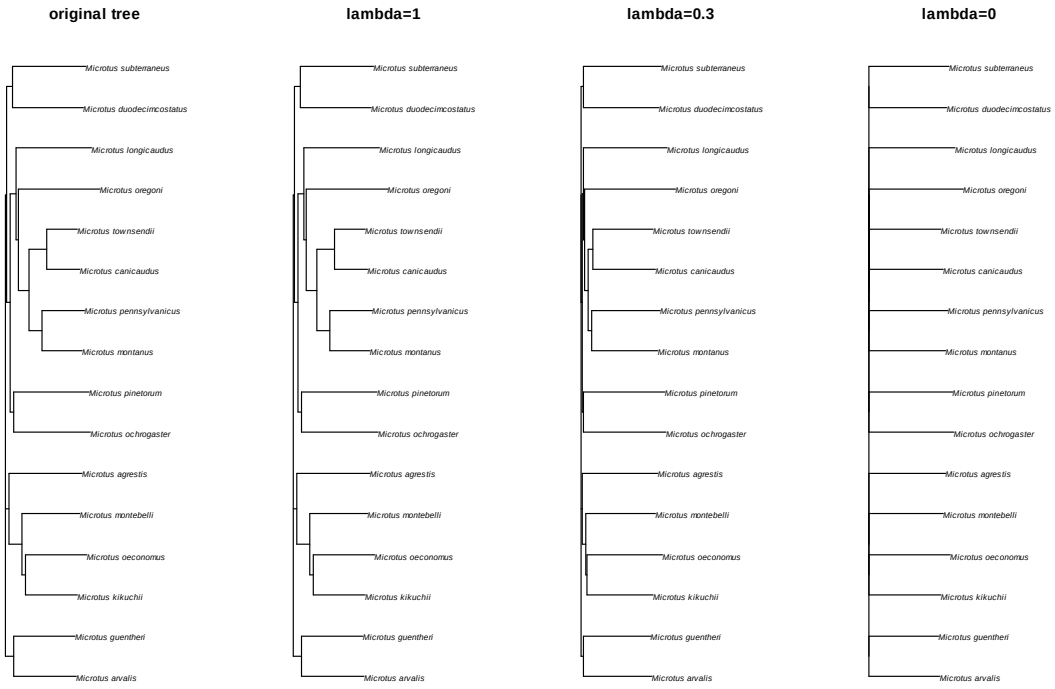


However, we don't know if this positive relationship between litter size and body size is independent of phylogenetic history. For example, a large rodent with a large litter may be the common ancestor to a number of species. If they are constrained from evolving different body sizes or litters, then the relationship is spurious–an artifact driven by ancestry, not selection. This may be less of a concern for today's practical since we are just considering one genus, *Microtus*, but if our range of taxa was larger, a spurious relationship may be a significant concern. But how can we tell if phylogeny is important for this relationship? We can calculate measures of phylogenetic signal on the traits themselves, and on the relationship between the traits.

1) Calculate measures of phylogenetic signal
We will now calculate measures of phylogenetic signal (λ and *K*) using **geiger** and/or **picante.**

Phylogenetic signal with Pagel's λ (lambda)
One method for testing phylogenetic signal is Pagel's lambda. Lambda is a tree transformation that shrinks the internal branches relative to the tip branches, making the tree more and more like a complete polytomy. If our estimated lambda = 0, then the traits are inferred to have no phylogenetic signal (i.e., traits evolve independent of phylogeny); lambda = 1 corresponds to a Brownian motion model (i.e., random genetic drift); 0 < lambda < 1 is in between. First let's look at what lambda does:

```
tree0 <- rescale(tree, model = "lambda", 0)
tree1 <- rescale(tree, model = "lambda", 1)
tree.3 <- rescale(tree, model = "lambda", 0.3)

par(mfcol = c(1, 4))
plot(tree)
title("original tree")
plot(tree1) #complete brownian motion
title("lambda=1")
plot(tree.3) #some weak signal
title("lambda=0.3")
plot(tree0) #star phylogeny (no phylo signal)
title("lambda=0")
```



We'll use phylosig from **phytools** to see if lambda is significantly different from 0. The method phylosig can also be used to estimate Blomberg's K (`method = "K"`), but more on that in the next section. Setting test = TRUE states we are testing the null hypothesis of no phylogenetic signal. A significant p-value tells you that there is significant phylogenetic signal – saying that close relatives are more similar than random pairs of species.

```
logbodysize <- dataset$log.body.size_g
names(logbodysize) <- rownames(dataset)
phylosig(tree, logbodysize, method = "lambda", test = TRUE)
```

Our output should be:

```
$lambda
[1] 6.610696e-05

$logL
[1] 8.822207

$logL0
[1] 8.822225

$P
[1] 1
```

With a lambda estimate close to zero and a p-value of 1, we can see there is essentially NO phylogenetic signal in body size among our 16 rodent species.

We can also compute lambda using the fitDiscrete or fitContinuous functions of **geiger**, depending on whether your data is discrete or continuous. To generate the maximum likelihood estimate of lambda for logbodysize:

```
lambda.bodysize<-fitContinuous(phy = tree, dat = logbodysize, model
= "lambda")
```

We can look at the output by typing `lambda.bodysize`. The output should look like this:

```
GEIGER-fitted comparative model of continuous data
 fitted 'lambda' model parameters:
        lambda = 0.000000
        sigsq = 0.286532
        z0 = 1.529960

 model summary:
        log-likelihood = 8.767121
        AIC = -11.534242
        AICc = -9.534242
        free parameters = 3

Convergence diagnostics:
        optimization iterations = 100
        failed iterations = 0
        frequency of best fit = 0.73

 object summary:
        'lik' -- likelihood function
        'bnd' -- bounds for likelihood search
        'res' -- optimization iteration summary
        'opt' -- maximum likelihood parameter estimates
```

This reaffirms the estimate from phylosig, that for log body size, lambda = 0.

Using phylosig for litter size:

```
littersize <- dataset$litter.size
names(littersize) <- rownames(dataset)
phylosig(tree, littersize, method = "lambda", test = T)
```

The output should be this:

```
$lambda
[1] 0.9999339

$logL
[1] -26.20789

$logL0
[1] -26.49946

$P
[1] 0.4450859
```

Hmm, so our lambda is very close to 1, but the p-value is not significant (we only have 16 species btw). Let's see what **geiger** says:

```
lambda.littersize<-fitContinuous(phy = tree, dat = littersize, model
= "lambda")
```

Output is as follows:

```
GEIGER-fitted comparative model of continuous data
 fitted 'lambda' model parameters:
        lambda = 1.000000
        sigsq = 26.163607
        z0 = 4.443859

 model summary:
        log-likelihood = -26.401584
        AIC = 58.803168
        AICc = 60.803168
        free parameters = 3

Convergence diagnostics:
        optimization iterations = 100
        failed iterations = 0
        frequency of best fit = 0.26

 object summary:
        'lik' -- likelihood function
        'bnd' -- bounds for likelihood search
        'res' -- optimization iteration summary
        'opt' -- maximum likelihood parameter estimates
```

So it appears prudent to consider that litter size may have phylogenetic signal when conducting our analyses.

Let's try our discrete variable (which I forced to have strong phylogenetic signal). The method phylosig only accepts continuous data, so we will use fitDiscrete using an equal rates (ER) model for belly color:

```
bellycol <- dataset$white.belly
names(bellycol) <- rownames(dataset)

lambda.bellycol<-fitDiscrete(phy = tree, dat = bellycol, model =
"ER", transform = "lambda")
```

We will get many warnings, but eventually there is this output after typing `lambda.bellycol`:

```
GEIGER-fitted comparative model of discrete data
 fitted Q matrix:
             black     brown     white
   black -2.622372  1.311186  1.311186
   brown  1.311186 -2.622372  1.311186
   white  1.311186  1.311186 -2.622372

 fitted 'lambda' model parameter:
       lambda = 1.000000

 model summary:
       log-likelihood = -11.883745
       AIC = 27.767489
       AICc = 28.690566
       free parameters = 2

Convergence diagnostics:
       optimization iterations = 100
       failed iterations = 40
       frequency of best fit = NA

 object summary:
       'lik' -- likelihood function
       'bnd' -- bounds for likelihood search
       'res' -- optimization iteration summary
       'opt' -- maximum likelihood parameter estimates
```

Phylogenetic signal with Bloomberg's K

Bloomberg's K (Blomberg et al. 2003) determines if closely related species resemble each other more or less than expected under Brownian motion models of evolution (random walk). Values of K range from 0 to infinity, with K=1 indicating Brownian motion evolution. K > 1 indicates that species are more similar than expected under random drift, and K < 1 indicates species are less similar (more divergent) than expected under random drift. Essentially, larger K indicates stronger phylogenetic signal.

We can quickly test our continuous traits (logbodysize, littersize) using phylosig:

```
phylosig(tree, logbodysize, method = "K", test = T)
```

Output:

```
$K
[1] 0.8799937

$P
[1] 0.513
```

```
      phylosig(tree, littersize, method = "K", test = T)
```

Output:

```
$K
[1] 0.9800898

$P
[1] 0.234
```

Both p-values are non-significant and indicate that there is no support of significant phylogenetic signal in our two traits (although litter size is close-ish).

2) Fit different models of trait evolution

If we expect a particular model of evolution for our traits, we can specify those models and then compare their AICc values with other models of evolution. Let's try Brownian motion (BM) model compared with Ornstein-Uhlenbeck (OU). BM expects a random walk, whereas OU expects a random walk around a central tendency, expected under stabilizing selection. First we fit the two models:

```
      bm.littersize <- fitContinuous(phy=tree, dat=littersize, model =
      "BM")
      ou.littersize <- fitContinuous(phy=tree, dat=littersize, model =
      "OU")
```

Then we can compare the two models' AICc values to find the lower value, indicating more support.

```
      bm.littersize$opt$aicc
[1] 57.72695
      ou.littersize$opt$aicc
[1] 60.76879
```

Our BM model has slightly better support than the OU model.

3) Perform independent contrasts analyses using caper

We will use the method of independent contrasts proposed by Felsenstein (1985) for estimating the evolutionary correlation between characters. Remember, contrasts are the differences between the trait values between species (and nodes). We'll use the **caper** package, which stands for Comparative Analyses of Phylogenetics and Evolution in R. First we need to combine the phylogeny and data into one object:

```
      rodent<- comparative.data(tree, dataset, names.col=Species,
      vcv=TRUE, na.omit=FALSE, warn.dropped=TRUE)
```

(NOTE: vcv=TRUE stores a variance-covariance matrix of our tree. We will use this for PGLS later)

We then use the function crunch to fit the independent contrast regression model:

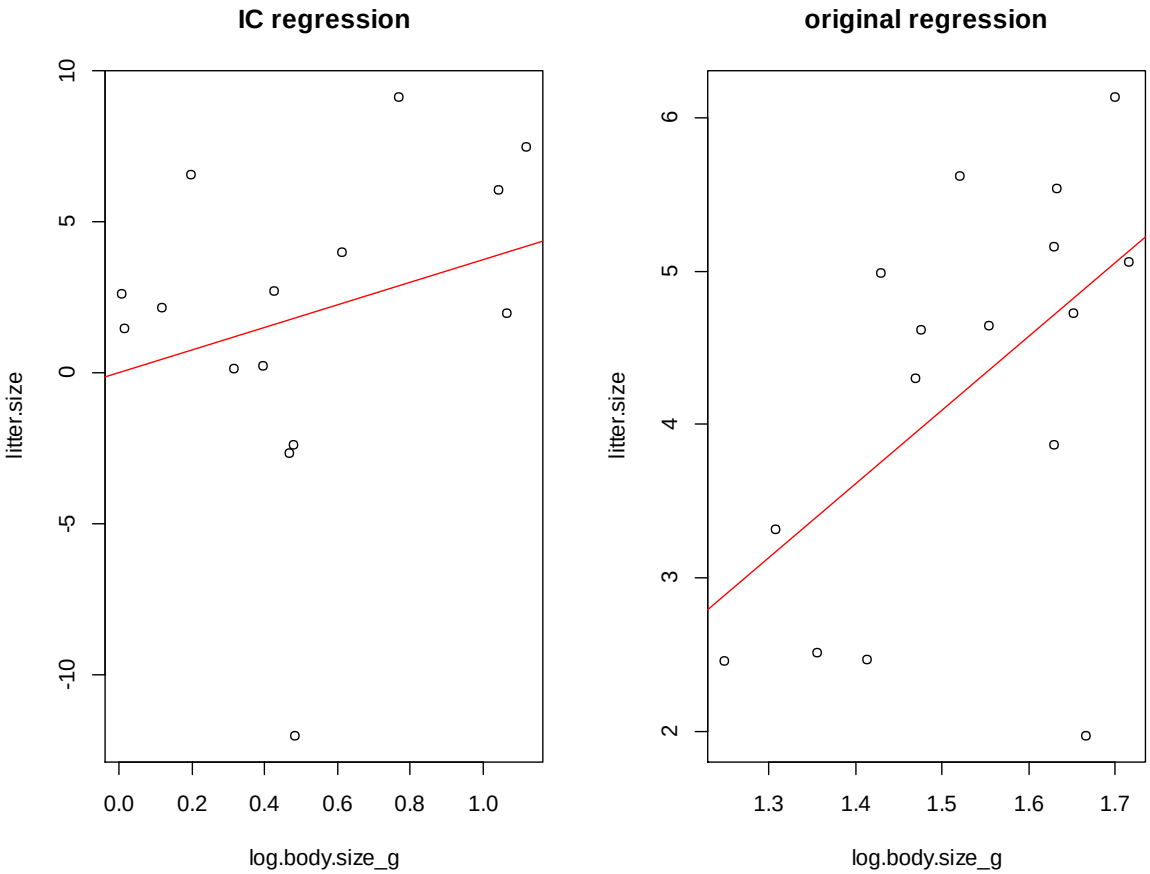
```
      rodent.ic<- crunch(litter.size~log.body.size_g, data=rodent)
      summary(rodent.ic)
```

And the output should look like this:

```
Call:
lm(litter.size ~ log.body.size_g - 1, data = contrData)

Residuals:
    Min      1Q  Median      3Q     Max
-13.840  -1.629   1.417   2.365   6.277

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
log.body.size_g    3.738      2.080   1.798   0.0938 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.929 on 14 degrees of freedom
Multiple R-squared:  0.1875,      Adjusted R-squared:  0.1295
F-statistic: 3.231 on 1 and 14 DF, p-value: 0.09384
```

We can visualize the contrasts in a scatterplot and compare with our original linear regression plot:

```
rodentcontr<-caic.table(rodent.ic)

par(mfrow=c(1,2))
plot(litter.size ~ log.body.size_g, data=rodentcontr)
abline(rodent.ic, col="red")
title("IC regression")
plot(litter.size ~ log.body.size_g, data=dataset)
abline(model1, col="red")
title("original regression")
```



4) Perform PGLS analyses using caper

Phylogenetic generalized least squares (PGLS) models are more flexible than independent contrast (IC) methods. The model of trait evolution can depart from a strict BM (lambda or K = 1) model. Different scaling parameters can be incorporated into the analysis (see PGLS in **caper** help for options), which can improve the fit of the model and thus improve the estimation of trait correlation. Another plus is that the intercept of the regression does not have to be forced to zero.

We will use our caper object that combined phylogeny and data:

```
rodent<- comparative.data(tree, dataset, names.col=Species,
vcv=TRUE, na.omit=FALSE, warn.dropped=TRUE)
```

We will now fit a model using the function `pgls` that uses the maximum likelihood estimate of lambda:

```
rodent.pgls<-pgls(litter.size~log.body.size_g, data=rodent,
lambda="ML")
summary(rodent.pgls)
```

Checking our output:

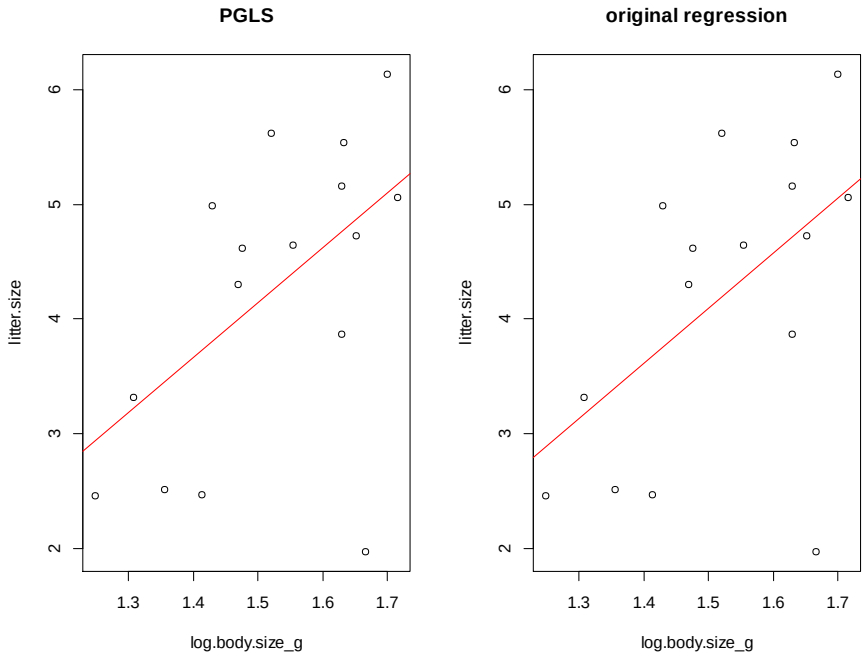
```
Call:
pgls(formula = litter.size ~ log.body.size_g, data = rodent,
    lambda = "ML")

Residuals:
     Min       1Q   Median       3Q      Max
-11.5780  -3.8210  -0.8256   1.2682   5.2497

Branch length transformations:

kappa  [Fix]  : 1.000
lambda [ ML]  : 0.000
   lower bound : 0.000, p = 1
   upper bound : 1.000, p = 0.22369
   95.0% CI   : (NA, NA)
delta  [Fix]  : 1.000

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      -3.0770     3.1829 -0.9667  0.35009
log.body.size_g   4.8132     2.0718  2.3231  0.03575 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.436 on 14 degrees of freedom
Multiple R-squared: 0.2782,    Adjusted R-squared: 0.2267
F-statistic: 5.397 on 1 and 14 DF,  p-value: 0.03575
```

So the maximum likelihood estimate of lambda was set to (0.000), and p-values from the likelihood ratio tests indicate that the ML estimate was not significantly different from 0 (p=1), nor from 1 (p=0.22369). Overall, it looks like litter size is indeed correlated with body size.

Plotting the results we can see that phylogenetic signal was not included in the pgls model:

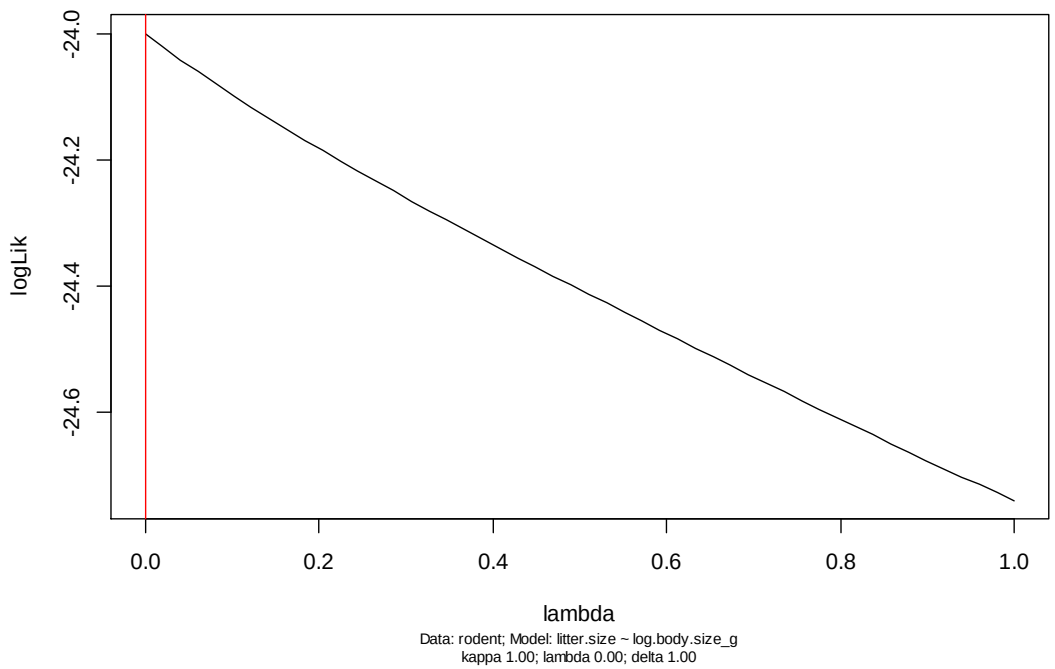
```
par(mfrow=c(1,2))
plot(litter.size ~ log.body.size_g, data=dataset)
abline(rodent.pgls, col="red")
```

```
title("PGLS")
plot(litter.size ~ log.body.size_g, data=dataset)
abline(model1, col="red")
title("original regression")
```



We can also plot the likelihood profile of our estimate of lambda:

```
par(mar=c(10,5,4,5)) #changing plot margins c(bottom,left,top,right)
lambda.prof <- pgls.profile(rodent.pgls, 'lambda')
plot(lambda.prof)
```



So we can clearly see that the most likely value for lambda is zero. But with other datasets you could expect to see a curve peaking with lambda between 0 and 1.

5) Solve the mystery

Finally, a chance to use what you've learned to find answers to an evolutionary question: **What is the evolutionary ancestral state for singing in voles?** Ok, the data are fake and for illustrative purposes only, but let's pretend that ~half of our 16 Microtus species can sing (or think they can), and the other half cannot. Did the ancestor to all these species sing? Or did it evolve independently multiple times?

Your challenge is to produce a compelling figure to illustrate your conclusions. The data are in the vole dataset (rodent.csv) under the column labeled "sing". You should first make sure R recognizes the variable as categorical by defining the data as a factor.

```
dataset$sing<-factor(dataset$sing)
str(dataset$sing)
```

The variable should now be recognized as factor with two levels "0" and "1".

HINTS: make sure your tree is fully resolved using `is.binary.tree()` and that it is rooted (**outgroup=2**) using `is.rooted()`. Make sure your species names and tree tips match EXACTLY, and that you have no missing data (NAs). You can do this by removing all rows with missing data using the command `complete.cases()` (but our dataset is complete; this is just for future reference).

```
complete<-complete.cases(dataset)
id<-which(complete == TRUE)
dataset.complete<-dataset[id,]
```

Have fun! Let me know if you have any questions and when you are finished!