

Data Analysis in Paleontology Using *R*

Session 5
2 Feb 2006

*Gene Hunt
Dept. of Paleobiology
NMNH, SI*

Phylogenetic Comparative Methods

Character evolution in the context of a
phylogenetic tree:

1. Estimate ancestral character states
2. Test relationships among variables,
accounting for dependence due to phylogeny

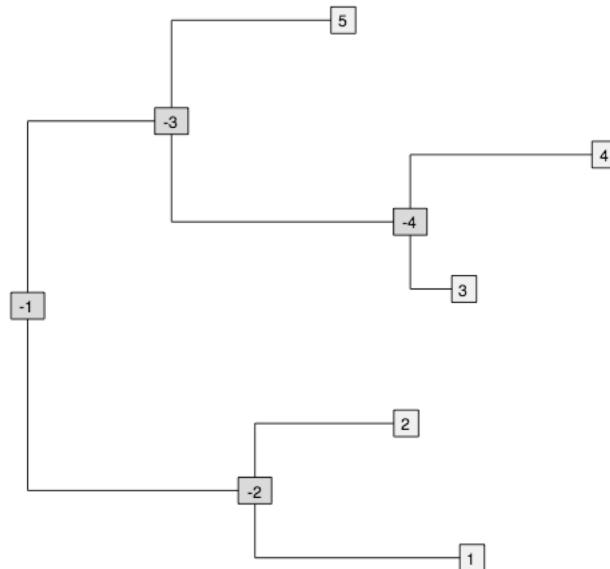
Plotting a tree

```
plot(tree)          # see ?plot.phylo for options
```

Making tree structure more obvious:

```
plot(tree, show.tip.label=F) # no taxon names  
nodelabels(text=-1:-4)     # shows node numbers  
tiplabels(text=1:5)        # shows tip numbers
```

Tree with nodes and tips labelled



Display character states

```
size <- c(1,1,2,3,5) # a continuous character
larv <- c(0,0,1,1,1) # a discrete character
```

```
plot(tree, show.tip.label=F)
tiplabels(text=size) # add text to tips
```

```
plot(tree, show.tip.label=F)
tiplabels(pch=19, cex=size) # symbols size based
# on continuous char.
```

```
plot(tree, show.tip.label=F)
tiplabels(pch=larv+21, bg=larv+1) # pch and bg based
# on discrete char.
```

Ancestral Character States

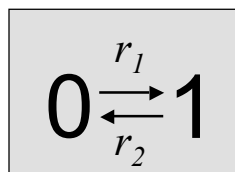
1. Discrete traits

Parsimony

- Minimize # char. state changes

Maximum Likelihood

- Maximize probability of the observed data
- Requires a model of character state change



A two-rate model

Ancestral Character States

2. Continuous traits

Squared Change Parsimony

- Minimize sum of squared char. state changes

Maximum Likelihood / GLS

- Based on Brownian motion model (= unbiased random walk)
- Prob (trait increase) = Prob (trait decrease)
- Very simple: 1 parameter (related to typical step size)

Ancestral states: `ace()`

```
ace (x, phy, type=, method=)
```

Arguments

x vector of trait values
phy phylog (tree) object
type “discrete” or “continuous”
method “ML”, “pic”, “GLS”

Warning: names of x must match exactly `phy$tip.label`.
If not, `ace()` assumes they are in the same order.
Assign names to a vector with the function `names()`

Ancestral states: ace()

1. Type="discrete"

The model argument can be used to specify any uniform model

At present (ape v. 1.8), this does not seem to work well, unless there are rather few changes on the tree.

2. Type="continuous"

method="ML" is the most likely used, and works well

method="GLS" should = ML, currently does not

method="pic" based on independent contrasts

Example: ace()

Estimate ancestral states

```
aa <- ace(size, tree) # default is ML(cont)
```

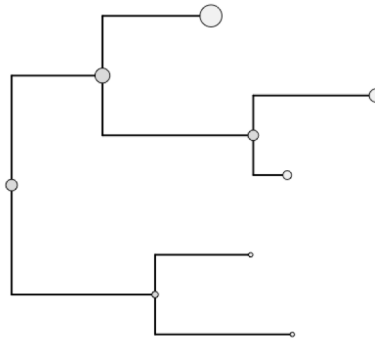
ace returns a list of:

loglik	Log-likelihood
ace	Vector of ancestral state estimates (nodes -1,-2, ...)
sigma2	Rate estimate (with SE)
CI95	95% CI on ancestral states

Visualizing ace() results

Plot tree with scales node & tip symbols

```
plot(tree, show.tip.label=F)  
tiplabels(pch=21, cex=size)  
nodelabels(pch=21, cex=aa$ace)
```



Relationships among variables

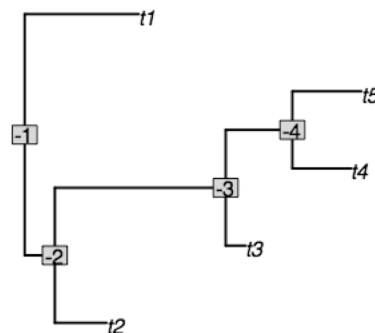
1. Phylogenetically independent contrasts

For a tree with N taxa, constructs $N-1$ “contrasts” that span each branch only once (Felsenstein, 1985)

Contrasts

- t4 to t5
- 4 to t3
- 3 to t2
- 2 to t1

Function `pic(x, phy)`



Relationships among variables

2. Generalized least squares (GLS)

Uses generalized linear models to handle non-independence among terminal taxa

Strength: can use regular statistical machinery of linear models to fit different models and choose among models

```
gls(y ~ x1 + x2, cor=corBrownian(1, tree))
```



Model
formula



Model dependence
assuming Brownian motion

Ape miscellanea

Analysis

compar.gee	Comparative methods using GEE's
compar.lynch	Lynch's (1991) comparative method
dist.phylo	Pairwise phylo distances between taxa
mst, nj	Tree making algorithms
Moran.I	Phylogenetic autocorrelation
evolve.phylo	Simulate Brownian evolution

Tree
functions

root	Root tree with taxon
rotate	Rotate branches on tree
consensus	Calculates consensus tree
di2mult	Collapses small branches
mult2di	Resolves multichotomies

Exercise 13. Comparative Methods

1. Create a random tree with 50 terminal taxa using `r tree()`. Type `?evolve.phylo` to see the help information for this function, which simulates the evolution of continuous characters on a given tree topology. Use `evolve.phylo` to simulate the evolution of two characters on this tree (you can do this by setting the `value` argument to a vector of two ancestral states). Look at the output created: notice that a tree is returned, with additional components for the trait values at the tips (`tip.character`) and the internal nodes (`node.character`).
2. Use the function `cor.test()` to test for a significant correlation between the tip values of the simulated variable. You'll need to extract the vectors out for each trait separately for `cor.test()`. Did you get a significant p-value? In fact, about 35% of tests here end up being significant ($p < 0.05$), even though the traits were simulated independently. These tests do so poorly because the species are not truly independent, and thus the sample size is overestimated.
3. Compute phylogenetically independent contrasts for the two traits using `pic()`, performing the operation on each trait separately. Test for a correlation between the contrasts. This test is based on truly independent observations, and thus should perform better.

Re-sampling methods

Goal:

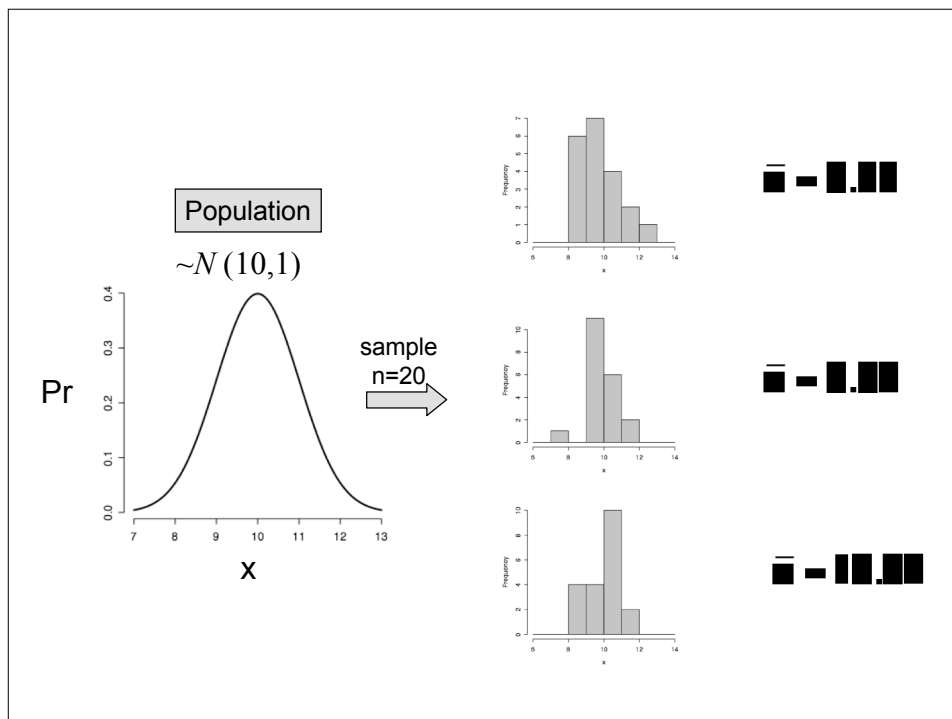
We are measuring some statistic in a data set, and wish to know how reliable we have estimated it (i.e., we want a standard error / confidence interval).

Usually, we rely on theory for this, but sometimes:

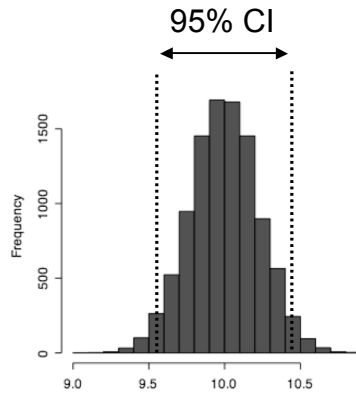
- (1) No one has worked out the theory for the statistic of interest
- (2) Our data violate the assumptions under which existing theory applies

Bootstrapping

- (1) Given X , a dataset with N observations, generate a bootstrap sample of X by randomly sampling **with replacement** N items from X
- (2) For the new bootstrap sample, calculate the statistic of interest
- (3) Repeat steps 1 & 2 lots of times to generate a distribution of statistic estimates. Standard errors and confidence intervals can be calculated from this distribution of bootstrap estimates.



The Sampling Distribution

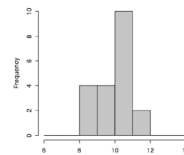
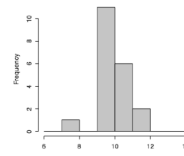
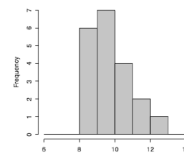
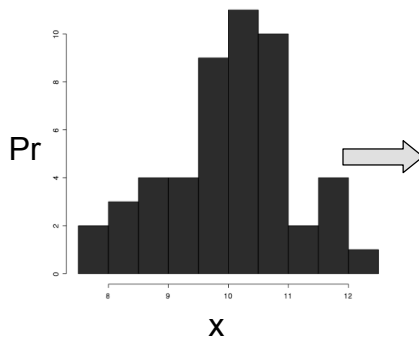


The standard deviation of this sampling distribution is called the **standard error** of the statistic

Confidence intervals can be computed from the percentiles of this distribution

Bootstrapping

Use the **observed** data to approximate the **population**



Bootstrapping is not a cure-all

1. If N is low, the data may not approximate the true underlying distribution very well
2. In some situations, the CI's and SE's are not very accurate (test by simulation)
3. Bootstrapping is nonparametric, but not assumption-free: observations are assumed to be *iid*

independent and identically distributed

Resampling in R

```
sample(x, size, replace = FALSE)
```

↑ ↑ ↑
Vector to be How many Whether or not to
resampled individuals to sample with
 resample replacement

By default, size = length(x)

```
x <- 1:10  
sample(x)            # permutes elements of x  
sample(x, replace=T)    # bootstrap sample of x  
sample(x, size=5, replace=F)    # subsample x
```

An example: sample mean

```
x                # data to analyze
x.mu <- mean(x)  # compute observed mean

boot.mu <- array(dim=100) # 100 bootstrap reps
for (i in 1:100)
{
  xboot <- sample(x, replace=TRUE) # create a boot sample
  boot.mu[i] <- mean(xboot)        # record mean
}
```

boot.mu approximates the sampling
distribution of the sample mean

What can we do with boot.mu?

```
mu.se <- sd(boot.mu) # standard error of mean
```

What about confidence intervals?

Can assume normality: 95% CI $\approx \mu \pm 2 \cdot \text{mu.se}$

Can sort boot.mu, pick out lower 2.5% and upper 2.5% cutoff points, or use `quantile()` function

```
mu.ci95 <- quantile(boot.mu, c(0.025, 0.975))
```

What about bootstrapping matrices?

Same idea, but use `sample()` on *indices*.

If X is a matrix with 20 rows:

```
wh <- sample(1:20, replace=T)
Xboot <- X[wh,] # bootstrap rows of X
```

Sampling ecological communities

Using `sample()` on community counts will not work, unless entries in the data represent **individuals**, rather than species **counts**.

Essentially, communities are multinomial distributions: there are a discrete number of possible outcomes (=taxa), with a probability associated with each outcome (=relative abundance).

Can use `rmultinom()` to simulate communities:

```
library(vegan)
data(BCI)
y<- BCI[1,]
```

`rmultinom(5, 100, prob=y)`

of samples to create N per sample Base relative abundances on 1st sample of BCI

Note: resulting matrix has samples in *columns*

Other Resampling Methods

1. Permutation Tests

Tests involving grouping; sampling distribution is created by permuting group membership

example: ANOSIM

2. Parametric Bootstrapping

Instead of approximating the underlying population with the real data (nonparametric bootstrapping), it is approximated by a generating model, which is sampled repeatedly to create bootstrap samples.

Final advice...

- You'll make a lot of mistakes in the beginning with syntax
- Use the email list, especially before starting a project
- Use the examples here as a guide
- If there seems like there should be an easy way to do something, there probably is

Exercise 14. Re-sampling Methods

1. Type the code for bootstrapping the sample mean into a source file. Generate sample data using `rnorm()` and bootstrap the sample mean for this random dataset. Compare the resulting standard error to the expectation from theory, which predicts the standard error of the sample mean to be equal to the standard deviation of the original sample, divided by the square root of N .
2. This one is more of an example than an exercise. The following function generates 95% bootstrap confidence intervals on RMA slopes; it uses the `rma()` function we created in session 4. This provides a good example on how to bootstrap indices when the data of interest do not consist of a single vector. To figure out what is going on, you may want to try individual lines out using the interactive R prompt.

```
rma.boot<- function(x,y, nreps=1000)
# function to compute bootstrap 95% CI on RMA slope
# by default, uses 1000 bootstrap samples
{
  nn<- length(x) # sample size
  boot.b1<- array(dim=nreps) # vector to hold slopes in boot samples
  for (i in 1:nreps)
  {
    wh<- sample(1:nn, replace=TRUE) # bootstrap sample of the indices
    xboot<- x[wh]
    yboot<- y[wh] # creates bootstrap sample of x & y
    w<- rma(xboot, yboot) # RMA of bootstrap sample
    boot.b1[i]<- w[2] # save resulting slope (w[1] is the intercept)
  }

  # now need to compute 2.5% and 97.5% quantiles of boot.b1
  ci95<- quantile(boot.b1, c(0.025, 0.975))
  return (ci95)
}
```

Exercise 14. Re-sampling Methods (cont's)

3. The first sample (row) of the BCI data has 448 individuals. Is the estimation of Shannon diversity very dependent on sample size for this community? To investigate this, simulate sampling from this community at two levels: $N=50$, and $N=300$. For each of these, generate a matrix of simulated communities called X_{50} and X_{300} ; generate 100 simulated communities for each sampling level. Use the `diversity()` function in `vegan` to calculate Shannon diversity for all simulated samples in both matrices (note: set `MARGIN=2` as an argument to `diversity` because the samples are in columns, instead of the usual rows). Plot the results using `boxplot()`, and do a `t.test()` test for significant differences. Does observed Shannon appear to be influenced by sampling?

Answers to Exercises [5]

Exercise 13. Comparative Methods

1.

```
tt <- rtree(50)
w <- evolve.phylo(tt, value=c(0,0), var=1) # exact values for value
# and var do not matter
w$tip.character is a matrix, with one column for each trait
```
2.

```
cor.test(w$tip.character[,1], w$tip.character[,2])
```
3.

```
pic.char1 <- pic(w$tip.character[,1], tt)
pic.char2 <- pic(w$tip.character[,2], tt)
cor.test(pic.char1, pic.char2) # accounts for phylogenetic dependence
```

Exercise 14. Re-sampling Methods

1.

```
x <- rnorm(100);
```

 If the bootstrap procedure described a few slides ago is followed, the `sd(boot.mu)` should approximately equal `sd(x)/sqrt(100)`.
2. Self explanatory.
3.

```
X50 <- rmultinom(100, 50, prob=BCI[1,])
X300 <- rmultinom(100, 300, prob=BCI[1,])
Sh50<- diversity(X50, MARGIN=2)
Sh300<- diversity(X300, MARGIN=2)
boxplot(Sh50, Sh300)
t.test(Sh50, Sh300)
```

 Shannon diversity is significantly higher in the better-sampled communities.