Exercices and Solutions to:

Analysis of Phylogenetics and Evolution With R (Second Edition)

Chapter 2

1. Start R and print the current working directory.

   ```
   > getwd()
   ```

   Suppose you want to read some data in three different files located in three different directories on your computer: describe two ways to do this.

   The first way is to change the working directory before and after reading each file:

   ```
   > od <- setwd("DATA1/")
   > X <- read.table("....
   > setwd(od)
   ```

   The function `setwd()` returns the value of the current working directory before changing it, so that it is easy to return to it after reading the data.

   The second way is to prefix the file names with their respective directory names:

   ```
   > X <- read.table("DATA1/....
   ```

   One way or the other may be prefered depending on the situation. If the directories have the same structure with the same file names, then the first solution is more convenient. The second solution is simpler if there is only one file to read in each directory.

2. Create a matrix with three columns and 1000 rows where each column contains a random variable that follows a Poisson distribution with rates 1, 5, and 10, respectively (see `?Poisson` for how to generate random Poisson values).

   We see here three solutions to the first part of this question. We first define the parameters of the simulation with a vector `lambda` containing the values of the Poisson rate, the sample size in `n`, and we extract the number of values in `lambda`. The first solution is to create the matrix (`X`) and fill its columns with a `for` loop:

   ```
   > lambda <- c(1, 5, 10)
   > n <- 1000
   > p <- length(lambda)
   > X <- matrix(NA, n, p)
   > for (i in 1:p) X[, i] <- rpois(n, lambda[i])
   ```

This is probably the simplest and most efficient solution. The second solution is similar to the first one except for the last two commands which are:

```
> X <- NULL
> for (i in 1:p) X <- cbind(X, rpois(n, lambda[i]))
```

This is probably only slightly less efficient than the first solution. The third solution is simpler (and does not need to extract `p`:

```
> lambda <- c(1, 5, 10)
> n <- 1000
> X <- sapply(lambda, function(l) rpois(n, l))
```

Find two ways to compute the means of each column of this matrix.

The means can be calculated directly with:

```
> apply(X, 2, mean)
[1] 0.965 4.900 9.936
```

Or with a `for` loop:

```
> means <- numeric(p)
> for (i in 1:p) means[i] <- mean(X[, i])
> means
[1] 0.965 4.900 9.936
```

3. Create a vector of 10 random normal values using the three following methods.

   (a) Create and concatenate successively the 10 random values with `c`.

   (b) Create a numeric vector of length 10 and change its values successively.

   (c) Use the most direct method.

   Compare the timings of these three methods (see `?system.time`) and explain the differences.

   The code below gives the three solutions after setting the value of `n` to 10:

```
> n <- 10
> ## (a)
> x <- NULL
> system.time(for (i in 1:n) x <- c(x, rnorm(1)))
   user  system elapsed
  0.002   0.000   0.002
> ## (b)
> x <- numeric(n)
> system.time(for (i in 1:n) x[i] <- rnorm(1))
   user  system elapsed
  0.002   0.000   0.002
```

```
> ## (c)
> system.time(x <- rnorm(n))
   user  system elapsed
      0       0       0
```

All three methods give similar results. The differences of timings are very small for this value of `n`.

Repeat this exercise with 10,000 values.

We only need to modify the first line of the previous code:

```
> n <- 1e4
> ## (a)
> x <- NULL
> system.time(for (i in 1:n) x <- c(x, rnorm(1)))
   user  system elapsed
  0.165   0.007   0.173
> ## (b)
> x <- numeric(n)
> system.time(for (i in 1:n) x[i] <- rnorm(1))
   user  system elapsed
  0.021   0.000   0.022
> ## (c)
> system.time(x <- rnorm(n))
   user  system elapsed
  0.001   0.000   0.001
```

4. Create the following text file:

```
Mus_musculus                   10
Homo_sapiens                70000
Balaenoptera_musculus   120000000
```

(a) Read this file with `read.table` using the default options. Look at the structure of the data frame and explain what happened. What option should have been used?

At the time of writing the book, `read.table()` had the option `header = TRUE` by default, so that the first row of the file was taken as names of the variables. With a recent version of R (4.2.1), the help page of this function explains the new behavior of this option:

> If missing, the value is determined from the file format: `header` is set to `TRUE` if and only if the first row contains one fewer field than the number of columns.

(b) From this file, create a data structure with the numeric values that you could then index with the species names, for example,

```
> x["Mus_musculus"]
[1] 10
```

Find two ways to do this, and explain the differences in the final result.

The first way uses a copy of the second column into the vector `x`, and setting its names with the first column:

```
> X <- read.table("data_ch2.dat")
> x <- X$V2
> names(x) <- X$V1
> x["Mus_musculus"]
Mus_musculus
          10
```

The second way uses the function `setNames` and we can check that the result is identical to the first one:

```
> y <- setNames(X$V2, X$V1)
> identical(x, y)
[1] TRUE
```

5. Create these two vectors (source: [13]):

```
Archaea <- c("Crenarchaea", "Euryarchaea")
Bacteria <- c("Cyanobacteria", "Spirochaetes",
              "Acidobacteria")
```

   (a) Create a list named `TreeOfLife` so that we can do `TreeOfLife$Archaea` to print the corresponding group.

```
> Archaea <- c("Crenarchaea", "Euryarchaea")
> Bacteria <- c("Cyanobacteria", "Spirochaetes",
+               "Acidobacteria")
> TreeOfLife <- list(Archaea = Archaea, Bacteria = Bacteria)
> TreeOfLife$Archaea
[1] "Crenarchaea" "Euryarchaea"
```

   (b) Update `TreeOfLife` by adding the following vector:

```
Eukaryotes <- c("Alveolates", "Cercozoa", "Plants",
                "Opisthokonts")
```

   It should appear at the same level as `Archaea` and `Bacteria`.

```
> Eukaryotes <- c("Alveolates", "Cercozoa", "Plants",
+                 "Opisthokonts")
> TreeOfLife$Eukaryotes <- Eukaryotes
```

(c) Update Archaea by adding "Actinobacteria".
Actually, Actinobacteria is part of Bacteria, so the correct solution is:

```
> TreeOfLife$Bacteria <- c(TreeOfLife$Bacteria, "Actinobacteria")
> TreeOfLife
$Archaea
[1] "Crenarchaea" "Euryarchaea"

$Bacteria
[1] "Cyanobacteria"  "Spirochaetes"   "Acidobacteria"
[4] "Actinobacteria"

$Eukaryotes
[1] "Alveolates"   "Cercozoa"     "Plants"
[4] "Opisthokonts"
```

(d) Print all the lowest-level taxa.

```
> unname(unlist(TreeOfLife))
 [1] "Crenarchaea"    "Euryarchaea"    "Cyanobacteria"
 [4] "Spirochaetes"   "Acidobacteria"  "Actinobacteria"
 [7] "Alveolates"     "Cercozoa"       "Plants"
[10] "Opisthokonts"
```

Chapter 3

1. Create a random tree with 10 tips.

```
> library(ape)
> tr <- rtree(10)
```

(a) Extract the branch lengths, and store them in a vector.

```
> x <- tr$edge.length
```

(b) Delete the branch lengths, and plot the tree.

```
> tr$edge.length <- NULL
> plot(tr)
```

(c) Give new, random branch lengths from a uniform distribution $U[0, 10]$. Do this in a way that works for any number of tips.

```
> tr$edge.length <- runif(Ntip(tr))
```

(d) Restore the original branch lengths of the tree.

```
> tr$edge.length <- x
```

2. Create a random tree with 5 tips, print it, and plot it.

```
> tr <- rtree(5)
> tr

Phylogenetic tree with 5 tips and 4 internal nodes.

Tip labels:
  t2, t3, t4, t5, t1

Rooted; includes branch lengths.
> plot(tr)
```

Find the way to delete the class of this object, and print it again.

```
> class(tr) <- NULL
> tr
$edge
     [,1] [,2]
[1,]    6    7
[2,]    7    8
[3,]    8    1
[4,]    8    2
[5,]    7    3
[6,]    6    9
[7,]    9    4
[8,]    9    5

$tip.label
[1] "t5" "t1" "t3" "t4" "t2"

$Nnode
[1] 4

$edge.length
[1] 0.88005925 0.16759254 0.43179953 0.01514436 0.36705115
[6] 0.27348595 0.39078180 0.97974093

attr(,"order")
[1] "cladewise"
```

Try to plot it again: comment on what happens.

```
> plot(tr)
Error in xy.coords(x, y, xlabel, ylabel, log) :
  'x' is a list, but does not have components 'x' and 'y'
```

`plot` is a generic function, thus is `tr` has the class `"phylo"` the command `plot(tr)` calls the function `plot.phylo()` to draw a phylogenetic tree. If this class is deleted, then `plot.default` is called and this one cannot find the expected data in `tr`.

The same comment can be made about `print()`, but in that case `print.default()` can print the "unclassed" `tr`.

Find a way to force the plot of the tree as before.

The only solution is to restore the class of `tr`:

```
> class(tr) <- "phylo"
> plot(tr)
```

3. Generate three random trees with 10 tips. Write them in a file. Read this file in R. Print a summary of each tree. Write a small program that will do these operations for any number of trees (say `N`) and any number of tips (`n`).

```
> TR <- rmtree(3, 10)
> write.tree(TR, "TR.tre")
> TRb <- read.tree("TR.tre")
```

For printing a summary of each tree, we can do it with the following command:

```
sapply(TRb, summary)
```

But this prints a somehow lengthy summary for each tree. We may also replace `summary` by `print` (which uses the `print.phylo` method). Here, we write our own code to print only the number of tips of each tree:

```
> for (i in 1:length(TRb))
+     cat("Tree", i, ": Number of tips:", Ntip(TRb[i]), "\n")
Tree 1 : Number of tips: 10
Tree 2 : Number of tips: 10
Tree 3 : Number of tips: 10
```

This can be customized easily by adding more code in `cat()` (e.g., `Nnode(TRb[i])`).

The code can be easily transformed into a small program by replacing the first command (`TR <- rmtree(3, 10)`) by:

```
N <- 3
n <- 10
TR <- rmtree(N, n)
```

4. (a) Write a function that will read trees from the Pfam database, so that so we can use it with:

```
read.pfamtree(accnum, type = "full")
```

where `accnum` is the accession number of the family, and `type` is the type of the alignment (see p. 44).

(b) Extract the tree #1000 in Pfam. Make three copies of this tree, and give them branch lengths (i) all equal to one, (ii) so that the node heights are proportional to the number of species, and (iii) randomly extracted from a uniform distribution $U[0, 0.1]$.

At the time of writing the book, the solution would have been:

```
read.pfamtree <- function(accnum, type = "full")
{
    if (!type %in% c("meta", "seed", "full", "ncbi"))
        stop("wrong value for argument 'type'")
    a <- "http://pfam.sanger.ac.uk/family/tree/"
    b <- paste0("download?alnType=", type, "&acc=", accnum)
    ref <- paste0(a, b)
    read.tree(ref)
}
```

However, the PFAM database is expected to be decommissioned in January 2023 with no planed replacement.[1]

5. Extract the sequences of the cytochrome *b* gene with the accession numbers U15717–U15724 (source: [116]).

The sequences are obviously deposited on GenBank, so we prepare the accession numbers in a vector with `paste` (or `paste0`) and pass them to `read.GenBank`:

```
> accnum <- paste0("U157", 17:24)
> X <- read.GenBank(accnum)
```

(a) Print the species names of each sequence.

```
> names(X)
[1] "U15717" "U15718" "U15719" "U15720" "U15721" "U15722"
[7] "U15723" "U15724"
```

`read.GenBank()` has been improved since the book was published; the species names are now returned in a specific attribute and a description in a separate attribute (see `?read.GenBank` for details):

```
> attr(X, "species")
[1] "Ramphocelus_passerinii"
[2] "Ramphocelus_sanguinolentus"
[3] "Ramphocelus_icteronotus"
[4] "Ramphocelus_costaricensis"
[5] "Ramphocelus_nigrogularis"
[6] "Ramphocelus_costaricensis"
[7] "Ramphocelus_carbo"
[8] "Ramphocelus_bresilius"
```

[1]`http://pfam.xfam.org/`; accessed 2022-10-19.

(b) Print, with a single command, the length of each sequence.

```
> lengths(X)
U15717 U15718 U15719 U15720 U15721 U15722 U15723 U15724
  1045   1045   1045   1045   1045   1045   1045   1045
```

(c) Arrange the data in a matrix.

```
> X <- as.matrix(X)
> X
8 DNA sequences in binary format stored in a matrix.

All sequences of same length: 1045

Labels:
U15717
U15718
U15719
U15720
U15721
U15722
...

Base composition:
    a     c     g     t
0.267 0.351 0.134 0.247
(Total: 8.36 kb)
```

This would give an error if the sequences are not of the same length.

(d) Extract and store in three matrices the first, the second, and the third codon positions of all sequences. Compute their base frequencies. What do you conclude?

The simplest way is to create three matrices with:

```
> POS1 <- X[, c(TRUE, FALSE, FALSE)]
> POS2 <- X[, c(FALSE, TRUE, FALSE)]
> POS3 <- X[, c(FALSE, FALSE, TRUE)]
```

An alternative, "programmatic" way is:

```
> POS <- vector("list", 3)
> names(POS) <- paste("Position", 1:3)
> for (i in 1:3) {
+     s <- logical(3)
+     s[i] <- TRUE
+     POS[[i]] <- X[, s]
+ }
```

This makes easier to compute the base frequencies:

```
> sapply(POS, base.freq)
  Position 1 Position 2 Position 3
a 0.37464183  0.2331178  0.1936063
c 0.51038682  0.2956178  0.2471264
g 0.03044413  0.2413793  0.1311063
t 0.08452722  0.2298851  0.4281609
```

There is a very clear difference with respect to the position.

(e) Save the three matrices in three different files. Read these files, and concatenate the three sets of sequences.

We prefer to use the list `POS`:

```
> for (i in 1:3)
+    write.dna(POS[[i]], paste0(names(POS)[i], ".fas"), "fasta")
```

Read these files, and concatenate the three sets of sequences.

```
> POSb <- vector("list", 3)
> for (i in 1:3)
+   POSb[[i]] <- read.dna(paste0(names(POS)[i], ".fas"), "fasta")
> Xb <- do.call(cbind, POSb)
> Xb
8 DNA sequences in binary format stored in a matrix.

All sequences of same length: 1045

Labels:
U15717
U15718
U15719
U15720
U15721
U15722
...

Base composition:
    a     c     g     t
0.267 0.351 0.134 0.247
(Total: 8.36 kb)
```

`X` and `Xb` have the same data but their columns are ordered differently.

6. (a) Write a program that will extract single nucleotide polymorphism (SNP) from a sequence alignment. The output will include the position of the SNPs along the sequence and the observed bases (alleles). You will include an option to output the sequence of the constant sites.

(b) Write a second program that will transform the above alignment into an object of class "loci".

Chapter 4

1. Draw Fig. 4.11 using a color scale in place of the grey one. The figure should include a legend.

2. Plot the phylogeny of avian orders, and color the Proaves in blue. Repeat this but only for the terminal branches of this clade.

3. Suppose you have a factor representing a character state for each node and each tip of a tree. Find a way to associate a color with each branch depending on the state at both ends of the branch.

4. Consider the trees `trc` and `trk`. Do a comparison of branching times as suggested above. This will include a bivariate plot with a dotted line $x = y$. You will also indicate the node by their numbers on the plot.

5. Create a list of trees simulated using a Yule process with $\lambda = 0.1$ (see Chapter 7). Sort the trees in increasing order of number of tips. Create an animation to visualize sequentially the tree on top and its lineage-through time plot (Chapter 6) on bottom.

```
> library(animation)
> TR <- replicate(20, rbdtree(0.1, 0), simplify = FALSE)
> TR <- TR[order(sapply(TR, Ntip))]
> saveHTML(
+     for (i in 1:length(TR)) {
+         layout(matrix(1:2))
+         plot(TR[[i]], show.tip.label = FALSE)
+         ltt.plot(TR[[i]])
+ })
```

Chapter 5

1. (a) Show that ultrametric distances are also Euclidean.

   (b) Simulate some data with `rTraitCont` and show that distances among these variables may be Euclidean. Compare with data generated with `rnorm`.

   (c) Show that DNA distances cannot be Euclidean. Find a distance method with continuous variables that shows the same property. See the formula in `?dist` and compare with what you know on how DNA distances are calculated.

2. Compare the seven methods available in `upgma`. You will draw a single figure with the seven UPGMA trees and the necessary annotations. You will take a data set of your choice.

3. Longer distances inferred from molecular sequences tend to have higher variances than the shorter ones. Derive a weighted version of the least squares formula by modifying equation 5.1 (p. 136) in order to give less importance to longer distances. Find a diagnostic plot that will confirm the rationale of this weighting scheme. (Hint: you may type `example(lm)` in R to find some inspiration.)

4. Consider a DNA sequence that evolves according to the Jukes–Cantor (JC69) model.

   (a) Build the corresponding rate matrix using for the overall rate of change the value 3×10^{-4}.

   The substitution rate is $\alpha = 10^{-4}$ and the diagonal elements are equal to minus the overall rate of change:

```
> Q <- matrix(1e-4, 4, 4)
> diag(Q) <- 0-3*1e-4
> Q
       [,1]   [,2]   [,3]   [,4]
[1,] -3e-04  1e-04  1e-04  1e-04
[2,]  1e-04 -3e-04  1e-04  1e-04
[3,]  1e-04  1e-04 -3e-04  1e-04
[4,]  1e-04  1e-04  1e-04 -3e-04
```

   (b) Compute, using two different approaches, the probability matrix for $t = 1$, $t = 1000$, and $t = 1 \times 10^6$. What do you observe? Was that expected?

```
> matexpo(Q * 1000)
           [,1]       [,2]       [,3]       [,4]
[1,] 0.75274003 0.08241999 0.08241999 0.08241999
[2,] 0.08241999 0.75274003 0.08241999 0.08241999
[3,] 0.08241999 0.08241999 0.75274003 0.08241999
[4,] 0.08241999 0.08241999 0.08241999 0.75274003
> matexpo(Q * 1e6)
     [,1] [,2] [,3] [,4]
[1,] 0.25 0.25 0.25 0.25
[2,] 0.25 0.25 0.25 0.25
[3,] 0.25 0.25 0.25 0.25
[4,] 0.25 0.25 0.25 0.25
```

   For longer times, the probabilities tend to the expected frequencies of each base as expected under a Markovian model.

   (c) What could you conclude about phylogeny estimation from this exercise?

   If the rate of substitution is large with respect to time, the sequences are expected to be “saturated”, so that signal of ancestry will be erased.

5. Consider a GTR model with the following parameters: $\alpha = 0.001$, $\beta = 5 \times 10^{-4}$, $\gamma = 2 \times 10^{-4}$, $\delta = 3 \times 10^{-4}$, $\epsilon = 1 \times 10^{-4}$, $\zeta = 5 \times 10^{-5}$, $\pi_A = 0.35$, $\pi_G = 0.17$, $\pi_C = 0.25$, and $\pi_T = 0.23$.

   (a) Build the corresponding rate matrix.

```
> rates <- c(alpha = 0.001, beta = 5e-4, gamma = 2e-4,
+            delta = 3e-4, epsilon = 1e-4, zeta = 5e-5)
> Q <- matrix(0, 4, 4)
> Q[lower.tri(Q)] <- rates
> Q <- t(Q)
> Q[lower.tri(Q)] <- rates
>
> isSymmetric(Q)
[1] TRUE
>
> pi <- c(A = 0.35, G = 0.17, C = 0.25, T = 0.23)
>
> sum(pi) == 1
[1] TRUE
>
> Q <- pi * Q
> Q
         [,1]    [,2]     [,3]     [,4]
[1,] 0.000000 3.5e-04 1.75e-04 7.00e-05
[2,] 0.000170 0.0e+00 5.10e-05 1.70e-05
[3,] 0.000125 7.5e-05 0.00e+00 1.25e-05
[4,] 0.000046 2.3e-05 1.15e-05 0.00e+00
> diag(Q)
[1] 0 0 0 0
> diag(Q) <- -apply(Q, 1, sum)
>
> apply(Q, 1, sum)
[1] -5.421011e-20  3.388132e-21  8.470329e-21  6.776264e-21
> all.equal(apply(Q, 1, sum), rep(0, 4), tol = 1e-16)
[1] TRUE
```

   (b) Compute the probability matrix for $t = 1$.

```
> P <- matexpo(Q)
> P
             [,1]         [,2]         [,3]         [,4]
[1,] 9.994052e-01 3.498616e-04 1.749387e-04 6.998043e-05
[2,] 1.699328e-04 9.997621e-01 5.100348e-05 1.700356e-05
[3,] 1.249562e-04 7.500512e-05 9.997875e-01 1.250318e-05
[4,] 4.598714e-05 2.300482e-05 1.150293e-05 9.999195e-01
> apply(P, 1, sum)
[1] 1 1 1 1
```

```
> library(expm)
> round(expm(Q) - matexpo(Q), 15)
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
[4,]    0    0    0    0
```

   (c) Find a method to simulate the evolution of a DNA sequence under this GTR model for an arbitrary t.

   (d) What are the expected base frequencies when t is very large?

6. Write R code to calculate the distance between two aligned nucleotide sequences with the GTR model using equation 5.8 (p. 144). (Hints: you will need the functions `base.freq`, `Ftab`, `eigen`, and `solve`. The 'trace' function is the sum of the diagonal elements of a matrix.)

7. (a) Give the R code to calculate `Py` in Section 5.2.2. Compare with `Px`. How this will affect subsequent calculations? Eventually try different values of α.

   (b) Why the likelihood values do not sum to one?

   (c) For site 2, why the likelihood for A is twice bigger than for T?

   (d) How many parameters are involved in these calculations?

   (e) Write an R function to calculate the likelihood of the (full) data; the arguments of this function will be these parameters.

8. Sketch a function doing Bayesian estimation of phylogeny. The code should include comments explaining the rationale of the choices.

9. Take the data prepared in Exercise 5 of Chapter 3.

   (a) Build saturation diagrams for the whole sequence, and for each codon position.

   (b) Examine graphically the effects of unequal transition and transversion rates and / or unequal base frequencies on the distance estimates for each data set (whole sequences and each codon position).

10. Generate 100 bootstrap trees from the woodmouse data (see p.176). Compute and plot the consensus network displaying the splits with frequency 0.1 or more. Compare with Fig. 5.13 and explain the differences. Repeat with 1000 bootstrap trees.

Chapter 6

1. Simulate for 99 time-steps two independent Brownian motion models with the same initial values. These variables should be taken as two species that have diverged after $t = 1$, and they should be stored in a two-column matrix.

(a) Simulate the divergence of each species in two daughter-species at $t = 100$ under the same model for 100 time-steps: the results should be stored in a four-column matrix. Plot the whole evolution for the 200 time-steps on a single graph.

```
X <- matrix(NA, 100, 2)
for (i in 1:2)
  X[, i] <- cumsum(c(0, rnorm(99)))
X2 <- matrix(NA, 100, 4)
for (i in 1:4)
  X2[, i] <- cumsum(c(X[100, ceiling(i/2)], rnorm(99)))
matplot(101:200, X2, type = "l", col = 1, xlim = c(1, 200))
matlines(X, col = 1)
```

(b) Repeat (a) but using an Ornstein–Uhlenbeck model with $\alpha = 0.2$, $\theta_1 = -1$ for the first pair of species, and $\theta_2 = 1$ for the second one.

```
X <- matrix(NA, 100, 2)
for (i in 1:2)
  X[, i] <- cumsum(c(0, rnorm(99)))
X2 <- matrix(NA, 101, 4)
X2[1, ] <- c(rep(X[100, 1], 2), rep(X[100, 2], 2))
a <- 0.2
t1 <- -20
t2 <- 20
for (j in 1:2) {
    e <- rnorm(100)
    for (i in 1:100)
      X2[i + 1, j] <- -a*(X2[i, j] - t1) + e[i]
    }
for (j in 3:4) {
    e <- rnorm(100)
    for (i in 1:100)
      X2[i + 1, j] <- -a*(X2[i, j] - t2) + e[i]
    }
#X2 <- X2[-1, ]
matplot(101:201, X2, type = "l", col = 1, xlim = c(1, 201))
matlines(X, col = 1)
```

(c) Repeat (b) with $\theta_1 = -20$ and $\theta_2 = 20$. Compare the results.

2. Repeat the analyses on the primate data with Moran's I (Section 6.1.2) using weights computed from the variance-covariance matrix (see `?vcv`).

3. Simulate two variables with `rnorm` and two with `rTraitCont`. You will analyze these pairs of variables with phylogenetically independent contrasts (Section 6.1.1) and with generalized least squares (Section 6.1.5); you will do all regressions with and without intercept. Find when the regression coefficients are the same with both methods.

4. Calculate the expected values of the Brownian motion and the Ornstein–Uhlenbeck models after 100 time-steps. Compare with the observed values from the simulations above.

5. Simulate a standard Brownian motion process with $\mathrm{d}t = 0.1$ until $t = 100$ (hint: there should be 999 iterations in the simulation). Calculate the variance among replicates of this process and compare with the one simulated above with $\mathrm{d}t = 1$. Which parameter of the latter we should modify to obtain similar variances?

6. Implement Desdevises et al.'s [54] method in R (see p. 216).

7. Consider the phylogenies estimated for *Sylvia* (Section 5.1.8). Compute the phylogenetically independent contrasts for migration distance using the following branch lengths:

   - The neighbor-joining estimates (Fig. 5.17);
   - From the chronogram estimated by penalized likelihood (Fig. 5.21);
   - Setting the node heights so that they are equal to the number of descendants (see `compute.brlen`);
   - All equal to one.

   Compare the results and comment on the assumptions underlying the use of each set of branch lengths.

8. Consider the neighbor-joining tree estimated for the genus *Sylvia* and the associated bootstrap values.

   (a) Compute the phylogenetically independent contrasts for the continuous variable (migratory distance, `mig.dist`) in the ecological data set.

   (b) We want to give more importance in the analysis to the contrasts associated with the nodes that are well supported by the bootstrap analysis. Propose a solution.

   (c) Compare the two sets of contrasts.

Chapter 7

1. Show that Aldous's branching model produces trees that tend to be balanced. Demonstrate this directly (i.e., without simulating trees) and by using tree simulations.

2. Simulate several discrete traits on a phylogenetic tree. Find some procedures to simulate a correlated evolution among these traits.

3. Simulate a sequence of nucleotides coding for a protein using as substitution rates 1, 0.8, and 2 for the three codon positions, respectively.

4. Simulate three continuous characters and three discrete ones along a phylogeny. Prepare a data frame with these six characters setting the rownames correctly.

5. Simulate some data in the same way than done to produce Fig. 7.3. Repeat the simulation outputting the ancestral values. Plot the two traits but displaying the nodes together with the tips. What analyses from Chapter 6 would you do on these data?

October 19, 2022