Short Course in Molecular Evolution and Phylogeny

Sebastian E. Ramos-Onsins
Ramos-Onsins Lab
Centre for Research in Agricultural Genomics (CRAG)

December 18, 2014

Contents

1 Objectives **3**

2 Few Basic concepts of Population Genetics useful for Phylogeny **4**
2.1 Mutation 4
2.2 Drift to Fixation 4
2.3 Selection and Fixation 5

3 Mutational Models and Sequence Evolution **7**
3.1 DNA sequence has "only" four different states (nucleotides) 7
3.2 The Jukes and Cantor model for correction of substitutions 8
3.2.1 The number of differences under the JC model 8
3.2.2 How can we infer the time of divergence using the observed differences? . 9
3.3 Including more realistic parameters 10
3.3.1 The HKY model: adding differences in Ts/Tv and in the frequency of nt. 10
3.3.2 Including heterogeneity across the sequence 11
3.3.3 Differences in the divergence estimation when we fail to use the correct model 13
3.4 Model Selection 14
3.4.1 The Likelihood of the data for a given model 15
3.4.2 The Likelihood Ratio Test (LRT) to estimate the best model 17

4 Phylogenetics **18**
4.1 Rooted and unrooted Trees 19
4.2 Species Trees and Gene Trees 19
4.3 Before starting a phylogenetic reconstruction 19
4.4 Create a history and simulate sequences 19
4.5 Methods of reconstruction of Phylogenetic Trees 22
4.5.1 Distance Matrices Methods 22
4.5.2 Methods based on Likelihood calculation 22
4.5.3 Practical Exercises 23
4.6 Support of the phylogenetic tree inferred 25
4.7 Phylogenomics: A way to obtain the Species Tree 26

5 Final Exercise **28**

6 References **29**

1 Objectives

In these sessions, the objective is the understanding of the theoretical bases and algorithms of Molecular Evolution and Phylogeny. We will learn basic concepts of Molecular Evolution and Phylogeny by mean of bioinformatics, that is, using computational methods to study and simulate evolutionary processes in molecules (focused on DNA sequences), and also the relationships between individuals or species.

You can review and obtain additional information from some general books, like Higgs and Attwood (2005) as well other more specifics, like the excellent book of Yang (2006).

2 Few Basic concepts of Population Genetics useful for Phylogeny

2.1 Mutation

The heritable information that contains the molecules of DNA can be modified by mutation events. The mutational process occurs randomly in a position of the strain with a given probability each generation. This random process can be modeled using a poisson distribution. Let's simulate a random process using the R package: Open the file "Bioinf_PopGenetics.R".

```
#In a population, the number of mutations depend on the mutation rate and the number of
individuals (and also the number of nucleotides per individual)
mu <- 1e-3 #mutations per generation per position
N <- 1e3 #individuals in the population
L <- 1e1 #positions per individual
expected.m <- mu * N * L
expected.m # is the average number of mutations in the population per generation in the
region of size L
#mutation is a random process, therefore for each generation we will have a random value
centered on expected.m
m <- rpois(n=1,lambda=mu*N*L)
m #will be for a specific case
```

2.2 Drift to Fixation

A mutation appear in a single individual and the fate of the mutation over generations is uncertain. A new mutation can disappear in the next generation with a high probability, depending on the effective population size $(1-1/N_e)$ or increase in frequency until fixation after many generations (given the change of frequencies by drift). Let's simulate a process of drift to fixation or loss:

```
#first a function to calculate the fixation time:
calculate.fixation.time.p <- function(initial.p,N,max.generations) {
   plotf <- array(0,dim=c(max.generations))
   p <- initial.p
   g <- 1
   plotf[g] <- p
   while(p > 0 && p < 1 && g < max.generations) {
     Pop <- rbinom(n=N,size=1,prob=p)
     p <- sum(Pop)/N
     g <- g + 1
     plotf[g] <- p
   }
   list(end.p=p,generations=g,plot.freqs=plotf)
}

# Then we start simulation
par(mfrow=c(1,3))
it <- 1e4
max.g <- 1e4
plotf <- array(0,dim=c(max.g))
plot(x=c(1:max.g),rep(0,max.g),ylim=c(0,1),type="l")
dist.fix <- array(0,dim=c(it))
n.fix <- 0
dist.los <- array(0,dim=c(it))
```

```
n.los <- 0

for(i in 1:it) {
   nm <- 1 #given a mutation, frequency in the initial population
   p <- nm/N

   fix.results <- calculate.fixation.time.p(initial.p=p,N=N,max.generations=max.g)
   plotf <- fix.results$plot.freqs
   g <- fix.results$generations
   p <- fix.results$end.p

   if(p == 1) {
     lines(x=c(1:g),y=plotf[1:g],type="l",col="black")
     dist.fix[n.fix+1] <- g
     n.fix <- n.fix + 1
   }
   if(p == 0) {
     lines(x=c(1:g),y=plotf[1:g],type="l",col="yellow")
     dist.los[n.los+1] <- g
     n.los <- n.los + 1
   }
}

hist(dist.fix[1:n.fix],breaks=30,main=sprintf("dist.fix:  n=%d; mean=%.3f", n.fix,mean(dist.fix[1:n.fix])),
freq=FALSE, xlim=c(0,max.g))
abline(v=mean(dist.fix[1:n.fix]),col="black")
n.fix
hist(dist.los[1:n.los],breaks=30,main=sprintf("dist.los:  n=%d; mean=%.3f", n.los,mean(dist.los[1:n.los])),
freq=FALSE, xlim=c(0,max.g))
abline(v=mean(dist.los[1:n.los]),col="yellow")
n.los

#proportion of fixed mutations
n.fix/(n.fix+n.los) #1/N under neutrality (total 1/N * it)
#number of generations to fix mutations
mean(dist.fix[1:n.fix]) #2*N under neutrality
```

2.3 Selection and Fixation

The speed and number of fixed mutations are affected by several processes, like the population size, the time generation but also selective events that increase or slow down the process of fixation. Let's simulate a selective process. In this case, we simulate that the mutation we study has some advantage over the wild variant that is present initially. Following classical population genetics theory, we assign a value $1+s$ of selection coefficient to this new variant. Thus, we have

Individuals	Wild (Np)	Mutant (Nq)
Next generation	Np/w	$Nq(1+s)/w$

where $w = p + q(1+s)$ is the total frequency and p/w and $q(1+s)/w$ are the frequencies of the wild and new variant at the next generation, respectively. In this case we modify the function that calculates the fixation time to consider selection: Highlighted in red are the modifications in respect to the previous function:

```
calculate.fixation.time.p.selection <- function(initial.p,N,max.generations,s=s) {
   plotf <- array(0,dim=c(max.generations))
```

```
    p <- initial.p
    g <- 1
    plotf[g] <- p
    while(p > 0 && p < 1 && g < max.generations) {
      p <- p*N*(1+s)/(p*N*(1+s) + (N-p*N))
      Pop <- rbinom(n=N,size=1,prob=p)
      p <- sum(Pop)/N
      g <- g + 1
      plotf[g] <- p
    }
    list(end.p=p,generations=g,plot.freqs=plotf)
}
```

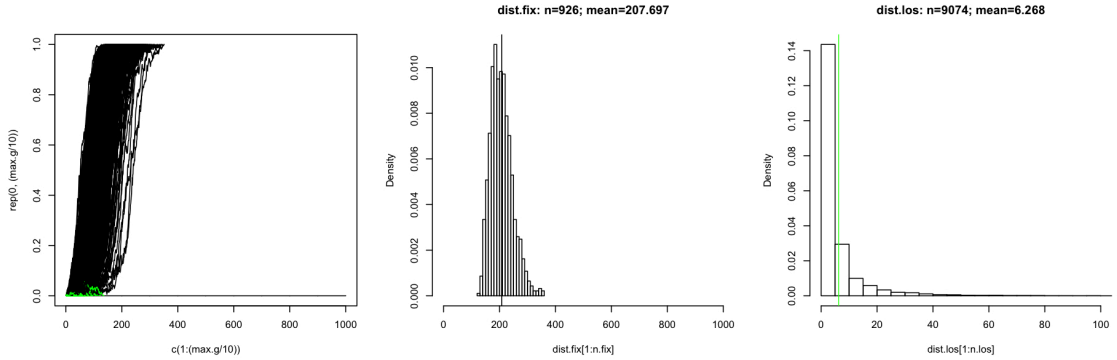

Figure 1: First, time to fix a selective mutation for $N = 1000$, $s = 0.05$ (in black) and to loss (in green). Second, distribution of the time of fixation events, Third, distribution of the time of loss events.

If we do a simulation using $s = 0.05$ and using the same parameters we used before the results are different (see Figure 1). In fact, the fraction of fixed mutations under the effect of strong selection ($|N_e s| > 1, |s| << 1$) is approximately $(1 - e^{(-2s)})/(1 - e^{(-2N_e s)})$ when $N_e s \gg 1$, while under no selection is $1/N_e$.

3 Mutational Models and Sequence Evolution

We are interested in studying the information that is contained in the DNA and the differences that are observed between individuals or organisms. Therefore, we will need to observe the data and consider a model to explain this data. Let's start with a simple two sequences of arbitrary length (1000 nt.). How many differences are between two sequences? How many differences we expect?. The first question seems easy to answer, but the second question needs the formulation of a model in order to have a better understanding of what happened.

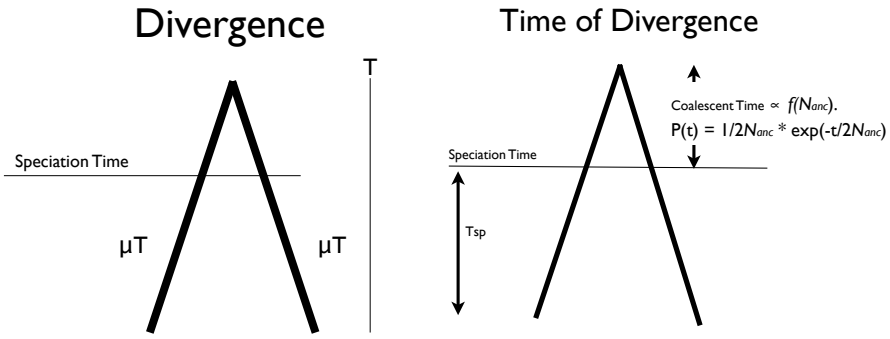

Figure 2: Representation of the divergence between two lineages. From the present to the past, a speciation time means the time since the species (or populations) split, while the time from the speciation time to the join of the branches represent the time the two lineages (not the species) diverged, which is a stochastic process. Here the equation is expressed for a diploid population.

Under the simplest model of constant mutation across generations, the number of expected mutations between two lineages are proportional to the number of generations since their common origin, That is,

$$m = 2\mu T_g \tag{1}$$

where m is the total number of mutations, T_g is the number of generations and μ is the mutation rate of the region per generation. The 2 means that we are counting from both sequences to the ancestor T_g ago. Under the neutral model, the substitution rate (λ) is equal to the mutation rate, as we observe in the previous section. Then, changing the terms we have $d = 2\lambda T_g$, where d is the number of substitutions between lineages.
Here we assume a infinite length sequence, a constant mutation rate across time, across lineage (implicitly that the time of generations is equal across lineages) and across sequence regions. Note also that T_g is not the time of split (or speciation) between two populations (or species), but the time of split between these two sequences, which would be larger than the split between populations. After speciation, the lineages coalesce some time afterwards, following a stochastic process that depends on the ancestral population size. Therefore, different regions in the genome may have relative different divergences.
Open the file "Bioinf_Molevol.R"and run the R commands following this section.

3.1 DNA sequence has "only" four different states (nucleotides)

Although we assume this simple model, we have to consider the fact that a DNA sequence contains four different nucleotides (A,C,G,T). In a finite length sequence, the mutations can accumulate over the same positions, and the mutations may revert to the ancestral state. Let's

simulate a sequence first:

```
#Let's generate a sequence:
Tnt <- c("A","C","G","T")
L <- 100
seq <- sample(x=Tnt,size=L,replace=TRUE,prob=c(0.25,0.25,0.25,0.25))
seq

#How many SUBSTITUTIONS we can see and how many substitutions really happened between 2
sequences that diverged G generations ago?
seq.ancestral <- seq
subst.rate = mu <- 1e-6 #assumption under neutrality:  rate of substitutions is equal to
mutation rate

G <- 1e6 #number of generations to the common ancestor
n.subst.lineage.1 <- rpois(n=1,lambda=subst.rate * L * G)
n.subst.lineage.1
n.subst.lineage.2 <- rpois(n=1,lambda=subst.rate * L * G)
n.subst.lineage.2
#Total mutations between the two sequences
n.subst.lineage.1 + n.subst.lineage.2
```

3.2 The Jukes and Cantor model for correction of substitutions

What is the probability that a given nucleotide (ex. A) mutate to another (ex. C)? We will use the simplest mutational model where each nucleotide has the same probability to mutate to another:

```
#we use a rate matrix for all combinations of mutations:
Q <- matrix(1/3,ncol=4,nrow=4,dimnames=list(c("A","C","G","T"),c("A","C","G","T")))
for(x in 1:4) {Q[x,x] <- 0; Q[x,x] <- -sum(Q[x,])}
Q #is the Substitution Rate matrix
```

3.2.1 The number of differences under the JC model

Let's calculate how many differences we really observe when compare two sequences assuming the Jukes and Cantor Model. We construct first a function:

```
substitutions.on.sequence <- function(seq,substitutions,Q,Tnt) {
   #seq is a vector of symbols (eg., A, T, A, G, C ...etc.)
   #substitutions is the total number of substitutions
   #Q is the substitution rate matrix (c x c)
   #Tnt is a vector of C values containing the possible symbols in the seq
   seq1 <- seq
   len <- length(seq)
   #where the substitutions fall on seq
   position.substitution <-sample(c(1:len),size=substitutions,replace=TRUE)
   number.symbols <- c(1:length(Tnt))
   for(ps in position.substitution) {
     nt <- which(Tnt==seq1[ps]) #which symbol is in seq[ps]
     symbol <- number.symbols[-nt] #count the possible variants (not the original)
     seq1[ps] <- sample(Tnt[-nt],1,prob=Q[nt,-nt]) #assign the new nucleotide
   }
   seq1 #the output is the new vector of symbols after including substitutions
}
```

The number of differences observed can be contrasted with the number of real mutation occurred:

```
#The observed differences are:
seq.lineage.1 <- substitutions.on.sequence(seq.ancestral,n.subst.lineage.1,Q,Tnt)
seq.lineage.2 <- substitutions.on.sequence(seq.ancestral,n.subst.lineage.2,Q,Tnt)

#The ancestral and new sequences:
seq.ancestral
seq.lineage.1
seq.lineage.2

#the number of observed and real differences differences:
Diff <- matrix(0,nrow=3,ncol=2,dimnames=list(c("1.vs.2","anc.vs.1","anc.vs.2"),c("OBS","REAL")))
Diff[1,1] <- sum(seq.lineage.1 != seq.lineage.2)
Diff[2,1] <- sum(seq.ancestral != seq.lineage.1)
Diff[3,1] <- sum(seq.ancestral != seq.lineage.2)
Diff[1,2] <- n.subst.lineage.1 + n.subst.lineage.2
Diff[2,2] <- n.subst.lineage.1
Diff[3,2] <- n.subst.lineage.2
Diff
```

3.2.2 How can we infer the time of divergence using the observed differences?

The relation between the observed divergence (number of observed differences divided by the total positions) and the real divergence is determined by the following equation:

$$d = -\frac{3}{4} ln(1 - \frac{4}{3}D) \tag{2}$$

where D is the observed divergence and d is the corrected ("real") divergence. This equation is obtained from the following reasoning: considering two sequences that diverged t generations ago, and we have q proportion of sites that are identical, the substitution rate per generation is λ. The proportion of sites that are invariant the next generation is $q_{t+1} = q_t(1-2\lambda) + (1-q_t)2/3\lambda$, in relation to sites that at generation t were equal (q_t) or different ($1 - q_t$) respectively. If we put together $q_{t+1} - qt = \frac{dq}{dt} = 2/3\lambda - 8/3q\lambda$ and solving the differential equation, we obtain $p = 1 - q = 3/4(1 - e^{-8/3\lambda t})$, which is also interpreted as the probability to have a variant position at time t. Finally, substitute $\lambda t = d/2$ and we obtain the equation 2.

First calculate the theoretical curves of the relationship between the observed divergence and the real divergence (or the time in generations):

```
subst.rate <- 1e-6
par(mfrow=c(1,1))
G <- seq(from=1e3,to=1e6,by=(1e6-1e3)/29) #we subset 30 different divergence times
y1 <- seq(from=0,to=2*subst.rate*G[30],by=2*subst.rate*G[30]/29)
plot(x=2*G,y=y1,ylim=c(0,2),type="l",col="red",xlab="Time",ylab="Divergence")
y2 <- c(3/4-3/4*exp(-4/3*subst.rate*2*G)) #theoretical curve
lines(x=2*G,y=y2,ylim=c(0,2),type="l",col="black")
```

Let's do the calculation of observed versus real number of substitutions:

```
#Simulate a new sequence of length 1000
L <- 1e3
seq.ancestral <- sample(x=Tnt,size=L,replace=TRUE,prob=c(0.25,0.25,0.25,0.25))
#Jukes and Cantor rate matrix
Q.JC <- matrix(1/3,ncol=4,nrow=4,dimnames=list(c("A","C","G","T"),c("A","C","G","T")))
for(x in 1:4) Q.JC[x,x] <- 0; Q.JC[x,x] <- -sum(Q.JC[x,])
```

```
#calculate divergence for different times (contained in the G vector)
D.obs <- array(0,30)
D.real <- array(0,30)
i <- 1
for(g in G) {
   n.subst.lineage.1 <- rpois(n=1,lambda=subst.rate * L * g)
   n.subst.lineage.2 <- rpois(n=1,lambda=subst.rate * L * g)
   seq.lineage.1 <- substitutions.on.sequence(seq.ancestral,n.subst.lineage.1,Q.JC,Tnt)
   seq.lineage.2 <- substitutions.on.sequence(seq.ancestral,n.subst.lineage.2,Q.JC,Tnt)
   D.obs[i] <- sum(seq.lineage.1 != seq.lineage.2)/length(seq.ancestral)
   D.real[i] <- (n.subst.lineage.1 + n.subst.lineage.2)/length(seq.ancestral)
   i <- i + 1
}
lines(x=2*G,y=D.obs ,type="p",col="black")
lines(x=2*G,y=D.real,type="p",col="red")
legend("topleft",legend=c("Real","Observed"),text.col=c("red","black"))
```

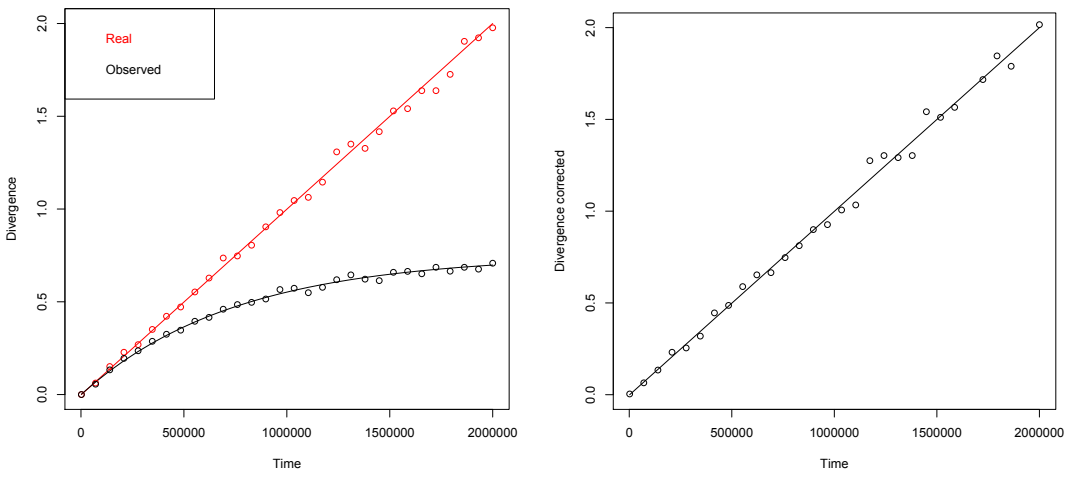

Figure 3: First, the observed (black dots) and real (red dots) divergence and their adjustment to predictions (lines). Second, linear relationship between corrected divergence and real divergence.

3.3 Including more realistic parameters

A number of models can be constructed taking into account more parameters to estimate the corrected divergence, as for example differences in the nucleotide proportion, different substitution rate for each type of mutation and also heterogeneity across the sequence length.

3.3.1 The HKY model: adding differences in Ts/Tv and in the frequency of nt.

Here we study the model HKY (Hasegawa, Kishino and Yano), where it is considered the different frequencies of nucleotides in the sequence as well as the possibility to have different substitution rate in transitions and transversions. Let's calculate the transition matrix using the HKY model for a given parameters:

$$P(t) = \begin{pmatrix} p_{AA}(t) & p_{GA}(t) & p_{CA}(t) & p_{TA}(t) \\ p_{AG}(t) & p_{GG}(t) & p_{CG}(t) & p_{TG}(t) \\ p_{AC}(t) & p_{GC}(t) & p_{CC}(t) & p_{TC}(t) \\ p_{AT}(t) & p_{GT}(t) & p_{CT}(t) & p_{TT}(t) \end{pmatrix}$$

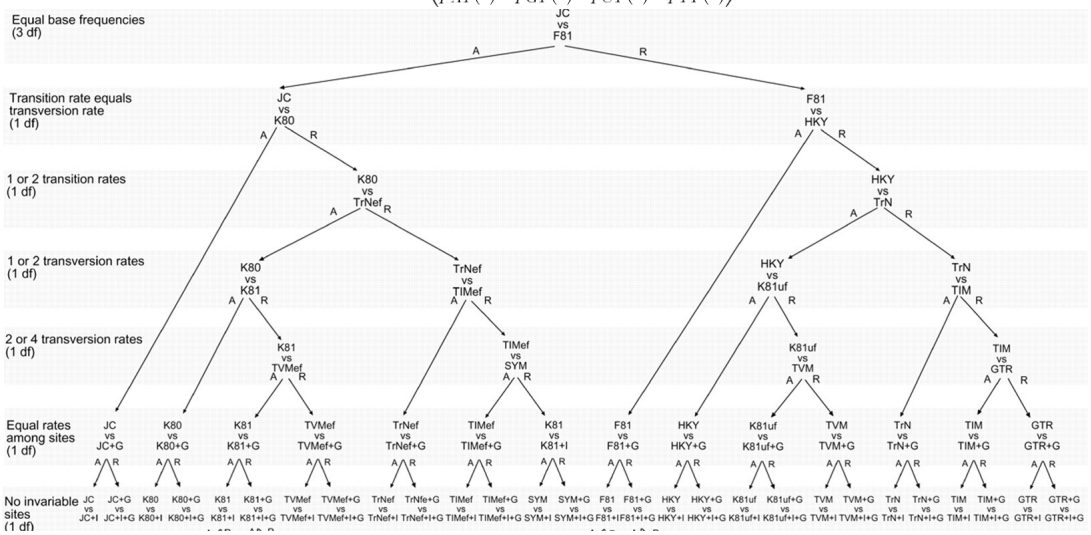

Figure 4: First, Transition matrix between the different nucleotide substitutions. Second, figure showing a number of different models taking into account different number of parameters, modified from Posada and Crandall (2001). See also the powerful software jModelTest2 (Darriba et al., 2012) .

```
#for example:
p.A <- 0.55
p.C <- 0.06
p.G <- 0.36
p.T <- 1- (p.A + p.C + p.G)
p.S <- 0.75
p.V <- 1 - p.S

Q.HKY <- c(0, p.C*p.V, p.G*p.S, p.T*p.V, p.A*p.V, 0, p.G*p.V, p.T*p.S, p.A*p.S, p.C*p.V
,0 , p.T*p.V, p.A*p.V, p.C*p.S, p.G*p.V, 0)
Q.HKY <- matrix(Q.HKY,ncol=4,nrow=4,byrow=TRUE,dimnames=list(c("A","C","G","T"), c("A","C","G","T")))
for(x in 1:4) Q.HKY[x,x] <- 0; Q.HKY[x,x] <- -sum(Q.HKY[x,])
Q.HKY #Substitution rate matrix for this specific parameters
```

3.3.2 Including heterogeneity across the sequence

The divergence can be wrongly corrected if we do not account with the possibly heterogeneity across the sequence length. For example, in case having regions with coding and intronic regions, the coding positions can be much more restricted to accumulate variants. Thus, the total positions that can "freely" change are smaller, and therefore the total positions we should consider for correcting the divergence is smaller. The consequence is that the correction is underestimated and we will fail to obtain the correct divergence.

First we do a function that consider invariant positions (positions that never have mutation). Highlighted in red are the R code that has been modified in relation to the previous function considered in the previous section:

```
#Function:  The number of mutations accumulate over the sequence with invariant positions:
substitutions.on.sequence.plusI <- function(seq,I,substitutions,Q,Tnt) {
   #seq is a vector of symbols (eg., A, T, A, G, C ...etc.)
   #I is the ratio of invariant positions
   #substitutions is the total number of substitutions
   #Q is the substitution rate matrix (C x C)
   #Tnt is a vector of C values containing the different symbols in the seq
   seq1 <- seq
   len <- floor(length(seq)*(1-I))
   #where the substitutions fall on seq
   position.substitution <- floor(runif(n=substitutions,min=1,max=len))
   number.symbols <- c(1:length(Tnt))
   for(ps in position.substitution) {
     nt <- which(Tnt==seq1[ps]) #which variant is in seq[ps]
     symbol <- number.symbols[-nt] #count the possible variants (not the original)
     seq1[ps] <- sample(Tnt[-nt],1,prob=Q[nt,-nt]) #assign the new nucleotide
   }
   seq1 #the output is the new vector of symbols after including substitutions
}
```

We now consider that we have heterogeneity in the positions and we model that effect using a gamma distribution. The gamma distribution can have very different shapes and it has two parameters, the shape (α) and the scale (β). Changing these parameters we can have Normal-like distributions to Exponential-like distributions. The mean of the gamma distribution is $E(x) = \alpha/\beta$ and variance $var(x) = \alpha/\beta^2$. If we fix $\beta = \alpha$ we have a distribution where $E(x) = 1$ and $var(x) = 1/\alpha$, so we can model the heterogeneity using a single parameter. For example, for the case of three different scale values of 100, 1 and 0.1 we have the following distributions:

```
par(mfrow=c(1,3))
par.gamma <- c(100,1,0.01)
for(i in 1:3) {
   alpha <- par.gamma[i]
   beta <- alpha
   d <- rgamma(n=10000,shape=alpha,rate=beta)
   text.plot <- sprintf("Gamma distibution\nmean=%.2f, var=%.3f",mean(d),var(d))
   hist(d,main=text.plot)
}
```

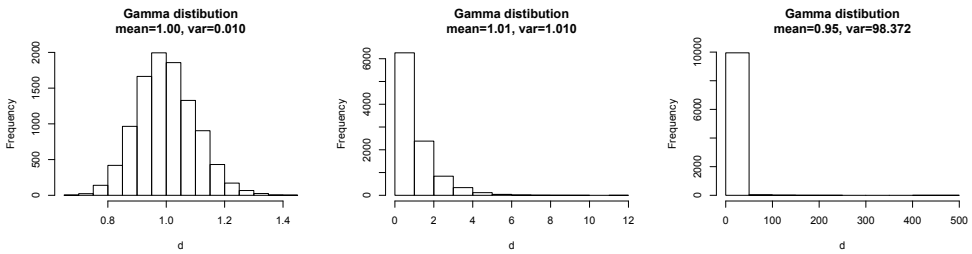

Figure 5: First, with shape=100 the distribution is very narow, almost no heterogeneity. Second, with shape=1 the distribution is an exponential like. third, shape=0.1 gives many 0 values close to zero and few with very high values.

To simulate a process containing heterogeneity modeld with a gamma distribution we modify the previous function. Highlighted in red are the modifications:

```
#Function:  The number of mutations accumulate over the sequence with invariant positions:
#and also a variable substitution rate given by a gamma distribution:
substitutions.on.sequence.plusI.Gamma <- function(seq,I,Gshape=0,substitutions,Q,Tnt) {
   #seq is a vector of symbols (eg., A, T, A, G, C ...etc.)
   #I is the ratio of invariant positions
   #Gshape is the parameter shape=scale in a gamma distribution
   #substitutions is the total number of substitutions
   #Q is the substitution rate matrix (C x C)
   #Tnt is a vector of C values containing the different symbols in the seq
   seq1 <- seq
   len <- floor(length(seq)*(1-I))
   #including gamma:
   if(Gshape > 0) {
     position.substitution <- array(,substitutions)
     pg <- rgamma(n=len,shape=Gshape,rate=Gshape) #each position has a value
     wt <- pg/sum(pg) #each position has a weight according to the gamma distribution
     #sample each position according its weight
     position.substitution <- sample(c(1:len),size=substitutions,replace=TRUE,prob=wt)
   } else {
     #where the substitutions fall on seq
     position.substitution <- floor(runif(n=substitutions,min=1,max=len))
   }
   number.symbols <- c(1:length(Tnt))
   for(ps in position.substitution) {
     nt <- which(Tnt==seq1[ps]) #which variant is in seq[ps]
     symbol <- number.symbols[-nt] #count the possible variants (not the original)
     seq1[ps] <- sample(Tnt[-nt],1,prob=Q[nt,-nt]) #assign the new nucleotide
   }
   seq1 #the output is the new vector of symbols after including substitutions
}
```

3.3.3 Differences in the divergence estimation when we fail to use the correct model

The right selection of the correct model is currently fundamental for accurate estimates of the divergence. Let's simulate data considering an evolution under the JC+I+G model and compare with the theoretical curve given JC model:

```
#Example Plot:  JC + gamma + Invariant positions
par(mfrow=c(1,1))
L <- 1e3
seq.ancestral <- sample(x=Tnt,size=L,replace=TRUE,prob=c(0.25,0.25,0.25,0.25))
subst.rate <- 1e-6
mu <- 1e-6
ratio.invariant <- 0.3
Gamma.shape <- c(1)

#we subset 30 different divergence times G <- seq(from=1e3,to=1e6,by=(1e6-1e3)/29) y1 <-
seq(from=0,to=2*subst.rate*G[30],by=2*subst.rate*G[30]/29)
plot(x=2*G,y=y1,ylim=c(0,2),type="l",col="black",xlab="Time",ylab="Divergence")
#theoretical curve JC
y2 <- c(3/4-3/4*exp(-4/3*subst.rate*2*G))
lines(x=2*G,y=y2,ylim=c(0,2),type="l",col="cyan")
#theoretical curve + invariant
y3 <- c(3/4*(1-ratio.invariant)-3/4*(1-ratio.invariant)*exp(-4/(3*(1-ratio.invariant))*subst.rate*2*G))
lines(x=2*G,y=y3,ylim=c(0,2),type="l",col="blue")
#theoretical curve + invariant + gamma
y4 <- c(3/4*(1-ratio.invariant)*(1-(4/(3*(1-ratio.invariant))*subst.rate*2*G / Gamma.shape+1)^-Gamma.shape))
lines(x=2*G,y=y4,ylim=c(0,2),type="l",col="red")
```

```
D.obs.JC.gamma <- array(0,30)
D.real <- array(0,30)
i <- 1
for(g in G) {
   #gamma is not included when calculating total mutations because the mean = subs.rate
   n.subst.lineage.1 <- rpois(n=1,lambda=subst.rate * L* (1-ratio.invariant) * g)
   n.subst.lineage.2 <- rpois(n=1,lambda=subst.rate * L* (1-ratio.invariant) * g)
   D.real[i] <- (n.subst.lineage.1 + n.subst.lineage.2)/(length(seq.ancestral)*(1-ratio.invariant))
   seq.lineage.1.JC <- substitutions.on.sequence.plusI.Gamma( seq.ancestral,I= ratio.invariant,
Gamma.shape, n.subst.lineage.  1, Q.JC ,Tnt)
   seq.lineage.2.JC <- substitutions.on.sequence.plusI.Gamma( seq.ancestral,I= ratio.invariant,
Gamma.shape, n.subst.lineage.  2, Q .JC,Tnt)
   D.obs.JC.gamma[i] <- sum(seq.lineage.1.JC != seq.lineage.2.JC) / (length(seq.ancestral)*(1-0))
   i <- i + 1
}
lines(x=2*G,y=D.obs.JC.gamma,type="p",col="red")
lines(x=2*G,y=D.real,type="p",col="black")
```

Here we want to show that the selection of a different model from the real one can give us very different estimates of corrected divergence (see Figure 6), which may seriously affect the results or the conclusions of our work. Therefore we have to be cautious choosing the right evolutionary model.

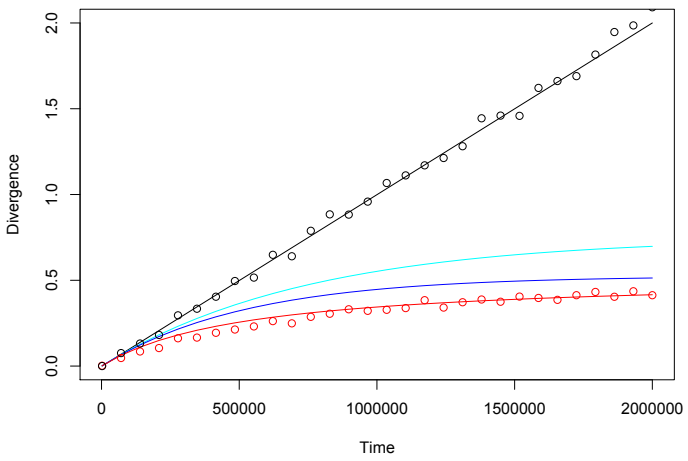

Figure 6: Plot showing the theoretical (lines) and simulated corrected divergence (circles) for JC (cyan), JC+I (blue), JC+I+G (red). Black lines shows the number of real mutations on the sequence

3.4 Model Selection

As it is commented before before, the selection of the correct model is fundamental for accurate estimates of the divergence. How can we select the best model? Given some data (for example two sequences from two different species), one way is calculating the probability of the data given a specific chosen model. If we chose different models we can calculate the probability (likelihood) for each model and then compare them. A way to do it is using the likelihood ratio test.

3.4.1 The Likelihood of the data for a given model

The likelihood must be calculated from the theoretical equations from each different model. The probability of a given model must be obtained over the best parameter(s) for this model. For example, in the Jukes and Cantor model, the only parameter is d, the real divergence of two sequences. The probability is obtained for each position independently and given by isolating D (difference or not in this position) in equation 2. $p_0(d)$ indicates the probability to have an invariant position given a divergence d. $p_1(d)$ indicates the probability to have a given variant at a unique position given a divergence d (3 different possibilities) .

$$p_0(d) = \frac{1}{4} + \frac{3}{4}e^{-\frac{4}{3}*d} \tag{3}$$

$$p_1(d) = \frac{1}{3}(1 - p_0(d)) = \frac{1}{4} - \frac{1}{4}e^{-\frac{4}{3}*d} \tag{4}$$

Given that there are two states (mutation or not), we use a binomial to calculate the global probability $f(x|d) = C(3p_1(d))^x(p_0(d))^{(L-x)}$, where C is a constant, L is the length of the sequence and x the number of mutations. The best parameter d is the one that has the highest probability. Finally, as a technical comment, the probability is redefined to consider all 4x4 type of mutations in order to compare other more complex models that consider transitions and transversions and the frequencies of the nucleotides (multiplied by 1/4 and use a multinomial of 16 cells). The logarithm of the function is calculated to facilitate the computation of $f(x|d)$.

```
#Observed data:
L <- 1e3
x <- 300 #D.obs = 300/1e3 = 0.3
nS <- 125 #number of transitions
nV <- x-nS #number of transversions

par(mfrow=c(1,3))
#In case JC, the probability that a position has a different nucleotide between 2 seqs is:
nit <- 1e5
D.real <- runif(nit) #Here we simulate 1e5 values of divergence given the observed values
logLik <- x * log(1/16 - 1/16*exp(-4/3*D.real)) + (L-x) * log(1/16 + 3/16*exp(-4/3*D.real))

p.ML <- which(logLik == max(logLik))
ML.1 <- max(logLik)
MLE.1 <- D.real[p.ML]
ML.1
MLE.1

#Also, the confident interval can be obtained by lowering the MLE by X-square(1dgf,5%)/2
= 1.92:  (max(logLik) - 1.92)
Ci <- logLik
Ci[which(Ci > max(logLik) - 1.92)] = min(logLik)

cil <- 1
cir <- 1
for(i in 1:nit) {
   if(D.real[i] < MLE.1 && Ci[i] > Ci[cil]) cil <- i
   if(D.real[i] > MLE.1 && Ci[i] > Ci[cir]) cir <- i
}

D.real.int.chi2.1 <- c(0,0)
D.real.int.chi2.1[1] <- D.real[cil]
D.real.int.chi2.1[2] <- D.real[cir]

plot(D.real,logLik,cex = .1,main="Model JC69")
```

```
abline(v=D.real.int.chi2.1[1],col="red")
abline(v=D.real.int.chi2.1[2],col="red")
text(D.real.int.chi2.1[1],min(logLik),sprintf("Dl=%.3f",D.real.int.chi2.1[1]),adj = c(1,0))
text(D.real.int.chi2.1[2],min(logLik),sprintf("Dr=%.3f",D.real.int.chi2.1[2]),adj = c(0,0))
abline(h=max(logLik) - 1.92,col="red")
text(min(D.real),max(logLik),sprintf("Lk=%.3f",max(logLik)),adj = c(0,0))

#We try our data with a model Kimura1980:  Parameters are the divergence and the ratio transition(S)
/ transversion(V)
#In case using the model K80, the probability that a position has a different nucleotide
between 2 seqs is:
#D.obs <- 1/2*log(1-2*S-V) - 1/4*log(1-2*V)
#The likelihood function is:

nit <- 1e5
D.real <- runif(nit)
k <- runif(nit,-1,1)
k <- 10 ^k
p0 <- 1/4+1/4*exp(-4*D.real/(k+2))+1/2*exp(-2*D.real*(k+1)/(k+2))
p1 <- 1/4+1/4*exp(-4*D.real/(k+2))-1/2*exp(-2*D.real*(k+1)/(k+2))
p2 <- 1/4-1/4*exp(-4*D.real/(k+2))
logLik <- (L-nS-nV) * log(p0/4) + nS * log(p1/4) + nV * log(p2/4)

p.ML <- which(logLik == max(logLik))
ML.2 <- max(logLik)
MLE.21 <- D.real[p.ML]
MLE.22 <- k[p.ML]
ML.2
MLE.21
MLE.22

D.real.int.chi2.2 <- c(0,0,0,0)
#The confident intervals can be obtained by lowering the MLE by X-square(1dgf,5%)/2 = 1.92:
(max(logLik) - 1.92)
Ci <- logLik
Ci[which(Ci > max(logLik) - 1.92)] = min(logLik)

#first parameter:
cil <- 1
cir <- 1
for(i in 1:nit) {
   if(D.real[i] < MLE.21 && Ci[i] > Ci[cil]) cil <- i
   if(D.real[i] > MLE.21 && Ci[i] > Ci[cir]) cir <- i
}

D.real.int.chi2.2[1] <- D.real[cil]
D.real.int.chi2.2[2] <- D.real[cir]

plot(D.real,logLik,cex = .1,main="Model K80")
abline(v=D.real.int.chi2.2[1],col="red")
abline(v=D.real.int.chi2.2[2],col="red")
text(D.real.int.chi2.2[1],min(logLik),sprintf("Dl=%.3f",D.real.int.chi2.2[1]),adj = c(1,0))
text(D.real.int.chi2.2[2],min(logLik),sprintf("Dr=%.3f",D.real.int.chi2.2[2]),adj = c(0,0))
abline(h=max(logLik) - 1.92,col="red")
text(min(D.real),max(logLik),sprintf("Lk=%.3f",max(logLik)),adj = c(0,0))

#second parameter:
cil <- 1
cir <- 1
for(i in 1:nit) {
   if(k[i] < MLE.22 && Ci[i] > Ci[cil]) cil <- i
   if(k[i] > MLE.22 && Ci[i] > Ci[cir]) cir <- i
```

```
}

D.real.int.chi2.2[3] <- k[cil]
D.real.int.chi2.2[4] <- k[cir]

plot(k,logLik,cex = .1,main="Model K80")
abline(v=D.real.int.chi2.2[3],col="red")
abline(v=D.real.int.chi2.2[4],col="red")
text(D.real.int.chi2.2[3],min(logLik),sprintf("kl=%.3f",D.real.int.chi2.2[3]),adj = c(1,0))
text(D.real.int.chi2.2[4],min(logLik),sprintf("kr=%.3f",D.real.int.chi2.2[4]),adj = c(0,0))
abline(h=max(logLik) - 1.92,col="red")
text(min(D.real),max(logLik),sprintf("Lk=%.3f",max(logLik)),adj = c(0,0))
```

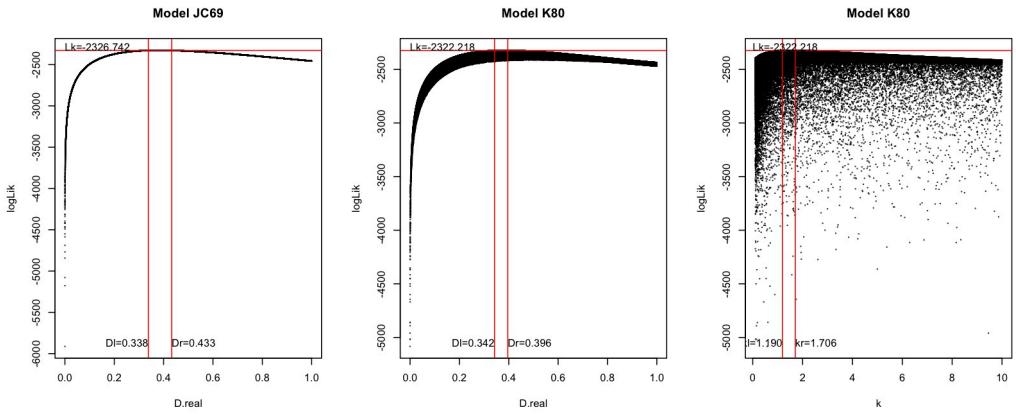

Figure 7: First. Likelihood plot of the observed data for the model JC and the confident intervals. Second and Third, Likelihood plot of the data for the model Kimura-2 for the parameters D and the ratio of transition/transversion (k), respectively.

3.4.2 The Likelihood Ratio Test (LRT) to estimate the best model

The LRT is a test that compares the Likelihood of two different models that are nested (that is, one model have more parameters than the other but the rest are common). The LRT distribution follows a χ^2 distribution with a number of degrees of freedom resulting of the difference of the number of parameters between the models. For non-nested models, another methods of comparison, like Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) are most used. Let's do a comparison of models using LRT:

```
#The significance can be obtained from a Chi-square with the difference between models as
degree of freedom
ML.1
ML.2
Xi2 <- 2*(ML.2-ML.1)
1-pchisq(Xi2,1) #Probability that the null model (JC) is accepted
```

4 Phylogenetics

This section explains some basic procedures for the construction of DNA sequence phylogenies, that is, the use of sequence DNA to construct evolutionary trees. A tree is a graphical representation of the relationships between the studied lineages using a tree structure in nodes and branches (see Figure 8).

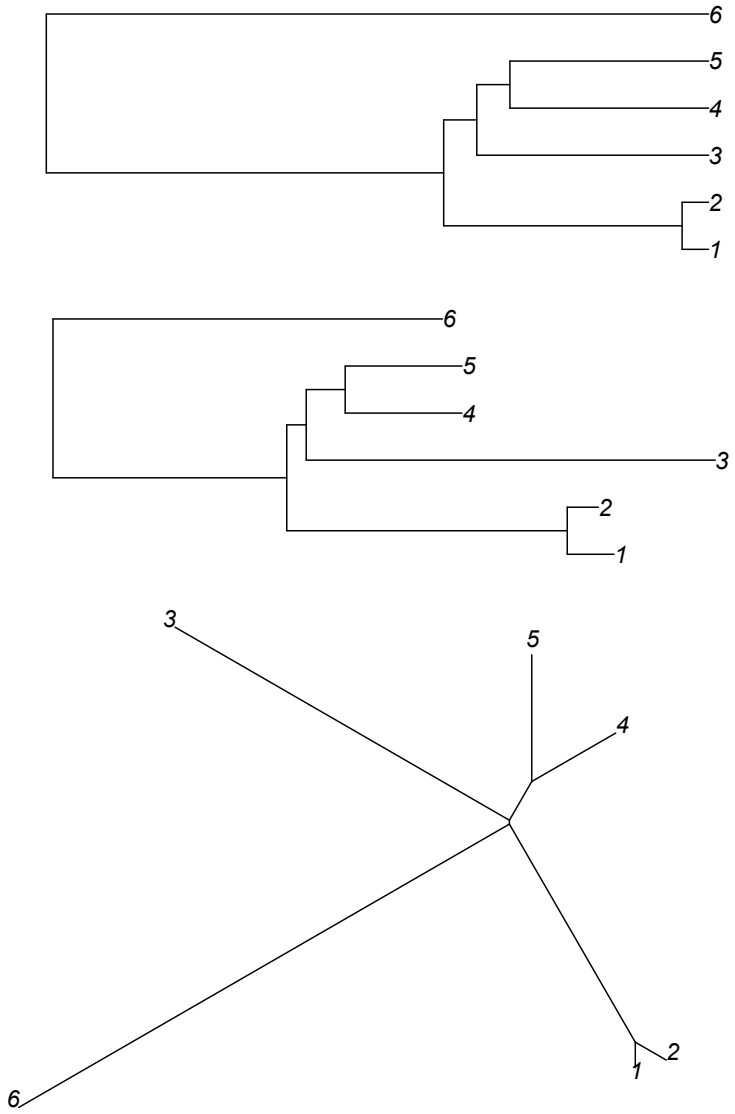

Figure 8: First, tree representation of six lineages, with different length branches representing the times to the common ancestral, represented by internal nodes. Second, representation of the same tree but showing the different substitution rates of each branch instead the time. Third, an unrooted tree; only the differences between lineages are represented, but not the time events.

4.1 Rooted and unrooted Trees

A rooted tree means that a factor related to the evolutionary time is represented over the axis of the tree, from the root (ancestral lineage) to the tips (the current lineages). An outgroup (farther lineage that join to the older node) is usually used to polarize the representation in the correct direction. The first two trees in the Figure 8 are rooted trees and the last one is an unrooted tree. An unrooted tree is a graphical representation of the relationship between lineages, but it does not give information about the ancestral times of the internal nodes (the common ancestors).

4.2 Species Trees and Gene Trees

The phylogeneticists are usually interested in inferring the relationship among the species, that is, a tree of the species. In molecular Evolution studies is common to represent phylogenetic trees using a region of the DNA sequence from the whole genome. This information is partial and may contain only the particular relationship of this region among the studied lineages (see Figure 2). A species tree should collect information from different sources to be a confident species tree. The phylogeneticists are also interested in evaluating the *monophyletic* features of groups. A monophyletic group is composed by the lineages that share a same common ancestor node.

4.3 Before starting a phylogenetic reconstruction

There are some basic steps before starting a phylogenetic reconstruction. First, we must be sure the sequences we are comparing are *homologous* sequences (the sequences have a common origin). In case of a species tree, it is important to have *orthologous* sequences (the sequences have a common origin by descendent, in contrast to *paralogous* sequences, related by duplication events). Also, once the alignment is done, it would be convenient to check the alignment and not use the regions that are too heterogeneous (see for example the program *Gblocks*, (Castresana, 2000; Talavera and Castresana, 2007)). Finally, the selection of the best evolutionary model is also fundamental for an accurate reconstruction, as we saw in the previous section (see for example *jModelTest2*, (Darriba et al., 2012)).
Open the file "Bioinf_Phylogeny.R"and run the R commands following this section.

4.4 Create a history and simulate sequences

Let's first start constructing a tree history. Imagine the following history given in the newick format:

```
#We want to study the following phylogeny:
#Newick format
tree.phyl <- "(((1,2),(3,(4,5))),6);"
tree.phyl.time <- "(((1:8e5,2:8e5):7.2e6,(3:7e6,(4:6e6,5:6e7):1e6):1e6):1.2e7,6:2e7);" #in
generations

par(mfrow=c(2,1))
library(ape) #plot the real species tree
tree.phyl.time.T <- "(((1:8e-1,2:8e-1):7.2e0,(3:7e0,(4:6e0,5:6e0):1e0):1e0):1.2e1,6:2e1);"
#in 1e6_generations
cat(tree.phyl.time.T, file = "tree.phyl.time.T.phy", sep = "\n")
tree.phyl.time.T <- read.tree("tree.phyl.time.T.phy") # load tree
unlink("tree.phyl.time.T.phy") #delete the file
plot(tree.phyl.time.T)
```

```
tree.phyl.time.D <- "(((1:0.024,2:0.016):0.144,(3:0.210,(4:0.06,5:0.06):0.02):0.01):0.12,6:0.2);"
#in divergence/nt
cat(tree.phyl.time.D, file = "tree.phyl.time.D.phy", sep = "\n")
tree.phyl.time.D <- read.tree("tree.phyl.time.D.phy") # load tree
unlink("tree.phyl.time.D.phy") # delete the file
plot(tree.phyl.time.D)
```

The example tree are the two first shown in the Figure 8. It is convenient to construct a matrix with the tree information and some more useful data for reconstructing the tree. In our example:

```
#Tree example:  include the nodes and relative times BUT ALSO mu and Ne:  That is, no real
molecular clock and the possibility of incomplete lineage sorting
n.lineages <- 6
tree.data <- matrix(c(
   1, 1, 7, 8e5, 3e-8, 1e4,
   2, 2, 7, 8e5, 2e-8, 1e3,
   3, 3, 9, 7e6, 3e-8, 1e4,
   4, 4, 8, 6e6, 1e-8, 1e4,
   5, 5, 8, 6e6, 1e-8, 1e5,
   6, 6, 11, 2e7, 1e-8, 1e4,
   1, 2, 10, 7.2e6, 2e-8, 5e3,
   4, 5, 9, 1e6, 2e-8, 1e5,
   3, 8, 10, 1e6, 1e-8, 1e5,
   7, 9, 11, 1.2e7, 1e-8, 1e5,
   6, 10, 0, 0, 1e-8, 1e4 #outgroup
   ),nrow=2*n.lineages-1,ncol=6,dimnames=list(c(sprintf("Node%02d", c(1:(2*n.lineages-1)))),
c("node1","node2","go.to.node","Time.sp","mu","Ne")), byrow=TRUE)
tree.data <- data.frame(cbind(tree.data[,1:6]))

tree.data
```

Calculate by simulation the divergence between the specific lineages (that is, the time of speciation plus the time of coalescent between lineages) and the total mutation for each branch. Finally construct the sequences for each lineage given the parameters included:

```
#counting the time period of coalescence of nodes (that is, from present to past, after
species join, count when the sequences coalesce)
#COALESCENCE of lineages in a population, prob=1/Ne in haploid population
#the expected time of coalescence follows an exponential distribution (the expected time
that something random occurs)
tree.data <- data.frame(cbind(tree.data[,c(1:6)],Time.seq=tree.data$Time.sp))

for(lg in (n.lineages+1):(2*n.lineages-1)) {
   #time speciation first ind.
   t1 <- tree.data$Time.seq[tree.data$node1[lg]]
   #time speciation second ind.
   t2 <- tree.data$Time.seq[tree.data$node2[lg]]
   #TIME COALESCENCE both sequences in the joined population
   time.coal <- rexp(n=1,rate=1/tree.data$Ne[lg])
   #adding time of coalescence to the first and second nodes
   tree.data$Time.seq[tree.data$node1[lg]] <- t1 + time.coal
   #adding time of coalescence to the first and second nodes
   tree.data$Time.seq[tree.data$node2[lg]] <- t2 + time.coal
}
tree.data

#counting the number of substitutions occurred in each branch
```

```
tree.data <- data.frame(cbind(tree.data[,c(1:7)],mutations=rep(NA,2*n.lineages-1)))
L <- 1e3 #The length of the sequence to study is L
tree.data$mutations <- rpois(n=2*n.lineages-1,lambda=tree.data$mu * tree.data$Time.seq *
L)
tree.data

## CONSTRUCT SEQUENCES ## #We want to know what are the sequences we have from this tree,
assuming a mutational model, for example JC69
Tnt <- c("A","C","G","T")
#Starting form a sequence, we include mutations
seq1 <- sample(x=Tnt,size=L,replace=TRUE,prob=c(0.25,0.25,0.25,0.25))
seq <- matrix(rep(seq1,2*n.lineages-1),ncol=L,byrow=TRUE)
#Start to include mutation to each node according the order of nodes, from past to present:
#Jukes and Cantor rate matrix
Q.JC <- matrix(1/3,ncol=4,nrow=4,dimnames=list(Tnt,Tnt))
for(x in 1:4) Q.JC[x,x] <- 0; Q.JC[x,x] <- -sum(Q.JC[x,])
for(node in (2*n.lineages-2):1) {
   seq[node,] <- substitutions.on.sequence.plusI.Gamma(seq=seq[node,],I=0, Gshape=0,
     substitutions=tree.data$mutations[node], Q=Q.JC, Tnt=Tnt)
   seq[tree.data$node1[node],] <- seq[node,]
   seq[tree.data$node2[node],] <- seq[node,]
}
```

The objective of this practical exercise is to reconstruct a phylogenetic tree and compare the real tree with the inferred trees obtained. We here choose the right model (Jukes and Cantor model) to calculate the divergence between the lineages (in this session the effect of selecting different models of evolution is not the objective).

```
#function calculating diff and divergence using JC69:
diffs.plus.jc69.div.matrix <- function(seq.matrix) {
   n.lineages <- (length(seq.matrix[,1])+1)/2
   L <- length(seq.matrix[1,])

   D <- matrix(0,nrow=n.lineages,ncol=n.lineages)
   #Calculate the matrix of differences between seqs
   for(i in 1:(n.lineages-1)) {
     for(j in (i+1):n.lineages) {
       D[i,j] <- sum(seq.matrix[i,] != seq.matrix[j,])
     }
   }
    #Calculate the divergence matrix assuming the model JC69:
   for(i in 1:(n.lineages-1)) {
     for(j in (i+1):n.lineages) {
       D[j,i] <- -3/4*log(1-4/3*D[i,j]/L) #Jukes and Cantor correction
     }
   }
   D
}

#Calculate the Divergence
D <- diffs.plus.jc69.div.matrix(seq)
D
#Fill in the symmetric matrix
for(i in 1:(length(D[1,])-1)) for(j in (i+1):length(D[,1])) D[i,j] <- D[j,i]
D
```

4.5 Methods of reconstruction of Phylogenetic Trees

There are many different methods of phylogenetic reconstruction. Here the methods that are commented are classified in the ones using distance matrices (for example, UPGMA, Neighbor-joining and others) or the ones using the entire information of the data (Likelihood and Bayesian methods). The Distance methods are typically fast and have unique solutions. On the other hand, Bayesian methods collect a number of compatible solutions, usually obtained by *heuristic methods* given the impossibility to calculate all possible solutions. A *consensus tree* (a compatible tree for all possible solutions with some unresolved nodes [more than two new branches]) is displayed as a solution. Parsimony methods are useful but not really used for sequence DNA data. Moreover, the effect of *Long Branch Attraction* (LBA) may easily distort the inference of a good phylogenetic tree using Parsimony.

4.5.1 Distance Matrices Methods

The distance methods must first calculate a matrix of distances between each pair of species. The matrix is used to reconstruct the phylogenetic tree. In general, the reconstruction involves the join of the closest nodes, recalculation of the matrix and repeat the process until finish. Typical distance methods are UPGMA, Least-square and Neighbour-joining methods.

The UPGMA method (Sokal and Sneath, 1963) is based on the assumption of a strict molecular clock evolution of the species. The obtained tree is always *ultrametric*, that is, in this rooted tree, the distance from the root to the tips is equal for whatever tip in the tree. The methodology seeks for the shortest distance of the matrix, then calculates the distance to the common ancestor as the half of the distance among both species. Then recalculate the matrix calculating the distance between the new cluster and the rest. This distance is calculated using the mean distance with all the species contained in the cluster. The process is repeated until having a single cluster. This method is very affected to the violation of the assumption of a strict molecular clock and is not very used.

The Neighbour-Joining method (Saitou and Nei, 1987), like the Least-square method (Cavalli-Sforza and Edwards, 1967) does not assume a molecular clock. That means the substitution rate may be different in the different branches. Also, this methods construct unrooted trees. In the case of Neighbour-joining (NJ) method, the *minimum evolution* criterion is used, that is, the tree with the smallest sum of branch lengths. The algorithm starts with a *star tree* and then look for two nodes that have the greatest reduction in tree length. The process is repeated until the tree is fully *resolved* (that is, each internal node bifurcates). The NJ method gives the right solution if the original distance matrix is *additive*. A distance matrix is additive if it is possible to find a tree where the pairwise distances will be equal to the sum of all branches that join the two given nodes. The NJ method is quite robust and gives reasonable trees in very short time. The concerns to this method are the possible no additivity of the distance matrix given, which may happen if the evolutionary method is incorrect, poor alignments or with the presence of horizontal transfer, but also works improperly for very divergent sequences, specially when substitution rates are high. Another algorithm, named BIONJ (Gascuel, 1997) solve this last situation and performs better with high and variable substitution rates.

4.5.2 Methods based on Likelihood calculation

A maximum likelihood (ML) criterion can be used to estimate tree topologies. This method looks for the highest probability of observing the data for a given parameters that define the topology

of the tree. Using heuristic methods, the whole parameters can be estimated and the ML tree is obtained. This method uses a large number of calculations and is generally slower than methods based on distances. On the other hand, the solutions are commonly very reasonable.

Bayesian methods to obtain a phylogenetic tree: the fundamental difference between the ML methods and the Bayesian methods is the concept of the uncertainty of the parameter. The parameter to be estimated has also a probability distribution that must to be inferred. The methodology to estimate the distribution of the parameters is based on the Bayes theorem. Priors to each parameter have to be defined and posterior distributions are obtained. A number (distribution) of compatible trees with the observed data are finally given. This methodology is very robust and has the advantage to obtain the possible compatible trees. A popular and very powerful software for inferring phylogenies using a Bayesian method is BEAST (Drummond et al., 2012).

4.5.3 Practical Exercises

The code that is commented here reconstruct a phylogenetic tree using basic clustering methods.

```
## HOW TO RECONSTRUCT A TREE? CLUSTERING METHODS
#From Distance matrix:
#UPGMA. Strict molecular clock
#Steps:  from distance matrix, find the shortest distance, recalculate the matrix and average
the closest, repeat until having 1x1 matrix
#how to keep data and represent data:  matrix with rows indicating the nodes and the distance
between them.
locate.min.lower.matrix <- function(D.tmp) {
   if(is.null(rownames(D.tmp))) rownames(D.tmp) <- c(1:length(D.tmp[,1]))
   if(is.null(colnames(D.tmp))) colnames(D.tmp) <- c(1:length(D.tmp[1,]))
   l <- length(D.tmp[1,])
   if(l<2) stop("too low dimensions")
   min.value <- D.tmp[2,1]
   row.value <- 2
   col.value <- 1
   for(i in 1:(l-1)) { #cols
     for(j in (i+1):l) { #rows
       if(min.value > D.tmp[j,i]) {
         min.value <- D.tmp[j,i]
         row.value <- j
         col.value <- i
       }
     }
   }
   # coordinates of the minimum value in the matrix
   i <- as.numeric(rownames(D.tmp)[row.value])
   j <- as.numeric(colnames(D.tmp)[col.value])
   c(i,j,min.value)
}

do.tree.matrix.upgma <- function(D) {
   n.lineages <- length(D[1,])
   #matrix of all relationships
   tree.matrix <- matrix(0,nrow=2*n.lineages-1, ncol=4,dimnames=list(c(1:(2*n.lineages-1)),
     c("node1","node2","go.to.node","Div")))
   tree.matrix[1:n.lineages,1:2] <- c(1:n.lineages) #init
   cluster <- matrix(0,nrow=n.lineages,ncol=n.lineages+1, dimnames=list(sprintf("line%d",c(1:n.lineages)),
     sprintf("cluster%d",c(0:(n.lineages)))))
   #clustering matrix
```

```
  cluster[1:n.lineages,1:2] <- c(1:n.lineages) #init
  D.tmp <- D
  rownames(D.tmp) <- c(1:length(D.tmp[,1]))
  colnames(D.tmp) <- c(1:length(D.tmp[1,]))
  l <- n.lineages
  n <- 2
  while(length(D.tmp[1,]) > 1) {
    Dmin <- locate.min.lower.matrix(D.tmp) #find the minimum value and their cluster coordinates
    join.clust <- c(which(cluster[,n] == Dmin[1]),which(cluster[,n] == Dmin[2]))
    #locate all lineages in the cluster
    node.number <- n+n.lineages-1
    cluster[join.clust,n] <- node.number #assign a new node to the new cluster
    cluster[,n+1] <- cluster[,n] #copy the column for next step
    #do the new matrix averaging the clustering lineages using the original D matrix
    D.tmp <- matrix(0,l-1,l-1)
    nodes <- sort(unique(cluster[,n])) #find the nodes that are currently present
    colnames(D.tmp) <- t(nodes)
    rownames(D.tmp) <- t(nodes)
    ii <- 1
    for(i in as.numeric(t(nodes))) {
      jj <- 1
      for(j in as.numeric(t(nodes))) {
        D.tmp[jj,ii] <- mean(D[which(cluster[,n]==j),which(cluster[,n]==i)])
        jj <- jj + 1
      }
      ii <- ii + 1
    }
    #D.tmp
    #calculating the new tree matrix
    tree.matrix[node.number,1:2] <- Dmin[1:2]
    nodes.prec <- sort(unique(cluster[which(cluster[,n]==node.number),n-1]))
    tree.matrix[nodes.prec , 3] <- node.number
    tree.matrix[nodes.prec, 4] <- Dmin[3]/2
    n <- n + 1
    l <- l - 1
  }
  #relative divergence for each node
  for(i in (2*n.lineages-2):(n.lineages+1)) {
    tree.matrix[i,4] <- tree.matrix[i,4] - tree.matrix[max(tree.matrix[i,1:2]),4]
  }
  #output a tree.data variable with the nodes and the distances and the clustering matrix
  list(tree.data=as.data.frame(tree.matrix),cluster=cluster[,-length(cluster[1,])])
}

#Finally, do the UPGMA tree using the matrix D
UPGMA.tree <- do.tree.matrix.upgma(D)
UPGMA.tree

#NJ: no molecular clock.  It is the best tree if distance data matrix is additive (d_ij
= d_ix + d_xj)
#Ploting using the libraries "ape" and "phangorn":  library(ape) library(phangorn)
par(mfrow=c(1,1))
UPGMA.D <- upgma(D)
str(UPGMA.D) #arguments in output
UPGMA.D$edge #relationship among nodes
plot(UPGMA.D) #inferred tree with UPGMA

Dd <- as.dist(lower.tri(D) * D)
NJ.D <- nj(Dd) #relationship among nodes
str(NJ.D) #arguments in output
NJ.D$edge
plot(NJ.D,"u") inferred unrooted tree using NJ
```

```
par(mfrow=c(1,1))
plottree(tree.phyl.time.T) #real tree
```

4.6 Support of the phylogenetic tree inferred

There are several methods to contrast the support of the reconstruction: methods that look at the congruence between different datasets, resampling methods like bootstrap or also base-model methods, like parametric bootstrap. Also, some statistical test to compare the compatibility of phylogenetic trees using a likelihood approach are available (see for example Kishino and Hasegawa, 1989). In this practical course, the non-parametric bootstrap is evaluated. The simplest bootstraping method consist in picking columns from the alignment sequence data with replacement and reconstruct a phylogeny with the new dataset. By doing it many times is possible to have an idea of the precision of the phylogeny. Each node is contrasted in relation to the obtained phylogeny, so we have a bootstrap value for each node. A value lower that 98-99% is usually considered not confident.

```
bootstrap.seq.matrix <- function(seq.matrix) {
   L <- length(seq.matrix[1,])
   post <- sample(x=c(1:L),size=L,replace=TRUE)
   seq.matrix <- seq.matrix[,post]
   seq.matrix
}

#recursive function to look for external nodes
look.for.external.nodes <- function(tree.observed,i) {
   external.nodes <- 0
   if(tree.observed$node2[tree.observed$node1[i]] == tree.observed$node1[i]) {
     external.nodes <- c(external.nodes,tree.observed$node1[i])
   } else {
     external.nodes <- c(external.nodes,look.for.external.nodes(tree.observed,tree.observed$node1[i]))
   }
   if(tree.observed$node2[tree.observed$node2[i]] == tree.observed$node1[i]) {
     external.nodes <- c(external.nodes,tree.observed$node2[i])
   } else {
     external.nodes <- c(external.nodes,look.for.external.nodes(tree.observed,tree.observed$node2[i]))
   }
   external.nodes[-1]
}

#function to compare internal tree nodes
compare.tree.nodes <- function(tree.observed,tree.check) {
   n.lineages <- length((tree.observed[,1]+1)/2)    #add a new field
   if(is.null(tree.observed$bootstrap)) tree.observed <- data.frame(cbind(tree.observed,bootstrap=0))
   for(i in (2*n.lineages-2):(n.lineages+1)) { #compare all internal nodes
     internal.nodes.obs <- sort(unique(look.for.external.nodes(tree.observed,i)))
     internal.nodes.chk <- sort(unique(look.for.external.nodes(tree.check,i)))
     if(length(internal.nodes.obs) == length(internal.nodes.chk)) {
       if(sum(internal.nodes.obs != internal.nodes.chk) == 0) {
         tree.observed$bootstrap[i] <- tree.observed$bootstrap[i] + 1
       }
     }
   }
   tree.observed
}
```

```
# DO BOOTSTRAP:
iter <- 1e3
for(i in 1:iter) {
   seq.boot <- bootstrap.seq.matrix(seq)
   #calculate the distance matrix
   D.boot <- diffs.plus.jc69.div.matrix(seq.boot)
   #do symmetric matrix
   for(i in 1:(length(D.boot[1,])-1))
     for(j in (i+1):length(D.boot[,1]))
       D.boot[i,j] <- D.boot[j,i]
   #calculate the number of equal clustered internal nodes in relation to the observed data
   UPGMA.boot <- do.tree.matrix.upgma(D.boot)
   #UPGMA.boot
   tree.data <- compare.tree.nodes(tree.observed=tree.data,tree.check=UPGMA.boot$tree.data)
}
tree.data$bootstrap <- tree.data$bootstrap/iter
tree.data #look at bootstrap support
```

4.7 Phylogenomics: A way to obtain the Species Tree

The phylogeny obtained based on a region of DNA is a gene tree, which may be a close approach to the species tree. Nevertheless, the obtained phylogeny is a particular history of the lineages in the specific region studied. Some nodes in the tree may have low support because the bifurcations are very close one each other and this low resolution may be affected by the particular DNA region studied. This regions are also called anomalous regions because they can be affected by the stochastic divergence of the lineages at the ancestral common population (in phylogeny this is related to the problem of *Incomplete Lineage Sorting*). One way to try to solve this problem is to use a phylogenomics approach, that is, the use of many DNA unlinked regions that allows the estimation of the split between species (not lineages). Essentially there are three approaches: (i) the use of a supermatrix, that is, merge all adta inone and calculate a unique tree; (ii) the calculation of a supertree by mean of many independent trees calculated around all the genome and (iii) the use of likelihood or bayesian approaches to estimate jointly some general parameters and remain others as particular of each locus. Supermatrix and supertrees are not convenient in many cases because they do not consider the heterogeneity across regions or because they inflate the variance and are not consistent methods. More, statistical support is not robust using these methods although they give an approximated idea about the species tree. Methods based on likelihood methods should be used although they are computationally expensive. The review about phylogenetic inference using whole genomes from Rannala and Yang explain some of the problems about phylogenetic reconstruction (Rannala and Yang (2008)).

```
#Calculate a NJ tree for each one using "phybase" library
#Then use the STAR algorithm by mean of star.sptree function in phybase to calculate the
consensus species tree.
#use star.sptree function:  STAR algorithm # Use the data "tree.data" from the previous
exercises

tree.data <- data.frame(cbind(tree.data[,1:6]))
tree.data <- data.frame(cbind(tree.data,Time.seq=0,mutations=0))
mu.vector <- tree.data$mu

n.loci <- 100
NJ.Dl <- array(,n.loci)
locus <- 1
```

```
while(locus <= n.loci) {
   tree.data$Time.seq <- tree.data$Time.sp
   #Calculate times
   for(lg in (n.lineages+1):(2*n.lineages-1)) {
     t1 <- tree.data$Time.seq[tree.data$node1[lg]] #time speciation first ind.
     t2 <- tree.data$Time.seq[tree.data$node2[lg]] #time speciation second ind.
     time.coal <- rexp(n=1,rate=1/tree.data$Ne[lg]) #TIME COALESCENCE both sequences
     tree.data$Time.seq[tree.data$node1[lg]] <- t1 + time.coal #adding time of coalescence
     tree.data$Time.seq[tree.data$node2[lg]] <- t2 + time.coal #adding time of coalescence
   }
   #Calculate number of mutations
   L <- floor(runif(1,400,2000)) #The length is different each locus.
   tree.data$mu <- mu.vector*10^(runif(1,-0.5,0.5)) #mutation is different each locus
   tree.data$mutations <- rpois(n=2*n.lineages-1,lambda=tree.data$mu*tree.data$Time.seq*L)
   Tnt <- c("A","C","G","T")
   seq1 <- sample(x=Tnt,size=L,replace=TRUE,prob=c(0.25,0.25,0.25,0.25))
   seq <- matrix(rep(seq1,2*n.lineages-1),ncol=L,byrow=TRUE)

   #include mutation according to a JC model:
   Q.JC <- matrix(1/3,ncol=4,nrow=4,dimnames=list(Tnt,Tnt)) #Jukes and Cantor rate matrix
   for(x in 1:4) Q.JC[x,x] <- 0; Q.JC[x,x] <- -sum(Q.JC[x,])
   for(node in (2*n.lineages-2):1) {
     seq[node,] <- substitutions.on.sequence.plusI.Gamma(seq=seq[node,], I=0,Gshape=0,
       substitutions=tree.data$mutations[node],Q=Q.JC,Tnt=Tnt)
     seq[tree.data$node1[node],] <- seq[node,]
     seq[tree.data$node2[node],] <- seq[node,]
   }
   #Calculate divergence using JC
   D <- diffs.plus.jc69.div.matrix(seq)
   #Calculate NJ tree
   if(sum(apply(D,c(1,2),is.nan)) == 0) {
     for(i in 1:(length(D[1,])-1)) for(j in (i+1):length(D[,1])) D[i,j] <- D[j,i]
     Dd <- as.dist(lower.tri(D) * D)
     NJ.Dl[locus] <- write.tree(root(phy=nj(Dd),outgroup=6),file="")
     locus <- locus + 1
   }
}

speciesname<-species.name(NJ.Dl[1])
taxaname<-speciesname
species.structure<-matrix(0,ncol=n.lineages,nrow=n.lineages)
diag(species.structure)<-1

#Finally, collect all trees and calculate the species tree
multilocus.tree <- star.sptree(NJ.Dl, speciesname, taxaname, species.structure,outgroup="6",method="nj")
#Plot the real and estimated trees
par(mfrow=c(2,1))
plot(tree.phyl.time.T)
plottree(multilocus.tree)
```

5 Final Exercise

Download, in fasta format, the dataset for the complete mitochondrial genome of a single individual of the several available primate species (Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates) and one outgroup (*Sus scrofa*). Here, use *Homo sapiens, Pan troglodytes, Lemur catta* and *Papio hamadryas*. Do a multiple alignment (for example using Tcoffe or MAFFT). Filter the poorly aligned positions and divergent regions (*e.g.* using Gblocks) and then estimate the best evolutionary model (for example using jModelTest). Do a ML phylogenetic reconstruction (*e.g.* using PhyML). Calculate bootstrap support for . Modify the plot using FigTree, iTol or another application and include the length of the branches and the bootstrap values.
Explain clearly the sequences downloaded, the methods used and the meaning of each analysis performed (multiple alignment method, filtering data, results of choosing the evolutionary model, results of the ML analysis). The information must be enough to replicate the analysis and understand why each analysis was performed. Evaluate the tree (divergence values, evolutionary model, molecular clock and bootstrap support). The results of this exercise must be presented in a PDF file containing a method section, the results (a tree) and a short discussion section evaluating the results.

6 References

Castresana, J. (2000, Apr). Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Mol Biol Evol 17*(4), 540–552.

Cavalli-Sforza, L. L. and A. W. Edwards (1967). Phylogenetic analysis. models and estimation procedures. *Am J Hum Genet 19*(3 Pt 1), 233–257.

Darriba, D., G. L. Taboada, R. Doallo, and D. Posada (2012, Aug). jmodeltest 2: more models, new heuristics and parallel computing. *Nat Methods 9*(8), 772.

Drummond, A. J., M. A. Suchard, D. Xie, and A. Rambaut (2012). Bayesian phylogenetics with beauti and the beast 1.7. *Mol Biol Evol 29*(8), 1969–1973.

Gascuel, O. (1997). Bionj: an improved version of the nj algorithm based on a simple model of sequence data. *Mol Biol Evol 14*(7), 685–695.

Higgs, P. and T. Attwood (2005). *Bioinformatics and Molecular Evolution*. Blackwell Publishing.

Kishino, H. and M. Hasegawa (1989). Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from dna sequence data, and the branching order in hominoidea. *J Mol Evol 29*(2), 170–179.

Liu, L. and L. Yu (2010, Apr). Phybase: an r package for species tree analysis. *Bioinformatics 26*(7), 962–963.

Liu, L., L. Yu, D. K. Pearl, and S. V. Edwards (2009, Oct). Estimating species phylogenies using coalescence times among sequences. *Syst Biol 58*(5), 468–477.

Posada, D. and K. A. Crandall (2001, Aug). Selecting the best-fit model of nucleotide substitution. *Syst Biol 50*(4), 580–601.

Rannala, B. and Z. Yang (2008). Phylogenetic inference using whole genomes. *Annu Rev Genomics Hum Genet 9*, 217–31.

Saitou, N. and M. Nei (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol 4*(4), 406–425.

Sokal, R. and P. Sneath (1963). *Principles of Numerical Taxonomy*. Freeman & Co. San Francisco.

Song, S., L. Liu, S. V. Edwards, and S. Wu (2012, Sep). Resolving conflict in eutherian mammal phylogeny using phylogenomics and the multispecies coalescent model. *Proc Natl Acad Sci U S A 109*(37), 14942–14947.

Talavera, G. and J. Castresana (2007, Aug). Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. *Syst Biol 56*(4), 564–577.

Yang, Z. (2006). *Computational Molecular Evolution*. Oxford Series in Ecology and Evolution. Oxford University Press, New York.