```
#Charger le package "ape" permettant de travailler avec des données phylogénétiques
#sur R.

library(ape)

#Charger le bon répertoire courant dans l'onglet fichier (soit le fichier où est enregistré le fichier
#texte contenant les données relatives à l'arbre phylogénétique étudié.

arbre<-read.tree("tree")

plot(arbre,main="Arbre phylogénétique")

#Commande donnant un vecteur contenant la longeur de chaque branche de l'arbre (temps):

t<-branching.times(arbre)

#Code pour la fonction de base; l=taux de spéciation, m=taux d'extinction et
#t=temps:

p<-function(l,m,t) {
num<-(l-m)^2*exp((l-m)*t)
den<- (l*exp((l-m)*t)-m)^2
return(num/den)}

#Code pour la fonction contenant les exceptions;
#-première exception: si l=m.
#-seconde exception: dans le cas où l=m et où l'on obtient NaN ou Infinite.

prob<-function(l,m,t) {
if(m==l) {
like<-log(1/((1+l*t)^2))}
else{ like<-log(p(l,m,t))}
if(is.nan(like)| is.infinite(like)) {return(-1000000000)}
else  { return(like)}
}

#Code pour la fonction complète; T=temps entre temps zéro et le premier ancêtre commun à toutes
#les espèces considérées dans l'arbre mais ne figurant pas sur celui-ci:

vrais<- function(l,m,t,T) {
a<- prob(l,m,T) + sum(sapply(t, prob, l=l, m=m)) + length(t)*log(l)
return(a)}

#Création d'une matrice contenant les valeurs de l et m que nous souhaitons tester:

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)#la commande length.out=x stipule que la longeur de la séquence générée doit être de x
m<-l

#Création d'une boucle afin de d'insérer la matrice dans la formule du maximum de vraissemblance:
```

```
for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)#Il faut avoir défini T (valeur fournie avec l'arbre)
}}

#Graphes:

contour(l,m,mat,xlab="l",ylab="m",main="Représentation en relief de la vraisemblance
des valeurs de l et m concernant arbre",col="blue")

persp(l,m,mat,main="Représentation tridimensionnelle de la vraisemblance
des valeurs de l et m concernant arbre",col="green",theta=30,phi=30)


#Génétation d'arbres pour lesquels on peut faire varier l, m, T ainsi que le nombre d'espèces
#afin de constater l'influence de ces paramètres sur les temps t (commande fournie par Nicolas Salamin):

createTree<-function(t=1, l=2, m=1, spp=20, plot=FALSE) {
  if(l<m) {
    stop("Pas d'arbre possible. Le taux de spéciation est plus petit que le taux d'extinction.\n",call.=FALSE);
  }

  if(l==0) {
    stop("Pas d'arbre possible. Le taux de spéciation est de zéro.\n",call.=FALSE);
  }

  require(ape);
  tree<-rtree(spp);
  nonodes<-(spp*2)-1;
  times<-genTimes(t, l, m, spp-1);

  j<-2;
  set<-c(1:spp,(spp+2):nonodes);
  for(i in set) {
    k<-which(tree$edge[,2]==i);
    if(i>spp) {
      tree$edge.length[k]<-times[j];
      j<-j+1;
    }
    else {
      tree$edge.length[k]<-0;
    }
  }

  edge<-tree$edge.length;
  for(i in 1:(nonodes-1)) {
    j<-tree$edge[i,1]; #it's the ancestor, I need to find which line as j as descendant to get its branch length

    if(j==(spp+1)) { #if j is the root, its time is times[1]
      anc<-times[1];
    }
```

```
    else { #otherwise, search the corresponding line
      k<-which(tree$edge[,2]==j);
      anc<-edge[k];
    }

    tree$edge.length[i]<-anc-edge[i];
  }

  if(plot) {
    plot(tree);
    axisPhylo();
  }

  return(list(tree=tree, T=times[1]));
}

genTimes<-function(t=1, l=2, m=1, spp=20) {
  if(l==m) {
    l<-l+0.000000001;
  }

lt<-l*t;
mt<-m*t;

myexp<-exp(lt-mt);

a<-0;
b<-(myexp - 1)/(l*myexp - m);

times<-sort(runif(spp, a, b), decreasing=TRUE);

  return((1/(l-m))*log((1-m*times)/(1-l*times)));
}

#Commande pour générer de tels arbres:

createTree(t=x, l=y, m=z, spp=w, plot=T)

#l=taux de spéciation, m=taux d'extinction, spp=nombre d’espèces
#observées au temps zéro, t=temps séparant le temps zéro du premier
#nœud ancestral n’étant pas compris dans l’arbre considéré (soit T)

#Commande pour faire des graphes à partir d'arbre générés par la fonction createTree:

arbre<-createTree(t=x, l=y, m=z, spp=w, plot=T)

T<-x

t<-branching.times(arbre$tree)

#...->suite de la procédure calssique avec les formules du maximimum de vraisemblance,
```

```
#la matrice et la boucle, puis les les graphes persp() et contour().

#Commande permettant d'obtenir, au moyen d'un processus itératif (Non Linear Minimization)
#les valeurs les plus probables de l et m pour expliquer les valeurs de t et T que nous
#commaissions initialement:

vrais2<-function(x){
return(-vrais(x[1],x[2],t,T))}

#la fonction nlm() de R donne par défaut un minimum. Comme dans notre cas nous
#recherchon un maximum, il nous faut donc prendre le résultat opposé, c'est
#pourquoi on introduit un signe "-".

nlm(vrais2,c(d,e))

#d et e représentent des valeurs numériques approximatives obtenues sur la base
#du graphe contour() pour respectivement l et m les plus probables.
```