Compte rendu du projet

Calcul des taux de spéciation et d'extinction

Charger la bibliothèque « ape » permettant de travailler avec des données phylogénétiques sur « R »[1].

```
library(ape)
```

Diriger « R » vers le répertoire contenant le(s) fichier(s) texte de l'arbre phylogénétique étudié (Fichier→Changer le répertoire courant).

Charger le fichier texte (par exemple « treefile ») contenant les données relatives à l'arbre phylogénétique que l'on souhaite étudier en lui donnant un nom (par exemple « arbre »).

```
arbre<-read.tree("tree")
```

Définir le vecteur (nommé par exemple « t ») contenant les temps propre à chacune des branches de notre arbre (pour être exact, ce vecteur contient les distances séparant chaque nœud de notre arbre du sommet des branches de ce dernier, autrement dit les distances séparant chaque nœud du temps zéro).

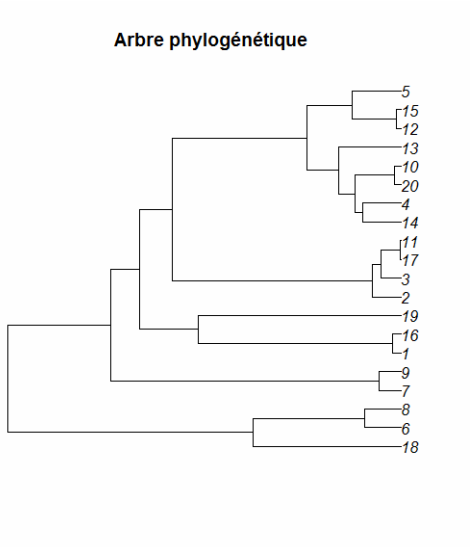
> La comparaison de séquences ADN appartenant à des organismes d'espèces différentes peut être utilisée dans le but d'estimer l'âge relatif du dernier ancêtre commun à ces espèces. On part alors de principe que les substitutions nucléotidiques survenues entre ces espèces se sont accumulées au hasard et de manière régulière si bien que leur nombre est proportionnel au temps s'étant écoulé depuis l'événement de spéciation.

```
t<-branching.times(arbre)
```

On peut maintenant réaliser une représentation graphique de notre arbre.

[1] Il est possible, dans le but de gagner un peu de temps, de créer un script source (fichier de type R) contenant les commandes principales que nous utilisons (comme les trois fonctions et le chargement de la librairie ape) et de le faire lire par « R » lorsque nous ouvrons la console (Fichier→Sourcer du code R…).

```
plot(arbre,main="Arbre phylogénétique")
```



Commande pour la fonction de base (nommée « p(l,m,t) ») que nous allons utiliser et qui est celle relative au processus dit de « Birth and Death ». Dans cette formule, l est défini comme le taux de spéciation, m comme le taux d'extinction et t comme les temps séparant chaque nœud de notre arbre du sommet des branches de ce dernier (à savoir ceux contenu dans notre vecteur « t »). Relevons que, quoi qu'il arrive, ces trois variables sont comprises dans l'ensemble [0 ;∞[.

> Birth and Death Process : $p_1(t)=[(l-m)^2 \times e^{(l-m)t}] / [(l \times e^{(l-m)t} - m)^2]$ → donne la probabilité que pour un temps t une espèce donnée ne connaisse ni événement de spéciation ni événement d'extinction.

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

Commande pour la fonction prenant en compte les exceptions mathématiques pouvant se produire dans notre fonction de base (et nommée « prob(l,m,t) »). Principalement il faut prendre en compte deux cas particulier :

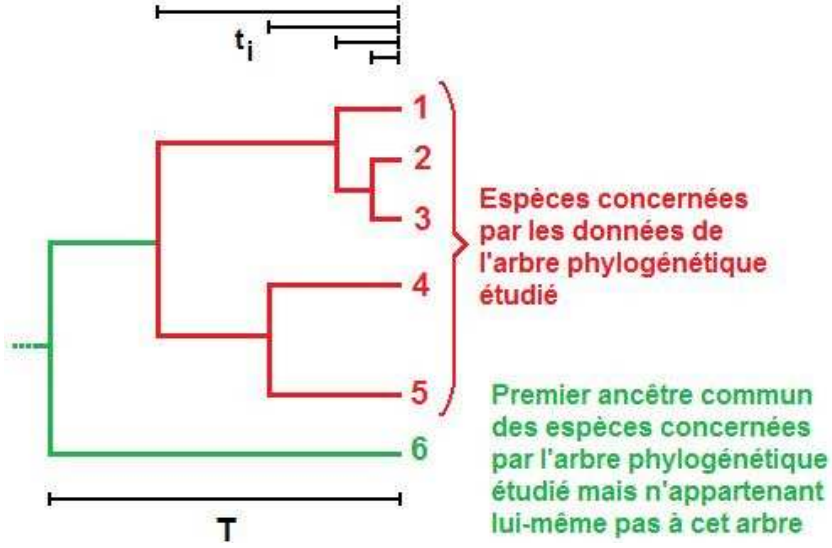
(a) Si l=m, alors le dénominateur vaut zéro. Dans ce cas, on souhaite utiliser une version simplifiée de la formule de processus de Birth and Death (nommée « like »).
(b) Si l=m et que le résultat obtenu est un « NaN » ou un « Infinite », alors on souhaite que le programme nous retourne une très grande valeur négative dans le but d'obtenir au final un probabilité nulle.

Suivant les valeurs de l, m et t introduites dans notre fonction de base, on peut obtenir des valeurs extrêmes (très petites et très grandes) qui sont de ce fait difficilement comparables et tout aussi difficilement représentables. C'est pourquoi on introduit dans cette fonction un logarithme en base dix.

```
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)}
}
```

Commande pour la formule dite du maximum de vraisemblance (Maximum likelihood estimation of speciation rates). Cette formule nous donne la probabilité que pour un l et un m donné le processus d'évolution ait effectivement produit le nombre d'espèces que l'on a dénombrées au temps présent (ou temps zéro, soit le sommet des branches de l'arbre). Une nouvelle variable apparaît dans cette formule, il s'agit de la variable T (cf. illustration ci-dessous). T représente la distance (~temps) séparant le sommet des branches de notre arbre (à savoir le temps zéro) du premier nœud ancestral n'étant pas compris dans l'arbre en question (la valeur de T nous est en principe transmise avec l'arbre étudié –et figure dans le fichier texte correspondant– et il est alors nécessaire de la définir dans « R »).



Maximum likelihood estimation of speciation rates (où a=nombre de taxa observé au temps présent et que l'on peut lier à la variable t de la manière suivante ; t_i=a-1):

$prob(a, t_1, t_2,\ldots, t_{a-1}| l, m, T) = p_1(T) \times l^{(a-1)} \times \mathbf{\Pi} p_1(t_i)$

Dans notre cas, l'introduction d'un logarithme entraîne la modification du produit en somme et des exposants en facteurs multiplicatifs :

$\log_{10}[prob(a, t_1, t_2,\ldots, t_{a-1}| l, m, T)]= p_1(T) + \mathbf{\Sigma} p_1(t_i)+ l(a-1)$

```
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)}
```

Nous avons maintenant codé la fonction d'intérêt (Maximum likelihood estimation of speciation rates), et nous connaissons les variables t et T. Le but est donc de générer une matrice contenant toute les combinaisons des valeurs de l et m que nous souhaitons tester afin de l'introduire dans notre fonction `vrais(l,m,t,T)` . La probabilité maximale obtenue suite à l'insertion de cette matrice dans la fonction nous donnera les valeurs de l et de m les

plus probables pour expliquer le nombre d'espèces dénombrées au temps présent ainsi que les différents t trouvés suite aux séquençages ADN. Il est important ici de noter que cette matrice est essentielle à la représentation graphique des données générées par notre fonction `vrais()`, en revanche elle n'est pas nécessaire au fonctionnement des commandes présentées par la suite et qui permettent de réaliser une estimation mathématiques des valeurs de l et m les plus probables (à savoir les fonctions `nlm()`, `optimum()` et `optimum2()`).

Création d'une matrice 100x100 et définitions des valeurs de l et m que nous souhaitons tester[2] (en l'occurrence nous allons tester des taux de spéciation et d'extinction allant de 0.1 à 10 en allant par pas de 0.1 afin d'obtenir des vecteurs de l et m ayant la même taille que notre matrice). Il est logique de noter que plus le nombre des l et m testé ainsi que la précision de ces valeurs (soit la longueur du pas introduite) est importante et plus les valeurs de l et de m obtenues par le maximum de vraisemblance pourront être considérées comme étant exactes.

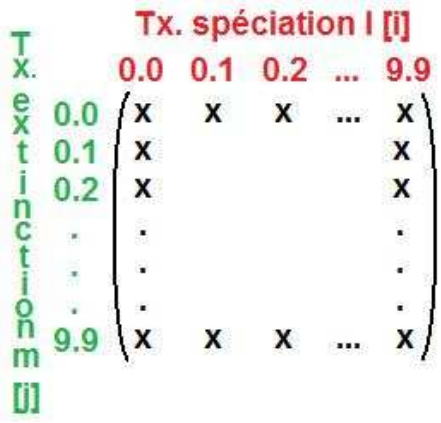
```
mat<-matrix(numeric(10000),ncol=100)

l<-seq(0.1,10,0.1)
m<-l
l<-seq(0.1,10,length.out=100)
m<-l                                    [ Vérif. :    length(l)
                                                      [1] 100 ]
```

> La commande `length.out=x` utilisée ici permet de stipuler que l'on souhaite que la séquence générée soit de longueur `x`.

Création d'une boucle permettant de tester à l'aide de notre fonction du maximum de vraisemblance toutes les combinaisons des valeurs l et m présentes dans la matrice précédemment crée.

```
for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)}}
```

Pour résumer, on a maintenant une matrice 100x100 se présentant ainsi :

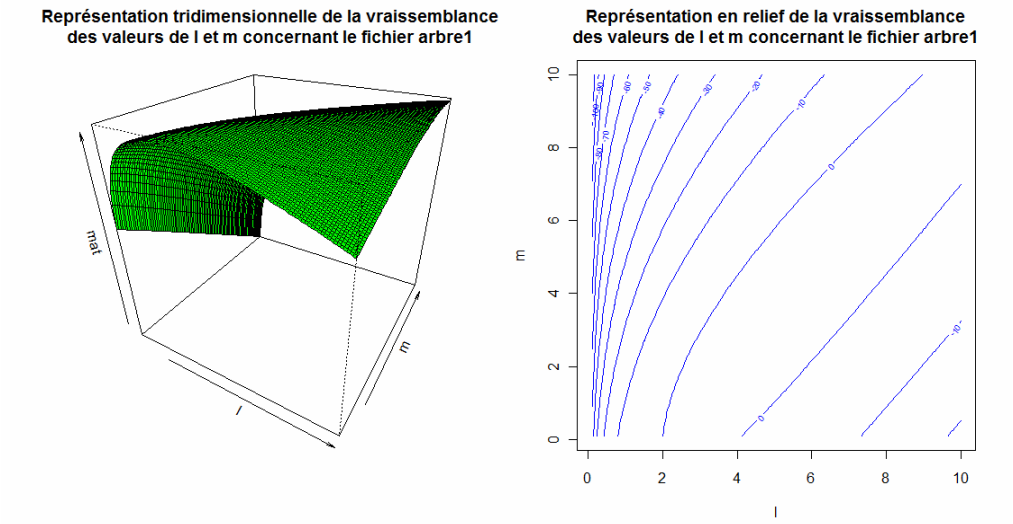


[2] Il est important de préciser ici que des taux de spéciation et d'extinction nulles ne sont biologiquement pas concevables. De ce fait, les valeurs l=0 et m=0 ne sont jamais testées.

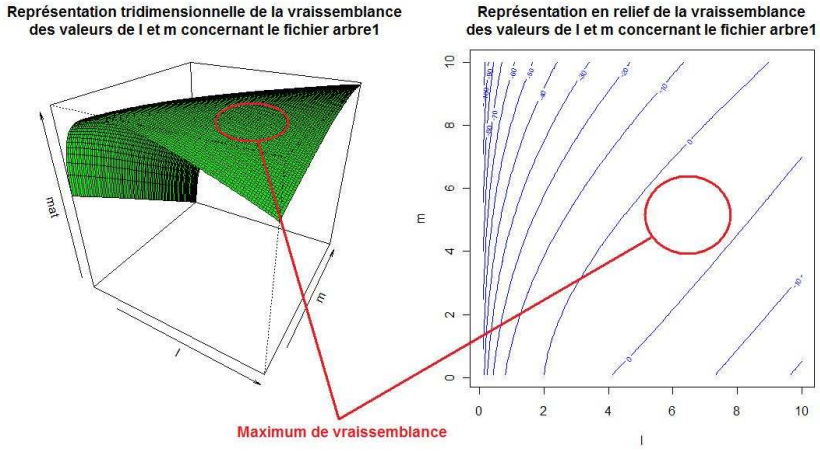
Où x_{ij} est la probabilité avec laquelle les valeurs des taux de spéciation l et d'extinction m correspondants expliquent les temps t (autrement dit x_{ij} est la vraisemblance de l_i et m_j).

On peut donc représenter ces valeurs (l, m et la vraisemblance qui y est maintenant associée) à l'aide de graphiques. Le premier graphe représente la surface de distribution des valeurs de vraisemblance en fonction de l et m. Le second est une représentation « en relief » (analogie avec les courbes de niveaux d'une carte topographique) où les valeurs de vraisemblance sont indiquées le long des courbes du graphe.

```
persp(l,m,mat,main="Représentation tridimensionnelle de la vraisemblance des valeurs de l et m concernant le fichier arbre1",col="green",theta=30,psi=30)

contour(l,m,mat,xlab="l",ylab="m",main= "Représentation en relief de la vraisemblance des valeurs de l et m concernant le fichier arbre1",col="blue")
```



A partir de ces représentations, on peut donc faire une estimation graphique des valeurs de l et m correspondantes au maximum de vraisemblance.



Les commandes suivantes vont nous permettre de passer d'une estimation graphique des valeurs de l et m correspondants au maximum de vraisemblance à des valeurs calculées mathématiquement. Pour ce faire, une des méthodes envisageable aurait été de calculer la dérivée de la formule donnant le maximum de vraisemblance et de l'égaliser à zéro afin de trouver les maximum de notre surface de distribution de probabilité (point de la surface présentant une pente nulle). Malgré le fait que cette méthode ait l'avantage de fournir des résultats exacts, elle s'avère difficilement applicable à notre cas (dérivation trop complexe). Nous avons donc opter pour l'utilisation d'un processus plus approximatif, à savoir l'application d'une méthode itérative reposant sur l'algorithme dit de « la méthode de Newton[3] ».

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

Cette première commande est nécessaire pour stipuler à R que l'on connaît les variables t et T (elles ont été définies précédemment) et que l'on souhaite qu'il nous retourne, lorsque l'on utilisera la fonction `vrais2()`, un vecteur de deux valeur `x[1],x[2]` qui seront les valeurs correspondant aux l et m qu'il aura calculé. Par la suite, la seconde commande que nous allons utiliser repose sur une méthode de minimisation itérative (Non Linear Minimization), or nous recherchons des maximums. C'est pourquoi il est important de définir `vrais2()` comme l'opposé de `vrais()`.

Cette seconde commande est l'application par R de la méthode de Newton, et repose sur la fonction `nlm()` (Non Linear Minimization). Dans cette commande, les deux valeurs numériques sont en fait les approximations obtenues graphiquement de respectivement l et m.

```
nlm(vrais2,c(6,4))
$minimum
[1] -2.928932

$estimate
[1] 5.766176 4.496907

$gradient
[1]  1.173420e-08 -1.210728e-08

$code
[1] 1

$iterations
[1] 9
```

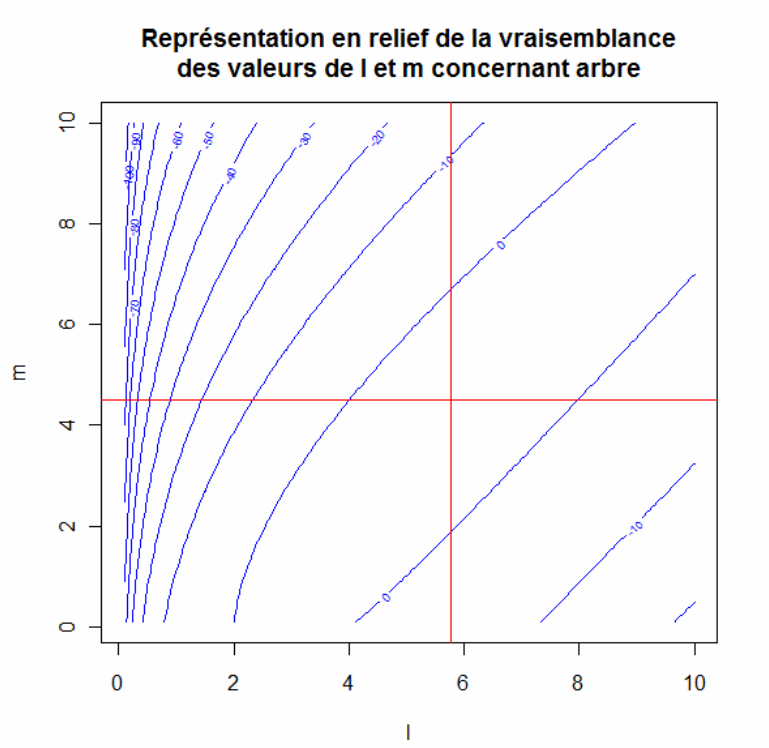
Le résultat obtenu sous `minimum` est en fait le logarithme de la probabilité que les valeurs de l et m calculées par la fonction `nlm()` soient effectivement les plus vraisemblables (il

[3] Algorithme efficace pour trouver des approximations d'un zéro d'une fonction d'une variable réelle à valeurs réelles. L'algorithme consiste à linéariser une fonction f en un point et de prendre le point d'annulation de cette linéarisation comme approximation du zéro recherché. Partant d'une valeur approximative raisonnable d'un zéro d'une fonction d'une variable réelle, on approxime au premier ordre la fonction par sa tangente en ce point. Cette tangente est une fonction affine dont on sait trouver l'unique zéro (analyse élémentaire). Ce zéro de la tangente sera généralement plus proche du zéro de la fonction. Par cette opération, on peut donc espérer améliorer l'approximation par itérations successives. *(Wikipedia)*

convient donc d'en prendre l'exponentiel pour obtenir la valeur exact de cette probabilité). Ces valeurs de l et m sont données dans l'ordre sous `estimate`.

On peut maintenant vérifier la pertinence de ce résultat en traçant sur le graphe `contour()` correspondant à l'arbre duquel découlent ces calculs des droites passant par les l et m obtenus.

```
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")

abline(h=4.496907,v=5.766176,col="red")
```



Afin de simplifier quelque que peu les choses, il est possible de compiler les fonctions `vrais2()` et `nlm()` en une seule fonction baptisée optimum :

```
optimum<-function(time,T,start) {
vrais2<-function(x){
return(-vrais(x[1],x[2],time,T))}
return(nlm(vrais2,start))
}
```

Il existe également une autre fonction de R utilisant aussi la méthode de Newton et qui permet de réaliser une estimation mathématique des valeurs de l et m les plus vraisemblables. Il s'agit de la commande `optim()` qui repose sur l'algorithme de Nelder.Mead et qui permet de minimiser une fonction dans un espace à plusieurs dimensions. Cette fonction, couplée à notre fonction `vrais2()`, permet donc d'obtenir (approximativement) le même résultat que celui donné par la commande `optimum()`. Nous avons regroupé ces commandes dans la fonction baptisée `optimum2()` :

```
optimum2<-function(time,T,start) {
vrais2<-function(x){
return(-vrais(x[1],x[2],time,T))}
return(optim(start,vrais2,method="L-BFGS-B",lower=c(0,0)))
}
```

Dans cette nouvelle fonction, l'argument `time` correspond au vecteur des longueurs de branches, `T` représente toujours la distance (~temps) séparant le sommet des branches de notre arbre (à savoir le temps zéro) du premier nœud ancestral n'étant pas compris dans l'arbre en question, et `start` est un vecteur de forme `c(x,y)` où `x` et `y` sont respectivement des estimations graphiques des valeurs de l et m.

Si on applique cette nouvelle fonction à l'arbre auquel nous nous sommes intéressé jusqu'à maintenant, nous obtenons :

```
optimum2(t,T,c(6,4))
$par
[1] 5.766176 4.496908

$value
[1] -2.928932

$counts
function gradient
      13       13

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Le résultat obtenu sous `$value` est en fait le logarithme de la probabilité que les valeurs de l et m calculées par la fonction `optimum2()` soient effectivement les plus vraisemblables (il convient donc d'en prendre l'exponentiel pour obtenir la valeur exact de cette probabilité). Ces valeurs de l et m sont données dans l'ordre sous `$par`.

Parenthèse sur la génération d'arbres avec « R »

La possibilité de générer des arbres phylogénétiques à partir de paramètres choisis mais selon un processus aléatoire devrait nous permettre de mieux comprendre l'influence des variables l (taux de spéciation), m (taux d'extinction), spp (nombre d'espèces observées au temps zéro) et T (temps) sur les mécanismes évolutifs.

Voici tout d'abord les commandes que nous a transmis Nicolas Salamin et qu'il est nécessaire d'introduire dans « R » afin de pouvoir utiliser la fonction permettant de générer des arbres phylogénétiques variables (possibilité de sourcer ces lignes de codes si un fichier de type R les contenant existe à l'aide du chemin Fichier→Sourcer du code R…).

```
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)));
}
```

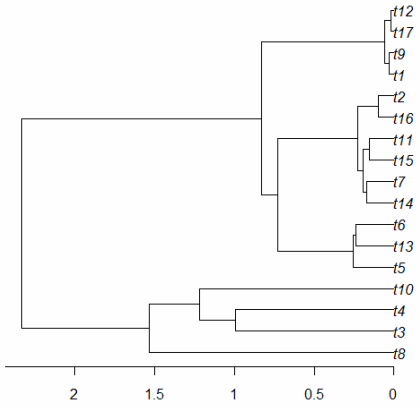
On peut maintenant générer des arbres phylogénétiques en faisant varier les variables l (taux de spéciation), m (taux d'extinction), spp (nombre d'espèces observées au temps zéro) et t[4] (temps séparant le temps zéro du premier nœud ancestral n'étant pas compris dans l'arbre considéré).

```
createTree(t=…, l=…, m=…, spp=…, plot=T)
```

Exemple :

```
createTree(t=4, l=2, m=1.3, spp=17, plot=T)
$tree

Phylogenetic tree with 17 tips and 16 internal nodes.

Tip labels:
        t8, t3, t4, t10, t5, t13, ...

Rooted; includes branch lengths.

$T
[1] 2.336935
```

[4] Attention, contrairement aux codes que nous avons écrits concernant le « Birth and Death process » ainsi que le « Maximum likelihood estimation of speciation rates » pour lesquels la variable t représentait les temps séparant chaque nœud de notre arbre du sommet des branches de ce dernier, dans la commande « createTree » la variable t représente le temps séparant le premier nœud ancestral n'étant pas compris dans l'arbre considéré du temps zéro (il s'agit donc de l'équivalent de notre T).



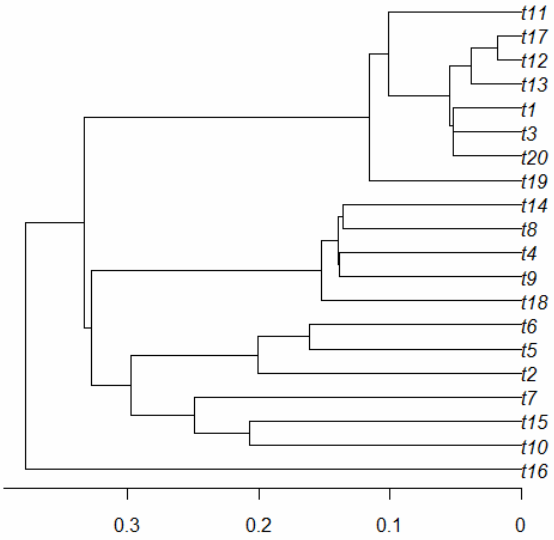
Comme nous sommes maintenant en mesure de générer des arbres pour des valeurs de l et m déterminées, nous allons pouvoir tester si la fonction `optimum2()` que nous avons encodées dans R permet de réaliser des estimations fiables des valeurs de l et m.

```
arbre<-createTree(t=2, l=5, m=2, spp=50, plot=T)
T<-2
t<-branching.times(arbre$tree)
start<-c(5,2)
optimum2(t,T,start)
$par
[1] 5.376554 3.133004

$value
[1] -14.73131

$counts
function gradient
      11       11

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

On voit donc que la fonction `optimum2()` permet de calculer l et m de manière relativement fiable. En effet, les valeurs qu'elle nous retourne, à savoir 5.376554 et 3.133004 pour respectivement l et m, sont proches de celles encodées avec la fonction `createTree()`, à savoir 5 et 2.

Illustration de l'influence de l et m sur la structure des arbres phylogénétiques

I. `createTree(t=4, l=5, m=0.1, spp=20, plot=T)`



Si l est beaucoup plus grand que m, alors les événements de spéciations ont du se produire relativement récemment. En effet, si tel n'était pas le cas et que les premières spéciations s'étaient produites beaucoup plus en arrière, de nombreux autres événement similaires auraient eu le temps de se produire générant ainsi un nombre significativement plus grand d'espèces observables au temps zéro que celui effectivement observé.

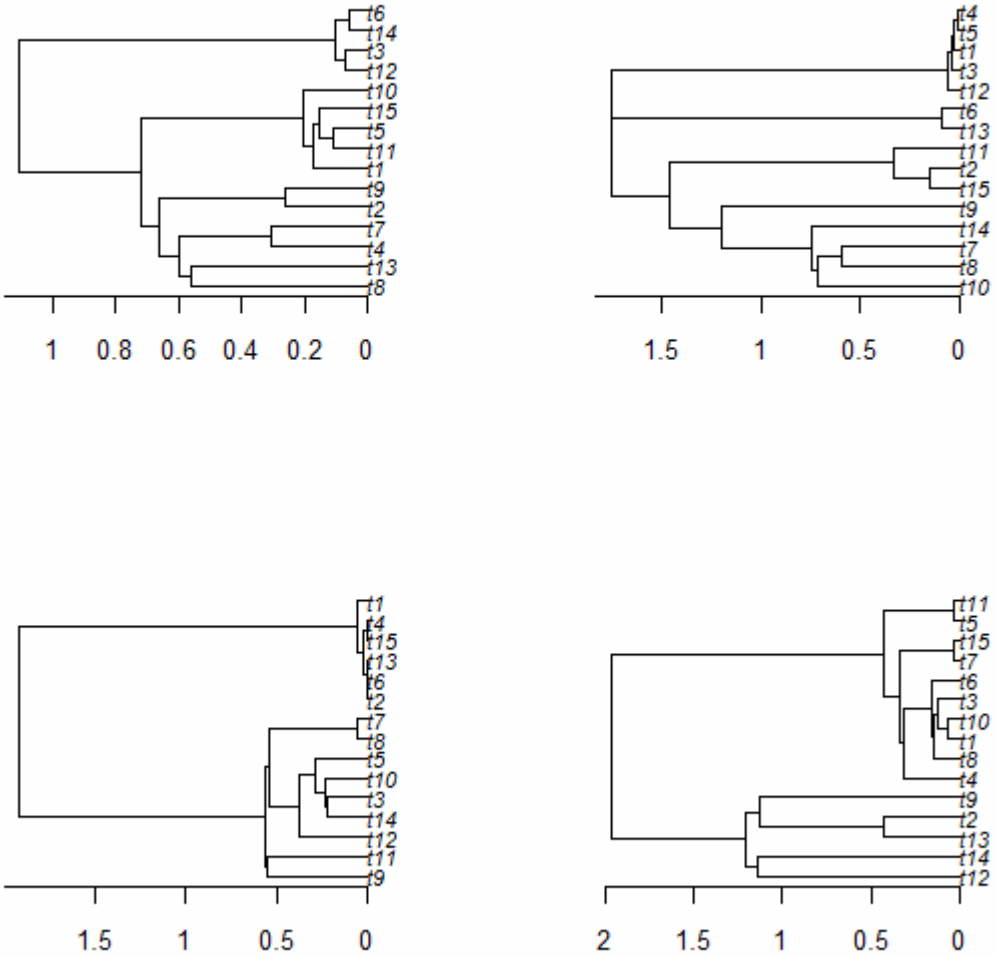
II. `createTree(t=4, l=1, m=3, spp=20, plot=T)`

```
Erreur : Pas d'arbre possible. Le taux de spéciation est plus petit que le
taux d'extinction.
```

IV. `createTree(t=4, l=0, m=0, spp=20, plot=T)`

```
Erreur : Pas d'arbre possible. Le taux de spéciation est de zéro.
```

V. `createTree(t=2, l=3, m=1.5, spp=15, plot=T)`



Il est intéressant de remarquer qu'en générant plusieurs arbres à l'aide de la même commande, on obtient des arbres différents. Cela met en évidence l'aspect stochastique des processus d'évolution.

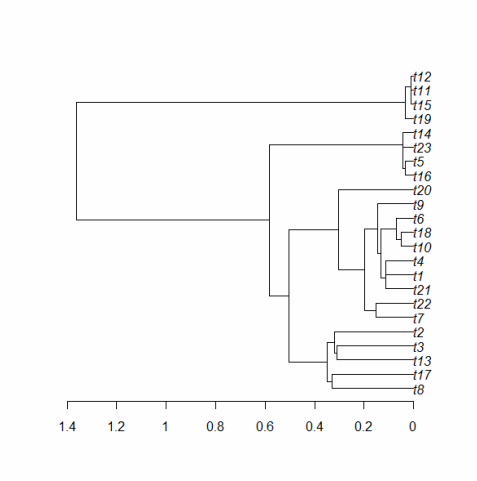
Illustration de l'influence du nombre d'espèces considérées sur l'estimation des taux de spéciation et d'extinction

Il est également possible, à partir d'un arbre phylogénétique généré à l'aide de la fonction `createTree()`, d'utiliser les valeurs de t qui en sont issues afin de les introduire dans notre système du maximum de vraisemblance. Voici quelques exemples :

I.

```
arbre<-createTree(l=3,m=1.5,t=2,spp=23,plot=T)
```

```
arbre
$tree

Phylogenetic tree with 23 tips and 22 internal nodes.

Tip labels:
        t8, t17, t13, t3, t2, t7, ...

Rooted; includes branch lengths.

$T
[1] 1.363510
```

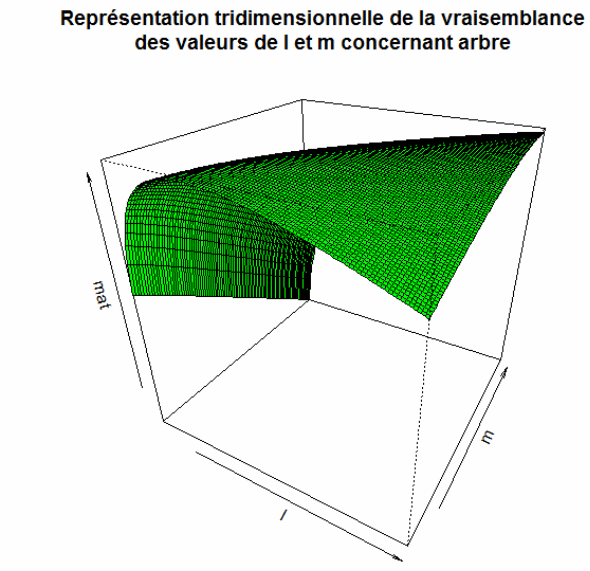


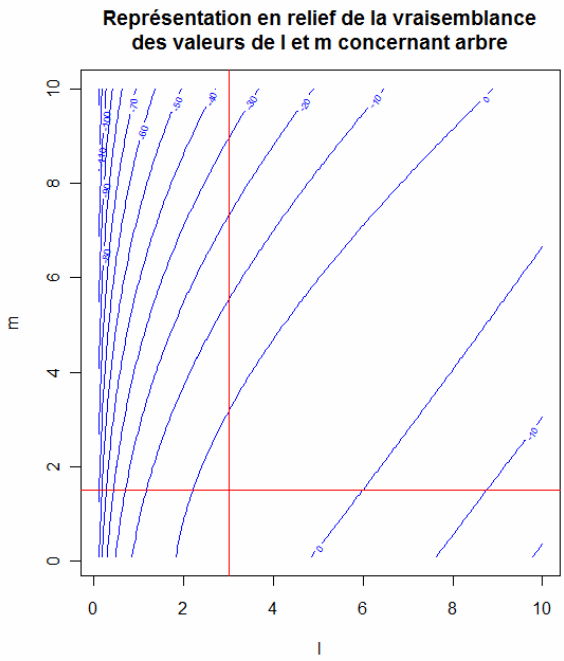

```
arbre$tree

Phylogenetic tree with 23 tips and 22 internal nodes.

Tip labels:
        t8, t17, t13, t3, t2, t7, ...

Rooted; includes branch lengths.

t<-branching.times(arbre$tree)
t
        24          25          26          27          28          29
30
1.36350979  0.58223469  0.50548960  0.35123400  0.32862170  0.32109104
0.30888106
        31          32          33          34          35          36
37
0.30502023  0.19750356  0.15373642  0.14509655  0.13166369  0.11246129
0.11139910
        38          39          40          41          42          43
44
0.07031668  0.04927957  0.04364270  0.04237863  0.03325612  0.03285059
0.01152181
        45
0.01145393
```

A ce stade, nous avons donc le vecteur t correspondant à l'arbre que nous avons généré, et nous connaissons également la valeur de T puisque c'est nous-même qui l'avons définie lors de l'utilisation de la fonction `createTree()`. On peut donc produire les graphes

correspondant à cet arbre à l'aide des formules que nous avons encodées concernant le maximum de vraisemblance.

```
T<-2

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100
m<-l


for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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

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")

abline(h=1.5,v=3,col="red")
```



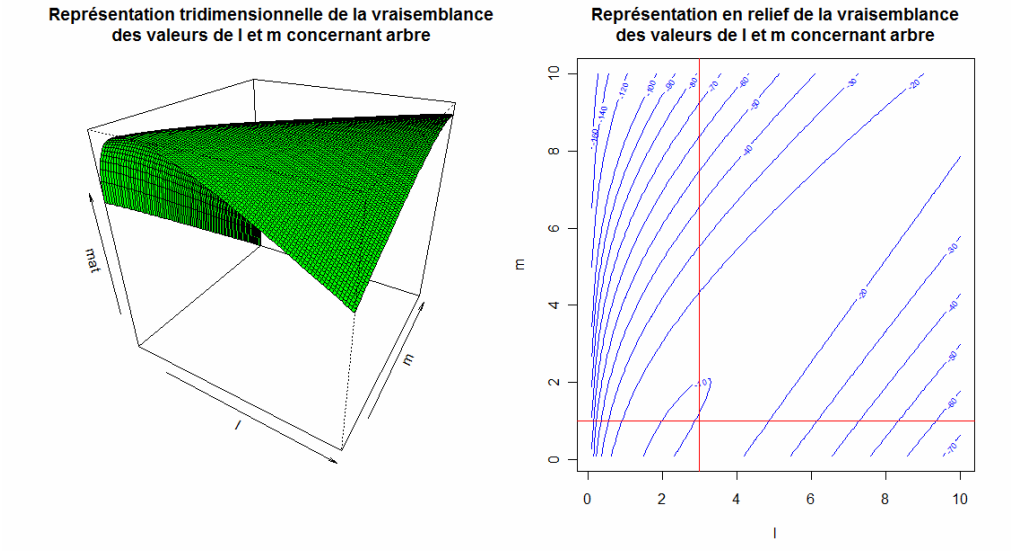


Lors de la génération de cet arbre avec la commande `createTree()`, nous avons nous-même défini les valeurs de m et l. Sachant cela et à l'aide de la commande `abline()`, nous pouvons donc représenter sur la graphique `contour()` la position réelle du maximum de

vraisemblance (intersection des droites rouges) et le comparer au résultat approximatif que nous fourni ce graphe (courbes bleues).

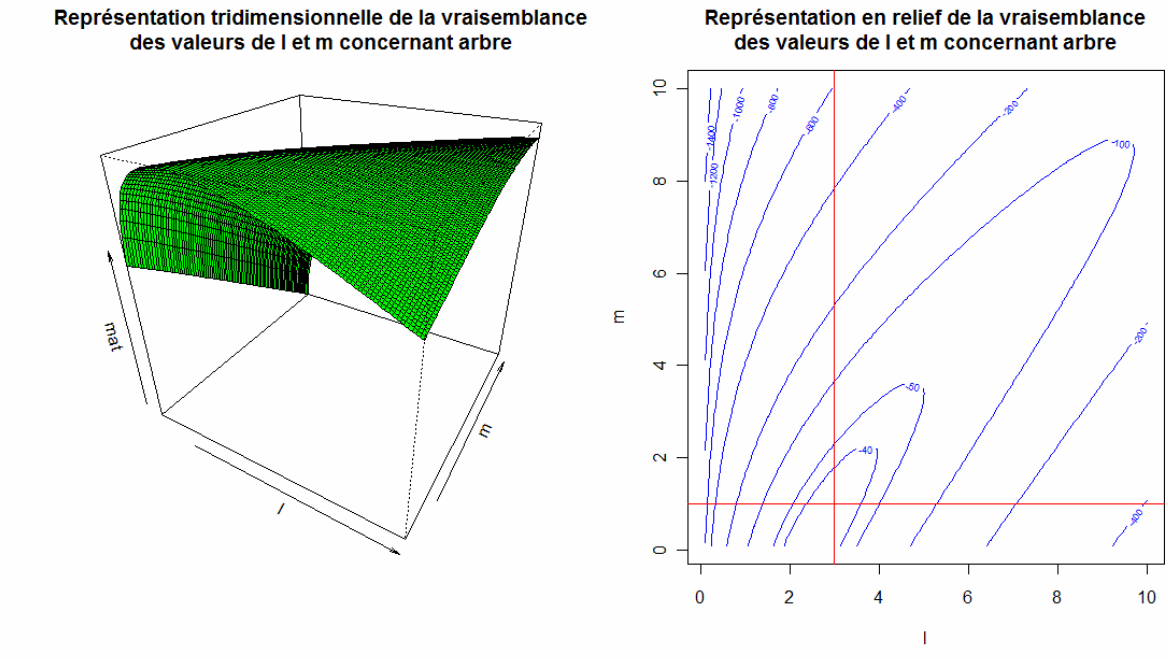
II.

```
arbre<-createTree(l=3,m=1,spp=25,plot=T,t=2)

T<-2

t<-branching.times(arbre$tree)

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)
m<-l

for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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

v<-c(0,-2,-4,-6,-8,-10,-20,-30,-40,-50,-60,-70,-80,-90,-100,-
120,-140,-160)

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",levels=v)

abline(h=1,v=3,col="red")
```



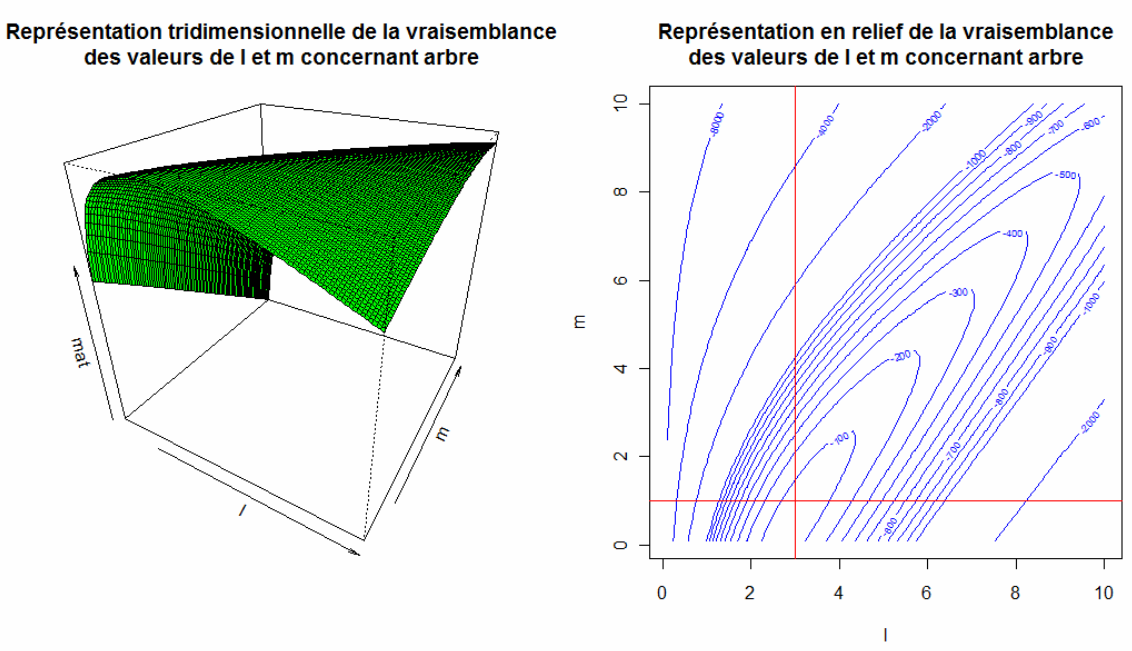
III.

```
arbre<-createTree(l=3,m=1,spp=250,plot=T,t=2)

T<-2

t<-branching.times(arbre$tree)

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)
m<-l

for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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

v<-c(0,-10,-20,-30,-40,-50,-100,-200,-400,-600,-800,-1000,-
1200,-1400)

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",levels=v)

abline(h=1,v=3,col="red")
```



IV.

```
arbre<-createTree(l=3,m=1,spp=2500,plot=T,t=2)

T<-2

t<-branching.times(arbre$tree)

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)
m<-l

for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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

v<-c(0,-10,-20,-30,-40,-50,-100,-200,-300,-400,-500,-600,-
700,-800,-900,-1000,-2000,-4000,-8000)

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",levels=v)

abline(h=1,v=3,col="red")
```



En comparant les exemples 2, 3, et 4 entre lesquels le nombre d'espèces considérées est accru d'un facteur 10 à chaque fois, on peut constater que, pour un T fixe et de valeur intermédiaire, plus le nombre d'espèce est grand et plus le maximum de vraisemblance apparaît avec précision.

Illustration de l'influence de T sur l'estimation des taux de spéciation et d'extinction

Pour que les commandes qui suivent fonctionnent correctement, il est nécessaire d'avoir au préalable sourcer les scripts *script source projet* et *script source génération arbres* ainsi que d'avoir charger la fonction `rep()`.

La fonction `rep()` utilise la fonction `createTree()` pour générer une série de n arbres (où n=nrep) ayant tous des paramètres identiques (l=l, m=m, t=T) mais que l'on peut faire varier. Ensuite, cette fonction récupère les longueurs de branches (vecteur t) de chacun des n arbres. Elle utilise alors la fonction `optimum2()` et la valeur de T que nous avons définie pour calculer les l et m les plus vraisemblables pour chacun de nos n arbres. Ces résultats sont alors stocké dans une matrice de nrep colonnes et de deux lignes (la première correspond aux valeurs de l et la seconde aux valeurs de m):

```
rep<-function(nreps=100,nsp=20,t=2,l=2,m=1) {
mat<-matrix(numeric(2*nreps),nrow=2,ncol=nreps);
for(i in 1:nreps) {
arbre<-createTree(t=t,l=l,m=m,spp=nsp);
a<-branching.times(arbre$tree);
param<-optimum2(a,t,c(l,m))$par
mat[,i]<-param;
}
return(mat);
}
```

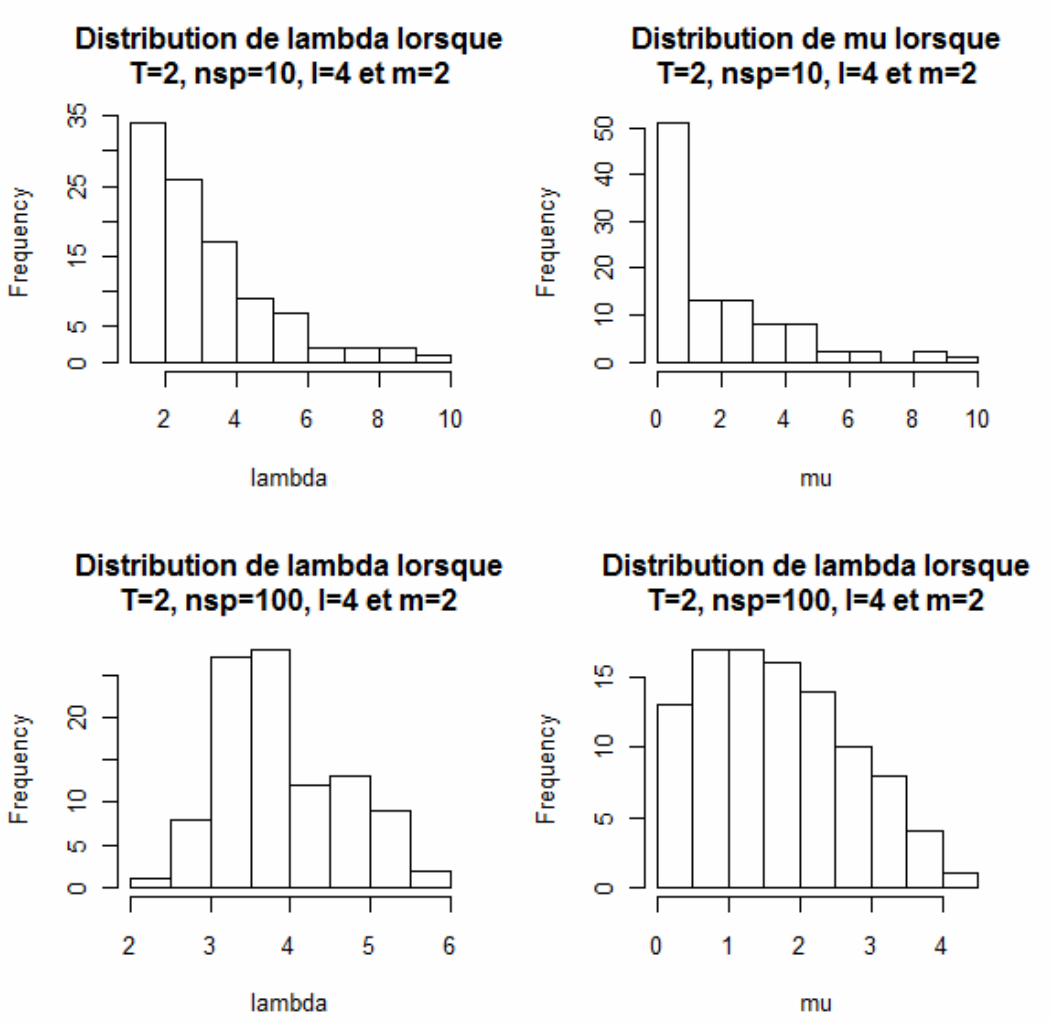
On peut facilement enregistrer les matrices générées par la fonction `rep()`. Cela permet alors de réaliser des représentations graphiques sous forme d'histogrammes des distributions de l et m relatives à des arbres pour lesquels nous avons défini les valeurs de T et du nombre d'espèces (nsp).

```
m<-rep(nreps=100,nsp=10,t=1)

par(mfrow=c(1,2))

hist(m[1,],xlab="lambda",main="Distribution de lambda lorsque
T=2, nsp=10, l=2 et m=1")
hist(m[2,],xlab="mu",main="Distribution de mu lorsque
T=2, nsp=10, l=2 et m=1")
```

On peut donc maintenant commencer par confirmer la théorie présentée à la section précédente et selon laquelle, pour un T fixe et de valeur intermédiaire, plus le nombre

d'espèce est grand et plus le maximum de vraisemblance apparaît avec précision (autrement dit plus les estimations de l et m réalisées par notre modèle seront proches de la réalité).

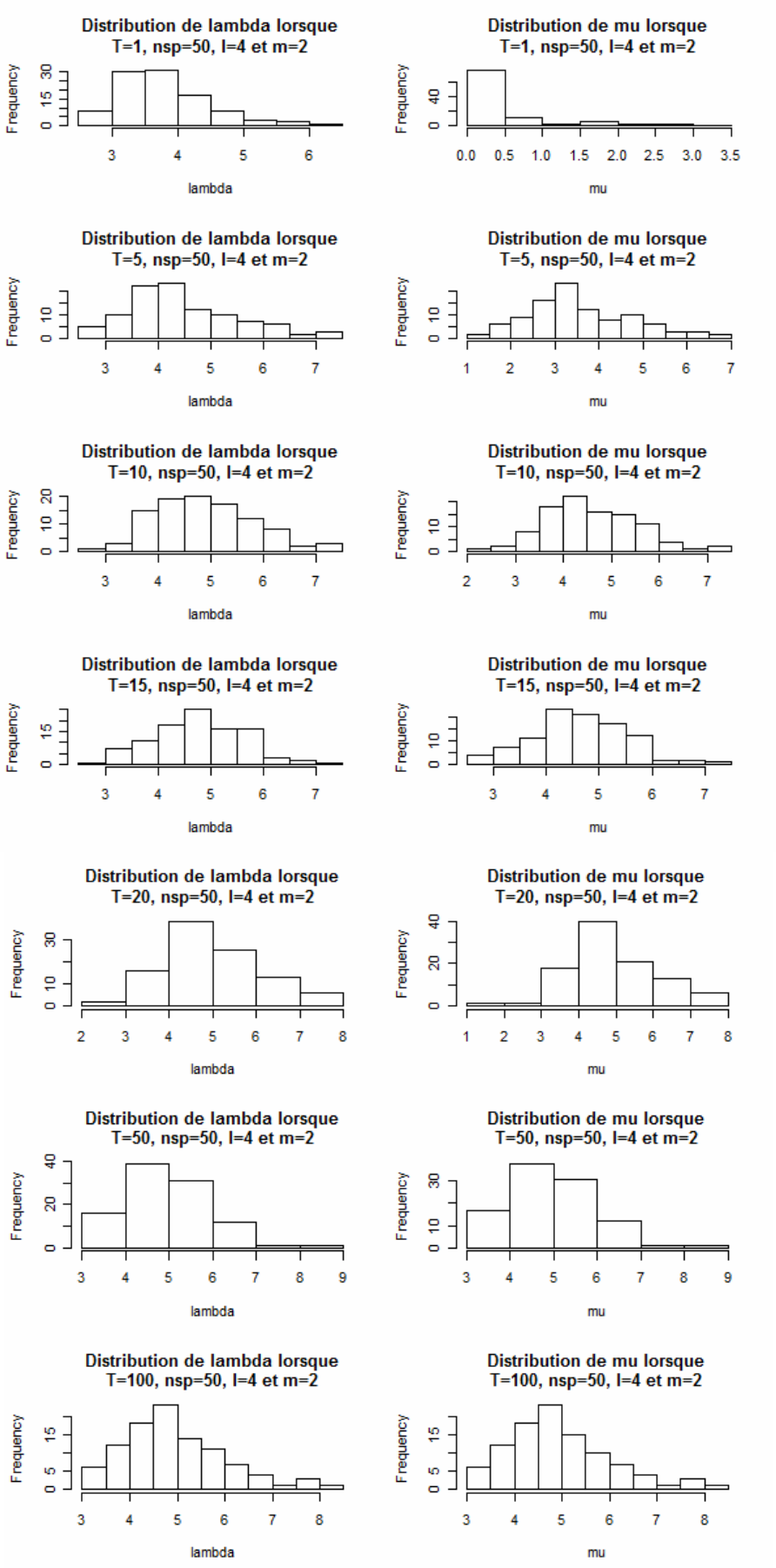
```
par(mfrow=c(2,2))

m<-rep(nreps=100,nsp=10,t=2,l=4,m=2)
n<-rep(nreps=100,nsp=100,t=2,l=4,m=2)

hist(m[1,],xlab="lambda",main="Distribution de lambda lorsque
T=2, nsp=10, l=4 et m=2")
hist(m[2,],xlab="mu",main="Distribution de mu lorsque
T=2, nsp=10, l=4 et m=2")

hist(n[1,],xlab="lambda",main="Distribution de lambda lorsque
T=2, nsp=100, l=4 et m=2")
 hist(n[2,],xlab="mu",main="Distribution de lambda lorsque
T=2, nsp=100, l=4 et m=2")
```



Ces graphes nous permettent en effet de constater que l'estimation de l et m est, pour une valeur de T intermédiaire, plus précise avec un grand nombre d'espèces.

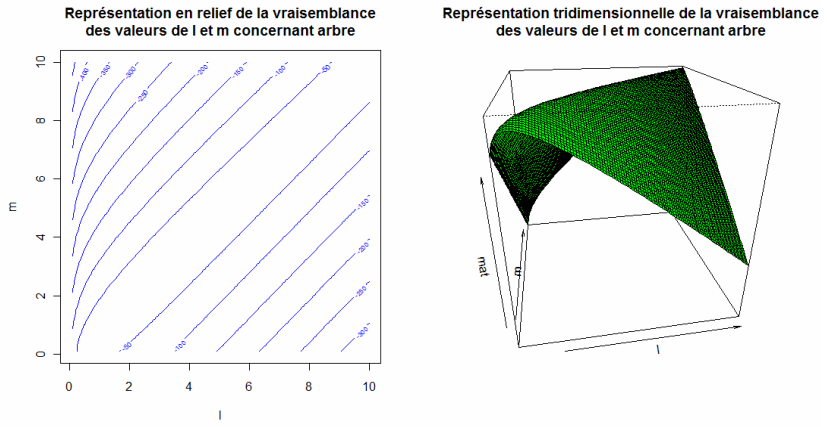
Faisons maintenant varier la valeur de T et observons ce qui se produit.

```
m<-rep(nreps=100,nsp=50,t=1,l=4,m=2)
n<-rep(nreps=100,nsp=50,t=5,l=4,m=2)
o<-rep(nreps=100,nsp=50,t=10,l=4,m=2)
b<-rep(nreps=100,nsp=50,t=15,l=4,m=2)
q<-rep(nreps=100,nsp=50,t=20,l=4,m=2)
r<-rep(nreps=100,nsp=50,t=50,l=4,m=2)
s<-rep(nreps=100,nsp=50,t=100,l=4,m=2)

par(mfrow=c(4,2))

hist(m[1,],xlab="lambda",main="Distribution de lambda lorsque
T=1, nsp=50, l=4 et m=2")
hist(m[2,],xlab="mu",main="Distribution de mu lorsque
T=1, nsp=50, l=4 et m=2")

hist(n[1,],xlab="lambda",main="Distribution de lambda lorsque
T=5, nsp=50, l=4 et m=2")
hist(n[2,],xlab="mu",main="Distribution de mu lorsque
T=5, nsp=50, l=4 et m=2")

hist(o[1,],xlab="lambda",main="Distribution de lambda lorsque
T=10, nsp=50, l=4 et m=2")
hist(o[2,],xlab="mu",main="Distribution de mu lorsque
T=10, nsp=50, l=4 et m=2")

hist(b[1,],xlab="lambda",main="Distribution de lambda lorsque
T=15, nsp=50, l=4 et m=2")
hist(b[2,],xlab="mu",main="Distribution de mu lorsque
T=15, nsp=50, l=4 et m=2")

par(mfrow=c(3,2))

hist(q[1,],xlab="lambda",main="Distribution de lambda lorsque
T=20, nsp=50, l=4 et m=2")
hist(q[2,],xlab="mu",main="Distribution de mu lorsque
T=20, nsp=50, l=4 et m=2")

hist(r[1,],xlab="lambda",main="Distribution de lambda lorsque
T=50, nsp=50, l=4 et m=2")
hist(r[2,],xlab="mu",main="Distribution de mu lorsque
T=50, nsp=50, l=4 et m=2")

hist(s[1,],xlab="lambda",main="Distribution de lambda lorsque
T=100, nsp=50, l=4 et m=2")
hist(s[2,],xlab="mu",main="Distribution de mu lorsque
T=100, nsp=50, l=4 et m=2")
```


Distribution de lambda lorsque T=1, nsp=50, l=4 et m=2
Distribution de mu lorsque T=1, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=5, nsp=50, l=4 et m=2
Distribution de mu lorsque T=5, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=10, nsp=50, l=4 et m=2
Distribution de mu lorsque T=10, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=15, nsp=50, l=4 et m=2
Distribution de mu lorsque T=15, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=20, nsp=50, l=4 et m=2
Distribution de mu lorsque T=20, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=50, nsp=50, l=4 et m=2
Distribution de mu lorsque T=50, nsp=50, l=4 et m=2
Distribution de lambda lorsque T=100, nsp=50, l=4 et m=2
Distribution de mu lorsque T=100, nsp=50, l=4 et m=2
Frequency
lambda
mu

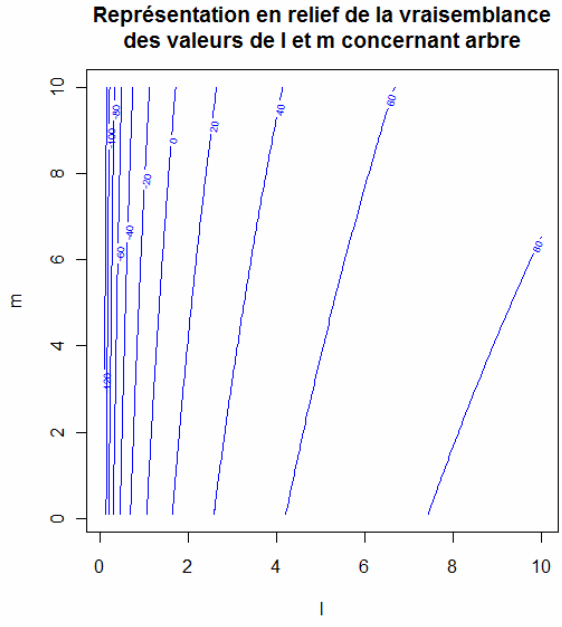
On peut constater qu'il existe une valeur de T intermédiaire (en pratique elle se situe généralement vers 10) qui permet de réaliser les estimations les plus exactes de l et m. Lorsque T est trop petit, notre modèle est forcé de postuler un taux d'extinction nulle car c'est pour lui le seul moyen d'expliquer que, malgré le fait que l'on considère un temps très court (puisque T est petit), on observe autant d'espèces au temps présent. A l'inverse, lorsque T est très grand, notre modèle postule des valeurs de l et m quasiment identiques car c'est pour lui la seule façon d'expliquer que sur un laps de temps très long (puisque T est grand) on observe pas plus d'espèces au temps présent que celles réellement dénombrées.
Ces deux situations – le cas d'un T trop grand ou à l'inverse celui d'un T trop petit – peuvent facilement être visualisé à l'aide des commandes graphiques `contour()` et `persp()`.

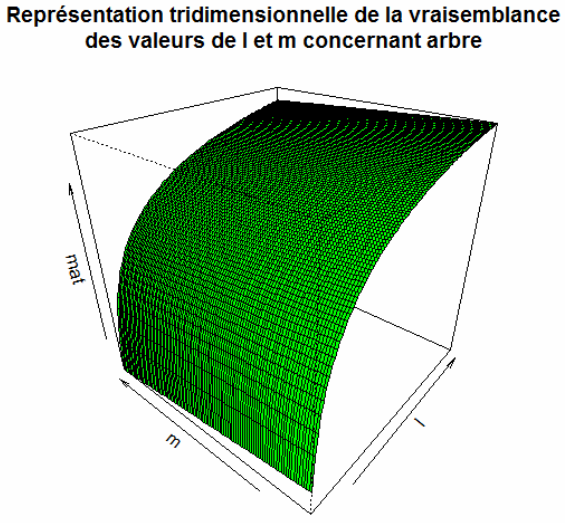
```
arbre<-createTree(t=30, l=4, m=2, spp=30, plot=T)

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)
m<-l

T<-30
t<-branching.times(arbre$tree)
start<-c(4,2)

optimum2(t,T,start)

for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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=-
10,phi=25)
```



Ces graphes montrent clairement que, lorsque T est trop grand, notre modèle est contraint de postuler des valeurs de l et m identiques, et ce malgré le fait qu'une telle conclusion soit biologiquement invraisemblable.

```
arbre<-createTree(t=0.1, l=4, m=2, spp=50, plot=T)

mat<-matrix(numeric(10000),ncol=100)
l<-seq(0.1,10,length.out=100)
m<-l

T<-0.1
t<-branching.times(arbre$tree)
start<-c(4,2)

optimum2(t,T,start)

for(i in 1:100) {
for(j in 1:100) {
mat[i,j]<-vrais(l[i], m[j], t, T)
}}

par(mfrow=c(1,2))

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=-
50,phi=30)
```





A l'inverse, ces graphes montrent, lorsque T est trop petit, notre modèle est contraint de postuler une valeurs m nulle, alors que cela est contraire aux processus évolutifs.

Application de la méthode de calcul des taux de spéciation et d'extinction à l'arbre phylogénétique des plantes Nematanthus et Chodonanthes

Chodonanthe et Nematanthus sont deux genres de plantes à fleurs tropicales qui sont caractérisées par une grande diversité de pollinisateurs (colibris, abeilles, papillons,…). Notre but est donc de calculer les valeurs de l et m ainsi que leurs vraisemblances pour les plantes de ces deux genres partageant le même type de pollinisateurs et de comparer les valeurs ainsi obtenues afin d'observer si le type de pollinisateurs a un effet sur des processus évolutifs telle que la vitesse de spéciation.

Pour ce faire, nous avons reçu un fichier de type « .RDAT » contenant les données phylogénétiques des plantes considérées ainsi que celles relatives à plusieurs out groups. Dans un premier temps, nous avons donc chargé ce ficher sur « R ».

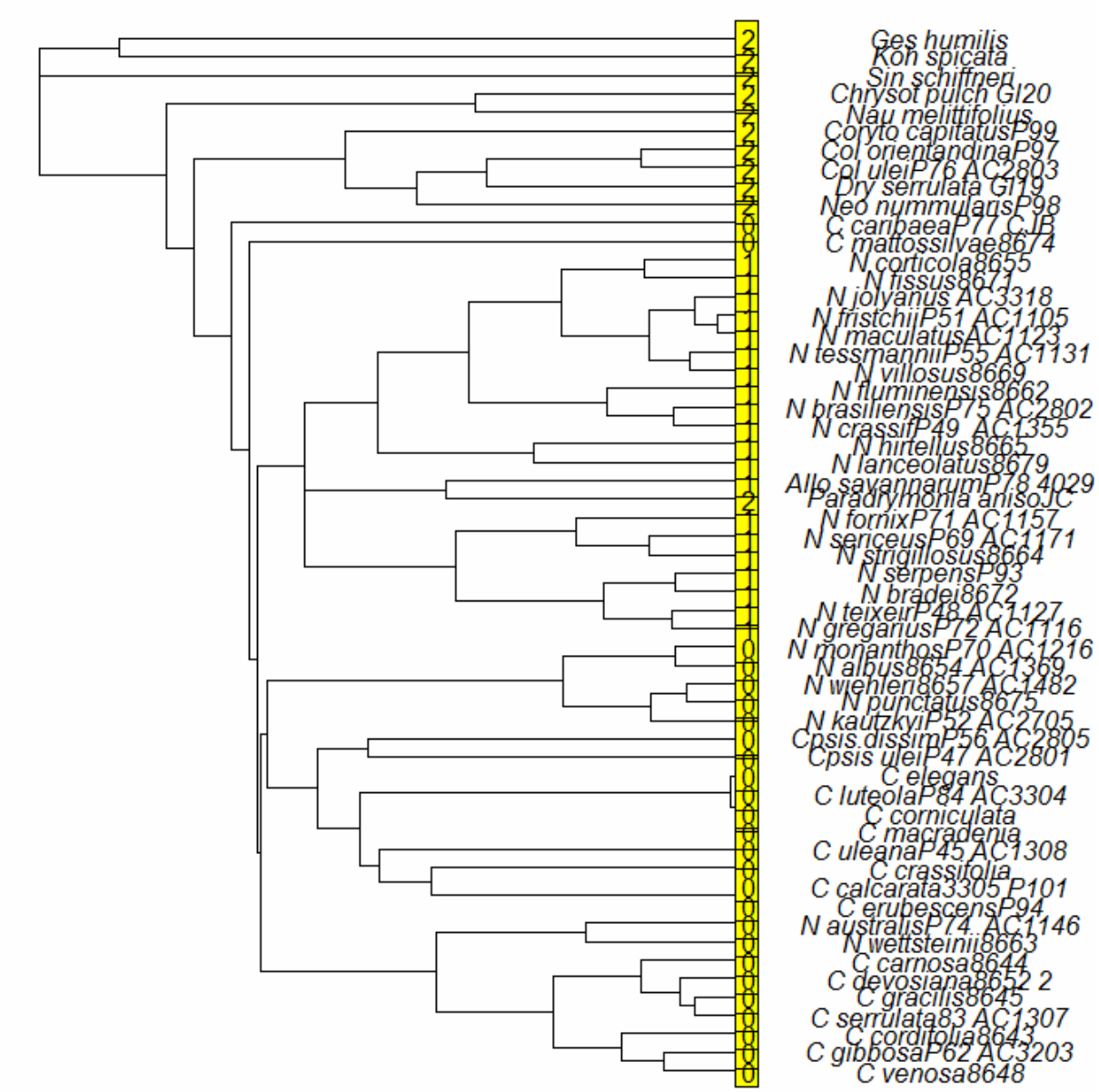
```
file<-load("Nematanthus&Chodonanthes.RDAT")
```

A ce moment, nous sommes donc en mesure d'afficher l'arbre correspondant à ces données.

```
plot(dated.tree,adj=0.5,no.margin=T)
```

On peut également faire figurer sur cet arbre le type de pollinisateur propre à chaque espèce, puisque cette information est contenue sous la rubrique `data` de notre fichier `dated.tree`. (On peut simplement faire figurer les noms et le type de pollinisateurs propres à chaque espèce en utilisant la commande `dated.tree$data`.)

```
tiplabels(dated.tree$data)
```



Relevons ici que l'indice 0 est attribué à toutes les espèces pollinisées par des abeilles, l'indice 1 à celles pollinisées par des colibris et l'indice 2 aux out groups.

Notre premier travail a consisté à modifier les données initiales dans le but d'obtenir un arbre contenant un seul out groups (C. caribaea) ainsi que nos deux catégories de plantes, à savoir celles pollinisées par des abeilles et celles pollinisées par des colibris. En observant les indices (en jaune) figurant sur l'arbre présenté ci-dessus, nous avons pu obtenir le nom des espèces à retirer des données initiales et réaliser ce retrait au moyen de la commande `drop.tip`.

```
dated.tree.all<-
drop.tip(dated.tree,c("Chrysot_pulch_Gl20","Ges_humilis","Koh_
spicata","Sin_schiffneri","Chrysot_pusch_Gl20","Nau_melittifol
ius","Coryto_capitatusP99","Col_orientandinaP97","Col_uleiP76_
AC2803","Dry_serrulata_Gl19","Neo_nummularisP98","C_mattossilv
ae8674"))
```

On peut alors réaliser une nouvelle représentation de cet arbre, et grâce à la commande `branching.times` (désormais classique), nous sommes également en mesure d'obtenir les différentes longueurs de branches qui lui sont propres.

```
plot(dated.tree.all)

tall<-branching.times(dated.tree.all)
```

De la même manière, on peut générer un arbre spécifique aux espèces pollinisées par des abeilles et un autre pour celles pollinisées par des colibris et obtenir pour chacun de ces arbres leurs longueurs de branches.

```
dated.tree.0<-
drop.tip(dated.tree.all,c("C_caribaeaP77_CJB","N_gregariusP72_
AC1116","N_teixeirP48_AC1127","N_bradei8672","N_serpensP93","N
_strigillosus8664","N_sericeusP69_AC1171","N_fornixP71_AC1157"
,"Paradrymonia_anisoJC","Allo_savannarumP78_4029","N_lanceolat
us8679","N_hirtellus8665","N_crassifP49__AC1355","N_brasiliens
isP75_AC2802","N_fluminensis8662","N_villosus8669","N_tessmann
iiP55_AC1131","N_maculatusAC1123","N_fristchiiP51_AC1105","N_j
olyanus_AC3318","N_fissus8671","N_corticola8655"))

t0<-branching.times(dated.tree.0)

dated.tree.1<-
drop.tip(dated.tree.all,c("C_caribaeaP77_CJB","C_mattossilvae8
674","C_venosa8648","C_gibbosaP62_AC3203","C_cordifolia8643","
C_serrulata83_AC1307","C_gracilis8645","C_devosiana8652_2","C_
carnosa8644","N_wettsteinii8663","N_australisP74__AC1146","N_k
autzkyiP52_AC2705","N_punctatus8675","N_wiehleri8657_AC1482","
N_albus8654_AC1369","N_monanthosP70_AC1216","Cpsis_uleiP47_AC2
801","Cpsis_dissimP56_AC2805","C_macradenia","C_corniculata","
C_luteolaP84_AC3304","C_elegans","C_uleanaP45_AC1308","C_crass
ifolia","C_erubescensP94","C_calcarata3305_P101"))

t1<-branching.times(dated.tree.1)
```

Ainsi, nous disposons maintenant de trois arbres phylogénétiques (un pour les plantes pollinisées par des abeilles, un pour celles pollinisées par des colibris et le dernier rassemblant toutes ces espèces de plantes). Et nous possédons pour chacun de ces arbres le vecteur t contenant leurs longueurs de branches. Il ne nous reste donc plus qu'à obtenir la valeur de T relative à chacun de ces arbres pour être en mesure d'appliquer à chacun d'entre eux notre fonction `optimum2()` qui nous permettra de déterminer leurs valeurs de l et m. Cela va être possible grâce au programme « FigTree » qui permet la visualisation d'arbres phylogénétiques.

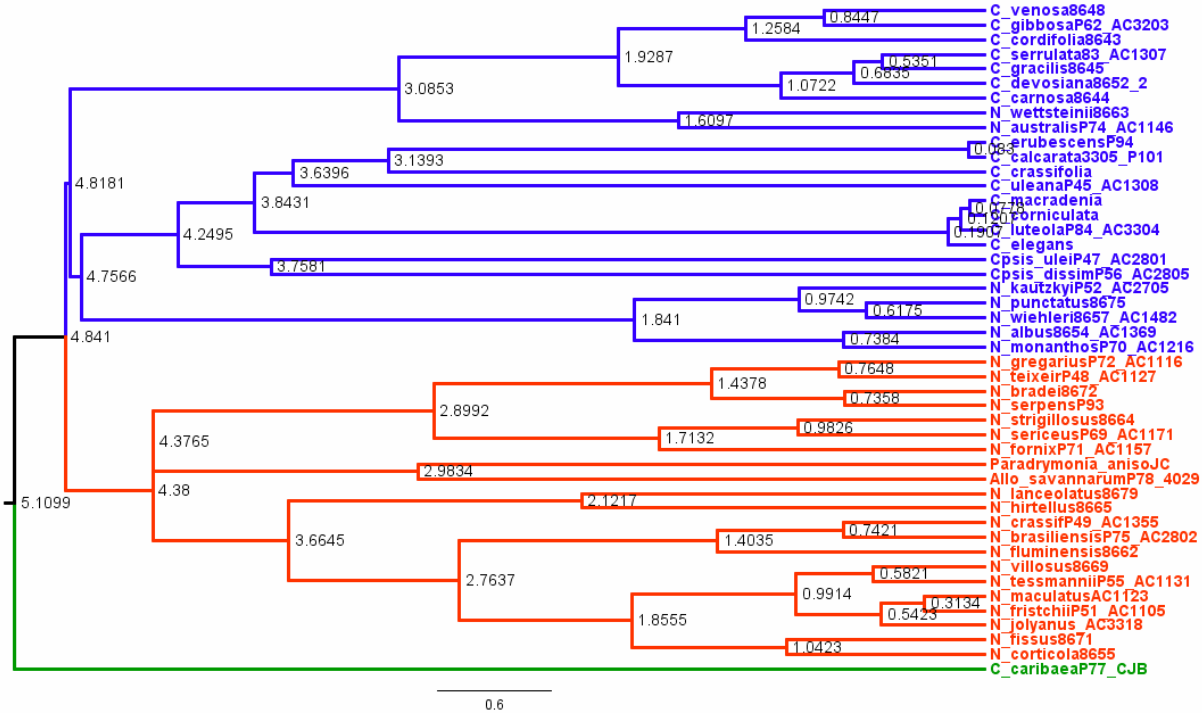
La commande `write.tree()` nous permet d'enregistrer les arbres que nous avons généré sur « R » dans un format qui sera lisible par « FigTree ».

```
write.tree(dated.tree.all,file="All.tre")
```

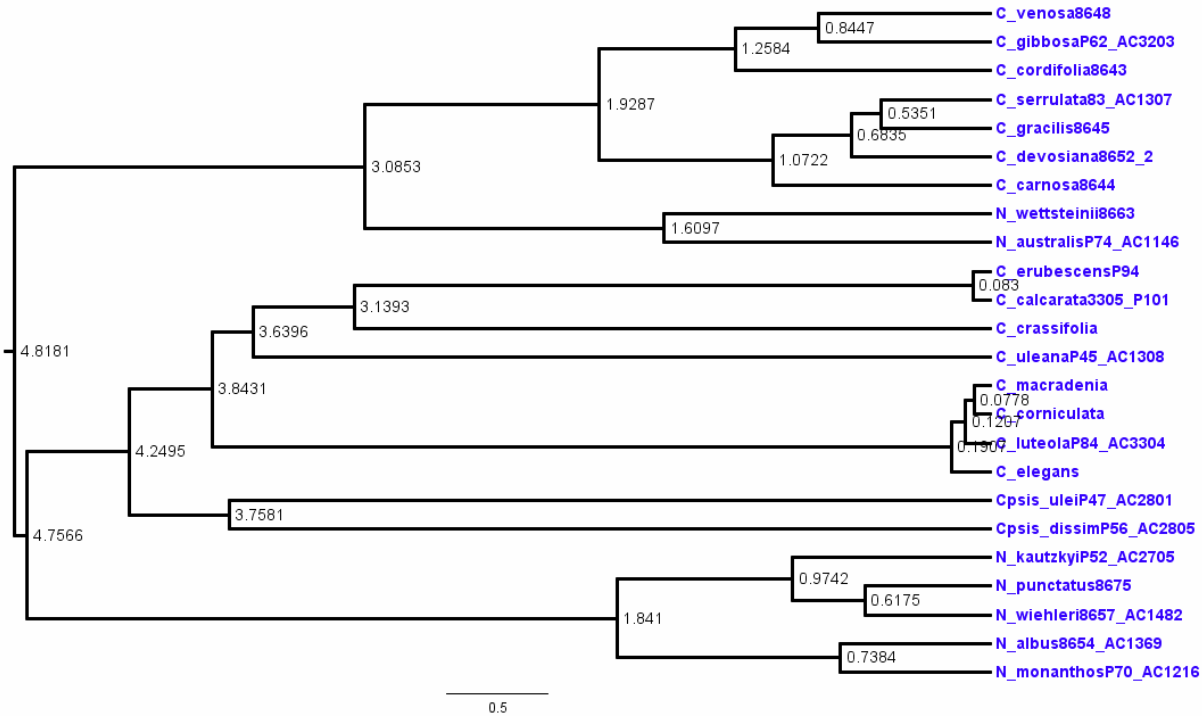
```
write.tree(dated.tree.0,file="Bee.tre")
write.tree(dated.tree.1,file="Colibris.tre")
```

On obtient alors les représentations suivantes:

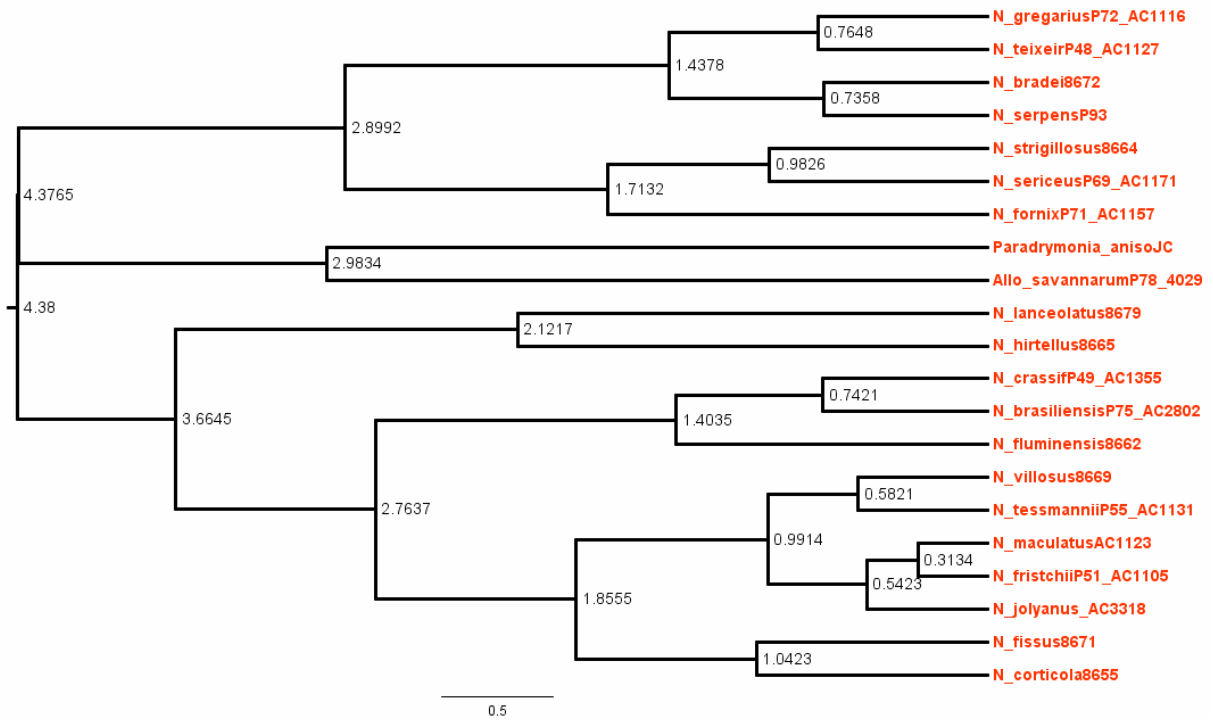
-Pour toutes les espèces considérées :



-Pour les espèces pollinisées par des abeilles :



-Pour les espèces pollinisées par des colibris :



En observant le premier arbre, on peut visuellement obtenir les valeurs de T suivantes :

- Pour l'ensemble des espèces considérées ; T = 5.1099
- Pour les espèces pollinisées par des abeilles ; T = 4.841
- Pour les espèces pollinisées par des colibris ; T = 4.841

On peut donc maintenant calculer les valeurs de l et m ainsi que les vraisemblances qui s'y rapportent pour chacun de nos trois arbres

- Pour l'ensemble des espèces considérées ;

```
optimum2(tall,Tall,c(6,3))
$par
[1] 0.4725805 0.0000000

$value
[1] 78.72969

$counts
function gradient
      18       18

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

- Pour les espèces pollinisées par des abeilles ;

```
optimum2(t0,T0,c(6,3))
$par
```

```
[1] 0.4722173 0.0000000

$value
[1] 40.25728

$counts
function gradient
      31       31

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

- Pour les espèces pollinisées par des colibris ;

```
optimum2(t1,T1,c(6,3))
$par
[1] 0.4861839 0.0000000

$value
[1] 34.42339

$counts
function gradient
      17       17

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Comme mentionné précédemment, afin de déterminer si, dans le cas étudié ici, le type de pollinisateurs à un effet sur l'évolution des espèces considérées, nous allons comparer les valeurs de vraisemblances (ce sont les valeurs soulignées et présentées sous la rubrique `$value`) obtenues avec les trois dernières commandes figurant ci-dessus.
Dans les faits, nous pourrons déduire un effet du pollinisateur si le résultat de l'addition des vraisemblances propres au clade des plantes pollinisées par des abeilles et à celui des plantes pollinisées par des colibris est significativement différent de la vraisemblance obtenue pour l'ensemble des espèces.

Nous commençons donc par extraire les valeurs de vraisemblance des trois modèles:

```
vrais.tot<-optimum2(tall,Tall,c(6,3))$value
vrais.abeilles<-optimum2(t0,T0,c(6,3))$value
vrais.colibri<-optimum2(t1,T1,c(6,3))$value
```

Ensuite, nous comparons ces valeurs à l'aide d'un test du chi^2 dans le but d'évaluer s'il existe une différence significative entre le résultat de l'addition des vraisemblances « par pollinisateurs » et la vraisemblance obtenue pour l'ensemble des espèces (notre hypothèse nulle est donc qu'il n'y pas de différence significative, ce qui signifie que les pollinisateurs n'ont pas d'effet sur les vraisemblances des différents groupes de pollinisateurs et sur les taux de spéciation et d'extinction):

```
chi2<-abs(-2*(vrais.tot-(vrais.abeilles+vrais.colibri)))
pchisq(chi2,2, lower=F)[5]
[1] 0.860397
```

La p-value étant supérieure au seuil de 5%, nous ne pouvons pas rejeter notre hypothèse nulle, et nous ne pouvons donc pas conclure à un effet significatif du type de pollinisateurs sur les valeurs de l et m et sur la vraisemblance.

Ce résultat ne paraît pas étonnant si l'on considère le fait que les valeurs des taux de spéciation pour nos trois groupes sont quasiment identiques, que chacun de ces groupes compte approximativement un nombre d'espèces similaire ou encore que les branches des arbres correspondants sont de longueurs comparables. En effet, ces trois observations tendent à signifier que le type de pollinisateurs n'influence pas de manière significative les processus évolutifs.

Il est à noter que si nous avons comparé les valeurs de vraisemblance et que nous n'avons pas utilisé les valeurs de l et m obtenues, c'est parce que, malgré nos efforts, celles-ci ne semblaient pas suffisamment fiables. En fait, la valeur du taux de spéciation, à savoir environ 0.5, paraît être une bonne estimation du taux réel. Cela signifierait donc que 0.5 espèces apparaissent par unité de temps, soit par million d'années (puisque l'échelle de nos arbres est le million d'années). En revanche, l'estimation du taux d'extinction, en l'occurrence 0, est inutilisable puisque l'on sait qu'un taux d'extinction nulle n'est biologiquement pas concevable. Malheureusement, cette erreur ne peut être corrigé par notre modèle. En effet, celle-ci résulte du biais introduit par la valeur de T, un paramètre indissociable de notre fonction (dans notre cas, le laps de temps étudié semble trop proche du présent pour que notre modèle puisse postuler une autre valeur que zéro pour le taux d'extinction).
La solution à ce problème serait d'utiliser une formule du maximum de vraisemblance qui ne serait plus fonction de T, soit la formule suivante :

$$prob(t_1, t_2, \ldots, t_{a-1} | n, l, m, T) = \mathbf{\Pi}[\, p_1(t_i) dt_i \,/\, \int_0^T p_1(u)\, du]$$

Le temps nous manque malheureusement pour coder une telle fonction dans « R ». C'est pourquoi Nicolas Salamin nous a fourni son script de la fonction en question. Ainsi nous allons pourvoir comparer les résultats obtenus avec les deux méthodes. Voici tout d'abord le script de cette fonction :

```
like<-function(times, T=1, like="given", algo="BFGS") {
     if(class(times)=="phylo") times<-branching.times(times);

     if(like=="data") {
     out<-optim(c(2, 1), function(p) -ldata(T, times, p[1],
p[2]), method=algo);
  }
  else {
          out<-optim(c(2, 1), function(p) -lgiven(times, p[1],
p[2]), method=algo);
```

[5] Notons que les degrés de liberté (dL) pour l'ensemble du modèle correspondent au nombre total de paramètres testés moins le nombre de groupes à comparer. Dans ce cas nous avons 4 – 2 dL, soit 2 dL.

```
          out$par<-sort(out$par, decreasing=TRUE);
     }

     print(list(speciation=out$par[1], extinction=out$par[2],
lnL=out$value));
}

prob<-function(t, s, e) {
     if(s==e) {
          like<-log(1)-2*log(1+s*t);
     }
     else {
          like<-log(((s-e)^2)*exp(s*t-e*t))-log((s*exp(s*t-
e*t)-e)^2);
     }

  if(is.nan(like) | is.infinite(like)) {
    return(-100000);
  }
  else {
     return(like);
  }
}

intprob<-function(t, s, e) {
     tol<-0.000000001;

     if(t>=250) {
          if(s>e) {
               if(s<tol) {
                    like<-log(1)-log(tol);
               }
               else {
                    like<-log(1)-log(s);
               }
          }
          else {
               if(e<tol) {
                    like<-log(1)-log(tol);
               }
               else {
                    like<-log(1)-log(e);
               }
          }
     }
     else {
          if(s==e) {
               like<-log(t)-log(s*t + 1);
          }
          else {
```

```
                like<-log(exp(s*t-e*t)-1)-log(s*exp(s*t-e*t)-e);
          }
     }

     return(like);
}

paradis<-function(t, s, e) {
  if(s<0 | e<0) return(-10000000);

     par.times<-c(NA, t);
  ntax=length(t)+1;

     return(-2*(sum(log((ntax-1):1))+(ntax-
2)*log(s)+s*sum(par.times[3:ntax])+ntax*log(1-e)-
2*sum(log(exp(s*par.times[2:ntax])-e))));
}

lgiven<-function(t, s, e) {
  if(s<0 | e<0) return(-10000000);

     ntimes<-length(t);
     T<-1000*max(t);

     return(-ntimes*intprob(T, s, e)+sum(sapply(t, prob, s=s,
e=e)));
}

ldata<-function(T, t, s, e) {
  if(s<0 | e<0) return(-10000000);

     tol<-0.000000000000001;
     ntimes<-length(t);

     if(s<tol) {
          return(ntimes*log(tol)+prob(tol, e, T)+sum(sapply(t,
prob, s=tol, e=e)));
     }

  return(ntimes*log(s)+prob(s, e, T)+sum(sapply(t, prob, s=s,
e=e)));
}

estimate.lgiven<-function(trees) {
     btimes<-lapply(trees, branching.times);
     speciation<-vector(length=length(trees));
     extinction<-vector(length=length(trees));
     for(i in 1:length(trees)) {
```

```
          out<-sort(optim(c(2,1), function(p) -
lgiven(btimes[[i]], p[1], p[2]), upper=c(100,100),
lower=c(0,0), method="L-BFGS-B")$par, decreasing=T);
          speciation[i]<-out[1];
          extinction[i]<-out[2];
     }

     return(data.frame(speciation=speciation,
extinction=extinction));
}

estimate.ldata<-function(trees, T) {
  btimes<-lapply(trees, branching.times);
     speciation<-vector(length=length(trees));
     extinction<-vector(length=length(trees));
     for(i in 1:length(trees)) {
          out<-sort(optim(c(2,1), function(p) -ldata(T,
btimes[[i]], p[1], p[2]), upper=c(100,100), lower=c(0,0),
method="L-BFGS-B")$par, decreasing=T);
          speciation[i]<-out[1];
          extinction[i]<-out[2];
     }

     return(data.frame(speciation=speciation,
extinction=extinction));
}

expectedTimeTransform<-function(T, s, e, n) {
  a<-0;
  b<-(exp(s*T-e*T)-1)/(s*exp(s*T-e*T)-e);
  lim<-(b-a)/n;
  y<-lim*(1:(n-1));
  return((1/(s-e))*log((1-e*y)/(1-s*y)));
}

calc.mat<-function(size, T, t) {
  mat<-matrix(numeric(size*size),nrow=size);
  l<-seq(0,10,length.out=size);
  m<-l;
  for(i in 1:size) {
    for(j in 1:size) {
      mat[i,j]<-ldata(T,t,l[i],m[j]);
    }
  }

  return(mat);
}

createTree<-function(t=1, l=2, m=1, spp=20, plot=FALSE) {
  if(l<m) {
```

```
    stop("Pas d'arbre possible. Le taux de spÃ©ciation est
plus petit que le taux d'extinction.\n",call.=FALSE);
  }

  if(l==0) {
    stop("Pas d'arbre possible. Le taux de spÃ©ciation est de
zÃ©ro.\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(tree);
}
```

```
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)));
}

plot2D<-function(t, l, m) {
  l<-m<-seq(0.000001, l, length=100);

  mat<-matrix(numeric(10000), nrow=100);

  for(i in 1:100) {
    for(j in (i+1):100) {
      mat[i,j]<-lgiven(t, l[i], m[j]);
    }
  }
}
```

Pour utiliser cette fonction, les commandes sont très simples, il suffit d'entrer `like(t, like= "given")`, où l'argument `t` représente (comme d'habitude) les longueurs des branches de notre arbre, pour que les valeurs des taux de spéciation (sous `$speciation`) et d'extinction (sous `$extinction`) ainsi que la vraisemblance (sous `$lnL`) nous soient retournées. Appliquons maintenant cette formule à notre arbre d'intérêt afin de comparer si les valeurs que nous obtiendrons pour l et m sont vraiment différentes de celle que nous avons obtenues en utilisant notre formule `optimum2()`.

Commençons par sourcer le code de cette nouvelle fonction ainsi que par charger la package « ape ».

```
source("C:\\Users\\Vincent\\Documents\\Vincent\\UNI\\BSc-
Biologie 2 UNIL\\Mathématiques\\Script source speciate.R")

library(ape)
```

Puis définissons le vecteur `t` comme les longueurs des branches de notre arbre.

```
file<-load("Nematanthus&Chodonanthes.RDAT")
```

```
dated.tree.all<-
drop.tip(dated.tree,c("Chrysot_pulch_Gl20","Ges_humilis","Koh_
spicata","Sin_schiffneri","Chrysot_pusch_Gl20","Nau_melittifol
ius","Coryto_capitatusP99","Col_orientandinaP97","Col_uleiP76_
AC2803","Dry_serrulata_Gl19","Neo_nummularisP98","C_mattossilv
ae8674"))

tall<-branching.times(dated.tree.all)
```

Et appliquons maintenant ces valeurs de t dans notre nouvelle fonction.

```
like(tall, like="given")
```

```
$speciation
[1] 0.4737472

$extinction
[1] 0.0007526793

$lnL
[1] 76.31502
```

On peut donc constater que l'estimation du taux de spéciation par notre fonction `optimum2()` (0.4725805) était tout à fait proche de celle obtenue ici (0.4737472). En revanche, la fonction `optimum2()` a comme nous l'avions prévu sous-estimé la valeur du taux d'extinction puisqu'on obtient ici un résultat différent de zéro, une différence qui s'explique comme nous l'avions mentionné par le biais introduit par la variable T.

Enfin, il est encore intéressant de constater que la valeur obtenue avec ce nouveau modèle pour le taux de spéciation est de plusieurs centaines de fois supérieure à celle du taux d'extinction (0.4737472/0.0007526793 = 629.4144). Or en principe, on sait par expérience que ce rapport n'excède pas 2 ou 3. Cela signifie donc que les plantes appartenant aux genres Nematanthus et Codonanthe sont effectivement caractérisées par un taux de spéciation bien plus élevé que la moyenne. Et comme nous avons mis en évidence le fait que cette valeur inhabituelle ne pouvait pas être expliquée par le type de pollinisateurs, il serait maintenant intéressant de tester d'autres paramètres susceptibles d'expliquer cette particularité (comme par exemple l'adaptation rapide à des niches écologiques spécialisées). Affaire à suivre…