TP1 : Simulation de données génétiques par coalescence et analyses standard

L'objectif de ce TP est d'apprendre à simuler des données génétiques par coalescence et à les analyser, avec le logiciel R. Pour cela on utilise la librairie coala, qui inclue notamment plusieurs algorithmes de simulation, ainsi que des fonctions permettant d'éxtraire différentes propriétés des données obtenues. Pour représenter les arbres de coalescence simulés, on utilise aussi la librairie R phyclust, qui est généralement utilisée en phylogénie.

```
library(coala)
library(phyclust)
```

Parmi les différents simulateurs disponibles, on choisit d'uitliser ms.

```
activate_ms(priority = 500)
```

Nous allons simuler un échantillon de 100 séquences génomiques, qui comprend 1000 locus indépendants de 1000 paires de bases chacun. Pour simuler ces données, nous allons considérer une population isolée et non structurée, et nous supposerons qu'il n'y a pas de recombinaison intra locus.

```
n=100 # taille de l'échantillon
loc=1000 # nombre de locus
L=1000 # taille de chaque locus
```

1. Population de taille constante

Considérons dans un premier temps une population de taille constante. Les paramètres du modèle biologiques sont

```
N=10000 # taille de la population
mu=2*10^(-8) # taux de mutation par base et par génération
```

Mais le modèle de coalescence correspondant n'a qu'un seul paramètre, $\theta = 2N\mu$.

```
theta=2*N*mu
```

Pour simuler les données, on définit d'abord le modèle choisi, en indiquant par les commandes de type *sumstat* les informations que l'algorithme doit retourner pour chaque échantillon simulé. Puis on effetue la simulation proprement dite.

```
model <- coal_model(sample_size=n,loci_number=loc, loci_length=L,ploidy=1) +
feat_mutation(rate=theta*L) +
sumstat_trees() + sumstat_seg_sites() + sumstat_nucleotide_div()
res <- simulate(model)
```

Analyse d'un locus

Commençons par étudier les résultats obtenus pour le premier locus (parmi les 1000 de l'échantillon simulé).

Arbre de coalescence

L'arbre de coalescence obtenu pour ce locus peut être visulisé à l'aide des commandes

```
tree=read.tree(text=res$trees[[1]])
plot(tree,show.tip.label=TRUE,no.margin=FALSE,direction="downwards")
```

Les temps des coalescences successives de cet arbre peuvent être obtenus par

```
btimes=branching.times(as.phylo(tree))
sort(btimes)
```

Diversité génétique

Les allèles portés par chaque séquence pour les sites polymorphes à ce locus sont stockés dans le tableau

```
tab <- res$seg_sites[[1]]
```

On peut en déduire par exemple le nombre de sites polymorphes,

```
p=dim(tab)[2]
```

les haplotypes existants à ce locus,

```
haps=apply(tab$snps,1,paste,collapse='')
uniq_haps=unique(haps)
```

et le nombre d'occurences de ces haplotypes dans l'échantillon.

```
h=length(uniq_haps)
f=rep(0,h)
for (i in 1:h){
    f[i]=sum(haps==uniq_haps[i])
}
```

A partir du nombre de sites polymorphes p, on peut déduire l'estimateur de Watterson basé sur ce locus,

$$\theta_W = \frac{p}{L}(\sum_{k=1}^{n-1}\frac{1}{k})^{-1}$$

qui est un estimateur du paramètre θ.

```
c=sum(1/(1:(n-1)))
w=p/(c*L)
```

Un autre estimateur de θ, l'estimateur de Tajima ou diversité nucléotidique, est calculé directement par la fonction *simulate* et s'obtient donc simplement par

```
t=res$pi[[1]]/L
```

(on divise par L car la fonction *simulate* le calcule au niveau du locus entier, et non par base). Cet estimateur correspond au nombre de différences moyen entre deux séquences de l'échantillon.

$$\theta_T = \pi_n = \frac{1}{L}\frac{1}{n(n-1)}\sum_{i\neq j}\Pi(i,j)$$

avec $\Pi(i,j)$ nombre de différences entre les séquences i et j.

Estimation de θ

En réalité, les estimations de θ que l'on peut obtenir à partir d'un locus de 1000 paires de base sont très imprécises. Pour obtenir une meilleure estimation, on va donc combiner les 1000 locus de l'échantillon simulé et considérer la moyenne des estimations

```
w=rep(0,loc)
for (i in 1:loc){
    tab <- res$seg_sites[[i]]
    w[i]=dim(tab)[2]/(c*L)
}
mean(w)
t=res$pi/L
mean(t)
```

On peut également calculer l'écart type de l'estimation pour un locus donné.

```
sqrt(var(w))
sqrt(var(t))
```

Ces résultats (moyenne et écart type) sont-ils en accord avec les propriétés des estimateurs démontrées dans le TD1?

Quelques fonctionnalités de coala

Un des intérêts de *coala* est qu'il permet de calculer directement un grand nombre de statistiques à partir des données simulées. Une de ces statistiques est le Site Frequency Spectrum (SFS), qui correspond au nombre de polymorphismes pour lesquels on observe i allèles dérivés parmi les n séquences, pour i de 1 à $n-1$. Le SFS est donc un vecteur de taille $n-1$, dont la somme correspond au nombre de polymorphismes total. D'autre part, *coala* peut retourner une transformation des statistiques, plutôt que les statistiques brutes. Ainsi les estimateurs de Watterson et Tajima dans un échantillon peuvent être obtenus de manière plus directe que ci-dessus, par la commande

```
model <- coal_model(sample_size=n,loci_number=loc, loci_length=L,ploidy=1) +
feat_mutation(rate=theta*L) +
sumstat_trees() + sumstat_seg_sites() +
sumstat_nucleotide_div(transformation=mean) + sumstat_sfs(transformation=sum)
res <- simulate(model)
w=res$sfs/(L*loc*c) # res$sfs est le nombre de polymorphismes pour l'ensemble des locus
t=res$pi/L
```

Nous utiliserons cette méthode dans la suite du TP, ainsi que dans le TP de demain.

Distribution du TMRCA

Nous avons vu précédemment que pour un locus donné, il était possible d'obtenir les instants des coalescences successives pour l'arbre simulé à ce locus. Le dernier de ces instants correspond au TMRCA (*Time to the Most Recent Common Ancestor*) du locus. En répétant cette opération pour l'ensemble des locus, on peut donc obtenir la distribution empirique du TMRCA pour le modèle considéré.

```
tmrca=rep(0,loc)
for (i in 1:loc){
    tree=read.tree(text=res$trees[[i]])
    btimes=branching.times(as.phylo(tree))
    tmrca[i]=max(btimes)
}
mean(tmrca)
hist(tmrca)
```

On remarque que le TMRCA moyen est ici deux fois plus court que l'espérance du TMRCA donnée en cours. Ceci est simplement dû au fait que dans *ms*, le temps est codé en untés de 2N générations, et non de N.

2. Population en expansion

Nous allons maintenant simuler des échantillons issus d'une population dont la taille augmente de manière exponentielle selon l'équation $N(t) = N(0) * exp(-0.0005 * t)$, où t est en générations et $t = 0$ correspond au présent. Ce taux de croissance implique que la taille de population est multipliée par deux en environ 1500 générations. Ce type de modèle démographique peut être simulé avec *ms*. On introduit simplement le paramètre de croissance en unités de coalescence

```
g=0.0005 # par génération
G=N*g
```

et on simule ce modèle en utilsant l'option *feat_growth* de *coala*

```
model <- coal_model(sample_size=n,loci_number=loc, loci_length=L,ploidy=1) +
feat_mutation(rate=theta*L) + feat_growth(G, time = 0) +
sumstat_trees() + sumstat_seg_sites() +
sumstat_nucleotide_div(transformation=mean) + sumstat_sfs(transformation=sum)
res <- simulate(model)
```

En utilisant les mêmes commandes que dans l'exercice 1 (modèle de taille constante), calculez les estimateurs de Watterson et Tajima pour ce jeu de données. Pourquoi les valeurs obtenues sont elles plus petites que précédemment?

On peut le vérifier en traçant la distribution du TMRCA, ou en calculant la moyenne de cette distribution, comme dans l'exercice 1.

Considérons maintenant le même scenario de croissance exponentielle mais avec une taille actuelle plus grande, c'est à dire $N(0) = 25000$ au lieu de 10000.

```
N=25000
theta=2*N*mu
G=N*g
```

Vérifiez que la moyenne de θ_W a de nouveau une valeur proche de celle de l'exercice 1. Est-ce le cas pour θ_T? Expliquez ce résultat en pensant à la forme des arbres de coalescence dans le cas d'une population croissante (cf cours) et au lien entre θ_T et les fréquences alléliques (cf TD1).

Conclusion : cet exercice montre qu'en combinant plusieurs caractéristiques d'un échantillon (par exemple θ_W et θ_T) on est a priori capable de dire si cet échantillon provient d'une population de taille variable ou constante. De manière générale, distinguer entre différents scénarios démographiques est un des objectfis centraux de la génétique des populations. Mais les modèles à tester sont souvent bien plus complexes que celui de ce TP (populations structurées, migration, recombinaison ...)!

3. Coalescent avec recombinaison

Nous avons supposé jusqu'ici que l'évolution au sein d'un locus de 1000 paires de base se faisait sans recombinaison. Cette hypothèse n'est en fait pas très réaliste car l'ordre de grandeur des taux de mutation et recombinaison par base est comparable, si bien qu'à partir du moment où on observe des mutations dans un locus, il est probable que ce locus ait aussi connu des recombinaisons. Heureusement, les algorithmes de coalescence tels que *ms* permettent de simuler des généalogies avec recombinaison. Avec *coala*, cela peut se faire typiquement comme dans l'exemple ci-dessous, où l'on considère un échantillon de 5 séquences et un locus de taille 100,000.

```
n=5
N=10000
L=100000
loc=1
mu=2*10^(-8)
theta=2*N*mu
r=10^(-8) # taux de recombinaison par génération et par paire de base
rho=2*N*r
model=coal_model(sample_size=n,loci_number=loc, loci_length=L,ploidy=1) +
feat_mutation(rate=theta*L) + feat_recombination(rate=rho*L) +
sumstat_trees() + sumstat_seg_sites() +
sumstat_nucleotide_div(transformation=mean) + sumstat_sfs(transformation=sum)
res=simulate(model)
```

La différence principale avec le cas sans recombinaison est que pour un locus donné, la généalogie n'est pas fixe pas varie en fonction de la position. Les commandes ci-dessous permettent de calculer le nombre de ces généalogies distinctes, et de les représenter (on ne regarde ici que les 4 premières, à titre d'exemple).

```
nbcoal=length(res$trees[[1]])
nbcoal
par(mfrow=c(1,4))
for (i in 1:4){
    tree=read.tree(text=res$trees[[1]][i])
    plot(tree,show.tip.label=TRUE,no.margin=FALSE,direction="downwards",y.lim=c(0,2),cex=1.5)
}
```

Les arbres successifs vous semblent ils complétement indépendants? Expliquez pourquoi en vous aidant du cours.