TP2 : Estimation de paramètres par une approche de type Approximate Bayesian Compuatation (ABC)

En statistique, l'estimation des paramètres d'un modèle à partir d'un échantillon de données observées passe souvent par le calcul de la vraisemblance de l'échantillon, c'est à dire de la probabilité d'observer cet échantillon sachant une certaine valeur des paramètres du modèle. En effet, la méthode du maximum de vraisemblance consiste à choisir comme estimateur la valeur des paramètres permettant de maximiser cette probabilité. Cependant, pour des modèles complexes tels que ceux généralement considérés en génétique des populations, on ne sait pas calculer cette vraisemblance. L'ABC est une méthode d'estimation adaptée à ce genre de cas. Plutôt que de calculer la vraisemblance pour une valeur des paramètres donnés, elle évalue, à partir de simulations intensives, si cette valeur permet de reproduire certaines caractéristiques des données, appelées statistiques résumantes.

1. Loi exponentielle.

Pour illustrer cette idée, supposons que l'on oberve un échantillon de $n = 100$ réalisations indépendantes $x_1, \ldots, x_n$ d'une loi exponentielle de paramètre $\lambda = 2$, et que l'on cherche à estimer λ à partir de cet échantillon.

```
n=100
lambda=2
obs=rexp(n,lambda)
```

La manière la plus naturelle de procéder est d'utiliser l'estimateur du maximum de vraisemblance, qui est ici connu, à savoir

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} x_i}$$

Mais imaginons que nous souhaitons utiliser l'approche ABC. La première chose à faire est de définir quelle(s) statistique(s) utiliser pour résumer les données. Prenons par exemple la moyenne $m = \frac{1}{n}\sum_{i=1}^{n} x_i$, qui est un choix assez standard.

```
s=mean(obs)
```

Puisque l'approche est Bayésienne, la deuxième chose à faire est de choisir une loi a priori pour le paramètre inconnu λ. Prenons par exemple une loi uniforme entre 0 et 10, autrement dit on pense a priori que le paramètre est entre 0 et 10 mais n'importe quelle valeur entre ces deux bornes semble aussi probable.

Ayant fait ces deux choix, nous allons maintenant simuler un grand nombre (par exemple 1000) d'échantillons de même taille que l'échantillon observé, en tirant λ au hasard dans dans notre loi a priori. Puis, pour chaque échantillon i, nous allons calculer la statistque résumante choisie, à savoir la moyenne m_i.

```
nb_rep=1000
abc_param=array(dim=c(nb_rep,1),data=0)
abc_sumstat=array(dim=c(nb_rep,1),data=0)
for (i in 1:nb_rep){
    abc_param[i,1]=runif(1,0,10) # on tire une valeur de lambda
    sim=rexp(n,abc_param[i,1]) # on simule l'échantillon de n observations
    abc_sumstat[i,1]=mean(sim) # on calcule m
}
```

```
colnames(abc_param)="lambda"
colnames(abc_sumstat)="moyenne"
```

Dans sa version la plus simple, appelée rejet, l'ABC consiste à choisir, parmi les 1000 valeurs m_i ainsi simulées, celles étant suffisamment proches de la vraie valeur m observée, et d'estimer la loi a posteriori de λ par l'histogramme des λ_i correspondants. Prenons par exemple les 10% de m_i les plus proches de m.

```
d=abs(abc_sumstat-s)
ind=order(d)[1:100] # idices des 100 distances les plus faibles.
hist(abc_param[ind,1])
```

Que pensez-vous de ce résultat?

Pour aller un peu plus loin, nous allons maintenant utiliser la librairie abc, qui propose différentes méthodes de type ABC et fournit plusieurs outils très utiles pour analyser les résultats. Avec cette librairie, l'estimation ci-dessus peut être obtenue par les commandes

```
library(abc)
abc_res = abc(s,abc_param,abc_sumstat,tol=0.1,method="rejection")
hist(abc_res)
```

On peut également étudier les principales caractéristiques de la distribution a posteriori avec la commande

```
summary(abc_res,param=abc_param)
```

Parmi les valeurs m_i retenues, certaines ne sont pas très proches de m et on peut donc penser que le paramètre λ_i associé n'est pas une très bonne estimation de λ. Une solution naturelle pour résoudre ce problème est de réduire la proportion de simulations conservées (la tolérance), mais il faut tout de même retenir un assez grand nombre de valeurs pour estimer la distribution a posteriori, et cette stratégie peut donc vite devenir très inefficace d'un point de vue calculatoire. Une solution alternative est de corriger les valeurs de λ_i retenues en tenant compte de l'écart entre m_i et m. La manière la plus simple d'effectuer cette correction implique une régression linéaire, et peut être effectuée par la commande

```
abc_res = abc(s,abc_param,abc_sumstat,tol=0.1,method="loclinear")
```

Comme précédemment, la distribution a posteriori obtenue peut être résumée par la commande

```
summary(abc_res,param=abc_param)
```

On peut également la représenter de manière graphique par la commande

```
plot(abc_res,abc_param)
```

Comme le montre le graphique en bas à gauche, la distribution postérieure obtenue par régression (en rouge) est nettement meilleure que celle obtenue par rejet (en noir). Lorque cela est possible, on utilise donc en pratique la régression. D'autres outils de régression plus élaborés sont également disponibles dans la librairie, mais nous ne les évoquerons pas ici.

Pour déterminer si la méthode ABC mise en place (choix des statistiques, nombre de répétitions, tolérance ...) permet d'estimer correctement λ, on peut évaluer l'erreur de prédiction attendue par validation croisée : parmi les *nb_rep* échantillons simulés, on en tire un au hasard, on estime son λ_i par notre procédure ABC basée sur les autres échantillons, et on le compare au vrai λ_i de cet échantillon. On obtient ainsi une estimation de l'erreur pour cet échantillon, et on peut calculer l'erreur moyenne en répétant l'opération un grand nombre de fois. Dans la librairie *abc*, la commande est

```
cv_res=cv4abc(abc_param,abc_sumstat,nval=100,tols=0.01,method="loclinear",statistic="mode")
summary(cv_res)
```

nval indique le nomnbre d'échantillons utilisés pour évaluer l'erreur, et *statistic="mode"* indique que pour chaque échantillon, λ est estimé par la valeur maximisant la distribution a posteriori. Si l'erreur de prédiction calculée par la librairie *abc* est proche de 1, cela signifie que notre procédure n'est pas bonne et que la distribution a posteriori est à peine meilleure que la distribution a priori. Plus elle est proche de 0, plus notre procédure d'estimation est bonne. Ainsi, l'intérêt principal de cette validation croisée est de comparer différentes procédures ABC basées sur le même jeu de données. Par exemple, on peut comparer différentes tolérances à l'aide de la commande

```
cv_res=cv4abc(abc_param,abc_sumstat,nval=100,tols=c(0.01,0.05,0.1,0.2),
method="loclinear",statistic="mode")
summary(cv_res)
```

Quel est la meilleure tolérance d'après ce résultat? On peut également comparer différentes statistiques. Par exemple, reprenez toute la procedure abc en utilisant comme statistique la variance des observations, et indiquez si le choix de cette statistique est approprié.

2. Génétique des populations

Nous allons maintenant appliquer cette stratégie ABC dans les modèles simples de génétique des populations étudiés au cours du TP1. Comme dans ce TP, on considére un échantillon de 100 séquences génomiques, qui comprend 1000 locus indépendants de 1000 paires de bases chacun. Soit p le nombre de sites polymorphes dans cet échantillon, et t le nombre de différences moyen entre deux séquences (ou estimateur de tajima).

Population de taille constante

On considère une population de taille constante, associée à une valeur de θ comprise entre 10^{-4} et 10^{-3}. Pour un échantillon de séquences issu de ce modèle, on a observé les statistiques p et t données dans le fichier *const_sumstat.txt* (chaque étudiant regarde une ligne différente).

Proposez une procédure ABC pour estimer θ à partir de p, et une autre à partir de t. Vous pourrez utiliser pour cela la fonction ci-dessous, qui simule des valeurs de θ, p et t pour le modèle de taille constante considéré ici, en utilisant *coala*.

```
simul_const=function(n,loc,L,nb_rep){
    # n nombre de séquences par échantillon
    # loc nombre de locus par échantillon
    # L taille de chaque locus (en paires de bases)
    # nb_rep nombre d'échantillons simulés
    model=coal_model(sample_size=n,loci_number=loc,loci_length=L,ploidy=1) +
        feat_mutation(rate=par_prior("theta",runif(1,0.1,1))) +
        sumstat_nucleotide_div(transformation=mean) +
        sumstat_sfs(transformation=sum)
    sim_res = simulate(model,nsim=nb_rep)
    abc_param = create_abc_param(sim_res, model)/L
    abc_sumstat = create_abc_sumstat(sim_res, model)
    abc_sumstat[,"pi"]=abc_sumstat[,"pi"]/L
    return(cbind(abc_param,abc_sumstat))
}
```

Comparez ces deux procédures par validation croisée. Chosissez la meilleure des deux et tracez la distribution a posteriori basée sur votre observation (valeur de p ou de t selon la méthode retenue). Comparez ce résultat avec les estimateurs de Watterson et Tajima pour cet échantillon.

Population de taille croissante

On considère maintenant une population dont la taille augmente ou diminue de manière exponentielle selon l'équation $N(t) = N(0) * exp(-G * t)$, où t est en unités de coalescence. La valeur de θ se situe entre 10^{-4} et 10^{-3}, et celle de G entre 0 et 10. Comme précédemment, on observe p et t pour un échantillon de séquences, et les valeurs observées sont cette fois à prendre dans *exp_sumstat.txt*.

Modifiez la fonction *simul_const* de manière à créer une fonction *simul_exp* qui simule des valeurs de θ, G, p et t pour ce modèle. En utilisant cette fonction, proposez une procédure ABC permettant d'estimer à la fois θ et G. Evaluez cette procédure par validation croisée et donnez la distribution a posteriori pour votre échantillon observé. Au vu de ces résultats, peut-on estimer à la fois θ et G à partir d'un échantillon de séquences génomiques? Pouvez vous exclure que le modèle soit en fait de taille constante?