Rapport d'étude d'un algorithme MCMC : l'algorithme HASH

Naila BOUTERFA, Thomas DINH, Wendy LEFEVRE

30 novembre, 2018

Table des matières

Introduction

Cataloguer la variation génétique humaine et comprendre son impact phénotypique est capital entre autres dans la compréhension des bases génétiques de maladies. Nous allons présenter dans ce rapport l'algorithme HASH ou *haplotype assembly for single human* un cas d'application d'algorithme *Markov chain Monte Carlo* (MCMC) à l'inférence d'haplotypes, présenté dans l'article *"An MCMC algorithm for haplotype assembly from whole-genome sequence data"* Bansal et al. (2008).

Dans le contexte d'assemblage d'haplotypes nous avons d'une part un génotype de référence et d'autre part de multiples séquences d'ADN issues d'un chromosome séquencé chez un individu. Chaque séquence lue représentant un fragment, une lecture qui recouvre plusieurs loci variants permet de révéler la combinaison d'alléles présente en ces sites sur le chromosome c'est à dire les haplotypes présents. En utilisant ses recouvrements sur des loci hétérozygotes (les loci homozygotes ne nous intéressent pas) on peut potentiellement assembler les deux haplotypes d'un individu pour une collection de fragments lus. Cette assemblage d'haplotypes représente un nouveau challenge computationel.

L'approche MCMC est une façon d'explorer l'espace des possibles haplotypes pour trouver des reconstructions d'haplotypes vraisemblantes et d'estimer la fiabilité des haplotypes reconstruit.

Nous verrons dans un premier temps comment cette situation est modélisée ensuite pour des données simulées dont nous connaissons le taux d'erreur nous calculerons la loi à posteriori exacte des haplotypes afin de la comparer dans un second temps à la loi à posteriori obtenue grace à l'algorithme MCMC.

Présentation du modèle

Description du Modèle

Nous allons définir le système de notation pour le modèle étudié :

h	Haplotype
H	Paire d'haplotypes composé d'un haplotype h et de son complémentaire $\overline{h}$
n	Nombre de loci hétérozygotes lus
N	Nombre de fragments lus
X_i	Fragment d'ADN, $X_i \in \{0, 1, -\}^n$, avec " $-$ " signifiant que le fragment ne couvre pas le locus correspondant 0 correspond au nucléotide référent et 1 au variant et $1 < i < N$
q	Probabilité d'erreur i.e : de lire un 0 au lieu d'un 1 et de lire un 1 au lieu d'un 0 dans l'estimation de des haplotypes

Simulation et variation des paramètres

Nous simulons pour commencer un premier exemple tiré de l'article (Bansal et al. (2008)) où nous avons n = 5 loci avec les deux haplotypes complémentaires suivants :

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nous avons également N = 6 vecteurs X_i, qui concaténés forment la matrice X, représentatifs des loci lus sur différents fragments d'ADN.

$$X = \begin{pmatrix} 0 & 1 & - & - & - \\ 1 & 0 & - & - & - \\ - & 0 & 0 & - & - \\ 0 & - & 1 & - & - \\ 1 & - & - & - & 1 \\ - & - & - & 0 & 0 \end{pmatrix}$$

Nous allons maintenant simuler la construction d'une matrice X avec des X_i tirés aléatoirement sur les deux haplotypes selon une matrice de design spécifique similaire à celui de la matrice précédente.

Nous pouvons donc observer les matrices simulées pour des taux d'erreur q de 0.8 et 0.01 respectivement.

$$X_{simulation1} = \begin{pmatrix} 0 & 1 & - & - & - \\ 0 & 1 & - & - & - \\ - & 1 & 1 & - & - \\ 1 & - & 0 & - & - \\ 1 & - & - & - & 0 \\ - & - & - & 1 & 1 \end{pmatrix} \qquad X_{simulation2} = \begin{pmatrix} 1 & 0 & - & - & - \\ 0 & 1 & - & - & - \\ - & 0 & 0 & - & - \\ 0 & - & 1 & - & - \\ 0 & - & - & - & 0 \\ - & - & - & 1 & 1 \end{pmatrix}$$

En reconstruisant les haplotypes correspondant aux deux matrices simulées, nous obtenons :

$$H_{simulation1} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & ? & 0 & ? & 0 \end{pmatrix} \qquad H_{simulation2} = \begin{pmatrix} 0 & 1 & 1 & ? & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nous pouvons apercevoir qu'avec un taux d'erreur élevé, la reconstruction des haplotypes se fait plus difficilement, il semblerait donc qu'il faille un q bas.

Loi à postériori

Loi à posteriori exacte

Dans cette partie nous allons expliquer comment se calcule la loi à postériori exacte $\mathbb{P}(H|X,q)$ et montrer l'impact de la variation du nombre de reads N et de la probabilité d'erreur q.

D'après la formule de Bayes, cette quantité se calcule de la façon suivante :

$$\mathbb{P}(H|X,q) = \frac{\mathbb{P}(H) \times \mathbb{P}(X|H,q)}{\sum_{H'} \mathbb{P}(H')\mathbb{P}(X|H',a)}$$

Dans notre modèle, on suppose que les différents H sont distribués uniformément. Par conséquent, $\mathbb{P}(H)$ est constant et vaut $\frac{1}{2^{n-1}}$.

En réalité, un haut désequilibre de liaison (LD) réduit le nombre d'haplotypes distincts possibles pour un individu permettant aux méthodes d'être plus précises par rapport aux cas de faible LD. Les probas d'haplotypes peuvent varier en raison de cela car l'association entre allèles de SNPs voisins (*Single Nucleotide Polymorphism*), se maintenant au bout d'un grand nombre de générations dans des régions courtes de génomes permet cela.

Le calcul de la loi à postériori revient donc à calculer $\mathbb{P}(X|H,q)$ et à le diviser par $\sum_{H'} \mathbb{P}(X|H',q)$ avec H' l'ensemble de tous les 2^{n-1} H possibles.

Nous cherchons donc à calculer $\mathbb{P}(X|H,q)$ pour chacun des H.

Etant donné que X est constitué de fragments X_i indépendants deux à deux on a :

$$\mathbb{P}(X|H,q) = \prod_{i=1}^{N} \mathbb{P}(X_i|H,q)$$

Comme H est composé de deux haplotypes h et $\overline{h}$ alors :

$$\mathbb{P}(X_i|H,q) = \frac{\mathbb{P}(X_i|h,q) + \mathbb{P}(X_i|\overline{h},q)}{2}$$

Il nous faut donc calculer $\mathbb{P}(X_i|h,q)$:

$$\mathbb{P}(X_i|h,q) = \prod_{j:X_i[j]\neq -} \delta_{jj}(1-q) + (1-\delta_{jj})q$$

Avec : $\delta_{jj} = \begin{cases} 1 & \text{si } X_i[j] = h[j] \\ 0 & \text{sinon.} \end{cases}$

En calculant toutes ces quantités, on peut retrouver de manière exacte la loi à postériori $\mathbb{P}(H|X,q)$.

Simulations avec loi à posteriori exacte :

Nous simulons dans un premier temps des calculs de loi à posteriori et observons le classement des haplotypes les plus probables pour des valeurs de q ou N différentes dont nous retrouverons les résultats en table 1.

TABLE 1: Lois à posteriori des haplotypes pour différentes valeurs de q et N

					q=0.01, N=6					q=0.2, N=6					q=0.2, N=18
0	1	1	0	0	0.9603843	0	1	1	0	0	0.3233838	0	1	1	0	0	0.8095272
1	0	0	0	0	0.0193997	1	0	0	0	0	0.1521806	1	0	0	0	0	0.0843635
0	1	1	1	0	0.0193997	0	1	1	1	0	0.1521806	0	1	1	1	0	0.0843635
0	1	0	0	0	0.0003919	0	1	0	0	0	0.0716144	0	1	0	0	0	0.0087918
1	0	0	1	0	0.0003919	1	0	0	1	0	0.0716144	1	0	0	1	0	0.0087918

Nous observons globalement que plus q est petit, plus grande est la probabilité associée au vrai haplotype, ce qui est normal la matrice X étant générée avec moins d'erreurs. Cela est également vrai pour N grand car on a plus de reads et donc plus d'informations sur le véritable haplotype.

En plus de la table 1 qui a permis d'observer un ordre d'haplotype les plus probables pour des paramètres différents, nous traçons deux courbes montrant l'évolution de la probabilité à posteriori du vrai haplotype en faisant varier q dans un premier temps, puis N dans un second temps.

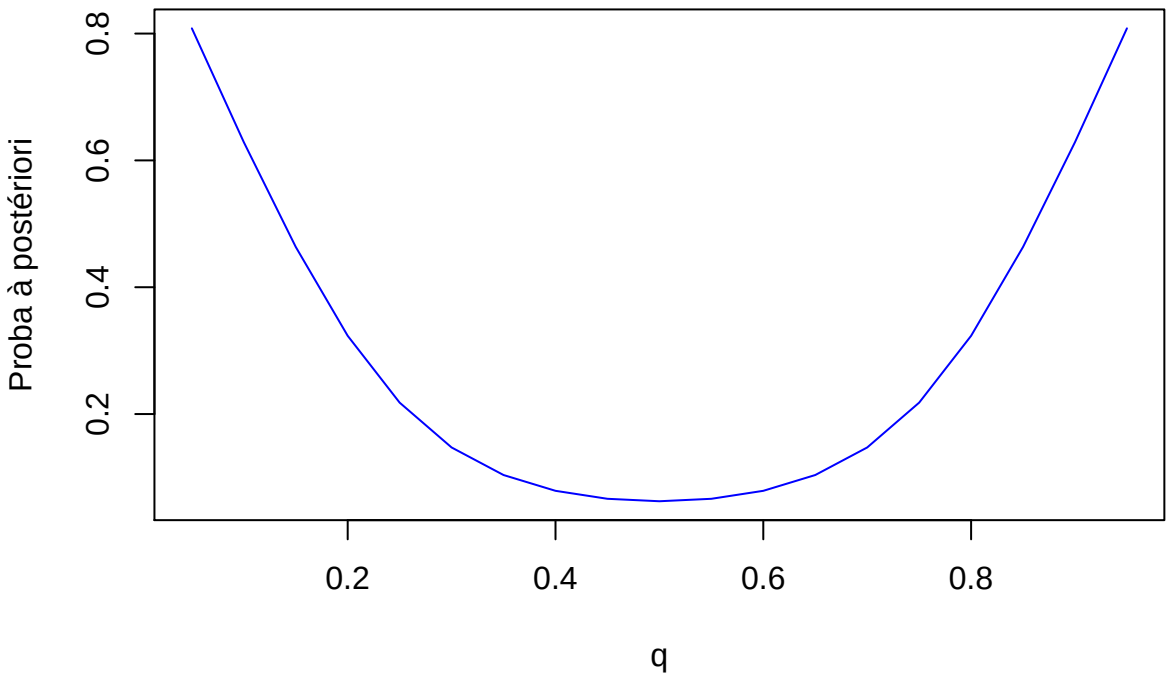

FIGURE 1 – Probabilité du vrai haplotype (0,1,1,0,0) en fonction de q (N=6)

Nous observons dans la figure 1 que plus l'erreur q est éloigné de 0.5, plus la probabilité à postériori du vrai h est grande et qu'avec l'augmentation du taux d'erreur, on retrouve également une grande probabilité à postériori pour le vrai h ce qui peut sembler contre-intuitif mais cela s'explique ici par le fait que notre H est construit de sorte à avoir un haplotype h et son complémentaire $\overline{h}$, par conséquent si q est grand on aura une grande probabilité de construire l'haplotype complémentaire à celui pris au hasard. Ce qui ne serait pas observable dans le cas d'haplotypes non complémentaires ou en l'absence de modalités binaires.

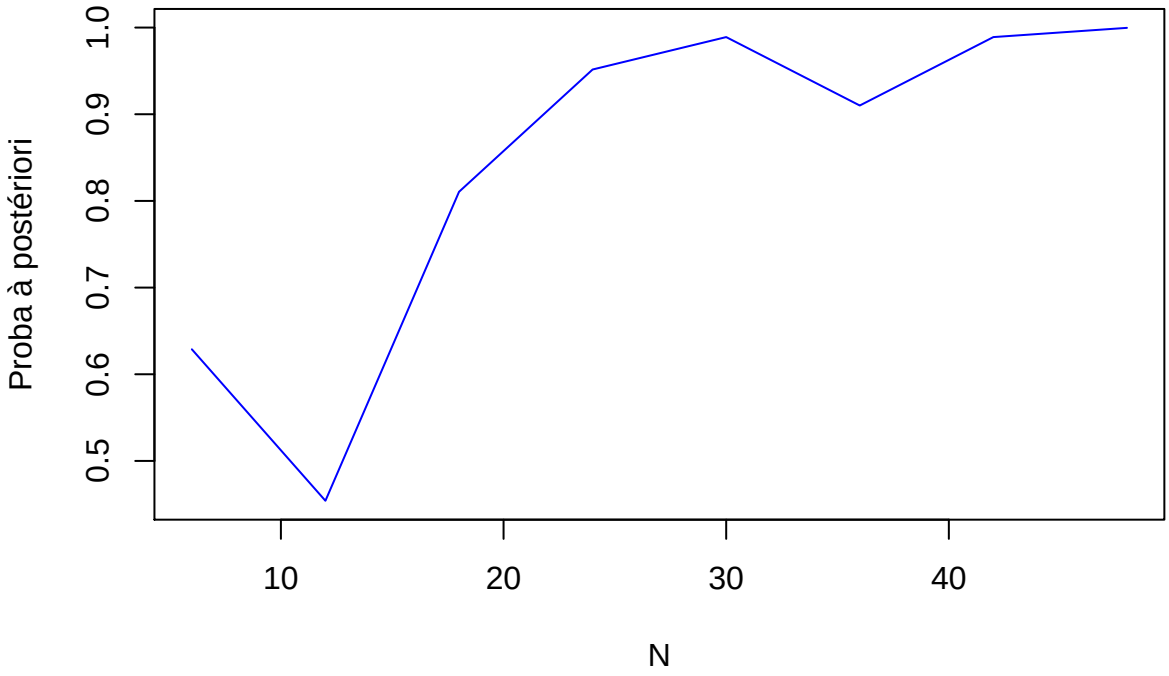

FIGURE 2 – Probabilité du vrai haplotype (0,1,1,0,0) en fonction de valeur de N (q = 0.1)

Nous remarquons également dans la figure 2 qu'augmenter le nombre de reads augmente cette probabilité ce qui est logique car on a plus d'informations sur le génotype.

Loi à postériori avec MCMC

Dans l'idéal, le calcul de la loi à postériori est toujours meilleur pour déterminer l'haplotype le plus probable selon les fragments d'ADN observés, cependant le coût en calcul s'élève très vite et est de l'ordre de 2^{n-1} donc lorsque le nombre de loci hétérozygotes à lire augmente, le temps de calcul de la loi à postériori risque de grandement augmenter.

L'algorithme MCMC consiste à comparer à chaque itération les lois à postériori d'un nouvel haplotype à l'ancien (l'étape *proposal*) et de choisir parmi les deux celui qu'on garde selon une probabilité α (l'étape *acceptance*). Ainsi il n'est donc pas nécessaire de faire une énorme quantité de calculs mais seulement de l'ordre du nombre d'itérations choisies.

Application au modèle de l'algorithme MCMC

Nous allons voir ici deux cas de matrices X de dimensions différents sur lesquels nous mettrons en pratique l'algorithme MCMC. Les possibilités d'haplotypes diffèrent énormement selon dimensions et le cardinal des haplotypes à considérer est de 2^{n-1}

On pose : "2" car on a le choix entre un haplotype référent ou variant (0 ou 1).

On pose : "n-1" au lieu "n" car un haplotype contient une information complémentaire à son symétrique.

Cas n°1

Nous allons, dans un premier temps travailler sur un cas simple et appliquer l'algorithme MCMC afin de comprendre ces mécanismes et de s'assurer de son fonctionnement. Nous avons $n = 5$ ce qui implique $2^4 = 16$ possibilités d'haplotypes différents et $N = 6$

Les données que nous avons sont présentées ainsi :

$$X = \begin{pmatrix} 0 & 1 & - & - & - \\ 1 & 0 & - & - & - \\ - & 0 & 0 & - & - \\ 0 & - & 1 & - & - \\ 1 & - & - & - & 1 \\ - & - & - & 0 & 0 \end{pmatrix}$$

Avec :

$$H = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Nous prenons pour paramètre q=0.05 avec 10000 itérations et en supprimant 2000 lignes pour ne pas altérer les résultats avec les premières itérations superficielles et non informatives, influencées par une condition initiale prise arbitrairement.

Comparaison de la loi à postériori exacte aux résultats de l'algorithme MCMC

La table 2 permet de comparer la loi à posteriori et l'algorithme MCMC. On y observe les haplotypes retenus et à côté la probabilité d'apparition correspondante.

TABLE 2: Tableau de probabilités décroissantes d'haplotypes inférés par MCMC et par calcul de loi à posteriori exacte

					Proba_à_postériori						Proba_MCMC
0	1	1	0	0	0.6287989	0	1	1	0	0	0.618000
1	0	0	0	0	0.1380290	1	0	0	0	0	0.153125
0	1	1	1	0	0.1380290	0	1	1	1	0	0.136750
0	1	0	0	0	0.0302991	0	1	0	0	0	0.029125
1	0	0	1	0	0.0302991	1	0	0	1	0	0.027625
0	0	0	0	0	0.0066510	0	0	0	0	0	0.010000
0	0	1	0	0	0.0066510	0	1	0	1	0	0.006875
1	0	1	0	0	0.0066510	0	0	1	0	0	0.005875
0	1	0	1	0	0.0066510	1	0	1	0	0	0.004750
1	1	0	0	0	0.0014600	0	0	0	1	0	0.002375
1	1	1	0	0	0.0014600	1	1	0	0	0	0.001875
0	0	0	1	0	0.0014600	0	0	1	1	0	0.001250
0	0	1	1	0	0.0014600	1	1	1	0	0	0.001125
1	0	1	1	0	0.0014600	1	0	1	1	0	0.000875
1	1	0	1	0	0.0003205	1	1	0	1	0	0.000250
1	1	1	1	0	0.0003205	1	1	1	1	0	0.000125

Nous remarquons que les probabilités associées au vrai haplotype sont quasiment similaires : 0.63 pour la loi à posteriori contre 0.62 pour l'algorithme MCMC. Aussi, les autres haplotypes sont pratiquement retrouvés dans le même ordre sauf pour les haplotypes les moins probables que de toute manière nous negligeons. On peut donc en déduire que la méthode de l'algorithme est relativement fiable.

Nous allons maintenant voir comment les résultats peuvent varier selon la condition initiale h. Pour cela nous définissons la notion de score : à chaque itération, on calcule le log de la proba de X sachant H et à chaque H, on associe un score qui lui est propre. On calcule le score cumulé à la fin de l'algorithme et si les scores de deux algorithmes différents sont les mêmes à la fin, cela signifie que le pourcentage des H qui sont apparus sont quasiment les mêmes et donc que les algorithmes ressortent à la fin la même loi à postériori.

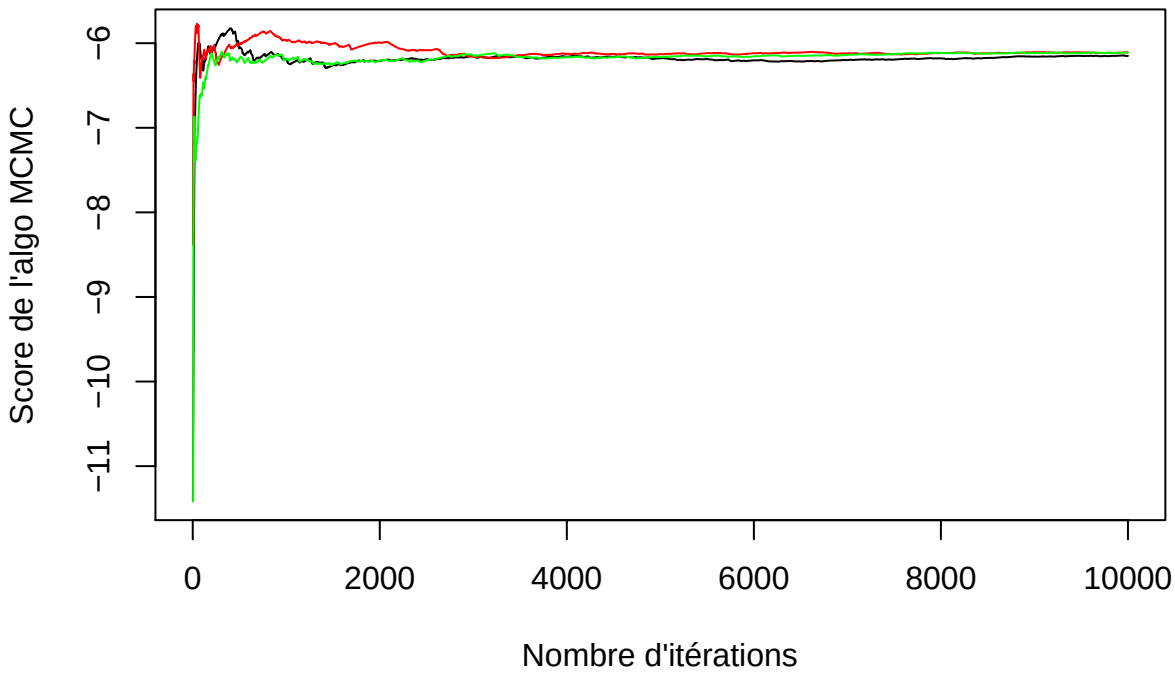

FIGURE 3 – Graphique de convergence de l'algorithme MCMC pour des haplotypes initiaux différents

Dans la figure 3, nous observons que quelque soit le h pris initialement, avec un nombre d'itérations suffisant l'algorithme MCMC finit par converger vers le même score et donc la même loi à postériori après un certain nombre d'itérations. Par conséquent, on peut penser que l'algorithme est stable.

Cas n°2

Ce deuxième exemple servira à mettre en application l'algorithme MCMC dans le cas d'un plus grand nombre n de loci lus dans l'haplotype, c'est un cas plus réaliste et pertinent d'application.

Dans notre exemple $n = 13$ est le nombre de loci hétérozygotes lus et $N = 16$ est le nombre de fragments lus. La différence est ici de taille par rapport au cas 1 car le nombre de possibilités grandit. Ici, on a $2^{13-1} = 4096$ possibilités d'haplotypes différents. Nous mettons de coté le calcul exact de la loi à postériori devenu fastidieux et appliquons uniquement l'algorithme MCMC.

Les données que nous avons sont présentées ainsi :

$$X = \begin{pmatrix} 0 & 0 & - & - & - & - & - & - & - & - & - & - & - \\ - & - & 1 & 1 & - & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & - & - & - & - & - & - & - & - & - \\ 1 & 1 & 1 & - & - & - & - & - & - & - & - & - & - \\ - & - & 0 & 0 & 1 & - & - & - & - & - & - & - & - \\ - & - & 0 & - & - & 0 & - & - & - & - & - & - & - \\ - & - & - & 0 & - & - & - & - & - & - & 1 & 1 & - \\ - & - & - & 1 & 0 & - & - & - & - & - & - & - & - \\ - & - & - & - & - & 0 & 0 & 0 & - & - & - & - & - \\ - & - & - & - & - & 1 & - & 0 & - & - & - & - & - \\ - & - & - & - & - & 0 & 0 & 1 & - & 0 & 0 & - & - \\ - & - & - & - & - & - & - & 0 & 0 & 0 & - & - & - \\ - & - & - & - & - & - & - & - & 1 & 1 & - & - & - \\ - & - & - & - & - & - & - & - & - & 0 & 0 & 0 & - \\ - & - & - & - & - & - & - & - & - & - & 0 & 0 & 0 \\ - & - & - & - & - & - & - & - & 1 & - & - & - & 1 \end{pmatrix}$$

Avec :

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

La table 3 nous montre les haplotypes les plus probables lors de l'exécution de l'algorithme MCMC. En effet, avec 100000 itérations et un q = 0.01, on remarque que le premier haplotype que l'algorithme a choisi est bien le vrai haplotype. L'algorithme a bien retrouvé ce que nous souhaitions. Comme précédement, nous allons vérifier la convergence de l'algorithme quelque soit le h pris initialement afin de nous assurer de son bon fonctionnement dans ce deuxième cas.

TABLE 3: Tableau de probabilités décroissantes d'haplotypes inférés par MCMC

0	0	0	0	1	1	1	1	1	1	0	0	0	0.6627750
0	0	0	0	0	1	1	1	1	1	0	0	0	0.3366500
1	1	1	1	0	0	0	0	0	0	1	1	0	0.0003375
0	0	0	0	0	1	0	1	1	1	0	0	0	0.0001375
1	1	1	1	1	0	0	0	0	0	1	1	0	0.0001000

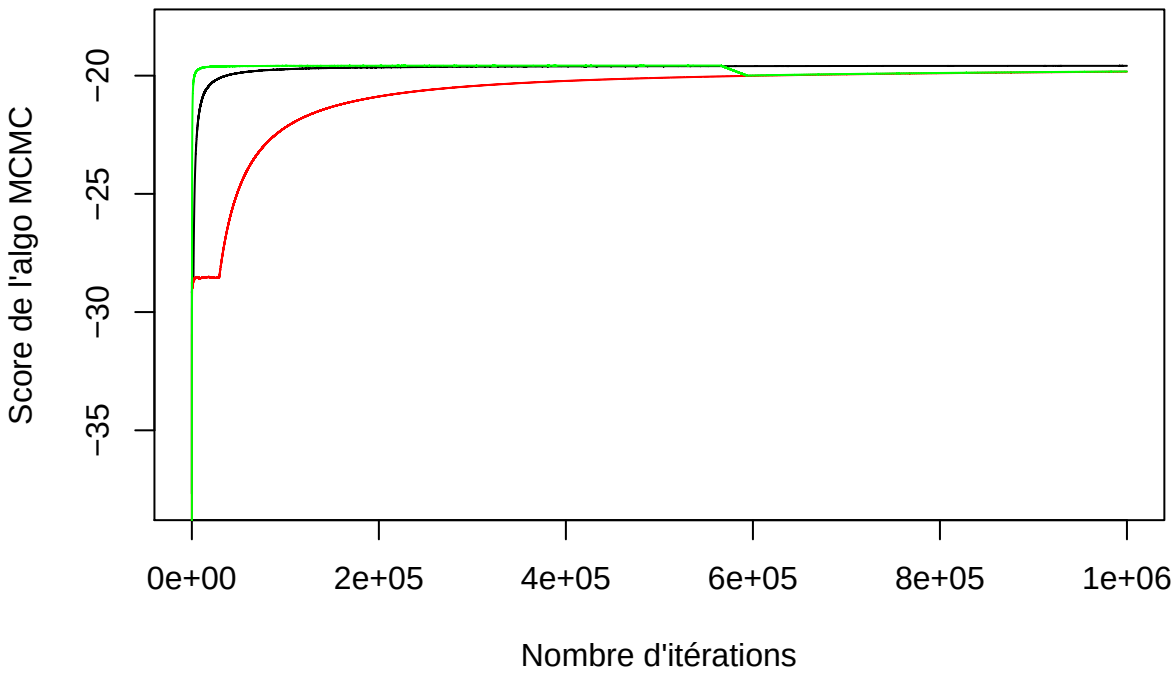

FIGURE 4 – Graphique de convergence de l'algorithme MCMC pour des haplotypes initiaux différents

Dans la figure 4, on observe bien la convergence du score quelque soit le h pris initialement. Cependant on voit qu'on met beaucoup plus de temps pour converger selon la condition initiale mais au final avec un nombre d'itérations suffisant, l'algorithme MCMC finit toujours par converger vers une même loi à postériori.

Conclusion

La comparaison rendue possible dans le premier cas nous a permis de constater l'efficacité de l'algorithme HASH dans l'inference d'haplotypes. Cela nous a donné l'occasion de vérifier si l'algorithme est fiable et de mesurer l'influence des différents paramètres sur l'algorithme. Etant donné que le nombre de possibilités d'haplotypes devient rapidement très grand selon le nombre de loci hétérozygotes lus, le calcul de la loi à posteriori exacte devient bien trop couteux et compliqué. Nous nous satisfaisons donc de l'algorithme MCMC qui est beaucoup plus rapide dans ce genre de calculs. Ainsi, l'algorithme utilisé ici probabilité q faible est un bon outil permettant la reconstruction d'haplotype plus rapide en calcul et qui se contente d'une d'initiatilisation par valeur h aléatoire.

Références

Bansal, Vikas, Aaron L Halpern, Nelson Axelrod, and Vineet Bafna. 2008. "An Mcmc Algorithm for Haplotype Assembly from Whole-Genome Sequence Data." *Genome Research* 18 (8). Cold Spring Harbor Lab : 1336–46.