Projet : Algorithme seqEM

M2 Ingénierie Mathématique et Biostatistique - Université Paris Descartes

Nadia Ouhssaine - Mohand Saïd Hidja

6 octobre 2019

Contents

1 Introduction

Nous allons mener une réflexion autour de l'article de Martin et al. (2010) qui étudie une nouvelle manière d'effectuer le génotypage d'individus à partir de données issues des technologies NGS (Next Generation Sequencing ou séquençage haut débit). Pour cela, les auteurs de l'article ont développé l'algorithme appelé seqEM qui applique l'algorithme Expectation-Maximization de Dempster, Laird, and Rubin (1977), qui est un algorithme itératif permettant de trouver les paramètres du maximum de vraisemblance d'un modèle probabiliste lorsque ce dernier dépend de variable latente non observables.

Pour aborder l'étude de cet article, nous allons dans un premier temps définir le contexte de l'étude et le modèle utilisé, puis nous allons simuler des données et implémenter l'algorithme seqEM, finalement nous nous engagerons dans des expérimentations afin d'en tirer des conclusions quant à l'efficacité de cet algorithme.

2 Contexte

Dans un premier temps, rappelons ce qu'est le séquençage des génomes et plus particulièrement le NGS.
Le séquençage des génomes consiste à lire l'ADN contenu dans les cellules d'un échantillon biologique afin de produire un texte numérique composé d'une succession des lettres A, C, G et T appelés nucléotides. Le NGS est une techonologie récente permettant le séquençage des génomes, il permet en effet de produire plusieurs séquences avec une certaines profondeur de lecture et une fois alignée sur un génome de référence, il permet de compter le nombre de nucléotides variants et le nombre de nucléotides de références.

Cependant, comme toute technologie, les NGS n'ont pas une précision de 100% et les lectures peuvent contenir des erreurs. Ainsi, deux lectures provenant du même locus peuvent être différente en raison de ces erreurs de séquençage. Elles devront donc être prise en compte pour déterminer le génotype des individus.

Les données NGS sont, en effet, utilisé pour le genotype calling de SNPs qui consiste à déterminer le génotype d'individus afin d'identifier les SNPs (= variation de séquence telle que deux chromosomes peuvent différer par un seul nucléotide A, T, C ou G). Pour améliorer le genotype calling, les auteurs de l'article ont donc utilisé l'algorithme appelé seqEM, qui applique l'algorithme EM et qui prend en compte l'erreur α de séquençage dont on a parlé précédement.

3 Modèle

Avant que nous abordions la simulation de l'étude, nous allons définir le modèle et les variables qui lui sont associées.

$\forall$ individu $i \in 1, ...,$n :

- Données initiales

$N_i \in \mathbb{N}^* =$ **Profondeur de lecture de l'individu i**

$N_i \sim \mathcal{P}(\lambda)$ ou $N_i \in \mathbb{N}$ fixé

$N = (N_1, N_2, ..., N_n)$

- Variables observées

$X_i \in \{0, 1, 2, ..., N_i\} =$ **Nombre de nucléotides variants au locus**

$X = (X_1, X_2, ..., X_n)$

$X_i \sim B(N_i\ , \alpha)$ si l'individu est homozygote RR

$X_i \sim B(N_i\ , 1 - \alpha)$ si l'individu est homozygote VV

$X_i \sim B(N_i\ , 0.5)$ si l'individu est hétérozygote RV

où α est le taux d'erreurs.

Notons que par soucis de simplicité, l'auteur de l'article a décidé que les erreurs causés par les données NGS, c'est à dire que V soit faussement appelé R et que R soit faussement appelé V, sont de probabilité égale.

- Variables non observées

$S_i \in \{1, 2, 3\}$ = **Génotype de l'individu**

$1 : RR, 2 : RV, 3 : VV$

$S = (S_1, S_2, ..., S_n)$

On note :

$\mathbb{P}\ (S_i = 1|p) = p_{rr} = 1 - p_{vv} - p_{rv}$

$\mathbb{P}\ (S_i = 2|p) = p_{rv}$

$\mathbb{P}\ (S_i = 3|p) = p_{vv}$

où p_{rr} est la proportion de génotype RR
p_{rv} est la proportion de génotype RV
p_{vv} est la proportion de génotype VV
$p = (p_{rr}, p_{rv}, p_{vv})$

Le modèle est également décrit par la loi conjointe suivante :

$$\mathbb{P}\ (X, S \mid \theta) \overset{\text{i indép.}}{=} \prod_i \mathbb{P}\ (X_i\ , S_i \mid \theta) = \prod_i \mathbb{P}\ (S_i \mid \theta)\ \mathbb{P}\ (X_i \mid S_i\ , \theta)$$

où $\theta = (\alpha, p_{vv}, p_{rv})$

Remarque : Il n'y a pas de p_{rr} dans θ car la condition tel que $p_{rr} + p_{rv} + p_{vv} = 1$ permet de substituer p_{rr} par $1 - p_{rv} - p_{vv}$.

D'après ce qui précède, on a :

$\mathbb{P}\ (X_i \mid S_i = 1\ , \theta) = \binom{N_i}{X_i} \alpha^{X_i} (1 - \alpha)^{N_i - X_i}$

$\mathbb{P}\ (X_i \mid S_i = 2\ , \theta) = \binom{N_i}{X_i} (\frac{1}{2})^{N_i}$

$\mathbb{P}\ (X_i \mid S_i = 3\ , \theta) = \binom{N_i}{X_i} (1 - \alpha)^{X_i} \alpha^{N_i - X_i}$

Par conséquent,

$\mathbb{P}\ (X_i\ , S_i = 1 \mid \theta) = \mathbb{P}\ (S_i = 1 \mid \theta)\ \mathbb{P}\ (X_i \mid S_i = 1\ , \theta) = \binom{N_i}{X_i} \alpha^{X_i} (1 - \alpha)^{N_i - X_i} p_{rr} = \binom{N_i}{X_i} \alpha^{X_i} (1 - \alpha)^{N_i - X_i} (1 - p_{rv} - p_{vv})$

$\mathbb{P}\ (X_i\ , S_i = 2 \mid \theta) = \mathbb{P}\ (S_i = 2 \mid \theta)\ \mathbb{P}\ (X_i \mid S_i = 2\ , \theta) = \binom{N_i}{X_i} (\frac{1}{2})^{N_i} p_{rv}$

$\mathbb{P}\ (X_i\ , S_i = 3 \mid \theta) = \mathbb{P}\ (S_i = 3 \mid \theta)\ \mathbb{P}\ (X_i \mid S_i = 3\ , \theta) = \binom{N_i}{X_i} (1 - \alpha)^{X_i} \alpha^{N_i - X_i} p_{vv}$

4 Simulation

Nous allons commencer à simuler la profondeur de lecture N en utilisant la loi de poisson et les génotypes S en tenant compte des proportions p_{rr}, p_{rv} et p_{vv} des différents génotypes puis nous terminerons par la simulation de X.

4.1 Simulation de S

Dans un premier temps, nous allons fixer arbitrairement les paramètres n, p, et choisir un taux d'erreur de séquençage α pour pouvoir simuler par la suite les variables observées X_i.

```
# simulation

set.seed(12345)

# Nombre d'individu
n=2000
# Erreur
alpha=0.15

# Paramètre arbitraire
p=c(0.80,0.15,0.05) # ordre RR:1 RV:2 VV:3

#profondeur de lecture
N=rpois(n,lambda=20)

#Génotype des individus
s=sample(c("RR","RV","VV"),size=n,prob=p,replace=TRUE)
```

Le tableau ci-dessous rassemble les proportions des génotypes générées par la simulation (S) et les vrais proportions des génotypes (p). Nous constatons que les proportions estimées sont très proche des vrais proportions.

Table 1: Verification simple.

	RR	RV	VV
Counts	1617.0000	294.000	89.0000
Estimate	0.8085	0.147	0.0445
True p	0.8000	0.150	0.0500

4.2 Simulation de X

Nous allons simuler X selon les génotypes des individus $S_{\text{i}} \in \{RR, RV, VV\}$:

- pour un individu hétérozygote S_{i}=RV, le nombre de variants $Xi \sim B(N_{\text{i}},1/2)$. La distribution binomiale du nombre de variants est symétrique et ne dépend que de la profondeur de lecture observée.
- pour un individu homozygote S_{i}=VV, le nombre de variants $Xi \sim B(N_{\text{i}},1-\alpha)$ et pour un individu homozygote S_{i}=RR,le nombre de variants $Xi \sim B(N_{\text{i}},\alpha)$. La distribution binomiale dépend donc du taux d'erreur α de lecture des nucléotides ainsi que de la profondeur de lecture N.

```
#création du vecteur X qui contient le nombre de variants par individu
x=rep(NA,n)

#index des RR dans s afin d'être réutilisé pour l'indexation de X
idx=(s=="RR")
```

```r
#simulation de X pour les RR selon la loi binomiale
x[idx]=rbinom(n=sum(idx),size=N[idx],prob=alpha)

#idem
idx=(s=="RV")
x[idx]=rbinom(n=sum(idx),size=N[idx],prob=0.5)

#idem
idx=(s=="VV")
x[idx]=rbinom(n=sum(idx),size=N[idx],prob=1-alpha)
```

Nous allons afficher la distribution des vrais génotypes d'après la simulation faite précédement.

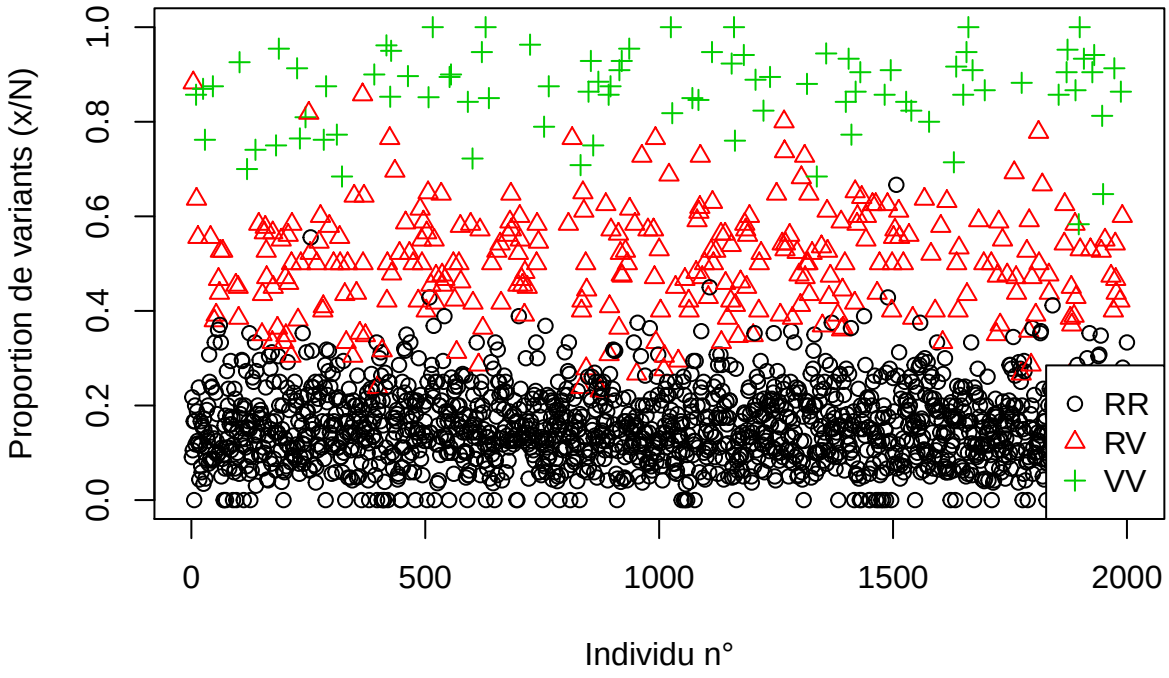

Figure 1: le ratio Xi/N en ordonné pour la simulation avec une profondeur de lecture suivant la loi de poisson de paramètre 20 et 15 pourcent de taux d'erreur. La couleur correspond au vrai genotype.

Dans un premier temps, nous constatons que les différents groupes de génotypes se distinguent très bien. Nous pouvons aussi lire, sans surprise, que le groupe des génotypes RR ont une faible proportion de variants (globalement entre 0 et 30%), le groupe des génotypes RV ont quant à eux une proportion moyenne de variants (globalement entre 40 et 60%), finalement le groupe les génotypes VV ont une forte proportion de variants (globalement entre 70 et 100%).

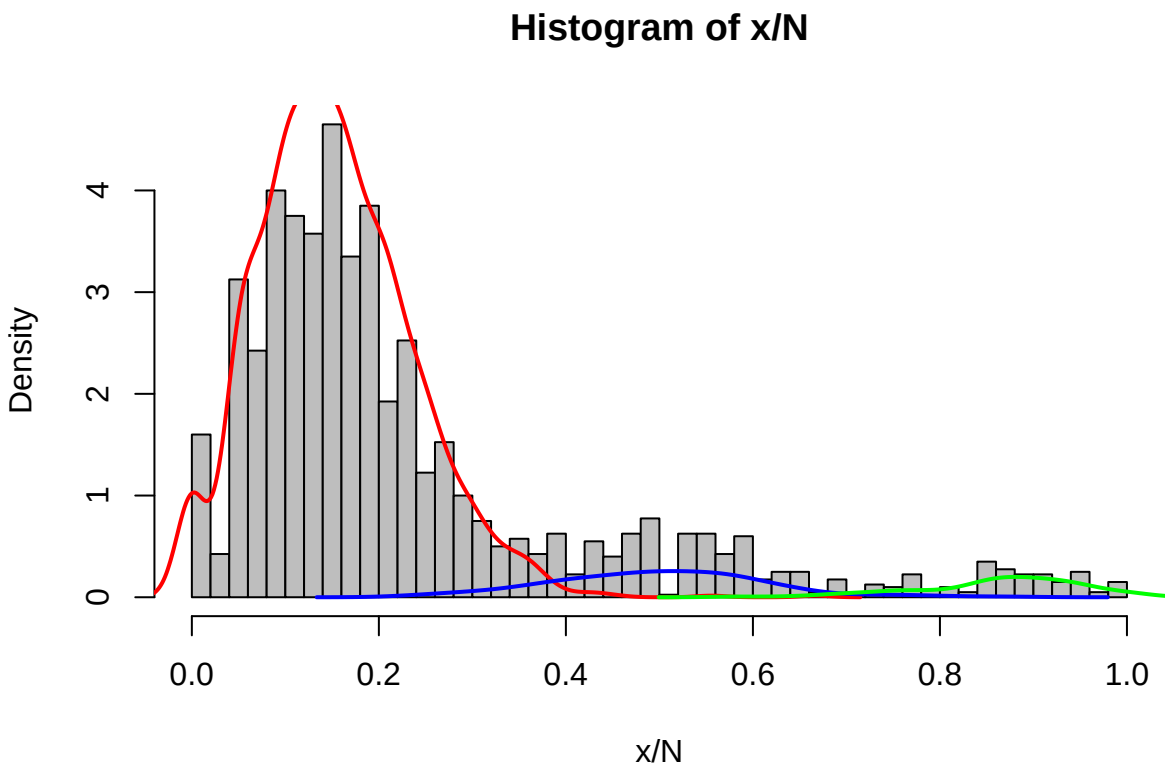

Figure 2: Densité empirique (bâtons) vs densité théorique (lignes) pour les génotypes RR, RV et VV

Nous pouvons voir sur l'histogramme la comparaison de la densité empirique (barres grises) et la densité théorique pour le génotype RR (ligne rouge), le génotype RV (ligne bleue) et le génotype VV (ligne verte) de nos données simulées.

Nous remarquons trois "bosse", la plus haute concerne le génotype RR, celle du milieu bien moins haute concerne le génotype RV, suivi d'une bosse encore plus basse qui concerne le génotype VV. La différence d'hauteur des bosses est simplement dû au déséquilibre de proportions entre les trois groupes de génotypes. Nous constatons par ailleurs que la densité empirique semble bien s'ajuster à la densité théorique dans le cas du génotype RR et VV (ligne rouge et verte), en revanche les données semblent plus dispersées pour le génotype RV et s'éloigne de la densité théorique.

Maintenant que nous avons définit et simulé le modèle, nous pouvons procéder à l'établissement de l'algorithme seqEM. Pour aborder cet algorithme, nous parlerons de la loi a posteriori qui nous sera très utile.

5 Loi a posteriori

La loi a posteriori est en effet utile pour deviner le génotype d'un individu à partir d'une observation.
Elle nous sera également utile afin de déterminer l'espérance de la vraisemblance $Q(\theta|\theta^{\text{old}})$ qui est une phase de notre algorithme seqEM.

D'après la formule de Bayes, on a la loi a posteriori suivante :

$$
\begin{aligned}
\Rightarrow \eta_{\mathrm{i}}(j) &= \mathbb{P}(S_{\mathrm{i}}|X_{\mathrm{i}},\theta) \\
&= \frac{\mathbb{P}(S_{\mathrm{i}}|\theta)\mathbb{P}(X_{\mathrm{i}}|S_{\mathrm{i}},\theta)}{\mathbb{P}(X_{\mathrm{i}}|\theta)} \\
&= \frac{p_{\mathrm{j}}\mathrm{dbinom}(X_{\mathrm{i}},N_{\mathrm{i}},prob \in (\alpha,1-\alpha,0.5))}{\sum_{\mathrm{k}} p_{\mathrm{k}}\mathrm{dbinom}(X_{\mathrm{i}},N_{\mathrm{i}},prob \in (\alpha,1-\alpha,0.5))}
\end{aligned}
$$

où $j,k \in \{1,2,3\}$ (ou $j,k \in \{RR,RV,VV\}$)

Pour comprendre davantage l'utilité de la loi a posteriori, observons les résultats obtenus dans la table ci-dessous pour 4 observations pour lesquelles nous avons choisis arbitrairement le nombre de variants au locus.
Nous pouvons voir avec le paramètre θ = (p = (0.8,0.15,0.05), α=0.15) qu'un individu ayant 5 variants au locus a 88,3% de chances d'avoir le génotype RR, tandis qu'un individu ayant 10 variants au locus a 91,7% de chances d'avoir le génotype RV et pour un individu ayant 20 variants au locus il a quasiment 100% de chances d'avoir le génotype VV.

Table 2: Loi a posteriori pour nos données simulées.

	P(Si = RR | Xi=x)	P(Si = RV | Xi=x)	P(Si = VV | Xi=x)
x=5	0.8830514	0.1169483	0.0000003
x=10	0.0826914	0.9173078	0.0000009
x=15	0.0000153	0.9943822	0.0056025
x=20	0.0000000	0.0002460	0.9997540

Pour approfondir cette réflexion au sujet de la loi a posteriori, remarquons ci-dessous, grâce à la table de contingence qui fait le lien entre les vrais génotypes et les génotypes estimés, le faible taux d'erreurs dans la repartition des génotypes :
Parmi les vrais génotypes "RV", nous avons estimé à tord 42 génotypes. Ce qui nous donne une erreur de 14%.
Parmi les vrais génotypes "RR", nous avons estimé à tord 17 génotypes. Ce qui nous donne une erreur de 1%.
Parmi les vrais génotypes "vv", nous avons estimé à tord 8 génotypes. Ce qui nous donne une erreur de 9%.

Table 3: Vrais génotypes vs génotypes estimés par la loi a posteriori.

	RR	RV	VV
RR*	1600	17	0
RV*	30	252	12
VV*	0	8	81

```
## On obtient au total 67 erreurs sur les 2000 individus
```

Les courbes ROC peuvent également bien illustrer ces dernières erreurs obtenus. Nous constatons, en effet, que les courbes ROC longent l'axes des ordonnées et le haut de graphique pour chaque groupe de génotypes, ce qui est signe d'une bonne estimation de la loi a posteriori.

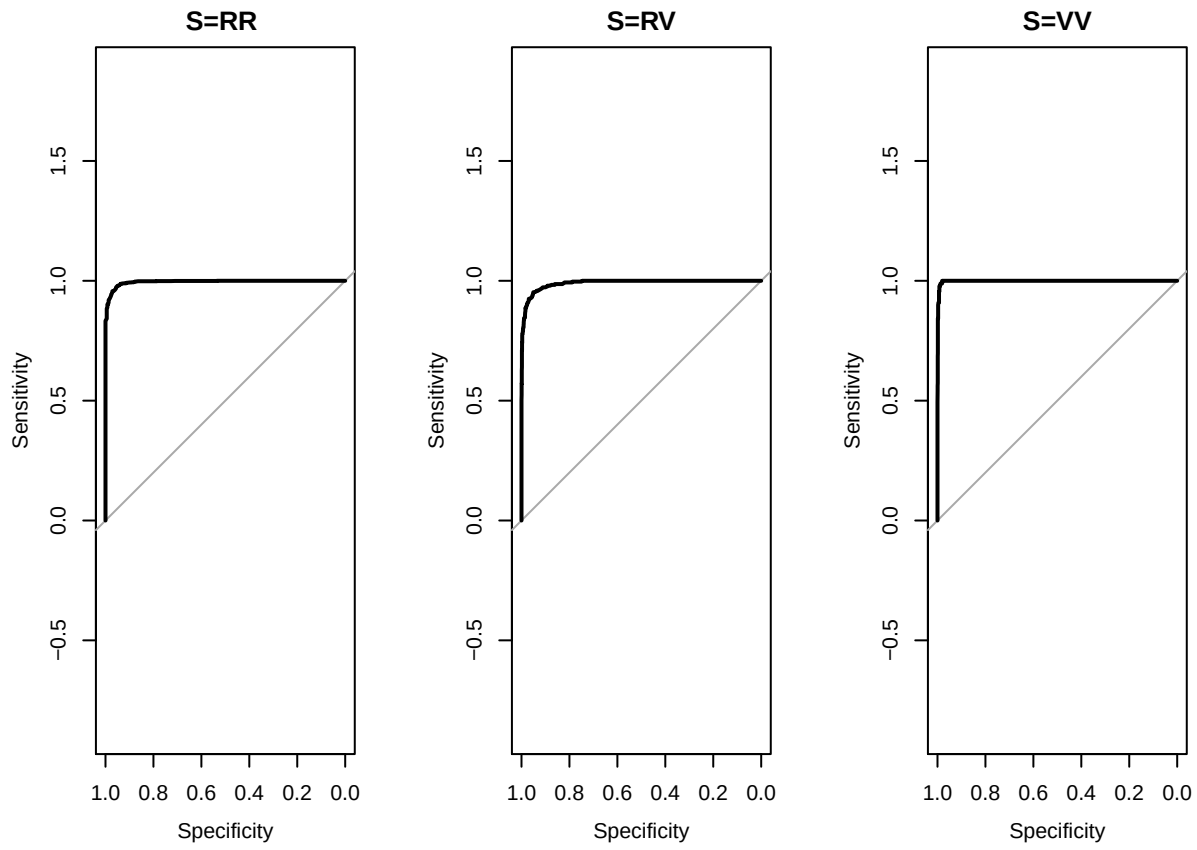

Figure 3: Courbe ROC pour l'estimation par la loi a posteriori des génotypes RR, RV et VV

Ces derniers résultats se retrouvent également graphiquement à l'aide de boxplot. Nous remarquons comme précédement que pour chaque boxplot représentant un groupe de génotype, la loi a posteriori a permis d'en deviner correctement une grande proportion. Par exemple, dans le premier boxplot qui représente le vrai groupe des génotypes RR, presque 100% d'entre eux ont été correctement deviné, nous remarquons que quelques uns ont été considéré comme RV et aucun comme des VV.

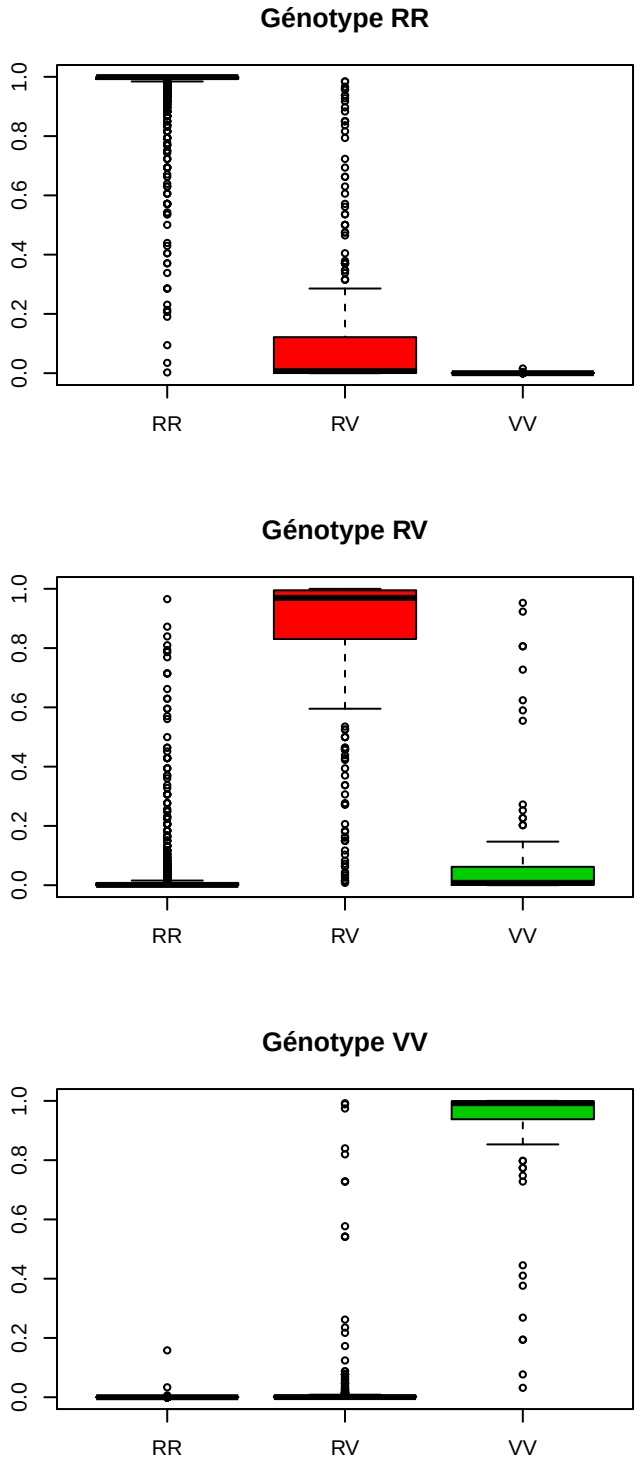

Figure 4: Proportion des génotypes estimés par la loi a posteriori des véritables génotype RR (boxplot 1), RV (boxplot 2) et VV (boxplot 3)

Toutes ces observations nous confirment bien l'utilité évidente de la loi a posteriori pour deviner le génotypes des individus à partir des observations.

6 Calcul et maximisation de $Q(\theta|\theta^{\mathbf{old}})$

Nous avons vu précédemment l'utilité de la loi a posteriori afin de l'utiliser efficacement dans l'élaboration de notre algorithme seqEM.
Afin de préparer son implémentation, nous allons calculer et maximiser l'espérance de la log-vraisemblance $Q(\theta|\theta^{\text{old}})$. Cette dernière nous permet d'estimer le paramètre θ.

6.1 Calcul de $Q(\theta|\theta^{\mathbf{old}})$

$$
\begin{aligned}
\Rightarrow Q(\theta|\theta^{\text{old}}) &= \sum_{\text{s}} \mathbb{P}(S|X,\theta^{\text{old}}) log(\mathbb{P}(X,S|\theta)) \\
&= \sum_{\text{s}} \mathbb{P}(S|X,\theta^{\text{old}}) log(\prod_{\text{i}} \mathbb{P}(S_{\text{i}}|\theta)\mathbb{P}(X_{\text{i}}|S_{\text{i}},\theta)) \text{ , d'après la loi conjointe (cf Modèle)} \\
&= \sum_{\text{i}} \sum_{\text{Si}} \mathbb{P}(S_{\text{i}}|X_{\text{i}},\theta^{\text{old}}) log(\mathbb{P}(S_{\text{i}}|\theta)\mathbb{P}(X_{\text{i}}|S_{\text{i}},\theta)) \\
&= \sum_{\text{i}} [\eta_{\text{i}}(RR) log(\binom{N_{\text{i}}}{X_{\text{i}}} \alpha^{X_{\text{i}}}(1-\alpha)^{N_{\text{i}}-X_{\text{i}}}(1-p_{\text{rv}}-p_{\text{vv}})) + \eta_{\text{i}}(RV) log(\binom{N_{\text{i}}}{X_{\text{i}}} (\frac{1}{2})^{N_{\text{i}}} p_{\text{rv}}) \\
&+ \eta_{\text{i}}(VV) log(\binom{N_{\text{i}}}{X_{\text{i}}} (1-\alpha)^{X_{\text{i}}} \alpha^{N_{\text{i}}-X_{\text{i}}} p_{\text{vv}})] \\
&= \sum_{\text{i}} [\eta_{\text{i}}(RR)(log\binom{N_{\text{i}}}{X_{\text{i}}} + X_{\text{i}} log(\alpha) + (N_{\text{i}}-X_{\text{i}}) log(1-\alpha) + log(1-p_{\text{rv}}-p_{\text{vv}})) \\
&+ \eta_{\text{i}}(RV)(log\binom{N_{\text{i}}}{X_{\text{i}}} + N_{\text{i}} log(\frac{1}{2}) + log(p_{\text{rv}})) + \eta_{\text{i}}(VV)(log\binom{N_{\text{i}}}{X_{\text{i}}} + X_{\text{i}} log(1-\alpha) + (N_{\text{i}}-X_{\text{i}}) log(\alpha) + log(p_{\text{vv}}))] \\
&= \underbrace{\sum_{\text{i}} [\eta_{\text{i}}(RR) log\binom{N_{\text{i}}}{X_{\text{i}}} + \eta_{\text{i}}(RV) log\binom{N_{\text{i}}}{X_{\text{i}}} + \eta_{\text{i}}(VV) log\binom{N_{\text{i}}}{X_{\text{i}}}]}_{\text{constante}} \\
&+ \sum_{\text{i}} [\eta_{\text{i}}(RR) X_{\text{i}} + \eta_{\text{i}}(VV)(N_{\text{i}}-X_{\text{i}})] log(\alpha) + \sum_{\text{i}} [\eta_{\text{i}}(RR)(N_{\text{i}}-X_{\text{i}}) + \eta_{\text{i}}(VV) X_{\text{i}}] log(1-\alpha) \\
&+ \sum_{\text{i}} \eta_{\text{i}}(RR) log(p_{\text{rr}}) + \sum_{\text{i}} \eta_{\text{i}}(RV) log(p_{\text{rv}}) + \sum_{\text{i}} \eta_{\text{i}}(VV) log(p_{\text{vv}}) \\
&= cst. + \sum_{\text{i}} [\eta_{\text{i}}(RR) X_{\text{i}} + \eta_{\text{i}}(VV)(N_{\text{i}}-X_{\text{i}})] log(\alpha) + \sum_{\text{i}} [\eta_{\text{i}}(RR)(N_{\text{i}}-X_{\text{i}}) + \eta_{\text{i}}(VV) X_{\text{i}}] log(1-\alpha) \\
&+ \sum_{\text{i}} \eta_{\text{i}}(RR) log(p_{\text{rr}}) + \sum_{\text{i}} \eta_{\text{i}}(RV) log(p_{\text{rv}}) + \sum_{\text{i}} \eta_{\text{i}}(VV) log(p_{\text{vv}})
\end{aligned}
$$

Nous posons :

$$SS_j = \sum_i \eta_i(j) \quad XX_j = \sum_i \eta_i(j) X_i \quad NN_j = \sum_i \eta_i(j) N_i$$

Nous obtenons alors :

$$\Rightarrow Q(\theta|\theta^{\text{old}}) = \text{cst.} + (XX_{\text{RR}} + NN_{\text{VV}} - XX_{\text{VV}}) \log \alpha + (XX_{\text{VV}} + NN_{\text{RR}} - XX_{\text{RR}}) \log(1-\alpha)$$
$$+ SS_{\text{RR}} \log p_{\text{rr}} + SS_{\text{RV}} \log p_{\text{rv}} + SS_{\text{VV}} \log p_{\text{vv}}$$

6.2 Maximisation de $Q(\theta|\theta^{\text{old}})$

Nous devons maximiser $Q(\theta|\theta^{\text{old}})$, cependant la condition $p_{\text{rr}} + p_{\text{rv}} + p_{\text{vv}} = 1$ concernant le paramètre p de θ doit être respecté. Ainsi, il s'agira plutôt de maximiser $f(\theta, \lambda) = Q(\theta|\theta^{\text{old}}) - \lambda(p_{\text{rr}} + p_{\text{rv}} + p_{\text{vv}} - 1)$

$\Rightarrow f(\theta, \lambda) = Q(\theta|\theta^{\text{old}}) - \lambda(p_{\text{rr}} + p_{\text{rv}} + p_{\text{vv}} - 1)$

$= \text{cst.} + (XX_{\text{RR}} + NN_{\text{VV}} - XX_{\text{VV}}) \log \alpha + (XX_{\text{VV}} + NN_{\text{RR}} - XX_{\text{RR}}) \log(1 - \alpha)$

$+SS_{\text{RR}} \log p_{\text{rr}} + SS_{\text{RV}} \log p_{\text{rv}} + SS_{\text{VV}} \log p_{\text{vv}} - \lambda(p_{\text{rr}} + p_{\text{rv}} + p_{\text{vv}} - 1)$

Effectuons les dérivées partielles selon α, p_{rr}, p_{rv} et p_{vv} :

$\frac{\partial f}{\partial \alpha}(\theta, \lambda) = \frac{XX_{\text{RR}} + NN_{\text{VV}} - XX_{\text{VV}}}{\alpha} - \frac{XX_{\text{VV}} + NN_{\text{RR}} - XX_{\text{RR}}}{1-\alpha} = 0$

$\Rightarrow \alpha = \frac{XX_{\text{RR}} + NN_{\text{VV}} - XX_{\text{VV}}}{NN_{\text{RR}} + NN_{\text{VV}}}$

$\frac{\partial f}{\partial p_{\text{rr}}}(\theta, \lambda) = \frac{SS_{\text{RR}}}{p_{\text{rr}}} - \lambda = 0$

$\Rightarrow p_{\text{rr}} = \frac{SS_{\text{RR}}}{n}$

$\frac{\partial f}{\partial p_{\text{rv}}}(\theta, \lambda) = \frac{SS_{\text{RV}}}{p_{\text{rv}}} - \lambda = 0$

$\Rightarrow p_{\text{rv}} = \frac{SS_{\text{RV}}}{n}$

$\frac{\partial f}{\partial p_{\text{vv}}}(\theta, \lambda) = \frac{SS_{\text{VV}}}{p_{\text{vv}}} - \lambda = 0$

$\Rightarrow p_{\text{vv}} = \frac{SS_{\text{VV}}}{n}$

Nous obtenons donc les formules de chaque paramètre de θ. Elles nous seront utile pour l'implémentation de l'algorithme seqEM.

7 Algorithme seqEM

Nous allons maintenant effectuer l'implémentation de l'algorithme seqEM à partir des données précédemment simulées avec le paramètre $\theta = (\alpha, p_{rr}, p_{rv}, p_{vv}) = (0.15, 0.8, 0.15, 0.05)$.

Rappelons les étapes de l'algorithme EM :

- Initialisation du paramètre θ
- Répéter pour i = $\{1, 2, \dots\}$ jusqu'à convergence de θ les phases E et M suivante :

Phase E : calculer l'espérance $Q(\theta|\theta_{\text{i-1}})$

Phase M : maximiser $Q(\theta|\theta_{\text{i-1}})$ pour obtenir $\theta_{\text{i}} = M(\theta_{\text{i-1}})$

Ainsi, à titre d'exemple, implémentons l'algorithme pour une initialisation $\theta = (\alpha, p_{rr}, p_{rv}, p_{vv}) = (0.1, 1/3, 1/3, 1/3)$

```
theta_star=list(alpha=alpha,p=p)
# initialization
alpha=0.10
p=rep(1/3,3)

# boucle principale
for (iter in 1:20) {
  # E-step
  eta=cbind(
    p[1]*dbinom(x,size=N,prob=alpha), # RR
    p[2]*dbinom(x,size=N,prob=0.5), # RV
    p[3]*dbinom(x,size=N,prob=1-alpha) # VV
```

```
  )
  loglik=sum(log(apply(eta,1,sum)))
  eta=eta/apply(eta,1,sum)
  # sufficient statistics
  NN=apply(eta*N,2,sum)
  XX=apply(eta*x,2,sum)
  SS=apply(eta,2,sum)
  # M-step
  alpha=(XX[1]+NN[3]-XX[3])/(NN[1]+NN[3])
  p=SS/n
  # verbose
  cat("iter=",iter,"loglik=",loglik,"alpha=",alpha,"p=",p,"\n")

}
```

```
## iter= 1 loglik= -5987.208 alpha= 0.1364181 p= 0.7256686 0.2291402 0.04519122
## iter= 2 loglik= -4857.677 alpha= 0.1462416 p= 0.788303 0.1687297 0.04296737
## iter= 3 loglik= -4811.26 alpha= 0.1492083 p= 0.8022949 0.1530311 0.04467402
## iter= 4 loglik= -4807.31 alpha= 0.1501409 p= 0.806226 0.1483285 0.04544548
## iter= 5 loglik= -4806.92 alpha= 0.1504396 p= 0.8074411 0.1468452 0.04571377
## iter= 6 loglik= -4806.88 alpha= 0.1505359 p= 0.8078285 0.1463695 0.04580205
## iter= 7 loglik= -4806.876 alpha= 0.150567 p= 0.8079532 0.1462161 0.04583072
## iter= 8 loglik= -4806.875 alpha= 0.150577 p= 0.8079934 0.1461666 0.04583999
## iter= 9 loglik= -4806.875 alpha= 0.1505802 p= 0.8080064 0.1461506 0.04584299
## iter= 10 loglik= -4806.875 alpha= 0.1505813 p= 0.8080106 0.1461454 0.04584396
## iter= 11 loglik= -4806.875 alpha= 0.1505816 p= 0.808012 0.1461437 0.04584427
## iter= 12 loglik= -4806.875 alpha= 0.1505817 p= 0.8080124 0.1461432 0.04584437
## iter= 13 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.146143 0.0458444
## iter= 14 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.146143 0.04584442
## iter= 15 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
## iter= 16 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
## iter= 17 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
## iter= 18 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
## iter= 19 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
## iter= 20 loglik= -4806.875 alpha= 0.1505818 p= 0.8080126 0.1461429 0.04584442
```

```
theta=list(alpha=alpha,p=p)
rbind(unlist(theta_star),unlist(theta))
```

```
##          alpha        p1        p2         p3
## [1,] 0.1500000 0.8000000 0.1500000 0.05000000
## [2,] 0.1505818 0.8080126 0.1461429 0.04584442
```

Rappelons que le but de l'algorithme seqEM est d'estimer au mieux le paramètre θ qui contient les vrais proportions des génotypes et qui par conséquent nous permet d'établir le génotype de chaque individu.
L'efficacité de l'algorithme se constate en regardant à partir de quelle itération les paramètres vont être le plus proche des vrais paramètres et vont converger.
Ici, nous constatons la convergence du paramètre θ à la 14ème itération et les paramètres sont bien estimés.

8 Expérimentations

Précédemment, nous nous étions chargé de simuler les données et implémenter l'algorithme seqEM en effectuant des verifications quant au bon fonctionnement de l'algorithme. Dans cette partie, nous allons effectuer des expérimentations en faisant varier nos différents paramètres afin de tirer des conclusions intéressante en ce

qui concerne l'algorithme seqEM.

Pour effectuer efficacement nos expérimentations, nous avons créer deux fonctions, simul() et estim(). Ces deux fonctions repennent tout notre cheminement afin de mettre en place l'algorithme. La fonction simul() simule nos données selon le nombre d'individus, le paramètre θ et la profondeur N. La fonction estim() effectue l'algorithme seqEM et s'arrête lorsque le paramètre θ converge.

Le code R de l'implémentation des fonctions simul() et estim() sont en annexe (Annexe 2)

Vérification simple :

```
simul1<-simul(2000,0.15,c(0.8,0.15,0.05),TRUE)
estim1<-estim(simul1$x,simul1$N,0.30,c(0.70,0.20,0.10),50,affiche=TRUE)
```

```
## iter= 1 loglik= -6829.5 alpha= 0.1712032 p= 0.8421655 0.08336632 0.07446813NULL
## iter= 2 loglik= -4960.594 alpha= 0.1558843 p= 0.8140551 0.1232414 0.06270353NULL
## iter= 3 loglik= -4892.361 alpha= 0.1510126 p= 0.7990406 0.1444069 0.05655251NULL
## iter= 4 loglik= -4882.3 alpha= 0.1493607 p= 0.7932257 0.1523631 0.05441123NULL
## iter= 5 loglik= -4881.077 alpha= 0.1488089 p= 0.7911827 0.1551137 0.05370366NULL
## iter= 6 loglik= -4880.939 alpha= 0.1486254 p= 0.790491 0.1560387 0.05347026NULL
## iter= 7 loglik= -4880.923 alpha= 0.1485645 p= 0.7902599 0.156347 0.05339306NULL
## iter= 8 loglik= -4880.922 alpha= 0.1485443 p= 0.7901831 0.1564494 0.05336749NULL
## iter= 9 loglik= -4880.921 alpha= 0.1485376 p= 0.7901576 0.1564834 0.05335901NULL
## iter= 10 loglik= -4880.921 alpha= 0.1485354 p= 0.7901491 0.1564947 0.05335619NULL
## iter= 11 loglik= -4880.921 alpha= 0.1485346 p= 0.7901463 0.1564984 0.05335526NULL
## iter= 12 loglik= -4880.921 alpha= 0.1485344 p= 0.7901454 0.1564997 0.05335495NULL
## iter= 13 loglik= -4880.921 alpha= 0.1485343 p= 0.7901451 0.1565001 0.05335485NULL
## iter= 14 loglik= -4880.921 alpha= 0.1485343 p= 0.790145 0.1565002 0.05335482NULL
## iter= 15 loglik= -4880.921 alpha= 0.1485343 p= 0.7901449 0.1565003 0.0533548NULL
## iter= 16 loglik= -4880.921 alpha= 0.1485343 p= 0.7901449 0.1565003 0.0533548NULL
```

```
rbind(unlist(simul1$theta),unlist(estim1$theta))
```

```
##          alpha        p1        p2        p3
## [1,] 0.1500000 0.8000000 0.1500000 0.0500000
## [2,] 0.1485343 0.7901449 0.1565003 0.0533548
```

La fonction estim() nous montre bien une convergence des paramètres qui sont aussi bien estimés.

8.1 Etude qualitative concernant la simulation des données

Nous allons dans cette partie effectuer des expérimentations concernant les différents paramètres du modèle. Le but est de savoir quels paramètres sont suceptible d'influer sur la qualité des données et donc de rendre l'estimation des génotypes plus difficile. Ainsi, dans chacune des sous parties ci-dessous, nous testerons différentes valeurs pour un paramètre tout en gardant tous les autres paramètres identique.

8.1.1 Variation du nombre d'individu n

Nous allons observer l'impact qu'à la variation du nombre d'individus. Dans un premier cas, nous simulerons les données pour 20 individus et dans un second cas, nous simulerons les données pour 20000 individus. Nous observerons ensuite dans une table la répartition génotypique de ses derniers.

```
simulnpetit<-simul(20,0.15,c(0.8,0.15,0.05),TRUE)
simulngrand<-simul(20000,0.15,c(0.8,0.15,0.05),TRUE)
```

Table 4: Proportions estimées des génotypes vs vrais proportions des génotypes pour 20 individus

	RR	RV	VV
Counts	15.00	4.00	1.00
Estimate	0.75	0.20	0.05
True p	0.80	0.15	0.05

Table 5: Proportions estimées des génotypes vs vrais proportions des génotypes pour 20000 individus

	RR	RV	VV
Counts	15962.0000	3037.00000	1.001e+03
Estimate	0.7981	0.15185	5.005e-02
True p	0.8000	0.15000	5.000e-02

Nous constatons une très mauvaise estimation des génotypes lorsque le nombre d'individu est petit. En revanche, lorsque le nombre d'individu est très élévé l'estimation est très bonne. Il faut donc prendre un grand nombre d'individu pour obtenir une bonne estimation.

8.1.2 Variation de la méthode de simulation de la profondeur de lecture N

Une des questions que l'on se pose pour la simulation de nos données est "Comment simuler la profondeur de lecture ?". Deux choix s'offre à nous, soit la profondeur de lecture est identique pour tous les individus ou bien elle suit la loi de poisson et varie autour d'un certain nombre (ainsi chaque individus a plus ou moins de fragments de lecture que les autres).

```
simulNfixe<-simul(2000,0.15,c(0.8,0.15,0.05),depth=300,FALSE)
simulNfixe2<-simul(2000,0.15,c(0.8,0.15,0.05),depth=10,FALSE)
simulNpoisson<-simul(2000,0.15,c(0.8,0.15,0.05),TRUE)
```

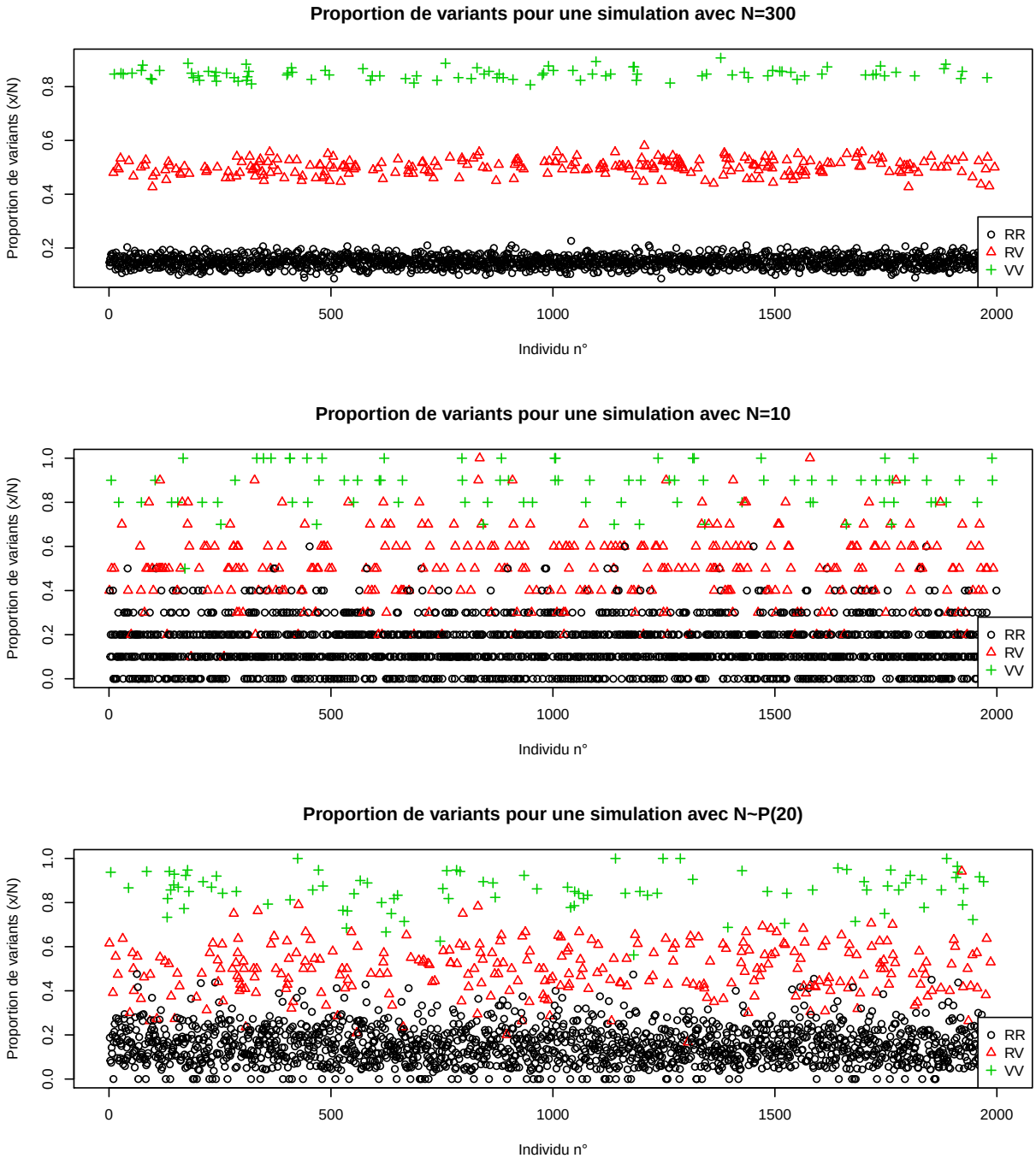

Figure 5: Proportion de variants pour une simulation avec N=300, N=10 et N P(20)

Nous constatons que le choix de fixer N pour tous les individus n'est pas un bon choix car si l'on prend par exemple un très grand N (N=300), les groupes sont trop distinct les uns aux autres et une telle distinction n'est pas représentative de la réalité, par ailleurs appliquer l'algorithme seqEM n'aurai aucune utilité car il est possible de distinguer les différents groupe à l'oeil nu. De même, si nous prenons un très petit N (N=10), les groupes se distingue moins mais ce n'est toujours pas représentatif de la réalité. En revanche, simuler N

selon une loi de poisson est un bon choix, nous distinguons très clairement les différents groupes de génotype, il n'y a néanmoins pas de sauts irréaliste d'un génotype à l'autre.

8.1.3 Variation sur l'erreur de séquençage

Il est évident que plus l'erreur de séquençage est élévé, moins les données sont de qualité et par conséquent l'estimation des génotypes est plus difficile. Ici, nous allons constater l'effet qu'à cette erreur de séquençage et à quelle échelle elle peut rendre notre estimation des génotypes difficile.

```
simulalpha1<-simul(2000,0.15,c(0.8,0.15,0.05),TRUE)
simulalpha2<-simul(2000,0.45,c(0.8,0.15,0.05),TRUE)
```

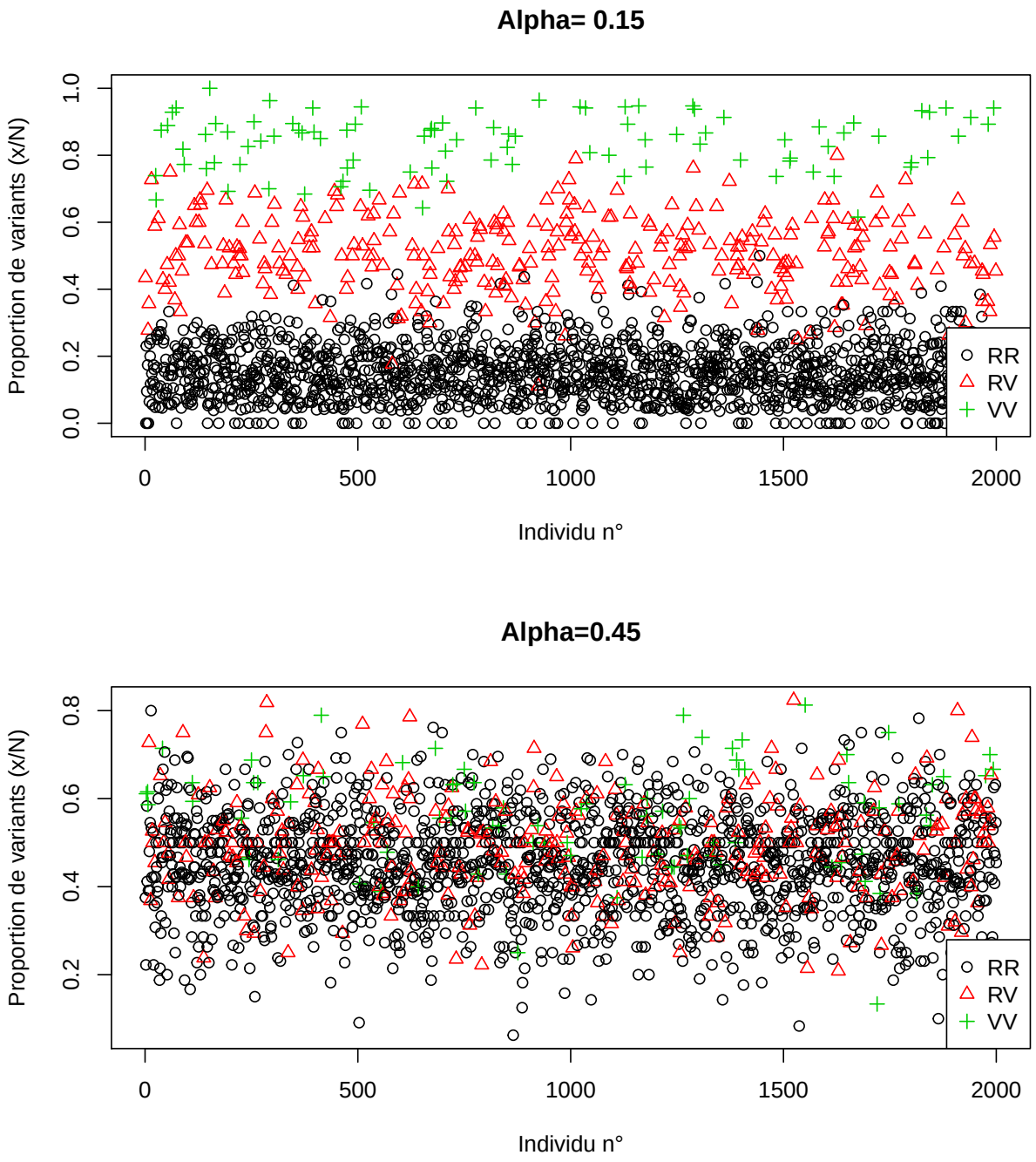

Figure 6: Proportion de variants pour une simulation avec alpha=0.15 et alpha=0.45

Le premier graphe représente la répartition des génotypes avec une erreur de séquençage de 15%, le second graphe représente la répartition des génotypes avec une erreur de séquençage de 45%. On constate sans appel que plus l'erreur de séquençage est élevé, plus la repartition génotypique est mauvaise.

8.2 Etude qualitative concernant l'estimation de θ

Nous allons dans cette partie effectuer des expérimentations concernant l'estimation par l'algorithme seqEM. Le but est de savoir si l'algorithme arrive facilement ou difficilement à estimer le paramètre θ. Pour cela, nous metterons en place deux scenarios qui pour chacun d'eux effectuera 2 séries de 50 simulations et estimations, la première série correspondra à un cas d'estimation dit "facile" et la seconde série correspondra à un cas d'estimation dit "difficile".
En effet, nous avons vu précédemment que les données peuvent être plus ou moins facile à estimer en fonction de plusieurs paramètres (s'il y a plus ou moins d'individus ou si le taux d'erreurs est plus ou moins élevé).

8.2.1 Scenario 1 : erreur de séquençage plus ou moins élevée

Nous allons expérimenter le scenario où nous avons d'une part un taux d'erreurs plutôt faible (α=0.15) et d'autre part un taux d'erreur assez élevé (α=0.3).
C'est deux cas constitueront respectivement le cas dit "facile" et le cas dit "difficile". Le but est d'observer à quelle itération l'algorithme converge, autrement dit nous observerons si l'algorithme a plus ou moins de mal à estimer les génotypes des individus.

Le code R pour effectuer les 50 simulations et estimations pour chacun des cas est en annexe (Annexe 2)

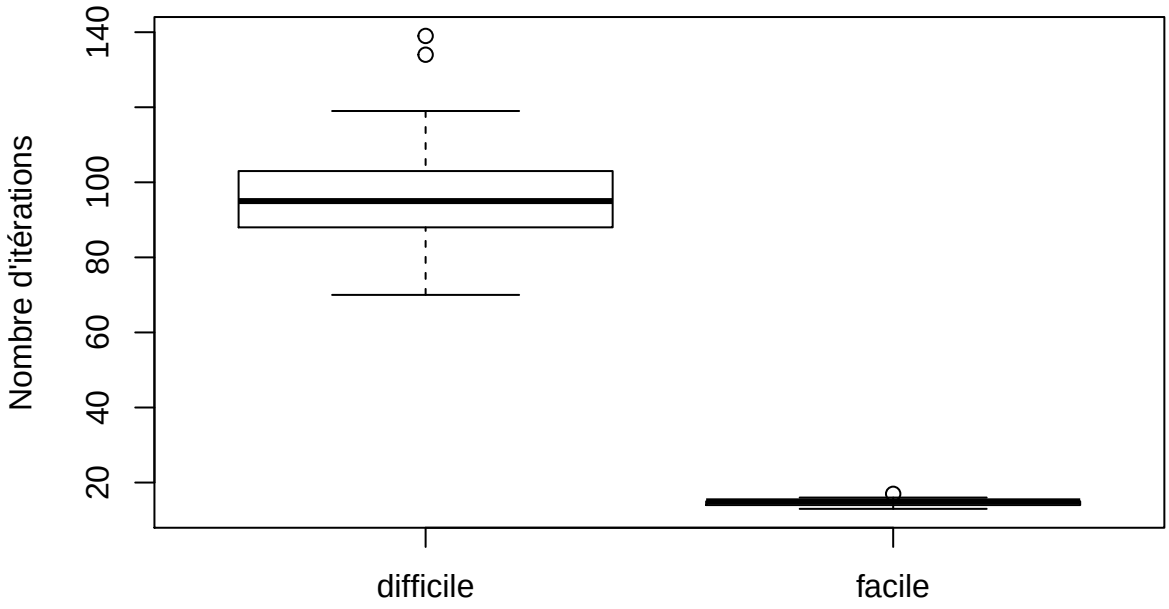

Figure 7: Nombres d'itérations pour un cas d'estimations 'difficile' et un cas d'estimations 'facile'

Nous constatons ici que lorsque nous essayons d'estimer le paramètre θ pour des données dites "difficile" car le taux d'erreurs est élevée alors l'algorithme met en moyenne une centaine d'itérations avant de converger. Tandis que lorsque nous essayons de l'estimer pour des données dites "facile" car le taux d'erreurs est faible alors l'algorithme met en moyenne une dizaine d'itérations avant de converger.

Observons, grâce aux tables ci-dessous, si les estimations pour le cas “difficile” et le cas “facile” sont proche de la vrai valeur de θ.

Table 6: Table de comparaison entre le vrai theta et la moyenne, max et min des 50 theta estimés pour le cas “difficile”.

	Alpha	Prr	Prv	Pvv
True	0.3000000	0.8000000	0.1500000	0.0500000
Mean estimate	0.2995313	0.8022535	0.1484078	0.0493386
Max estimate	0.3080992	0.8490550	0.1870476	0.0657044
Min estimate	0.2872146	0.7704983	0.0971797	0.0365780

Table 7: Table de comparaison entre le vrai theta et la moyenne des 50 theta estimés pour le cas “facile”.

	Alpha	Prr	Prv	Pvv
True	0.1500000	0.8000000	0.1500000	0.0500000
Mean estimate	0.1500847	0.7972919	0.1523575	0.0503506
Max estimate	0.1540679	0.8169761	0.1717950	0.0607011
Min estimate	0.1466206	0.7779511	0.1310697	0.0391515

Nous remarquons qu’en effet les estimations sont de bonne qualités. Le cas “difficile” nécessite plus de temps (car plus d’itérations) mais arrive néanmoins à une correct estimation de θ.

8.2.2 Scenario 2 : Nombre d’individu plus ou moins important

Nous allons expérimenter le scenario où nous avons d’une part un nombre d’individus est plutôt faible (n=10) et d’autre part un nombre d’individus assez élevé (n=2000). C’est deux cas constitueront respectivement le cas dit “difficile” et le cas dit “facile”. A nouveau, le but est d’observer à quelle itération l’algorithme converge, autrement dit on observera si l’algorithme a plus ou moins de mal à estimer les génotypes des individus.

Le code R pour effectuer les 50 simulations et estimations pour chacun des cas est en annexe (Annexe 2)

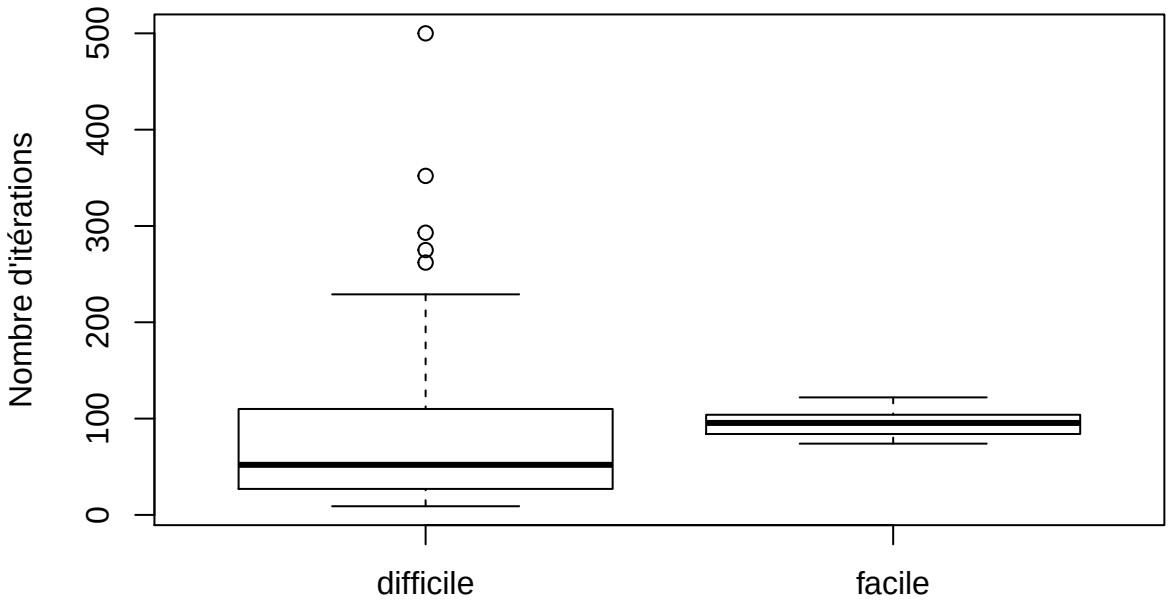

Figure 8: Nombres d'itérations pour un cas d'estimations 'difficile' et un cas d'estimations 'facile'

Nous constatons ici que lorsque nous essayons d'estimer le paramètre θ pour des données dites "difficile" car le nombre d'individus est faible alors l'algorithme met en moyenne 40 itérations avant de converger et peut parfois prendre 200, 300 voir 500 itérations avant de converger. Tandis que lorsque nous essayons de l'estimer pour des données dites "facile" car le nombre d'individus est élevé alors l'algorithme met systèmatiquement moins de 100 itérations avant de converger.

Observons, grâce aux tables ci-dessous, si les estimations pour le cas "difficile" et le cas "facile" sont proche de la vrai valeur de θ.

Table 8: Table de comparaison entre le vrai theta et la moyenne des 50 theta estimés pour le cas "difficile".

	Alpha	Prr	Prv	Pvv
True	0.3000000	0.8000000	0.1500000	0.0500000
Mean estimate	0.2996603	0.0041380	0.0005900	0.0002719
Max estimate	0.3631839	0.0050000	0.0029077	0.0015762
Min estimate	0.2075517	0.0020923	0.0000000	0.0000000

Table 9: Table de comparaison entre le vrai theta et la moyenne, max et min des 50 theta estimés pour le cas “facile”.

	Alpha	Prr	Prv	Pvv
True	0.3000000	0.8000000	0.1500000	0.0500000
Mean estimate	0.2994326	0.8015637	0.1498027	0.0486336
Max estimate	0.3061338	0.8429537	0.1973069	0.0661409
Min estimate	0.2927526	0.7627482	0.0992589	0.0358366

Nous remarquons que les estimations pour le cas “facile” sont de bonne qualité. Cependant, le cas “difficile” qui nécessitait moins de temps (car moins d’itérations) ne nous donne finalement pas d’estimations juste. L’algorithme converge effectivement mais ne converge pas vers les bonnes valeurs.

Ainsi, dans ce scenario où l’on prend un petit nombre d’individu, l’algorithme n’arrive pas à faire correctement son travail. On en conclu donc qu’une limite de l’algorithme serait de prendre un faible nombre d’individus. Cependant, l’algorithme parvient à contrôler un grand taux d’erreurs malgré que cela implique plus d’itérations et par conséquent plus de temps.

9 Conclusion

Théoriquement, l’algorithme seqEM qu’utilise Martin et al. (2010) dans son article semble être un bon moyen d’effectuer le génotypage d’individus à partir de données NGS qui implique un taux d’erreurs de séquençage. Le but de cet algorithme était donc d’estimer non seulement les proportions de chaque génotype mais aussi ce taux d’erreur.
Nos expérimentations nous ont permis de connaitre les limites de telles données et ce qu’elles peuvent impliquer sur l’efficacité de l’algorithme. En effet, nous avons vu qu’avoir un faible nombre d’individus, un grand taux d’erreur de séquençage ou une profondeur de lecture non réaliste rendaient nos données de mauvaise qualité. Nous avons finalement vu que l’estimation sur de telles données dites “difficiles” ont rendu notre algorithme moins efficace et même inefficace (avec une estimation erronée malgré la convergence) dans le cas où l’on a très peu d’individus.

10 Annexes

10.1 Annexe 1 : Implémentation de l'algorithme seqEM

```
# simulation
simul=function(n,
               alpha,
               p=c(0.80,0.15,0.05),depth=25,poisson=TRUE
               ) {

  # Choix sur la simulation de la profondeur de lecture (fixe ou poisson)
  if(poisson == TRUE){
    N=rpois(n,lambda=20)
  }else{
    N=rep(depth,n)
  }

  #Tirage aléatoire avec remise du génotype de chaque individu
  s=sample(c("RR","RV","VV"),size=n,prob=p,replace=TRUE)

  #Affectation du nombre de variant pour chaque individu en fonction du génotype
  #Le nombre de variant est déterminer par la loi binomial dont les paramètres sont propre au génotype
  x=rep(NA,n)
  idx=(s=="RR")
  x[idx]=rbinom(n=sum(idx),size=N[idx],prob=alpha)
  idx=(s=="RV")
  x[idx]=rbinom(n=sum(idx),size=N[idx],prob=0.5)
  idx=(s=="VV")
  x[idx]=rbinom(n=sum(idx),size=N[idx],prob=1-alpha)

  return(list(
    theta=list(alpha=alpha,p=p),
    N=N,
    x=x,
    s=s,#utile pour faire notre vérification simple pour etude de la variation de n
    n=n #idem
  ))
}

# estimation
estim=function(x,N,
               alpha0,
               p0=c(0.80,0.15,0.05),
               itermax=500,affiche=FALSE) {

  for (iter in 1:itermax) {


    # E-step : Etablir la vraisemblance
    eta=cbind(
      p0[1]*dbinom(x,size=N,prob=alpha0), # RR
      p0[2]*dbinom(x,size=N,prob=0.5), # RV
      p0[3]*dbinom(x,size=N,prob=1-alpha0) # VV
    )
```

```
    loglik=sum(log(apply(eta,1,sum)))
    eta=eta/apply(eta,1,sum)
    # sufficient statistics
    NN=apply(eta*N,2,sum)
    XX=apply(eta*x,2,sum)
    SS=apply(eta,2,sum)

    # M-step : maximisation de la vraisemblance et estimation du nouveau paramètre theta
    #avec condition d'arrêt quand theta converge et affichage si TRUE en parametre
    if(round(alpha0,7)==round((XX[1]+NN[3]-XX[3])/(NN[1]+NN[3]),7) && round(p0,7)==round(SS/n,7)){
      if(affiche==TRUE)
        print(cat("iter=",iter,"loglik=",loglik,"alpha=",alpha0,"p=",p0))
      break
    }else{
      alpha0=(XX[1]+NN[3]-XX[3])/(NN[1]+NN[3])
      p0=SS/n
      if(affiche==TRUE)print(cat("iter=",iter,"loglik=",loglik,"alpha=",alpha0,"p=",p0))
    }
  }
  list(
    theta=list(alpha0=alpha0,p0=p0),
    iter=iter,
    loglik=loglik,
    eta=eta
  )


}
```

10.2 Annexe 2 : Implémentation de l'expérience des 50 simulations + estimations

```
#Cas difficile avec alpha = 0.3
nrep=50 #Nombre de simulations
sim1=vector(nrep,mode='list') #List qui comportera tte les simulations
est1=vector(nrep,mode='list') #List qui comportera tte les estimations
itr=c() #vecteur qui comportera tte les itérations
par1=c() #vecteur qui comportera ts les theta estimés
for (rep in 1:nrep) {
  sim1[[rep]]=simul(2000,0.3,c(0.8,0.15,0.05),TRUE)
  est1[[rep]]=estim(sim1[[rep]]$x,sim1[[rep]]$N,alpha0 = 0.1,c(1/3,1/3,1/3),itermax = 500, affiche=FALSE
  itr=cbind(itr,est1[[rep]]$iter)
  par1=rbind(par1,c(est1[[rep]]$theta$alpha0,est1[[rep]]$theta$p0))
}

#Cas facile avec alpha = 0.15
nrep=50
sim2=vector(nrep,mode='list')
est2=vector(nrep,mode='list')
itr2=c()
par2=c()
for (rep in 1:nrep) {
  sim2[[rep]]=simul(2000,0.15,c(0.8,0.15,0.05),TRUE)
```

```
  est2[[rep]]=estim(sim2[[rep]]$x,sim2[[rep]]$N,alpha0 = 0.1,c(1/3,1/3,1/3),itermax = 500, affiche=FALSE
  itr2=cbind(itr2,est2[[rep]]$iter)
  par2=rbind(par2,c(est2[[rep]]$theta$alpha0,est2[[rep]]$theta$p0))
}
```

```
#Cas difficile avec n = 10
nrep=50
sim3=vector(nrep,mode='list')
est3=vector(nrep,mode='list')
itr3=c()
par3=c()
for (rep in 1:nrep) {
  sim3[[rep]]=simul(10,0.3,c(0.8,0.15,0.05),TRUE)
  est3[[rep]]=estim(sim3[[rep]]$x,sim3[[rep]]$N,alpha0 = 0.1,c(1/3,1/3,1/3),itermax = 500, affiche=FALSE
  itr3=cbind(itr3,est3[[rep]]$iter)
  par3=rbind(par3,c(est3[[rep]]$theta$alpha0,est3[[rep]]$theta$p0))
}

#Cas facile avec n = 2000
nrep=50
sim4=vector(nrep,mode='list')
est4=vector(nrep,mode='list')
itr4=c()
par4=c()
for (rep in 1:nrep) {
  sim4[[rep]]=simul(2000,0.3,c(0.8,0.15,0.05),TRUE)
  est4[[rep]]=estim(sim4[[rep]]$x,sim4[[rep]]$N,alpha0 = 0.1,c(1/3,1/3,1/3),itermax = 500, affiche=FALSE
  itr4=cbind(itr4,est4[[rep]]$iter)
  par4=rbind(par4,c(est4[[rep]]$theta$alpha0,est4[[rep]]$theta$p0))
}
```

Références

Dempster, Arthur P, Nan M Laird, and Donald B Rubin. 1977. "Maximum Likelihood from Incomplete Data via the Em Algorithm." *Journal of the Royal Statistical Society: Series B (Methodological)* 39 (1). Wiley Online Library: 1–22.

Martin, Eden R, DD Kinnamon, Michael A Schmidt, EH Powell, S Zuchner, and RW Morris. 2010. "SeqEM: An Adaptive Genotype-Calling Approach for Next-Generation Sequencing Studies." *Bioinformatics* 26 (22). Oxford University Press: 2803–10.