Rapport de stage

Institut national de la recherche agronomique (INRA)
Unité MaIAGE - Jouy-en-Josas

Étude de la distribution des longueurs des lectures produites par les technologies de séquençage de 3e génération

Amina Ghoul

Encadrant : M. Jean-François Gibrat

1er Avril 2019 — 15 Juillet 2019

Remerciements

Avant tout développement sur cette expérience professionnelle, il apparaît opportun de commencer ce rapport de stage par des remerciements, à ceux qui m'ont beaucoup appris au cours de ce stage, et même à ceux qui ont eu la gentillesse de faire de ce stage un moment très profitable.

Aussi, je remercie Monsieur Jean-François Gibrat, mon maître de stage qui m'a formé et accompagné tout au long de cette expérience professionnelle avec beaucoup de patience et de pédagogie. Enfin, je remercie l'ensemble des membres de l'unité MaIAGE pour les conseils qu'ils ont pu me prodiguer au cours de ce stage.

Table des matières

1 Introduction

Dans le cadre de mon master 1 "mathématiques et interactions" de l'université d'Evry-Val-d'Essonne, j'ai effectué mon stage pendant 3 mois et demi dans l'unité MaIAGE de l'Institut national de la recherche agronomique (INRA) situé à Jouy-en-Josas.
Le sujet du stage a concerné l'étude de la distribution des longueurs des lectures produites par les technologies de séquençage de 3^{e} génération.
Une lecture (en anglais read) correspond à la partie séquencé des fragments d'ADN obtenus par fragmentation du génome (shotgun sequencing).
Contrairement aux précédentes générations qui fournissent des lectures de longueur constante et petite (d'au plus 400 paires de bases (bp)), les technologies de troisième génération produisent quant à elles, une distribution de longueurs avec une médiane d'environ 12 kbp (kilo paires de bases) et une queue de distribution qui peut aller jusqu'à 100 kbp pour les lectures les plus longues. Les séquenceurs qui nous ont intéressés lors de ce stage sont le RSII de Pacific Biosciences (Pacbio) et le MinION d'Oxford Nanopore Technologies (MinION)

Ainsi, l'objectif ce stage est de trouver une expression analytique de la distribution des longueurs des lectures issues du séquençage de troisième génération.

Dans un premier temps, il s'agira de présenter l'INRA ainsi que l'équipe au sein de laquelle j'ai effectué mon stage. Et dans un second temps, je décrirai les missions et les activités réalisées lors de ce stage.

2 Contexte du stage

Mon stage a été réalisé à l'INRA (Institut National de la Recherche Agronomique) dans l'unité de recherche "Mathématiques et Informatique Appliquées du Génome à l'Environnement" (MaIAGE), dans l'équipe StatInfOmics (Statistique et Bioinformatique des données Omiques).
Cet institut, fondé en 1946, est un organisme français de recherche en agronomie ayant le statut d'établissement public à caractère scientifique et technologique, et sous la double tutelle du ministère chargé de la Recherche et du ministère chargé de l'Agriculture. Il est le premier institut de recherche agronomique en Europe avec 8417 chercheurs, ingénieurs et techniciens, et deuxième dans le monde en nombre de publications en sciences agricoles et en sciences de la plante et de l'animal. L'INRA contribue à la production de connaissances et à l'innovation dans l'alimentation, l'agriculture et l'environnement.
Il est composé de plus de 200 unités de recherche implantées dans 17 centres en région. L'ambition de l'INRA est, dans une perspective mondiale, de contribuer à assurer une alimentation saine et de qualité, une agriculture compétitive et durable ainsi qu'un environnement préservé et valorisé.
J'ai effectué mon stage dans le centre Ile-de-France de Jouy-en-Josas qui comprend 29 unités dont l'unité MaIAGE. L'unité de recherche MaIAGE regroupe des mathématiciens, des informaticiens, des bioinformaticiens et des biologistes autour de questions de biologie et agro-écologie, allant de l'échelle moléculaire à l'échelle du paysage en passant par l'étude de l'individu, de populations ou d'écosystèmes.
L'unité développe des méthodes mathématiques et informatiques originales de portée générique ou motivées par des problèmes biologiques précis. Elle s'implique aussi dans la mise à disposition de bases de données et de logiciels permettant aux biologistes d'utiliser les outils dans de bonnes conditions ou d'exploiter automatiquement la littérature scientifique.
L'inférence statistique et la modélisation dynamique sont des compétences fortes de l'unité, auxquelles s'ajoutent la bioinformatique, l'automatique et l'algorithmique. Les activités de recherche et d'ingénierie s'appuient également sur une forte implication dans les disciplines destinatrices : écologie, environnement, biologie moléculaire et biologie des systèmes.

3 Présentation de l'équipe StatInfOmics

L'équipe StatInfOmics de l'unité MaIAGE, vise à développer et mettre en oeuvre des méthodes statistiques et bioinformatiques dédiées à l'analyse de données "omiques". D'un point de vue biologique, les questions abordées concernent principalement l'annotation structurale et fonctionnelle des génomes, les régulations géniques, la dynamique évolutive des génomes, et la caractérisation d'écosystèmes microbiens en terme de diversité et de fonctions présentes ; une cible commune étant la relation entre génotype et phénotype. Une part de plus en plus importante de l'activité de cette équipe est relative à l'intégration de données "omiques" hétérogènes pour en extraire de l'information pertinente et aussi prédire des processus biologiques.
D'un point de vue méthodologique, les travaux de cette équipe sont essentiellement d'ordre statistique : estimation de distributions, inférence de modèles à variables latentes, prédiction de relations entre jeux de variables, segmentation, visualisation et classification, avec une attention particulière au cadre de la grande dimension qui caractérise la majorité des jeux de données "omiques" étudiés. Ces recherches s'appuient souvent sur une ingénierie bioinformatique très forte.

4 Missions effectuées

Comme indiqué précédemment, l'objectif du stage est de trouver une expression analytique de la distribution des longueurs des lectures produites par le séquençage de troisième génération.
J'ai effectué plusieurs missions :
Dans un premier temps, j'ai dû me familiariser avec certaines notions concernant le séquençage. Ensuite, j'ai modélisé la distribution de manière empirique, en utilisant le langage python. Enfin, je me suis intéressée à trouver une expression mathématique de cette distribution.

4.1 Le séquençage de l'ADN

Ma première mission a été de comprendre le séquençage de l'ADN et plus particulièrement,le fonctionnement des séquenceurs de troisième génération suivants : Pacific Biosciences RS II(Pacbio) et Oxford Nanopore (MinION). En effet, il était nécessaire de comprendre le principe de ces technologies afin de pouvoir entamer ce stage et répondre à la problématique posée.
Dans un premier temps, l'ADN que l'on souhaite séquencer est cloné, puis il est fragmenté aléatoirement : il s'agit de la méthode shotgun. Ces fragments sont ensuite séquencés. Lors de ce stage nous nous sommes intéressés à la distribution des longueurs des fragments séquencés.

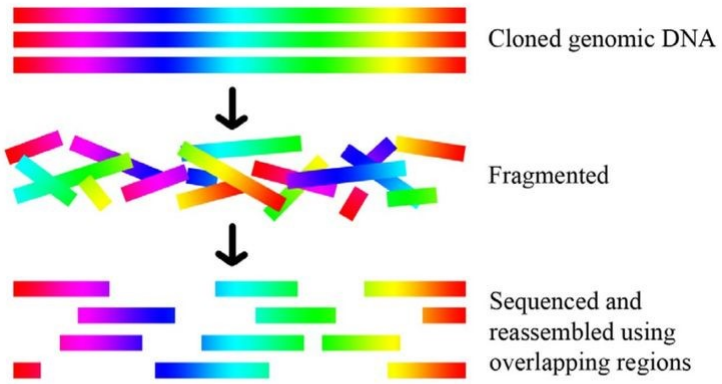

FIGURE 1 – Méthode shotgun

Dans notre cas, les fragments sont séquencés par le Pacbio ou le MinION.
D'une part, la société Pacific Biosciences (Pacbio) a développé une méthode permettant de séquencer des fragments d'ADN. Une enzyme (ADN polymérase) responsable de l'élongation de l'ADN est fixée au fond de puits (Figure 2). En présence d'une molécule d'ADN simple brin, celle-ci va être capable d'incorporer des nucléotides libres complémentaires au brin à analyser, un par un. Les nucléotides ont la particularité d'être marqués par une molécule fluorescente différente (bleu pour G, jaune pour A, vert pour T et rouge pour G). Lorsque l'enzyme incorpore un nucléotide marqué, elle le retient suffisamment longtemps pour qu'un signal soit détecté sous la forme d'un plateau (Figure 2). Le fluophore est ensuite clivé avant l'incorporation d'un autre nucléotide.

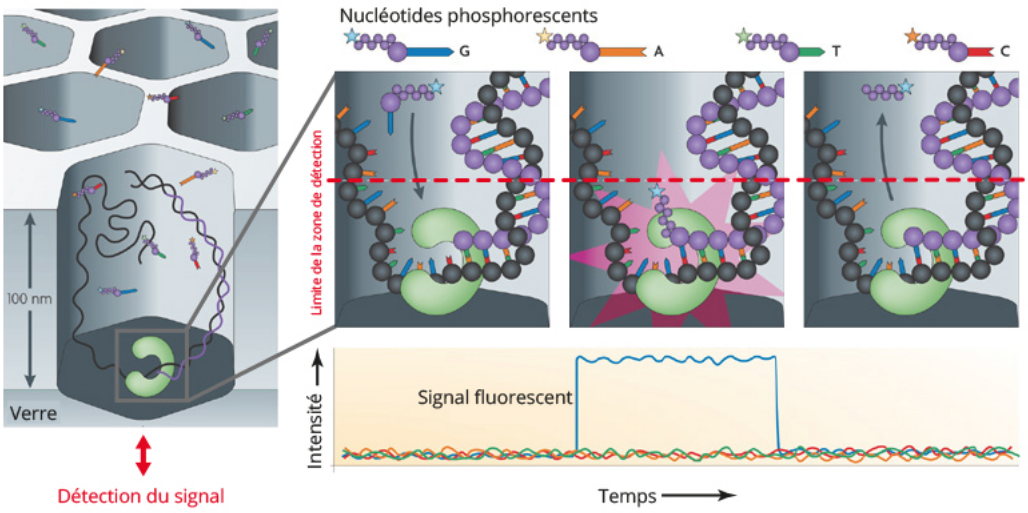

FIGURE 2 – Fonctionnement du Pacbio

La société Oxford Nanopore (MinION), quant à elle, utilise le séquençage par nanopore biologique(Figure 3). L'ADN à séquencer est capté par une enzyme (en bleu) qui extrude l'ADN simple brin à une vitesse réduite. Le nanopore lui-même (en rouge) constitue une ouverture dans la membrane lipidique (en vert) dans laquelle le champ électrique (flèches bleues) force le passage de l'ADN simple brin (en marron). Ce dernier est ralenti au passage par la constriction du nanopore (encart du bas) et permet l'identification des bases individuelles par mesure de l'intensité du courant électrique traversant le pore à un instant donné (encart du haut). Le trait pointillé bleu représente le courant à vide du nanopore.

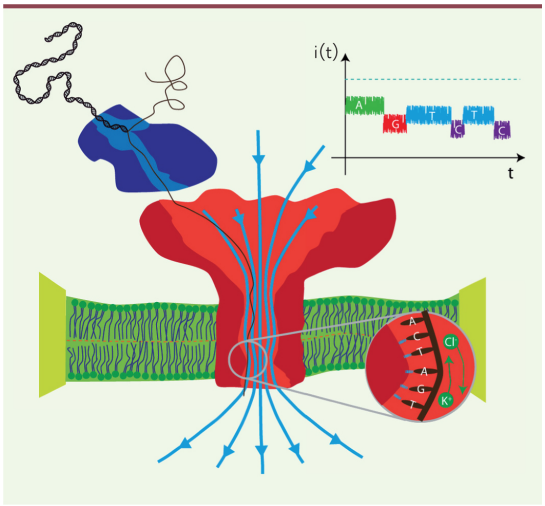

FIGURE 3 – Fonctionnement du MinION

Dans le but de me familiariser avec ces nouvelles notions, j'ai eu l'occasion de suivre une partie d'une formation sur le séquençage enseigné par mon tuteur de stage. J'ai également utilisé Internet pour me renseigner sur ce sujet. De plus, j'ai eu l'occasion de rencontrer des biologistes du CNRS(Centre National de la Recherche Scientifique) qui utilisent le séquenceur MinION, et d'avoir eu une conversation téléphonique avec une biologiste de l'INRA de Clermont-Ferrand qui utilise le séquenceur Pacbio.
D'après les biologistes rencontrés, les séquenceurs ont des spécificités. En effet, d'après leurs ob-

servations, il semble que dans certains cas, le MinION et le Pacbio ne parviennent pas à arriver au bout de la lecture. Soit l'ADN polymérase se décroche pour le Pacbio, soit la séquence reste coincée dans le nanopore pour le MinION.

4.2 Simulation du séquençage

Ma deuxième tâche a consisté à simuler par ordinateur le séquençage d'un brin d'ADN, et d'observer la distribution des longueurs des lectures obtenue. Afin de vérifier les idées générales avant de se lancer dans les calculs théoriques. J'ai utilisé le langage python (le package numpy, random) pour réaliser cette simulation.
Avant d'être séquencé, l'ADN est fragmenté aléatoirement en N morceaux, ce qui est semblable au cas d'un bâton coupé en N morceaux aléatoirement.
Ainsi, on sait que la loi suivie par la distribution des longueurs des morceaux est une loi bêta de paramètres $\alpha = 1$ et $\beta = N$.
De densité :
$p_F(x) = \frac{\Gamma(1+N)}{\Gamma(1)\Gamma(N)} x^0 (1-x)^{N-1}$ pour $x \in [0,1]$
$p_F(x) = \frac{N!}{(N-1)!}(1-x)^{N-1}$
$p_F(x) = N(1-x)^{N-1}$
où Γ est la fonction Gamma et x = $\frac{y}{G}$ avec y : la taille des fragments (en bp) et G : la taille du génome (en bp).

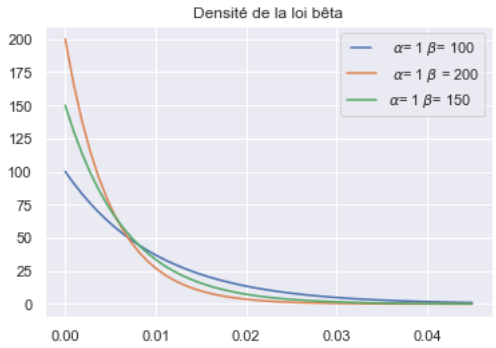

FIGURE 4 – Densité de la loi bêta(1,N) pour plusieurs valeurs de N

Dans notre cas, N est également une variable aléatoire suivant une loi à déterminer.
En effet, il y a plusieurs génomes à fragmenter avant le séquençage (Figure 1) et il n'y a aucune raison que tous les génomes soient découpés en même nombre de morceaux N. D'où le fait que N soit une variable aléatoire avec une loi à déterminer. De plus, pour une loi donnée, la moyenne et l'écart-type vont varier en fonction de la personne qui prépare la librairie de fragments. Une personne qui manipule bien aura tendance à produire des N avec une moyenne et un écart-type plus petits que si cette personne ne manipule pas soigneusement.
Ainsi, nous avons pensé tout d'abord à la loi normale de paramètres $\mu > 0$ et $\sigma^2 > 0$. Si μ et σ^2 sont petits, cela veut dire que la personne chargée de fragmenter l'ADN, travaille soigneusement, et inversement.
Ensuite, nous avons considéré la loi de poisson de paramètre $\lambda > 0$. Même si cette loi caractérise les évènements rares, cette loi est néanmoins plus facile à intégrer et peut être approchée par la loi normale pour $\lambda > 25$, ce qui est notre cas ici.
Enfin, nous avons choisi la loi uniforme de paramètres a>0 et b>0. En effet, on peut considérer

que le nombre de fragment est distribué de manière uniforme.

Le but de cette simulation est de comparer la distribution obtenue par simulation et la distribution des longueurs des lectures issue de données expérimentales afin d'avoir une idée de la loi appropriée pour N et de l'influence du séquenceur sur la distribution.
Les données expérimentales proviennent du site E.N.A (European Nucleotide Archive). Je ne me suis intéressée qu'aux données de bactéries dont l'ADN a été séquencé par le MinION ou le Pacbio.

4.2.1 Minion

J'ai d'abord représenté la distribution des longueurs des lectures (en bp), de deux jeux de données expérimentales, après séquençage de l'ADN avec le MinION.

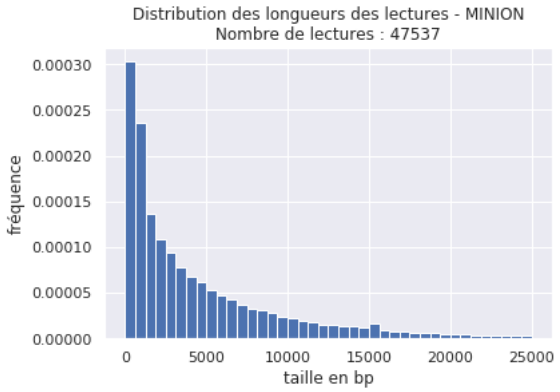

FIGURE 5 – Histogramme de la distribution expérimentale des longueurs des lectures avec le MinION

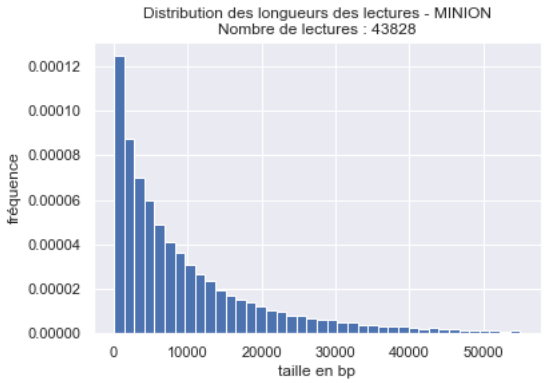

FIGURE 6 – Histogramme de la distribution expérimentale des longueurs des lectures avec le MinION

On remarque que l'allure de cette distribution décroit rapidement vers zéro, elle ressemble à la loi bêta mentionnée précédemment.
Ainsi, j'ai considéré 1000 génomes de taille G, que j'ai coupé aléatoirement en N morceaux, en tirant N selon soit une loi de poisson, une loi uniforme ou une loi normale.
Ensuite, dans le cas du MinION, j'ai considéré que le séquenceur découpe aléatoirement les fragments indépendamment de leur taille avec une probabilité très faible. On trouve alors une distribution semblable à la distribution des longueurs des fragments de l'ADN séquencé par le MinION pour les trois lois de N. Le code python se trouve en annexe 1.
Ci-dessous, les résultats de la simulation du séquençage de 1000 génomes de taille G = 1000000 pour N suivant :

- une loi normale de paramètres : $\mu = 350$ $\sigma = 10$
- une loi uniforme de paramètres : a = 300, b = 350
- une loi de poisson de paramètres : $\lambda = 350$

Les valeurs des paramètres ont été trouvées par essais-erreur.

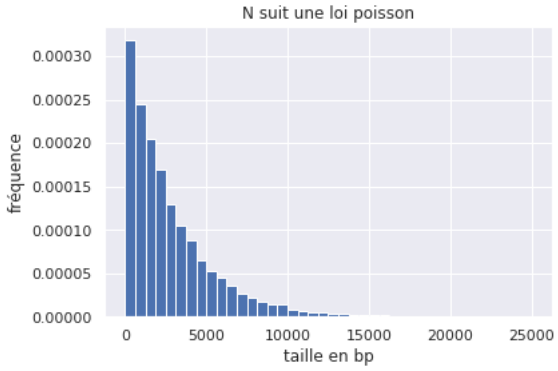

FIGURE 7 – Histogramme de la distribution des longueurs des lectures N loi de poisson (MinION)

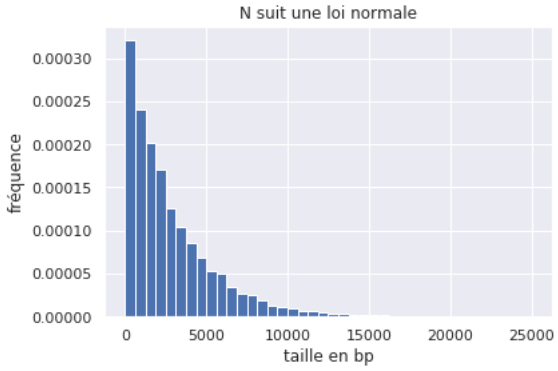

FIGURE 8 – Histogramme de la distribution des longueurs des lectures N loi normale (MinION)

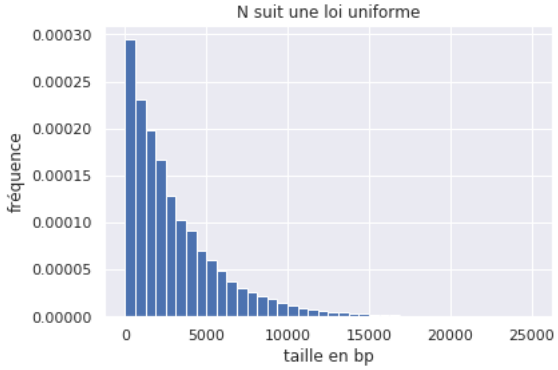

FIGURE 9 – Histogramme de la distribution des longueurs des lectures N loi uniforme (MinION)

J'ai également réalisé un test de Kolmogorov-Smirnov avec :
H_0 : les deux jeux de données expérimentales (Figure 5) et simulées sont issus d'une même loi
H_1 : les deux jeux de données expérimentales (Figure 5) et simulées ne sont pas issus d'une même loi
Pour ce faire, j'ai utilisé la fonction ks2samp du package scipy.stats.
J'ai trouvé des p-value > 0.01
si N suit une loi normale : p-value $= 0.0123$, si N suit une loi uniforme : p-value $= 0.0106$ et si N suit une loi de poisson : p-value $= 0.0119$
Par conséquent, pour $\alpha = 0.01$, dans les trois cas, on ne rejette pas H_0.

4.2.2 Pacbio

J'ai également représenté la distribution après séquençage de l'ADN avec le Pacbio.

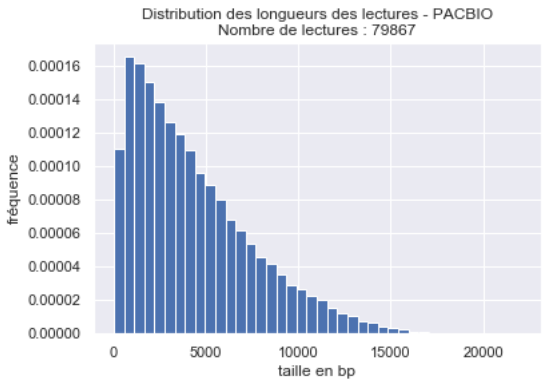

FIGURE 10 – Histogramme de la distribution des longueurs des lectures avec le Pacbio

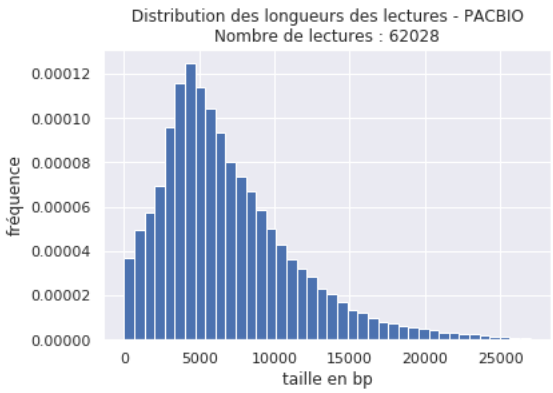

FIGURE 11 – Histogramme de la distribution des longueurs des lectures avec le Pacbio

On observe une distribution différente de celle du Minion, on voit que l'allure de cette distribution est croissante puis décroissante.
Afin de simuler une distribution similaire, j'ai également considéré 1000 génomes de taille G, que j'ai coupé aléatoirement en N morceaux, où N suit soit une loi de poisson, une loi uniforme ou une loi normale.
En revanche, dans le cas du Pacbio, j'ai considéré que le séquenceur découpe préférentiellement et aléatoirement les grands fragments. On trouve alors une distribution semblable à la distribution des longueurs des fragments de l'ADN séquencé par le Pacbio pour les trois lois de N. Le code python correspondant se trouve en annexe 1.
On trouve alors les résultats de la simulation du séquençage de 1000 génomes de taille G = 1000000 pour N suivant :

— une loi normale de paramètres : $\mu = 270$ $\sigma = 10$
— une loi uniforme de paramètres : a = 270, b = 280
— une loi de poisson de paramètres : $\lambda = 270$

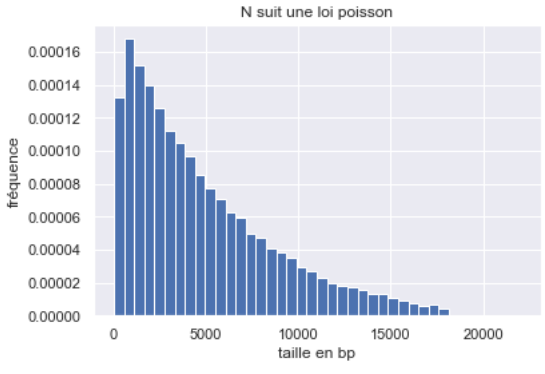

FIGURE 12 – Histogramme de la distribution des longueurs des lectures N loi de poisson (Pacbio)

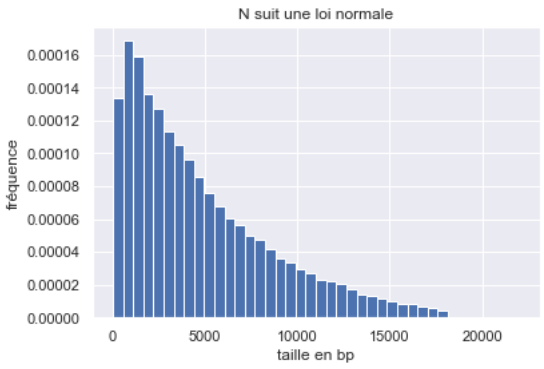

FIGURE 13 – Histogramme de la distribution des longueurs des lectures N loi normale (Pacbio)

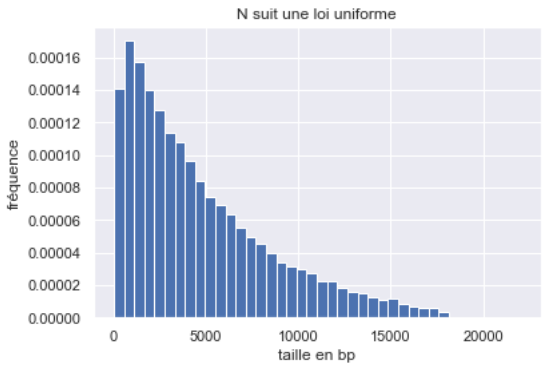

FIGURE 14 – Histogramme de la distribution des longueurs des lectures N loi uniforme (Pacbio)

De même j'ai réalisé un test de Kolmogorov-Smirnov avec :
H_0 : les deux jeux de données expérimentales (Figure 10) et simulées sont issus d'une même loi
H_1 : les deux jeux de données expérimentales (Figure 10) et simulées ne sont pas issus d'une même loi
Pour ce faire, j'ai utilisé la fonction ks2samp du package scipy.stats.
J'ai trouvé des p-value > 0.01
si N suit une loi normale : p-value = 0.0165, si N suit une loi uniforme : p-value = 0.0111 et si N suit une loi de poisson : p-value = 0.0201
Par conséquent, pour $\alpha = 0.01$, dans les trois cas, on ne rejette pas H_0.

D'après nos observations, on peut alors se dire que le séquenceur MinION coupe effectivement les fragments indépendamment de leur taille, le séquenceur Pacbio coupe préférentiellement les grands fragments et que la loi de N peut être une des lois mentionnées.

4.3 Formule théorique

La tâche suivante a été d'obtenir une formule explicite de la loi composée de la loi bêta et de la loi de N. Puis de trouver la formule finale décrivant la distribution des longueurs des lectures en prenant en compte le fait que le séquenceur ne parvienne pas à lire toutes les bases pour certaines lectures, il fournit donc une version tronquée de la lecture.

4.3.1 Loi composée bêta - loi de N

Tout d'abord, on sait que après fragmentation de l'ADN avant séquençage, la distribution des tailles des fragments suit une loi de bêta de paramètres α=1 et β=N, où N correspond au nombre de fragments.
Or étant donné que dans notre cas N est lui aussi une variable aléatoire, la distribution des longueurs des fragments qui nous intéresse est une loi composée de la loi bêta et de la loi de N, dont la formule est :
$p_H(x) = \int p_F(x|n)p_G(n)dn$ où
p_H : densité de la loi composée, p_G : densité de la loi de N
p_F : densité de la loi bêta de paramètres $\alpha = 1$ et $\beta = N$.

Avec $p_F(x) = N(1-x)^{N-1}$
pour $x \in [0,1]$
où x = $\frac{y}{G}$ avec y : la taille des fragments (en bp) et G : la taille du génome (en bp).

Loi composée bêta - poisson On considère dans un premier temps le cas où N suit une loi de poisson de paramètre λ et de densité de probabilité : $p_G(N=n) = \frac{\lambda^n e^{-\lambda}}{n!}$
avec $\lambda > 0$ et n est un entier naturel.
On calcule alors la loi composée bêta-poisson p_H.
$p_H(x) = \sum_{i=0}^{+\infty} n(1-x)^{n-1} \frac{\lambda^n e^{-\lambda}}{n!} = \lambda e^{-\lambda} \sum_{i=0}^{+\infty} \frac{(\lambda(1-x))^{n-1}}{(n-1)!}$
$p_H(x) = \lambda e^{-\lambda} \sum_{i=1}^{+\infty} \frac{(\lambda(1-x))^{n}}{(n)!} + \frac{\lambda^2 e^{-\lambda}}{1-x} = \lambda e^{-\lambda x} + \frac{\lambda^2 e^{-\lambda}}{1-x}$
$p_H(x) = \sum_{n=0}^{+\infty} n(1-x)^{n-1} \frac{\lambda^n e^{-\lambda}}{n!} = \sum_{n=1}^{+\infty} n(1-x)^{n-1} \frac{\lambda^n e^{-\lambda}}{n!} + 0$
$p_H(x) = \lambda e^{-\lambda} \sum_{k=0}^{+\infty} \frac{((1-x)\lambda)^k}{(k)!} = \lambda e^{-\lambda} e^{(1-x)\lambda}$
Finalement, on trouve :

$p_H(x) = \lambda e^{-\lambda x}$, $x \in [0,1]$

On reconnait la densité de la loi exponentielle de paramètres $\lambda > 0$.

Loi composée bêta - uniforme On fait de même en composant la loi bêta avec la loi uniforme de paramètres : a, b
$p_H(x) = \int_a^b n(1-x)^{n-1} \frac{1}{b-a} dn$

En intégrant, on trouve : $p_H(x) = \frac{e^{-log(1-x)}}{b-a}(\frac{be^{blog(1-x)} - ae^{alog(1-x)}}{log(1-x)} + \frac{e^{alog(1-x)} - e^{blog(1-x)}}{(log(1-x))^2})$, $x \in]0,1[$

Cette densité n'est pas défini pour x = 0 ni pour x = 1.
En effet, on peut supposer que l'ADN à séquencer (de taille g), ne peut pas être fragmenté en morceaux de taille 0 (i.e x =0), ni en morceaux de taille G (i.e x=1).

Loi composée bêta - normale Enfin, pour N suivant la loi normale de paramètres μ et σ^2, on a :
$p_H(x) = \int_0^{+\infty} n(1-x)^{n-1} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(n-\mu)^2}{2\sigma^2}} dn = \frac{1}{\sigma\sqrt{2\pi}} \int_0^{+\infty} n e^{u(n)}$
avec $u(n) = \frac{-1}{2\sigma^2} n^2 + n(log(1-x) + \frac{\mu}{\sigma^2}) - (log(1-x) + \frac{\mu^2}{2\sigma^2})$
on pose $a = \frac{1}{2\sigma^2}$, $b(x) = log(1-x) + \frac{\mu}{\sigma^2}$, $c(x) = log(1-x) + \frac{\mu^2}{2\sigma^2}$
$p_H(x) = \frac{e^{-c(x)}}{\sigma\sqrt{2\pi}} \int_0^{+\infty} n e^{-an^2 + b(x)n} dn = \frac{e^{-c(x)}}{\sigma\sqrt{2\pi}} J$
avec $J = \int_0^{+\infty} n e^{-an^2 + b(x)n} dn$
on peut écrire, $n = \frac{-1}{2a}(b(x) - 2an) + \frac{b(x)}{2a}$
d'où, $J = \frac{b(x)}{2a} \int_0^{+\infty} e^{b(x)n - an^2} dn + \frac{1}{2a} \int_0^{+\infty} (2an - b(x)) e^{b(x)n - an^2} dn = \frac{b(x)}{2a} L + \frac{1}{2a} K$
avec $L = \int_0^{+\infty} e^{b(x)n - an^2} dn$ et $K = \int_0^{+\infty} (2an - b(x)) e^{b(x)n - an^2} dn$
on résout K et on trouve $K = -e^{b(x)n - an^2}$
pour L, on a $L = \int_0^{+\infty} e^{\frac{b(x)^2}{4a} - (\sqrt{a}n - \frac{b(x)}{2\sqrt{a}})^2} dn$
on fait un changement de variable : $u = \frac{2an - b(x)}{2\sqrt{a}}$, $du = \sqrt{a} dn$

$L = \frac{\pi e^{\frac{b(x)^2}{4a}}}{2\sqrt{a}}(\int_{\frac{-b(x)}{2\sqrt{a}}}^{0} \frac{2e^{-u^2}}{\sqrt{\pi}}du + \int_{0}^{+\infty} \frac{2e^{-u^2}}{\sqrt{\pi}}du)$
on reconnait la fonction erreur $erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$, on a donc :
$L = \frac{\pi e^{\frac{b(x)^2}{4a}}}{2\sqrt{a}}(erf(\frac{b(x)}{2\sqrt{a}}) + 1)$
Finalement, $p_H(x) = \frac{e^{-c(x)}}{\sigma\sqrt{2\pi}}(\frac{b(x)}{2a}L + \frac{1}{2a}K) = \frac{e^{-c(x)}}{\sigma\sqrt{2\pi}}(\frac{b(x)}{2a}\frac{\pi e^{\frac{b(x)^2}{4a}}}{2\sqrt{a}}(erf(\frac{b(x)}{2\sqrt{a}}) + 1) - \frac{e^{b(x)n-an^2}}{2a})$
On considère que n tend vers $+\infty$, on remplace les valeurs de a, b(x) et c(x) et on trouve :

$$p_H(x) = \frac{\sigma\pi(\sigma^2 log(1-x)+\mu)e^{\sigma^2 log(1-x)(2(\mu-1)+\sigma^2 log(1-x))+2\mu^2}}{2\sqrt{\pi}}(erf(\frac{\sqrt{2}(\sigma^2 log(1-x)+\mu)}{2\sigma}) + 1)$$

On trace les fonctions obtenus pour différentes valeurs de leur paramètres :

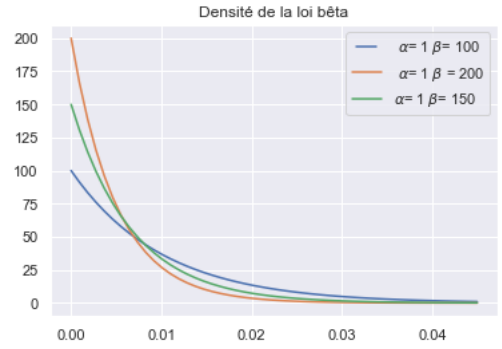

FIGURE 15 – Densité de la loi bêta

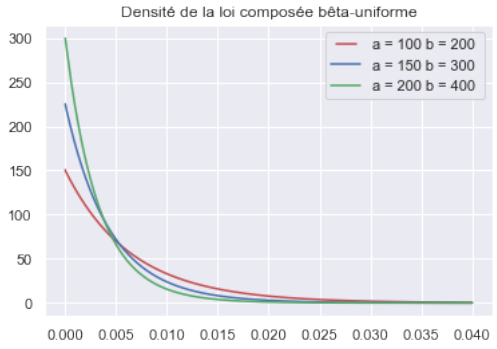

FIGURE 16 – Densité de la loi bêta-uniforme

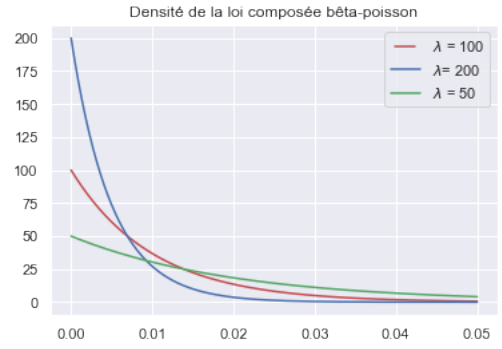

FIGURE 17 – Densité de la loi bêta-poisson

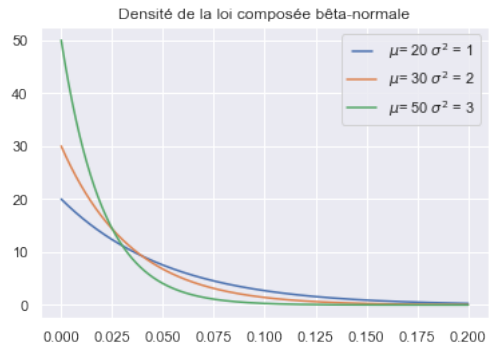

FIGURE 18 – Densité de la loi bêta-normale

On remarque que pour les trois lois composées obtenues, l'allure de cette densité est assez semblable à celle de la densité de la loi bêta.
En outre, on observe que les espérances des lois de N se retrouvent graphiquement lorsque x tend vers 0.
En effet, E(N)= λ, lorsque N suit une loi de poisson et lorsque x tend vers 0, $p_H(x) = \lambda$.
E(N)= μ, lorsque N suit une loi normale et lorsque x tend vers 0, $p_H(x) = \mu$.
E(N)= $\frac{a+b}{2}$, lorsque N suit une loi uniforme et lorsque x tend vers 0, $p_H(x) = \frac{a+b}{2}$.

4.3.2 Loi du séquenceur et expression finale

Après avoir déterminé la loi composée de bêta avec la loi de N, la mission suivante a été de trouver une loi f permettant de modéliser la distribution des longueurs des lectures, lorsque celles-ci passent dans le séquenceur.
En effet, comme mentionné précédemment, les séquenceurs Minion et Pacbio ne parviennent pas à lire toutes les bases pour certaines lectures, d'après les biologistes que nous avons rencontrés. Ainsi, pour le Minion, la loi géométrique nous a semblé approprié pour décrire la distribution des longueurs des fragments fourni par ce séquenceur.
$f(x) = p(1-p)^{x-1}$ où x représente la longueur d'un fragment (en bp)
En effet, cette loi nous donne la probabilité $f(x)$ d'obtenir dans une succession de x épreuves de Bernoulli, x-1 échecs suivis d'un succès.
Ici, le succès a une probabilité p faible d'interrompre en cours de route le séquençage d'un fragment d'ADN. Cette probabilité est indépendante de la longueur du fragment introduit dans le séquenceur dans le cas de la loi géométrique. Ainsi, à chaque nucléotide, la probabilité p est constante.
Ainsi, on calcule la fonction G décrivant la loi de la distribution des longueurs des fragments en prenant en considération la particularité du séquenceur. G(x) donne la probabilité d'avoir un fragment de taille x après être passé dans le séquenceur.

On a : $G(x) = \int_x^G f(x) * p_H(l)dl$

En effet, après passage dans le séquenceur, la lecture de taille x, ne peut pas être plus longue que le fragment initial de taille l donc x<l<G. Sinon la densité de probabilité vaut 0.
Étant donné que la fonction composée bêta-poisson a une expression plus simple que les autres fonctions composées calculées précédemment, et que leurs représentations sont similaires, pour simplifier les calculs, nous avons décidé de considérer $p_H(x) = \lambda e^{-\lambda x}$

$G(x) = p(1-p)^{x-1}(e^{-\lambda x} - e^{-\lambda G})$ avec G : taille du génome.

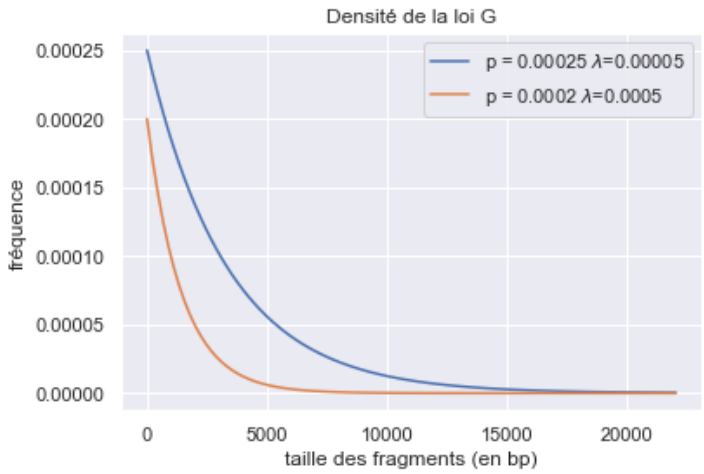

FIGURE 19 – Représentation de la loi G-MINION

En représentant graphiquement cette fonction G, on a remarqué que l'on obtient une courbe de même allure que la distribution des longueurs des fragments pour l'ADN séquencé avec le MinION. Il s'agit d'une courbe qui décroit vers 0.

On estime les paramètres p et λ à l'aide de la log-vraisemblance :
$L(x) = \sum_{i=1}^{n} log(p(1-p)^{x_i-1}(e^{-\lambda x_i}) - e^{-\lambda G})$
On trouve alors les expressions :

$p = \frac{n}{\sum_{i=1}^{n} x_i}$
$\sum_{i=1}^{n} \frac{-x_i e^{-\lambda x_i} + G e^{-\lambda G}}{e^{-\lambda x_i} - e^{-\lambda G}} = 0$
on peut réécrire : $\sum_{i=1}^{n} (G - x_i) \frac{e^{\lambda(G-x_i)}}{e^{\lambda(G-x_i)}-1} = nG$
Ce dernier paramètre étant difficile à isoler, j'ai alors trouver une valeur de λ pour laquelle la courbe de la fonction G a la même allure que des données théoriques.
Ensuite à l'aide des paramètres initiaux trouvés, je les ai optimisé en minimisant la log-vraisemblance négative à l'aide du package scipy.optimize sur python. J'ai utilisé la fonction fmin qui minimise une fonction en utilisant l'algorithme du simplexe.
J'ai trouvé comme paramètres pour un jeu de données issues du séquençage MinION :
$p = 9.37 * 10^{-5}$ et $\lambda = 2 * 10^{-5}$
Ainsi, en représentant la fonction G et les données expérimentales, on trouve :

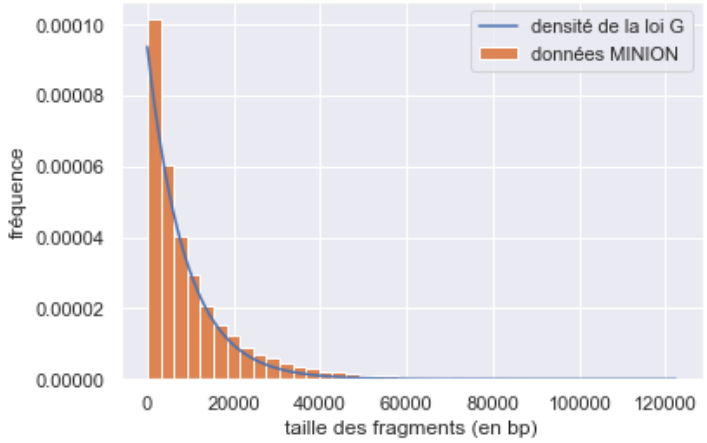

FIGURE 20 – Représentation de la loi G et données MINION

Dans le cas du séquençage effectué à l'aide du Pacbio, en revanche, la fonction G décrite ci-dessous n'est pas appropriée. On a supposé, que la probabilité d'interruption du séquençage du fragment introduit croît avec la taille du fragment. On a alors modélisé ce cas en utilisant la loi de Weibull.

$f(x) = \frac{k}{\alpha}(\frac{x}{\alpha})^{k-1} e^{-(\frac{x}{\alpha})^k}$

où k>0 et $\alpha > 0$
On posera $\mu = \frac{1}{\alpha}$
Cette loi est utilisée généralement lorsque le taux de défaillance évolue comme une puissance du temps.
Dans notre cas : le taux de défaillance correspond au taux d'interruption par le séquenceur, et cette loi est en fonction de la taille en bp et non du temps. De plus, le paramètre k est supérieur à 1, c'est-à-dire que le taux d'interruption croît avec la taille.
La fonction G vaut alors :
$G(x) = \int_x^G f(x) * p_H(l) dl$

$G(x) = k\mu(x\mu)^{k-1}e^{-(x\mu)^k}(e^{-\lambda x} - e^{-\lambda G})$

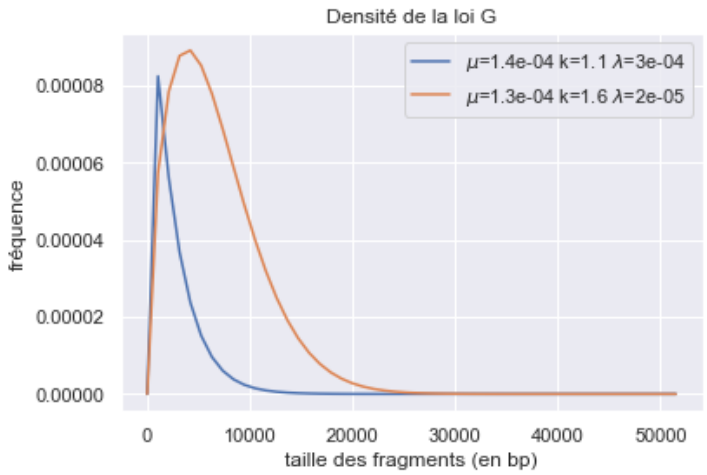

FIGURE 21 – Représentation de la loi G-PACBIO

On estime les paramètres à l'aide de la log-vraisemblance :
$L(x) = \sum_{i=1}^{n} log(k\mu(x_i\mu)^{k-1}e^{-(x_i\mu)^k}(e^{-\lambda x_i} - e^{-\lambda G}))$
$\frac{\partial L(x)}{\partial \mu} = \frac{nk - k\mu^k \sum_{i=1}^{n} x_i^k}{\mu}$
$\frac{\partial L(x)}{\partial k} = \frac{n}{\mu} nlog(\mu) + \sum_{i=1}^{n} log(x_i) - \sum_{i=1}^{n} log(\mu x_i) e^{klog(\mu x_i)}$
$\frac{\partial L(x)}{\partial \lambda} = \sum_{i=1}^{n} \frac{-x_i e^{-\lambda x_i} + Ge^{-\lambda G}}{e^{-\lambda x_i} - e^{-\lambda G}}$
On trouve alors les expressions suivantes :

$\mu = \sqrt[k]{\frac{n}{\sum_{i=1}^{n} x_i^k}}$
$-\frac{n}{k} + \sum_{i=1}^{n} log(\mu x_i) e^{klog(\mu x_i} = nlog(\mu) + \sum_{i=1}^{n} log(x_i)$
$\sum_{i=1}^{n} \frac{-x_i e^{-\lambda x_i} + Ge^{-\lambda G}}{e^{-\lambda x_i} - e^{-\lambda G}} = 0$
on peut réécrire : $\sum_{i=1}^{n} (G - x_i) \frac{e^{\lambda(G-x_i)}}{e^{\lambda(G-x_i)} - 1} = nG$

Comme précédemment, j'ai estimé des paramètres initiaux à l'aide des expressions ci-dessus, puis j'ai optimisé les paramètres. J'ai trouvé pour le jeu de données représenté ci-dessous $\mu = 1.3e-04$, k , $\lambda = 5e-06$.

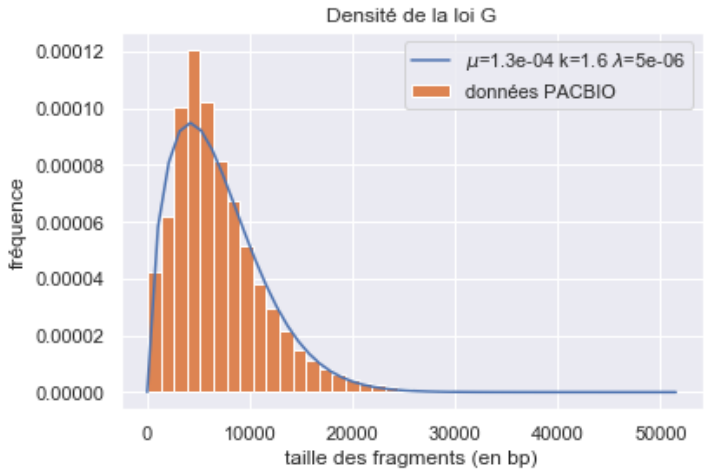

FIGURE 22 – Représentation de la loi G et données PACBIO

On retrouve la loi du Pacbio si on utilise la loi de Weibull.

5 Conclusion - Perspectives

En conclusion, le séquençage d'ADN par les technologies de troisième génération nous donne une distribution des longueurs des lectures dont il a fallu trouver l'expression explicite pendant ce stage.
Ainsi, concernant le séquenceur Minion on a :
$G(x) = p(1-p)^{x-1}(e^{-\lambda x} - e^{-\lambda G})$
avec $\lambda > 0$, $p \in [0,1]$, G : taille du génome et x : taille en bp

Et pour le Pacbio, nous avons trouvé l'expression suivante :
$G(x) = k\mu(x\mu)^{k-1}e^{-(x\mu)^k}(e^{-\lambda x} - e^{-\lambda G})$
avec $\lambda > 0$, k>1, $\mu > 0$, G : taille du génome et x : taille en bp

Ce stage a été très enrichissant pour moi car il m'a permis de découvrir le secteur de la recherche scientifique, et il m'a permis de participer concrètement à ses enjeux au travers de mes missions. Ce stage m'a aussi permis d'utiliser les mathématiques appliquées à un domaine qui m'intéresse particulièrement, en effet, je souhaiterais exercer un métier en rapport avec celui-ci.
J'ai appris des notions nouvelles sur la génomique. Pour répondre à la problématique de ce stage, j'ai dû travailler sur l'aspect théorique en déterminant et en composant des lois de probabilités, ainsi que d'extraire et d'étudier des données expérimentales. Aussi, j'ai du faire preuve d'autonomie, ainsi, plusieurs essais et recherches ont été nécessaires afin de pouvoir accomplir mes missions.
Les principales difficultés rencontrées ont été, dans un premier temps, de comprendre clairement l'objectif du stage et de me familiariser avec le séquençage d'ADN au début du stage. De plus, trouver une expression de la loi bêta-normale et déterminer la fonction f représentant la loi du séquenceur ont également été des difficultés techniques que j'ai rencontrées.

Ainsi, les expressions trouvées permettront par la suite de pouvoir revisiter les travaux Lander-Watermann(1988) qui s'intéressent aux propriétés statistiques du séquençage shotgun. Dont le but serait de savoir combien de lectures sont nécessaires pour recouvrir tous le génome, étant donné que la taille des lectures n'est plus fixe mais suit une loi de fonction de densité G(x).

Liste des figures

A Annexes

A.1 Code python de la simulation du séquençage d'ADN

In [ ]:

```python
def fracture(taille_genome, loi='normale', source = None):
    """
    Fracture une liste d'échantillons en utilisant un nombre de fractures variables suivant la loi d
e N
    :param taille_genome : taille des génomes à fragmenter
    :param loi: loi de N : normale, poisson ou uniforme
    :param source : séquenceur PACBIO ou MINION
    :return : un tableau de tailles de fragment
    """
    #paramètres des lois normale, uniforme, poisson choisis arbitrairement
    parametres =  dict(
        normale=[270, 10],
        uniforme=[270, 280],
        poisson=[270],
    )[loi]

    echantillons = [taille_genome]*1000
    #len(echantillons) correspond au nombre de génome que l'on souhaite fragmenter ici on considère
 1000 génomes
    #On génère len(echantillons) nombres suivant la loi normale de paramètres parametres
    if loi == 'normale':
        n, sigma = parametres
        nfrag = n + np.random.randn(len(echantillons)) * sigma

    #On génère len(echantillons) nombres suivant la loi uniforme de paramètres parametres
    elif loi == 'uniforme':
        nmin, nmax = parametres
        nfrag = np.random.uniform(nmin, nmax, len(echantillons))

    #On génère len(echantillons) nombres suivant la loi poisson de paramètres parametres
    elif loi == 'poisson':
        n = parametres
        nfrag = np.random.poisson(n, len(echantillons))

    else:
        raise NotImplementedError(loi)
    echantillons_fractures = []
    echantillon_fracture = []

    # On prends chaque génome et lui associe un nombre de fractures, converti en int à la volée
    for index, (nombre_fragments, taille_echantillon) in enumerate(zip(map(int, nfrag), echantillons
)):

        if nombre_fragments > taille_echantillon:
            raise ValueError(
                '\n\n\n'
                'ERREUR: Le nombre de fragments demandé est trop grand, il est impossible de continu
er\n'
                'Vous pouvez: \n'
                '\t- Réduire le nombre de fragments demandé, en jouant sur N et sigma\n'
                '\t- Augmenter la taille des échantillons à fracturer\n'
            )
        # Il y aura n - 1 fractures pour obtenir n elements
        nombre_fractures = nombre_fragments - 1

        # Une fracture sera placée avant l'index désigné, donc une fracture
        #   localisée à l'index 1 donnera un morceau de 1 et un de taille - 1
        fractures = set()
        while len(fractures) < nombre_fractures:
            fractures.update(np.random.randint(1, taille_echantillon - 1, nombre_fractures))

        if len(fractures) != nombre_fractures:
            if len(fractures) < nombre_fractures:
                raise RuntimeError('ERREUR: Une erreur s\'est produite: pas assez de fragments')
            fractures = set(list(fractures)[:nombre_fractures])

        fractures = list(fractures)
        fractures = [0] + list(sorted(fractures)) + [taille_echantillon]
        echantillon_fracture = [
            fractures[i + 1] - fractures[i] for i in range(len(fractures) - 1)
        ]
        if sum(echantillon_fracture) != taille_echantillon:
            raise RuntimeError(
                "ERREUR: Une erreur s'est produite lors de la fragmentation, \n"
                f"Nous avons {abs(sum(echantillon_fracture) - taille_echantillon)}"
                f" fragments de différence entre ce qu'on attendais et ce qu'on a"
                f" ({sum(echantillon_fracture)} contre {taille_echantillon})"
            )
        echantillon_fracture = list(i for i in sorted(echantillon_fracture))
        echantillons_fractures.append(echantillon_fracture)

    fragments = list(chain.from_iterable(echantillons_fractures))
    if source.upper() == 'MINION':
        frag_seq = []
        maximum = max(fragments)
        for i in fragments:
            cut = random.randint(1, (i*1000) - 1) # prends nombre entre 1 et (i*1000) - 1
            i = min(cut, i) # si min(cut,i)= i, cela veut dire que le fragment n'a pas été découpé
            frag_seq.append(i)
        return frag_seq

    elif source.upper() == 'PACBIO':
        frag_seq = []
        maximum = max(fragments)
        minimum = min(fragments)
        for i in fragments:
            cut = random.randint(500, 18000 - 1)
            i = min(cut, i) # on considère que les grands fragments sont coupés plus fréquemment que
les petits
            frag_seq.append(i)
        return frag_seq
```

B Bibliographie

Sites internet consultés

- www.inra.fr
- www.medecinesciences.org/en/articles/medsci/full_html/2018/02/medsci20183402p161/medsci20183402p161.html
- www.napoleome.ch/fr/projet/le-sequencage/methodes/
- en.m.wikipedia.org/wiki/Compound_probability_distribution
- www.sciencebeta.com/whole-exome-sequencing/

Articles

- Jared C. Roach, "Random subcloning", 1995