

Stockage d'information sur ADN

Ariane Badual

▶ **To cite this version:**

Ariane Badual. Stockage d'information sur ADN. Biotechnologie. 2019. hal-02401641

HAL Id: hal-02401641
https://inria.hal.science/hal-02401641

Submitted on 10 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.







Master 1 de Bioinformatique

Rapport de stage

Version du 18 juin 2019

Stockage d'information sur ADN

Encadrant :

M. Dominique Lavenier

Auteure :

M^me^ Ariane Badoual

Tutrice :

M^me^ Annabelle Monnier

INRIA - IRISA Rennes

Campus de Beaulieu,

263 Avenue Général Leclerc,

35042 Rennes

Table des matières

1 Introduction **1**

2 Matériels et Méthodes **2**

2.1 Codage de l'information 2

2.1.1 Codage en ternaire 3

2.1.2 Ajout d'une séquence aléatoire 3

2.1.3 Ajout du code de correction des erreurs 3

2.1.4 Codage en bases ADN 5

2.1.5 Ajout d'une en-tête et d'une fin 5

2.2 Séquençage de l'ADN 5

2.2.1 Séquençage MinION 5

2.2.2 Simulation avec DeepSimulator 6

2.3 Correction des erreurs et récupération de l'information 7

2.4 Génération des séquences 8

3 Résultats **10**

3.1 Séquençage des séquences 10

3.1.1 Résultats de la simulation avec DeepSimulator 10

3.1.2 Résultat avec le séquençage MinION 14

4 Discussion **14**

5 Conclusion **16**

Références **17**

Annexes **19**

1 Introduction

Depuis toujours, l'humain génère et accumule de nouvelles informations. Au fur et à mesure du temps, la quantité de données numériques a augmenté de manière exponentielle. Selon IDC (International Data Corporation), le volume global d'information dans le monde devrait passer de 33 zettaoctet en 2018 à 175 zettaoctet en 2025 (1 zettaoctet = 10^{21} octets) (Reinsel et al, 2018 [11]). Au cours de l'histoire, diverses technologies ont été développées afin de stocker les informations de manière optimale, allant de la disquette aux data centers en passant par les disques durs. Néanmoins, malgré les nombreuses solutions de stockage qui existent, des problématiques restent à résoudre. Dans un monde où l'information augmente toujours, son stockage coûte de plus en plus cher et demande de plus en plus de place et de ressources. Les innovations sont nombreuses pour développer un support d'information toujours plus performant tout en essayant de réduire sa taille au maximum bien que la durée de vie de ces supports reste toujours limitée.

Afin de résoudre ces problématiques, de nombreuses études ont développé l'idée de stocker des informations sur un support ADN. De nos jours, il est tout à fait possible de synthétiser et de séquencer des séquences d'ADN. Bien sûr, pour le moment, le fait d'écrire et de lire l'information stockée sur de l'ADN est beaucoup plus long que d'écrire et lire l'information stockée sur des supports de stockage classique. Mais le stockage sur ce genre de support comporte d'autres avantages. L'ADN est une molécule très stable qui possède une longue espérance de vie, ce qui est idéal pour le stockage à long terme (Willerslev et al, 2007 [13]). De plus, il est capable de stocker énormément d'informations dans très peu de matière : un simple brin a une capacité théorique de stockage de 455 exaoctets (1 exaoctet = 10^{18} octets) par gramme (Church et al, 2012 [1]). Ainsi, en utilisant un support ADN en tant que stockage d'informations il serait possible de tirer profit des avantages de celui-ci. De cette manière, il pourrait être envisageable de stocker beaucoup d'information dans peu d'espace. Les avantages qu'offre cette molécule seraient utiles pour le stockage d'information à long terme comme, par exemple, pour les archives (O'Discoll et al, 2013 [10]). Bien qu'il serait long d'y accéder, ce genre de stockage offrirait la possibilité de stocker beaucoup d'information dans un volume réduit et pendant longtemps. Ce stockage à long terme pourrait permettre, notamment, de transmettre des informations aux futures générations (Cox et al, 2001 [2]). De plus, de nos jours, les data centers qui se chargent du stockage et de la gestion de l'information dépensent énormément d'énergie (Glanz et al, 2012 [5]). Stocker l'information sur un support ADN serait plus écologique puisque cela prendrait moins de place et utiliserait beaucoup moins d'énergie.

De nombreux essais ont déjà été menés afin de stocker l'information sur ADN. Le premier essai date de 1988. Joe Davis, en collaboration avec Dana Boyd, une biologiste moléculaire de Harvard, a inclut une image codée sur 35 bits représentant une rune germanique signifiant la terre femelle, dans le génome de E. Coli (Davis et al, 1996 [4]). Par la suite, d'autres équipes dans le monde ont réussi à synthétiser avec succès des séquences d'ADN stockant de l'information (images, messages, etc…). Ce fut le cas, par exemple, d'un groupe de chercheurs en 2013. Ils ont stocké 5 types de documents sur de l'ADN (Goldman et al, 2013 [6]).

Afin de retranscrire l'information sur l'ADN, plusieurs étapes sont effectuées. Tout d'abord le message est codé en trits (l'unité du codage ternaire). Il est ensuite transcrit en nucléotides (Adénine, Cytosine, Thymine, Guanine). La séquence obtenue est enfin synthétisée.

Un projet appelé "DNA-storage" a été mis en place par les équipes GenScale de l'INRIA/I-RISA à Rennes et IAS du Lab-STICC à Lorient. Son objectif est d'étudier l'information stockée sur un support ADN et plus précisément de se concentrer sur la restitution d'information après un séquençage réalisé par le séquenceur MinION de la société Oxford Nanopore Technology. Cette technologie de troisième génération permet de séquencer de l'ADN grâce à un pore traversé par un courant ionique. En fonction du signal, il est ainsi possible de connaître la composition de la séquence. Cependant, ce séquenceur présente un taux d'erreur d'environ 10% (Jordan et al, 2017 [7]). L'objectif du stage est de générer un environnement bioinformatique permettant de récupérer la totalité de l'information stockée après le séquençage. Pour ce faire, des simulations de séquençage ont été réalisées via l'outil DeepSimulator (Li et al, 2018 [8]) afin d'anticiper les résultats produits avec le séquenceur MinION et, ainsi, pouvoir vérifier la fiabilité de l'information obtenue après séquençage et après correction des erreurs. Par la suite, un séquençage réel avec le séquenceur MinION sera réalisé sur une séquence synthétisée et nous vérifierons la fiabilité de l'information récupérée en regardant si nous sommes capables de la corriger.

2 Matériels et Méthodes

2.1 Codage de l'information

Le codage en séquences d'ADN d'un texte en ASCII, ou d'une toute autre information, consiste à passer par plusieurs étapes. Dans un premier temps le code décimal du code ASCII est récupéré, puis traduit en ternaire. Après l'ajout d'informations supplémentaires, le code est transcrit en bases nucléiques afin de générer une séquence ADN. C'est l'équipe du Lab-STICC

qui est chargée de réaliser cette étape de codage.

2.1.1 Codage en ternaire

La première étape consiste à passer de l'information initiale à une séquence ternaire. Pour ce faire, nous utiliserons le code décimal du code ASCII des lettres du message que nous souhaitons coder. Ensuite, le code décimal est transformé en code ternaire. Pour réaliser cette étape, il faut diviser le nombre décimal par 3 puis continuer en divisant le résultat jusqu'à arriver à un résultat qui est égal à 0. Le résultat de la conversion est formé en concaténant les restes des divisions. Utiliser un codage en trits plutôt qu'en bits permet de prévenir les répétitions lors du codage en bases ADN (cf paragraphe *"Codage en bases ADN"*).

ex : 10 / 3 = 3 et il reste 1

3 / 3 = 1 et il reste 0

1 / 3 = 0 et il reste 1

le code binaire est donc 101

2.1.2 Ajout d'une séquence aléatoire

Il est possible que certaines sous-séquences se répètent à certains endroits de la séquence une fois transposée en séquence ADN. Cela est dû à des mots ou des phrases similaires dans le texte. Or, la synthèse de l'ADN est réalisée en fractionnant la séquence fournie en plusieurs sous-séquences appelées oligos. Ces oligos seront ensuite ré-assemblés pour récupérer la séquence voulue. Cependant, si certains oligos sont similaires, il sera difficile de connaître l'emplacement des séquences et donc le ré-assemblage ne se passera pas correctement. Afin de pallier ce phénomène, on ajoute une chaîne ternaire d'une longueur de 1000 paires de bases générée aléatoirement à la séquence initiale en ternaire. Ce résultat sera au modulo 3 afin de garder une séquence ternaire (c'est à dire que l'on effectue une division euclidienne sur chaque caractère et que l'on ne garde que le reste de cette division). L'ajout de cette séquence permet d'introduire un caractère aléatoire à la séquence finale ce qui permettra d'éviter des répétions et ainsi avoir une synthèse fiable de l'ADN.

2.1.3 Ajout du code de correction des erreurs

La future séquence ADN sera synthétisée pour le stockage de l'information. Mais, pour lire les informations qu'elle contient, elle sera amenée à être séquencée. Ce séquençage est réalisé par un séquenceur MinION. Cependant, le séquençage d'ADN entraîne des erreurs : le message

récupéré ne sera pas le même que le message codé. Le séquençage introduit des erreurs de substitutions, de délétions et d'insertions. Le rôle du Lab-STICC est réaliser un codage pour la correction des erreurs. Pour le moment, le codage pour les erreurs de délétion et d'insertion est en train d'être réalisé, celui permettant de corriger les substitutions n'a, quant à lui, pas encore été codé. Nous ajouterons donc, lorsque cela sera possible, un code de correction des erreurs de délétion et d'insertion à la séquence ternaire que nous avons obtenu. Celui-ci est basé sur le code de correction de Varshamov-Tenengolts, appelé VT code. La séquence initiale est séparée en plusieurs blocs de même taille (mots de code). L'encodage et le décodage utilisent deux variables appelée A et B que nous fixons et qui permettent de trouver l'emplacement des erreurs de séquençage et la manière de les corriger. L'encodage se déroule en plusieurs étapes (Tenengolts et al, 1984 [12]).

Pour prévenir les erreurs de délétions et d'insertions, la première étape consiste à encoder la séquence, c'est-à-dire en prendre une partie et l'inclure dans les mots de code. Pour ce faire, on génère un ensemble de mots qui respectent les contraintes A et B. On génère d'abord une séquence appelée séquence auxiliaire. Cette séquence est composée de 1 et de 0 en fonction de la valeur du trit par rapport au précédent. Si le premier trit est plus grand que le deuxième alors nous mettons un 0, sinon nous mettons un 1. Par exemple la séquence auxiliaire de 0121 est 110. La séquence auxiliaire est la même pour chacun des mots de code.

Calcul de la séquence auxiliaire : $\alpha_i = \begin{cases} 1, \text{ si } \alpha_i \geq \alpha_{i-1} \\ 0, \text{ si } \alpha_i < \alpha_{i-1} \end{cases}$

Ensuite, on calcule le syndrome de la séquence auxiliaire α. Le syndrome correspond à la somme de chaque trit α_i multiplié par sa position i. Cette somme est mise au modulo n (la longueur du mot). Ce syndrome correspondra à la variable A.

Calcul du syndrome : $A = Syn(\alpha) = \sum_{i=0}^{n} i \times \alpha_i \; mod(n)$

Le syndrome est le même pour chaque bloc de la séquence initiale : il correspond à la variable A que nous avons fixé et qui est constante. Nous calculons également la somme du mot modulo 3 (la base utilisée est une base ternaire) de la séquence initiale a. Cette somme correspond, quant à elle, à la variable B.

Calcul de la somme : $B = Sum(a) = \sum_{i=0}^{n} a_i \; mod(3)$

2.1.4 Codage en bases ADN

Enfin, nous pouvons coder la séquence ternaire en séquence ADN. Afin de réaliser cette étape, il faut prendre en compte le caractère précédant chacun des caractères que nous souhaitons traduire en suivant un tableau (voir la table 1). L'intérêt de travailler à partir d'une base ternaire (et non à partir d'une base binaire ou quaternaire) est de faire en sorte que le caractère suivant dépende du précédant et ainsi d'éviter les répétitions d'un caractère. En génomique, les répétitions d'une même base sont appelées homopolymères et empêchent le bon séquençage de l'ADN par les séquenceurs comme MinION. Les homopolymères favorisent l'apparition d'insertions ou de délétions lors du séquençage (Lu et al, 2016 [9]). Par exemple, la base qui succédera à un A sera soit un T, soit un C, soit un G. Il n'est donc pas possible d'avoir deux A successifs. Pour le codage du premier trit, nous le faisons comme si le caractère précédent avait été un A.

caractère précédent	trits de la chaîne		
	0	1	2
A	C	G	T
C	G	T	A
G	T	A	C
T	A	C	G

Table 1 – Tableau de conversion des trits en nucléotides

2.1.5 Ajout d'une en-tête et d'une fin

Enfin, nous réalisons l'ajout d'une en-tête et d'une fin connue permettant de reconnaître, après séquençage, s'il s'agit du brin reverse ou non et ainsi de savoir quels brins décoder. Cet ajout d'information ne sera pas utile pour le décodage à proprement parlé de l'information.

2.2 Séquençage de l'ADN

2.2.1 Séquençage MinION

Pour récupérer l'information stockée dans l'ADN, nous réaliserons un séquençage avec le séquenceur MinION de Oxford Nanopore Technologies (Lu et al, 2016 [9]). Cette technologie repose sur un nanopore. La molécule d'ADN est préparée en échantillon puis passe ensuite à travers le nanopore. Le passage des bases par le pore entraînera une variation de courant spécifique qui varie en fonction du 6-mer. De cette manière, le signal de sortie est spécifique à la séquence d'ADN. Le signal est ensuite transmis à Albacore, un basecaller, qui l'analyse pour reconstituer la séquence d'ADN.

Le séquençage minION présente l'avantage d'avoir la possibilité de séquencer des read longs, ce qui est très intéressant dans le stockage d'informations sur support ADN. Cependant, le séquençage MinION possède une précision de 90 %. Ce qui laisse la place à des erreurs de séquençage qui peuvent compromettre l'information stockée. De plus, la technologie MinION montre quelques défauts en ce qui concerne le séquençage pour les séquences présentant des homopolymères. La présence d'homopolymères entraîne un taux de délétion et d'insertion plus élevé pour les séquences (Lu et al , 2016 [9]).

2.2.2 Simulation avec DeepSimulator

Dans un premier temps, avant de réaliser un séquençage réel, nous avons travaillé sur un simulateur de séquençage appelé DeepSimulator (Li et al, 2018 [8]) . Ainsi, nous avons pu appréhender correctement le séquençage mais aussi connaître les conditions optimales pour la génération de la séquence synthétisée afin que l'information qu'elle contient soit conservée le mieux possible. Contrairement aux logiciels déjà existants en ce qui concerne la simulation de séquençage Minion, DeepSimulator n'utilise pas de probabilités mais reproduit chacune des étapes du séquençage Minion (voir la figure **1**).

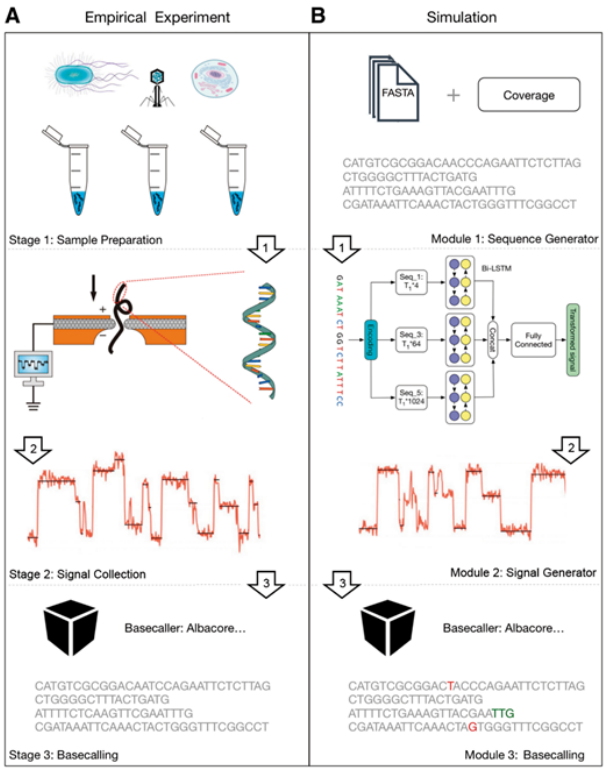


Figure 1 – "A. la procédure de séquençage par nanopore. B. Le flux de travail principal de DeepSimulator. Il simule l'intégralité du pipeline de l'expérience de séquençage par Nanopore, en produisant à la fois les signaux simulés et les lectures simulées finales. De plus, DeepSimulator est hautement modulaire, ce qui signifie qu'il peut être personnalisé et mis à jour facilement pour suivre le rythme de développement des technologies de séquençage Nanopore. [...] Dans les lectures simulées sur le bas de la figure, les bases de couleur rouge sont les incompatibilités. Les bases de couleur verte indiquent qu'il y a des indel (insertion et suppression) devant eux" (Li et al, 2018 [8])

Dans un premier temps, le simulateur va séparer la séquence d'entrée en read. Cette étape est appelée l'étape de génération de séquences, et vise à reproduire l'étape de préparation d'échan-

tillon réalisée lors d'un séquençage réel. La longueur et le nombre de reads sont spécifiés par l'utilisateur. Pour l'utilisation que nous voulons en faire, nous avons choisis de simuler un seul read.

L'étape suivante est appelée l'étape de génération des signaux. DeepSimulator modélise le passage des reads par le pore et émule ainsi la génération des signaux. Il utilise un modèle de pores. Deux types de modèle sont proposés par le logiciel : le modèle de pores indépendant du contexte, qui est le modèle de pore officiel, et un autre dépendant du contexte du pore. Nous avons choisi d'utiliser le modèle indépendant du contexte puisqu'il prend en compte les 6-mers. Les paramètres pour ce modèle sont modifiables : l'écart-type du bruit du signal, le cut-off du filtre passe-bas et le taux d'échantillonage. La modification de ces paramètres permet de générer un modèle le plus proche de ce que nous attendons. Le modèle génère d'abord un signal pour chacun des k-mers puis générera un ensemble de signaux pour l'ensemble de la séquence. Lors d'un séquençage réel, l'acquisition du signal brut est plus rapide que le passage de la séquence ADN par le pore, ce qui fait que certains 6-mers sont traduit en signaux plusieurs fois. Pour cette raison, il y a une répétition de certains signaux. Ensuite, il y a la mise en place d'un filtre passe-bas. Ce filtre transmet les signaux qui sont inférieurs au cut-off (seuil) défini et atténue les fréquences du signal brut qui sont au-dessus de ce cut-off. Par défaut, il est défini à 850 Hz. Enfin, le modèle ajoute du bruit gaussien au signal afin de modéliser l'environnement du modèle. Selon David et al, l'environnement de séquençage produit beaucoup de bruit qui s'ajoute au signal brut [3]. Ainsi, le pore modèle génère une liste de signaux correspondant aux signaux qu'auraient potentiellement générés le séquenceur minION.

Enfin, l'étape de basecalling permet de transmettre le signal généré à Albacore. Albacore est le logiciel utilisé lors des séquençages avec le séquenceur minION. L'utilisation de ce logiciel modélise donc de manière optimale l'étape de basecalling lors d'un séquençage réel.

2.3 Correction des erreurs et récupération de l'information

Une fois le message transmis, le décodeur effectuera les mêmes calculs du syndrome et de la somme que lors de l'encodage du signal. Une nouvelle séquence auxiliaire est générée, elle ressemble à la première séquence auxiliaire initiale avec un caractère en moins ou en plus dans le cas d'une délétion ou d'une insertion. La somme de la séquence auxiliaire sera notée ω. Par la suite, les étapes seront différentes s'il s'agit d'une délétion ou d'une insertion (Tenengolts et al, 1984 [12]).

Pour la détection et la correction d'une délétion, les calculs de S1 et S2 seront effectués. S1 permet de trouver la valeur du caractère perdu.

Calcul de S1 : $S1 = B - \sum_{i=0}^{n} a'_i \ \ mod(q)$

Le deuxième calcul, appelé S2, sert à retrouver la séquence auxiliaire initiale.

Calcul de S2 : $S1 = A - \sum_{i=0}^{n} i \times a'_i \ \ mod(n+1)$

Si $S2 \geq \omega$ alors alors c'est un 1 qui a été perdu, sinon c'est un 0. La différence de S2 avec oméga permet de connaître la position du caractère manquant : si $S2 < \omega$ alors le chiffre correspond au nombre de zéro à gauche du caractère, sinon le chiffre S2 correspond au nombre de 1 à droite du caractère manquant. Ensuite, l'algorithme essaie l'ensemble des possibilités de placement pour le caractère afin que la séquence corresponde à la séquence auxiliaire.

Pour la détection et la correction d'une insertion, les calculs de S1 et S2 sont effectués différemment. Dans ce cas, le résultat de S1 correspond au caractère qui a été ajouté.

Calcul de S1 : $S1 = \sum_{i=0}^{n} a'_i \ \ mod(q) - B$

Le résultat de S2, quant à lui, est utilisé pour retrouver la séquence auxiliaire initiale.

Calcul de S2 : $S1 = \sum_{i=0}^{n} i \times a'_i \ \ mod(n+1) - A$

Si $S2 = 0$ alors le caractère à enlever est à la fin de la séquence auxiliaire. Si $0 < S2 < \omega - 1$ alors il faut retirer le 0 de la séquence auxiliaire afin de faire en sorte de laisser un nombre de 1 à droite du symbole qui est égal à S2. Si $S2 = \omega - 1$, alors il faut retirer le premier caractère de la séquence auxiliaire et si $S2 > \omega - 1$ alors il faut retirer le 1 de la séquence auxiliaire de façon à laisser un nombre de 0 à droite du symbole qui est égal à S2. De la même manière que précédemment, l'algorithme essaie l'ensemble des possibilités de placement pour le caractère afin que la séquence corresponde à la séquence auxiliaire.

2.4 Génération des séquences

Trois types de séquences ont été générés de trois longueurs différentes (500, 8 000 et 16 000 paires de bases).

Les séquences d'une longueur de 500 et de 8 000 ont été obtenues en codant le début du texte du petit chaperon rouge de Charles Perrault (cf figure **2**). Les séquences d'une longueur de 16 000 paires de bases ont été obtenues en doublant la séquence d'une longueur de 8 000 pb.

Pour le moment, les séquences n'ont pas été enrichies du code de correction d'erreurs qui est encore en train d'être ajusté par l'équipe du Lab-STIIC.

A. "Il était une fois une petite fille de village, la plus jolie qu'on eût "

B. "Il était une fois une petite fille de village, la plus jolie qu'on eût su voir ; sa mère en était folle, et sa mère-grand plus folle encore. Cette bonne femme lui fit faire un petit chaperon rouge, qui lui seyait si bien que partout on l'appelait le petit Chaperon rouge.
Un jour sa mère, ayant cuit et fait des galettes, lui dit : « Va voir comme se porte ta mère-grand, car on m'a dit qu'elle était malade, porte-lui une galette et ce petit pot de beurre. » Le petit Chaperon rouge partit aussitôt pour aller chez sa mère-grand, qui demeurait dans un autre village. En passant dans un bois elle rencontra compère le loup, qui eut bien envie de la manger ; mais il n'osa, à cause de quelques bûcherons qui étaient dans la forêt. Il lui demanda où elle allait ; la pauvre enfant, qui ne savait pas qu'il est dangereux de s'arrêter à écouter un loup, lui dit : « Je vais voir ma mère-grand, et lui porter une galette avec un petit pot de beurre que ma mère lui envoie. »
— Demeure-t-elle bien loin ? lui dit le loup.
— Oh ! oui, dit le petit Chaperon rouge, c'est par-delà le moulin que vous voyez tout là-bas, à la première maison du village.
— Hé bien, dit le loup, je veux l'aller voir aussi ; je m'y en vais par ce chemin ici, et toi par ce chemin-là, et nous"

Figure 2 – Texte du petit chaperon rouge de Charles Perrault, utilisé pour l'encodage des différentes séquences de longueurs de 500 (A) et 8 000 (B)

Différentes séquences ont été générées en variant leur composition. Un première séquence sans modification a été générée. La seconde séquence a été générée en ajoutant des homopolymères à la première. Enfin, une troisième séquence a été produite selon un postulat technique de la technologie minION : dans le cas où l'on veut séquencer une séquence d'ADN ordinaire nous ne pouvons pas modifier la composition de la séquence. Mais dans notre cas, où nous synthétisons la séquence de notre choix, il est possible de choisir les bases qui composeront la séquence. D'un point de vue technique, lors de la génération des signaux électriques pendant le passage de l'ADN par le pore, les 6-mers émettent des signaux différents les uns des autres. Certains signaux peuvent être plus proches que d'autres et nous pouvons émettre le postulat que lorsque deux 6-mers émettant un signal proche se suivent, cela peut avoir une incidence sur l'assimilation du signal par Albacore et donc cela pourrait entraîner des erreurs de lecture de la séquence. Afin de palier ce phénomène, il a été imaginé par le lab-STICC un algorithme qui optimise les distances entre les fréquences de deux K-mers consécutifs. Ainsi, l'algorithme calcule, à chaque pas, la distance entre deux 6-mers successifs et la compare à un seuil (5 pA). De cette façon, si la distance entre les deux bases est inférieure au seuil, alors la deuxième base sera remplacée par une autre en suivant un ordre établi. Le code sera ensuite modifié afin de pouvoir retrouver la séquence initiale et pouvoir décoder le message correctement.

3 Résultats

3.1 Séquençage des séquences

3.1.1 Résultats de la simulation avec DeepSimulator

Afin de se rapprocher un maximum des résultats que nous obtiendrons lors d'un séquençage réel avec minION, nous avons tester plusieurs paramètres sous DeepSimulator. Dans leurs travaux pour la création du simulateur, Li et al ont pu montrer que certains paramètres s'approchaient le plus d'un séquençage minION (Li et al, 2018 [8]). Ainsi, nous avons choisis de régler les paramètres du bruit du signal à 1.2. Nous avons par la suite réalisé plusieurs tests en variant le cut-off du filtre passe bas.

Les taux d'identités, d'insertions/délétions et de substitutions ont été calculés ainsi :

$$taux\ d'identités = \frac{Nombre\ de\ matchs}{taille\ de\ l'alignement} \quad taux\ d'insertion/délétion = \frac{Nombre\ de\ gaps}{taille\ de\ l'alignement}$$

$$taux\ de\ substitutions = \frac{Nombre\ de\ substitutions}{taille\ de\ l'alignement}$$

$$taux\ d'erreurs = \frac{Nombre\ de\ substitutions + nombre\ d'insertions/délétions}{taille\ de\ l'alignement}$$

Ces calculs permettent de comparer les séquences entre elles et de déterminer laquelle est celle qui génère le moins d'erreurs lors du séquençage. Cela permettra également de vérifier quels paramètres permettent au mieux de simuler un séquençage par minION. Dans l'idéal, nous recherchons à avoir des taux d'erreurs, de substitutions et de délétions/insertions bas et un taux d'identité élevé.

Pour comparer les différents paramètres des séquences, 100 simulations ont été réalisées pour chacune des séquences avec les paramètres que nous souhaitions étudier. Nous avons réalisé un alignement global entre la séquence initiale et la séquence obtenue après la simulation. Cet alignement a été réalisé avec le logiciel fasta, permettant de réaliser un alignement global. Ensuite, un script python a été créé afin de réaliser une analyse des différents résultats. Dans un premier temps, le script prend en entrée le fichier texte dans lequel est stocké le résultat de l'alignement pour chaque simulation et pour chaque longueur. Les deux séquences sont stockées dans deux listes et les listes sont ensuite comparées afin de calculer différents taux (taux d'identités, d'insertions/délétions, de substitution et d'erreur) sous différentes conditions. Dans un premier temps, le script réalise le calcul des taux pour l'ensemble de l'alignement. Dans un second temps, il calcule les taux pour une fenêtre d'une longueur de 100 prise aléatoirement dans la séquence. Enfin, les taux sont calculés le long de la séquence grâce à une fenêtre glis-

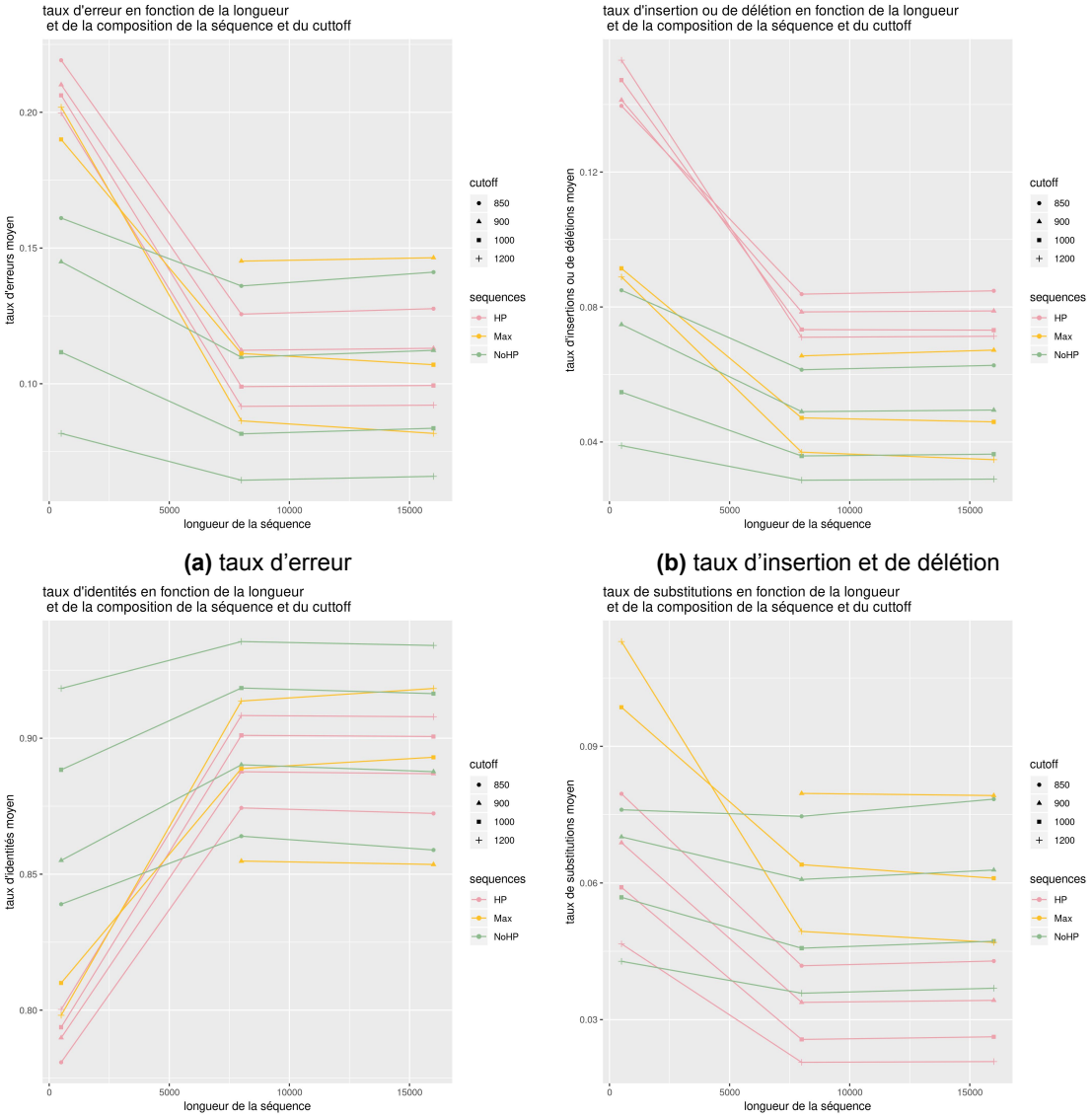
sante afin de voir l'évolution des erreurs le long de la séquence. De ce fait, nous pourrons voir l'endroit où le séquençage réalise le plus d'erreur. Il a été pris un pas de 50 et une fenêtre de 100 pour les séquences de 500 paires de bases, un pas de 800 et une fenêtre de 1 600 pour les séquences de 1600 paires de bases et un pas de 1 600 et une fenêtre de 3 200 pour les séquences de 16 000 paires de bases, de façon à avoir des résultats proportionnels les uns aux autres.

Grâce au logiciel R, nous avons fait une moyenne de chaque taux pour chacun des groupes de simulations puis avons créé des graphiques avec ggplot. Nous avons testé différents cut-off afin de dégager les tendances des différentes séquences en fonction de leur longueur. Ainsi, nous avons essayé de simuler le séquençage avec un cut-off de 850 (qui est le paramètre par défaut de l'application et représente une précision de 92 %), de 900, de 1 000 et de 1 200.

La figure **3** visualise les courbes présentant les différents taux sur l'ensemble de l'alignement global en fonction des différents cut-off et en fonction de la longueur et de la composition de la séquence. La couleur des courbes permet de distinguer les séquences en fonction de leur composition (avec homopolymères (HP), sans homopolymères (NoHP) ou avec des distances optimisée (Max)), la forme permet de distinguer la valeur du cut-off du filtre passe bas choisi pour les simulations. On peut remarquer que plus le cut-off augmente et plus les taux s'améliorent. On remarque que la composition de la séquence influe aussi sur les erreurs, puisque les séquences aux distances maximisées ont des taux d'erreurs plus grands que les autres séquences. On voit aussi que le taux d'insertions et de délétions pour les séquences qui possèdent des homopolymères est plus haut que pour les autres séquences. Lorsque que le cut-off est réglé à 850, les homopolymères ont malgré tout un taux d'identités plus haut que les autres. En revanche, en augmentant le cut-off on voit néanmoins que la tendance s'inverse en faveur des séquences qui ne présentent pas d'homopolymères. On peut également remarquer, sur ces graphiques, que les séquences de 8 000 paires de bases et de 16 000 paires de bases ne possèdent pas beaucoup de différences entre elles en ce qui concerne les différents taux. Ce qui n'est cependant pas le cas pour les séquences de 500 paires de bases qui possèdent des taux moins intéressants.

Afin de savoir si cette différence est due à une différence de séquençage qui a pour origine la longueur de la séquence ou à un déséquilibre des longueurs lors du calcul des taux, nous avons réalisé un calcul des taux sur une fenêtre aléatoire dans l'alignement. Les résultats sont visibles sur la figure **4**.

De cette manière, nous visualisons les courbes représentant les différents taux sur sur une séquence de 100 paire de bases choisie aléatoirement le long de l'alignement global en fonction



(a) taux d'erreur

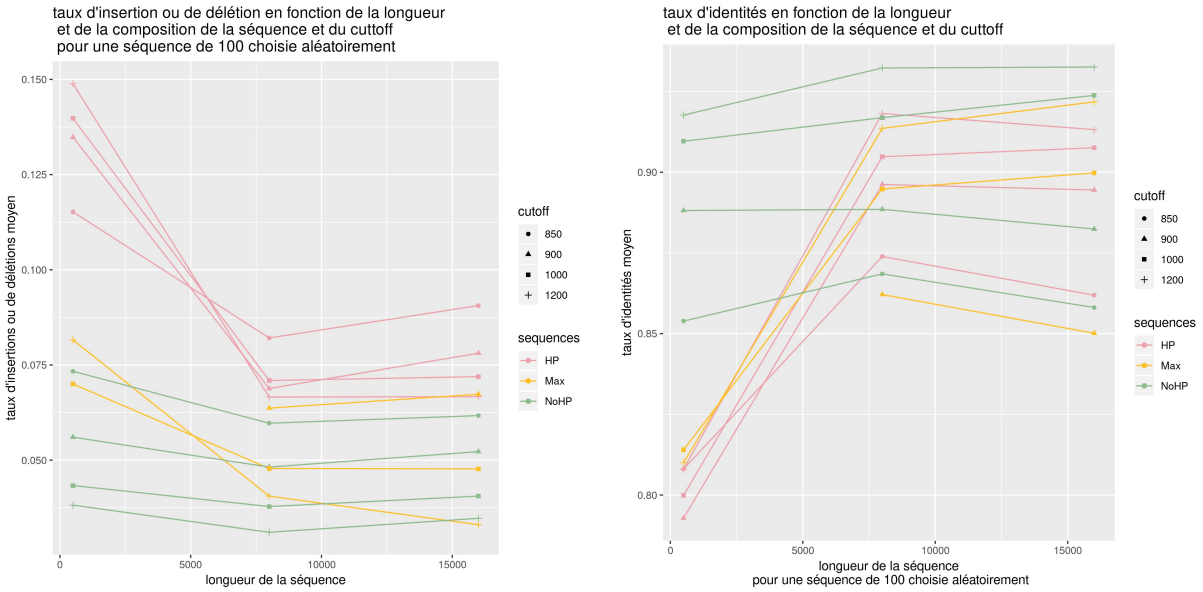
(b) taux d'insertion et de délétion

(c) taux d'identités

(d) taux de substitution

Figure 3 – Graphiques des différents taux en fonction de la longueur de la séquence pour l'ensemble de l'alignement global selon la composition de la fréquence et la fréquence du cut-off du filtre passe-bas. Les couleurs diffèrent selon la composition de la séquence, les formes selon la fréquence du cut-off (en Hertz) entrée en paramètres de la simulation.

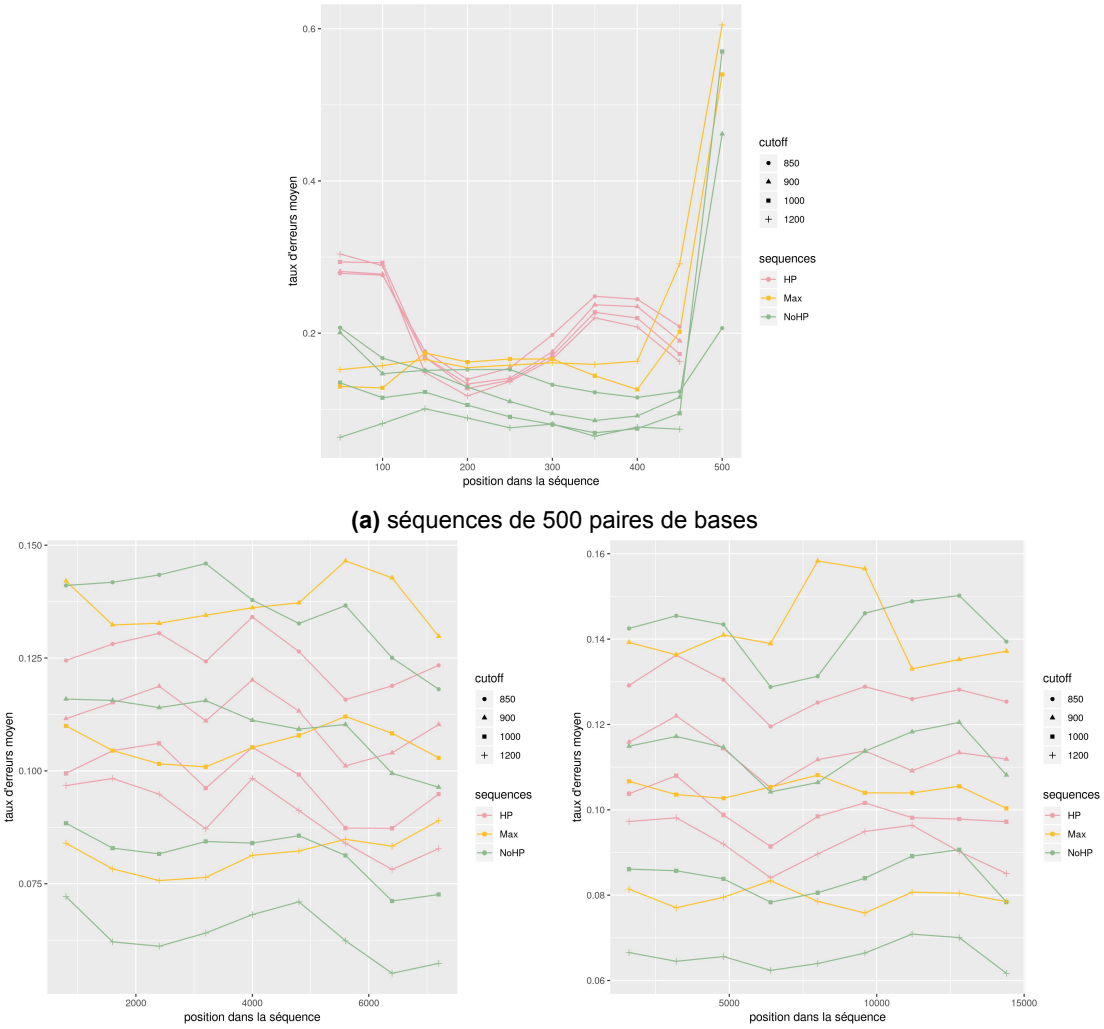
des différents cut-off et en fonction de la longueur et de la composition de la séquence. On peut voir que, comme lors des calculs pour l'ensemble de l'alignement, le séquençage sur des séquences de 500 paires de bases génère plus d'erreurs que le séquençage sur des séquences plus longues. Ces graphiques présentent les mêmes tendances que les graphiques précédents en ce qui concerne la variation de la composition des séquences et l'augmentation du cut-off. Cependant, les séquences sans homopolymères ont tendance à montrer une différence moins marquée entre les séquences de longueur de 500 paires de bases et les séquences plus longues.



(a) taux d'insertion et de délétion **(b)** taux d'identités

Figure 4 – Graphiques du taux d'insertions et de délétions et du taux d'identité en fonction de la longueur de la séquence pour une fenêtre aléatoire de l'alignement global selon la composition de la fréquence et la fréquence du cut-off du filtre passe-bas. Les couleurs diffèrent selon la composition de la séquence, les formes selon la fréquence du cut-off (en Hertz) entrée en paramètres de la simulation.

La figure **5** visualise les courbes représentant le taux d'erreur au fur et à mesure de l'alignement global en fonction des différents cut-off et de la composition de la séquence pour chacune des trois longueurs. Cette figure permet de se rendre compte de l'évolution des erreurs le long de la séquence et de détailler ainsi l'évolution des erreurs. On remarque que les erreurs sont plus importantes aux extrémités pour les séquences de 500 paires de bases. Les séquences plus longues possèdent des variations plus continues. Il est cependant difficile d'analyser ces variations car le pas et la fenêtre sont grandes et que les calculs des taux dépendent de l'emplacement de la fenêtre dans la séquence.



(a) séquences de 500 paires de bases

(b) séquences de 8 000 paires de bases

(c) séquences de 16 000 paires de bases

Figure 5 – Graphiques représentant le taux d'erreur le long de la séquence pour chacune des longueurs (500, 8000 et 16000 paires de bases). Les couleurs diffèrent selon la composition de la séquence, les formes selon la fréquence du cut-off (en Hertz) entrée en paramètres de la simulation.

3.1.2 Résultat avec le séquençage MinION

Le séquençage réalisé avec le séquenceur MinION permettra par la suite de valider les résultats obtenus par la simulation de séquençage. Les données obtenues à la suite de cette étape ne sont cependant pas encore disponibles.

4 Discussion

Dans un premier temps, nous avons cherché à déterminer les paramètres importants à indiquer lors des simulations de séquençage afin d'avoir un séquençage le plus proche de la réalité. Nous savons que les séquences qui présentent beaucoup d'homopolymères ont normalement un taux d'insertions ou de délétions plus élevé lors du séquençage que les séquences qui en

contiennent moins. Nos graphiques ont permis de voir que, quel que soit la fréquence choisie pour la simulation, les séquences qui contiennent des homopolymères présentent toujours le taux d'insertions et de délétions le plus élevé. DeepSimulator permet donc bien de se rendre compte de ce point. Cependant, le taux d'erreurs des séquences comportant des homopolymères est plus faible pour un cut-off de 850 Hz par rapport aux séquences sans homopolymères. Cela est du au fait que les séquences sans homopolymères possède un taux de substitutions plus élevé pour un cut-off à 850. Il serait intéressant de vérifier si cette tendance est du à la simulation ou si nous la retrouverions lors d'un séquençage réel.

Les graphiques nous permettent également de nous faire une idée sur l'intérêt de l'optimisation des distances pour des séquences ADN. L'optimisation des distances permet d'avoir, pendant la génération des signaux, lors du séquençage, une grande différence entre les signaux de deux bases consécutives. Nous avons émis l'hypothèse que de courtes distances entre deux bases pouvaient générer plus d'erreurs lors de la lecture du signal. Les graphiques laissent cependant penser que le basecaller n'est pas pénalisé par les petites distances. Au contraire, on peut se supposer qu'il est adapté au fait que les distances ne soient pas optimisées puisque les taux pour les séquences avec des distances optimisées sont moins bons que pour les autres séquences.

De plus, nous avons pu remarquer que la taille de la séquence avait une influence sur le taux d'erreurs. Les petites séquences présentent plus d'erreurs. Nous aurions pu penser que cela était dû aux en-tête, mais les taux restent sensiblement les mêmes lorsque l'on choisi de les faire sur une fenêtre aléatoire dans la séquence. Ce phénomène peut avoir comme origine le fait que le séquençage minION est normalement plus adapté à la génération de long reads plutôt que de reads court, 500 paires de bases étant une longueur très basse pour un séquenceur capable de générer des reads de plusieurs dizaines voire centaines de kb.

En ce qui concerne les erreurs repérées le long de la séquence par la fenêtre glissante, on peut remarquer de grandes fluctuations pour les séquences de 500 paires de bases. En effet, les taux d'erreurs sont plus hauts au début et à la fin de la séquences. Certaines séquences de 500 paires de bases continue au delà de 500 sur l'axe des abscisses, cela est dû à l'alignement. Celui-ci n'a pas une longueur de 500 paires de bases, il est généralement plus grand puisque qu'il y a eu des insertions ou des délétions et donc l'apparition de gaps. Les graphes présentent la moyenne des taux pour les 100 simulations réalisées et il s'avère qu'il n'y avait que certaines simulations pour lesquelles nous avons calculés un taux d'erreurs au delà de 500 paires de bases. Cela pourrait potentiellement générer un biais lors de la mise en place des graphiques et pourrait expliquer le pic au niveau du taux d'erreurs sur la fin de la séquence.

Par rapport aux erreurs qui se situent au début de la séquence, nous pouvons supposer que le simulateur est susceptible de réaliser plus d'erreurs au début du séquençage. Il est possible que cela ne se remarque pas pour les séquences des autres longueurs puisqu'elles sont plus longues et que nous avons choisis une fenêtre et un pas plus grand. Il est également possible que ce soit pour cette raison que les taux d'erreurs des séquences de 500 paires de bases soient en général plus élevé que les taux d'erreurs des séquences de 8 000 et 16 000 paires de bases. Pour valider ces hypothèses, il serait intéressant par la suite de réaliser à nouveau un calcul de taux sur les 500 ou 1 000 premières et dernières paires de bases de la séquence pour analyser et comparer ces taux d'erreurs avec ceux de la séquences de 500 paires de bases pour s'assurer de savoir s'il y a bien une différence entre les longueurs en ce qui concerne le séquençage des extrémités. De plus, la différence de pas et de fenêtre pour les séquences plus grandes provoque possiblement une atténuation des erreurs de début mais aussi des atténuations des erreurs le long de la séquences. Il serait intéressant de faire des graphiques avec des pas de 50 et des fenêtres de 100 pour ces longueurs également afin de comprendre en détail la fluctuation des erreurs lors du séquençage de long reads.

5 Conclusion

Les résultats suggèrent que la simulation de séquençage avec DeepSimulator correspond à ce que nous pouvions attendre d'un séquençage réel avec la technologie minION notamment en ce qui concerne les erreurs pour les séquences contenant des homopolymères. De plus, nous pouvons écarter l'idée d'utiliser un algorithme pour optimiser les distances pour la lecture des séquences d'ADN. Nous avons pu dégager les types d'erreurs de séquençage et leur localisation grâce à DeepSimulator. Ces paramètres pourront être utilisés afin d'optimiser un maximum la génération des séquences dans le cas où le séquenceur modélise correctement un séquençage réel avec minION. Par la suite, nous comparerons les résultats des simulations avec les données obtenues lors d'un séquençage réel que nous analyserons. Si DeepSimulator permet une bonne modélisation des résultats, nous pourrons utiliser les conclusions de cette étude afin de générer des séquences optimales pour minimiser les erreurs produites lors du séquençage et nous permettre de récupérer de l'information qui ne soit pas corrompue. Nous pourrons également réaliser des simulations afin de tester l'algorithme de correction d'erreurs et vérifier que nous retrouvons la totalité de l'information. Ainsi, le stockage d'information sur un support ADN se ferait sans risque de perte d'information.

Références

[1] George M. Church, Yuan Gao et Sriram Kosuri : Next-generation digital information storage in DNA. *Science (New York, N.Y.)*, 337(6102) :1628, septembre 2012.

[2] Jonathan P. L Cox : Long-term data storage in DNA. *Trends in Biotechnology*, 19(7) :247–250, juillet 2001.

[3] Matei David, L. J. Dursi, Delia Yao, Paul C. Boutros et Jared T. Simpson : Nanocall : an open source basecaller for Oxford Nanopore sequencing data. *Bioinformatics*, 33(1) :49–55, janvier 2017.

[4] Joe Davis : Microvenus. *Art Journal*, 55(1) :70–74, mars 1996.

[5] James Glanz : Data Centers Waste Vast Amounts of Energy, Belying Industry Image. *The New York Times*, septembre 2012.

[6] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M. LeProust, Botond Sipos et Ewan Birney : Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435) :77–80, février 2013.

[7] Bertrand Jordan : Séquençage d'ADN : l'offensive des nanopores - Chroniques génomiques. *médecine/sciences*, 33(8-9) :801–804, août 2017.

[8] Yu Li, Renmin Han, Chongwei Bi, Mo Li, Sheng Wang et Xin Gao : DeepSimulator : a deep simulator for Nanopore sequencing. *Bioinformatics*, 34(17) :2899–2908, septembre 2018.

[9] Hengyun Lu, Francesca Giordano et Zemin Ning : Oxford Nanopore MinION Sequencing and Genome Assembly. *Genomics, Proteomics & Bioinformatics*, 14(5) :265–279, octobre 2016.

[10] Aisling O' Driscoll et Roy D. Sleator : Synthetic DNA. *Bioengineered*, 4(3) :123–125, mai 2013.

[11] David Reinsel, John Gantz et John Rydning : The Digitization of the World from Edge to Core. page 28, 2018.

[12] G. Tenengolts : Nonbinary codes, correcting single deletion or insertion (Corresp.). *IEEE Transactions on Information Theory*, 30(5) :766–769, septembre 1984.

[13] Eske Willerslev, Enrico Cappellini, Wouter Boomsma, Rasmus Nielsen, Martin B. Hebsgaard, Tina B. Brand, Michael Hofreiter, Michael Bunce, Hendrik N. Poinar, Dorthe Dahl-Jensen, Sigfus Johnsen, Jørgen Peder Steffensen, Ole Bennike, Jean-Luc Schwenninger,

Roger Nathan, Simon Armitage, Cees-Jan de Hoog, Vasily Alfimov, Marcus Christl, Juerg Beer, Raimund Muscheler, Joel Barker, Martin Sharp, Kirsty E. H. Penkman, James Haile, Pierre Taberlet, M. Thomas P. Gilbert, Antonella Casoli, Elisa Campani et Matthew J. Collins : Ancient biomolecules from deep ice cores reveal a forested southern Greenland. *Science (New York, N.Y.)*, 317(5834) :111–114, juillet 2007.

Résumé : Le monde génère beaucoup d'informations et les supports de stockage de ces informations sont une problématique actuelle. Depuis quelques années, la possibilité du stockage sur un support ADN commence à émerger. Ce type de stockage permettrait de stocker beaucoup d'informations en prenant très peu de place mais aussi de stocker ces informations sur le long terme puisque l'ADN est une molécule très stable dans le temps. Notre projet vise à étudier le stockage de l'information sur de l'ADN, et plus précisément de se concentrer sur la restitution de l'information après un séquençage par le séquenceur MinION. De ce fait, ce stage se concentre sur la préparation d'un environnement bioinformatique permettant de modéliser le séquençage de l'information et d'en prévoir les erreurs afin de mieux les corriger. Après avoir codé du texte en séquences ADN, nous avons utilisé le simulateur DeepSimulator afin de simuler le séquençage des séquences générées. Nous vérifions d'abord que le séquenceur permet de donner une idée exacte de la réalité du séquençage par nanopore puis nous analysons les types d'erreurs afin de prévoir les erreurs générées lors d'un séquençage réel et adapter les séquences afin de produire le minimum d'erreurs possible. Par la suite, une comparaison entre la simulation du séquençage et le séquençage réel se fera afin de pouvoir vérifier les hypothèses. Si le simulateur répond bien aux attentes, nous pourrons l'utiliser afin de tester les codes correcteurs d'erreurs pour vérifier que nous serons bien capables de récupérer toute l'information.
Mots clés : Séquençage de 3ème génération, MinION, code correcteur d'erreurs, simulation

Abstract : The world generates lot of informations and storage supports for these informations are a current problematic. Since few years, possibility of storage information on a DNA support begin to emerges. his type of storage enable to store lot of informations in a few space but enable also to stor these informations for a long time because DNA is a stable molecule over time. Our project aim for study DNA storage, and more precisly, we focus on the restitution of information after a sequencing by minION sequencer. Hence, this internship focus on preparation of bioinformatic environment enable to model sequencing of information and to predict errors for correct them better. After encoding text to DNA, we use DeepSimulator (a sequencing simulator) to simulate sequencing of generated sequences. In a first time, we check that sequencer can give a good idea of a real sequencing, then, we analyze types of erros to predict errors generated during a real sequencing et to adapt sequences to produce a minimum of errors. Therafter, a comparison between simulation and reality have to be realised for checked hypothesis and if simulator delivers the goods, We can use it for verify that we are be able to recover all the information.
Mots clés : 3rd generation sequencing, MinION, error correcting code, simulation