Alignement multiple et séquençage de troisième génération

Coralie Rohmer

Université de Lille

Thèse

pour obtenir le grade de :

Docteur de l'Université de Lille

par

Coralie ROHMER

Alignement multiple et séquençage de troisième génération

Thèse soutenue le 4 décembre 2023 devant le jury composé de :

Hélène TOUZET	Directrice de recherche CNRS, CRIStAL	(Directrice de thèse)
Antoine LIMASSET	Chargé de recherche CNRS, CRIStAL	(Encadrant)
Matthias ZYTNICKI	Chargé de recherche INRAE, MIAT	(Rapporteur)
Sèverine BERARD	Maîtresse de conférences Université de Montpellier, ISEM	(Rapportrice)
François BOULIER	Professeur des universités Université de Lille, CRIStAL	(Examinateur)
Thierry LECROQ	Professeur des universités Université de Rouen-Normandie, LITIS	(Examinateur)

CRIStAL - UMR 9189
Bâtiment ESPRIT Cité Scientifique 59650 VILLENEUVE D'ASCQ
2022-2023

Résumé

Le séquençage d'ADN n'a cessé d'évoluer ces dernières décennies, notamment avec l'arrivée du séquençage à haut-débit. La troisième génération de séquenceurs a produit de nouvelles données, que l'on nomme *long reads*, qui permettent d'accéder à de nouvelles informations biologiques en surmontant les contraintes des générations précédentes, telles que la faible longueur et les biais de composition des séquences. Néanmoins, du fait de leurs forts taux et profil d'erreur, ces long reads posent également de nouvelles questions d'analyses de données. Cette thèse s'inscrit dans cette problématique et traite plus précisément du sujet de l'alignement multiple des long reads.

L'alignement multiple permet, comme son nom le suggère, d'aligner plusieurs séquences d'ADN entre elles. Ce domaine joue un rôle très utile dans l'analyse de séquences. Cela permet, entre autres, d'identifier des domaines fonctionnels partagés entre espèces proches, d'identifier des variations entre différents individus ou d'analyser des gènes issus d'une même famille génétique pour en retracer l'histoire évolutive en phylogénie. C'est dans ce cadre que la plupart des méthodes d'alignement multiple ont vu le jour. L'application aux long reads est particulière, puisqu'il s'agit de détecter et corriger des erreurs commises lors du séquençage mais aussi d'identifier les variations au sein de l'ADN entre différents individus.

Par conséquent, l'objectif de cette thèse est de vérifier s'il est possible d'appliquer les outils déjà existants, utilisant différentes méthodes d'alignement multiple, sur les long reads. Pour cela, j'ai développé un pipeline automatisé permettant la comparaison d'outils d'alignement multiples, ainsi qu'un benchmark original sur lequel j'ai pu mener l'évaluation de neuf outils d'alignement, de manière reproductible.

Abstract

DNA sequencing has continuously evolved over recent decades, especially with the advent of high-throughput sequencing. The third generation of sequencers has produced new data, referred to as long reads, which provide access to new biological information while overcoming the constraints of previous generations, such as short sequence length and sequence composition biases. However, due to their high error rates and error profiles, these long reads also pose new data analysis challenges. This thesis addresses this issue and specifically deals with the topic of multiple sequence alignment of long reads.

Multiple sequence alignment, as its name suggests, allows for the alignment of several DNA sequences with each other. This field plays a very useful role in sequence analysis. Among other things, it allows for the identification of functional domains shared between closely related species, the identification of variants between different individuals, and the analysis of genes from the same genetic family to trace their evolutionary history in phylogeny. It is in this context that most multiple sequence alignment methods have been developed. The application to long reads is unique, as it involves detecting and correcting errors made during sequencing, as well as identifying variations within the DNA between different individuals.

In this context, the objective of this thesis is to determine whether it is possible to apply existing tools, which use various multiple sequence alignment methods, to long reads. For this purpose, I developed an automated pipeline for comparing such tools, as well as an original benchmark on which I was able to conduct a reproducible evaluation of nine alignment tools.

Remerciements

Je tiens tout d'abord à remercier Hélène Touzet et Antoine Limasset de m'avoir permis de faire cette thèse.

Je n'aurais pu espérer avoir meilleure directrice de thèse qu'Hélène. Ses conseils et sa grande expérience m'ont toujours été d'un grand secours. Ses réflexions et ses connaissances m'ont permis de faire avancer mes travaux. Toujours à l'écoute, elle a su me guider avec bienveillance jusqu'à la fin de ce projet tout en me laissant faire mes propres échecs pour apprendre par moi-même. Elle a été mon plus grand soutien tant sur le plan professionnel que sur le plan humain.

Je remercie Antoine pour son encadrement de qualité. Son expertise et sa logique m'ont toujours bluffée. Il a joué un grand rôle dans l'avancement de mes travaux. Son enthousiasme débordant et contagieux pour son domaine a été une source importante de motivation. Il a toujours réussi à se rendre disponible alors même qu'il a vu sa vie être bouleversée avec la naissance de ses deux enfants. Merci pour les heures passées à jouer au Mahjong avec le reste de l'équipe grâce à ton amour pour ce jeu et merci pour ta sympathie et ta bonne humeur.

Je tiens aussi à remercier l'intégralité du reste de l'équipe BONSAI du laboratoire CRIStAL pour m'avoir supportée ces 4 dernières années ainsi que pour leurs humours et leurs gentillesses.

Merci à Areski, à Stéphane, à Camille, à Laurent, à Maude, à Mikaël et à Jean-Stéphane. Merci surtout à Bastien d'avoir toujours pris le temps de répondre à mes innombrables questions sur l'enseignement, sans soupirer une seule fois.

Merci aux autres doctorants d'avoir supporté mes cris de désespoir face à cette thèse qui n'en finissait pas. J'espère ne pas vous avoir démoralisés et je vous souhaite à tous d'avoir un jour à écrire les remerciements de vos propres thèses respectives comme je suis actuellement en train de le faire.

Merci à Lilian et Pierre d'avoir apporter un vent de fraîcheur en remplissant mon bureau, qui était vide jusque là, avec leur humour, leurs bonnes humeurs et nos (trop) nombreuses discussions. Ça a était un réel plaisir de connaître Lilian et un grand déplaisir de connaître Pierre. (Ne vous inquiétez pas pour Pierre, "déplaisir" est devenu un code pour lui dire que je l'apprécie.)

Merci à Léa d'avoir partager mon bureau, mon addiction aux sodas et mon désespoir de thésarde. Merci de ne pas m'avoir jugée à chaque fois que j'arrivais en retard.

Merci à Thomas pour toutes les discussions improbables et pour toutes ces fois où tu es venu perturber la tranquillité de notre bureau d'un coup de chaise roulante.

Merci à Timothée, Caleb et Igor pour les discussions aux pauses cafés et bon courage pour le début de vos thèses respectives.

Je remercie aussi la plateforme BILILLE pour tout ces repas partagés et pour tous leurs conseils professionnels.

Merci aussi au laboratoire CRIStAL et à l'intégralité de son secrétariat.

Je remercie aussi mes amis de m'avoir toujours écoutée patiemment me plaindre. Merci à Nicolas pour toutes ses heures passées à jouer qui m'ont permise de sortir la tête de la thèse. Merci à Marine, Sarah et Christelle pour toutes nos discussions, nos blagues sur la bioinformatique et pour ne jamais avoir posé la question "Alors la thèse, ça avance ?" mais plutôt attendu patiemment que je vous donne des nouvelles. Merci à Caroline, Justine et Chloé pour toutes nos soirées qui m'ont toujours fait un bien fou tant je me sens dans un cocon de bienveillance avec vous.

Et pour finir, cela vous semblera peut-être un peu orgueilleux mais je voudrais me remercier moi. En effet, s'il y a bien une personne qu'on ne ménage pas durant sa thèse, c'est soi-même. Merci Coralie d'avoir tenu ces quatre années. Désolée aussi pour toutes ces fois où j'ai dit que tu étais stupide. De toute évidence, si je suis en train de rédiger les remerciements de ma thèse, c'est que tu ne l'es pas tant que ça. Peut-être qu'une thèse est aussi le moment de faire la paix avec soi-même.

Non-remerciements

Je tiens spécifiquement à ne pas remercier l'épidémie de COVID-19 qui a eu lieu durant ma deuxième année de thèse et qui l'a impactée négativement. J'ai aussi une pensée négative pour la personne qui a volé mon ordinateur durant cette même année. Vraiment pas cool mec.

Sommaire

Préambule

Ayant grandi dans un milieu non-scientifique, je suis consciente du décalage qui existe parfois entre la population et les sciences. La médiation scientifique a toujours été au cœur de mes préoccupations.

Durant mes quatre ans de thèse, j'ai activement participé à faire comprendre mon domaine. J'ai tout d'abord enseigné durant trois ans l'informatique à l'école Polytech Lille. J'ai eu la chance d'encadrer à plusieurs reprises les stages d'Informatique au Féminin qui visent à promouvoir l'informatique auprès des jeunes filles. J'ai aussi eu l'occasion durant la semaine NSI (Numérique et Sciences Informatiques) de présenter le domaine de la bioinformatique à des collégiens.

Vous trouverez peut-être que cette thèse manque parfois de termes techniques et de complexité. Sachez que c'est un choix conscient et pleinement assumé. J'ose espérer avoir réussi à rendre ma thèse accessible aux plus grands nombres et pas uniquement aux personnes issues de mon domaine d'étude.

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Ceci est l'introduction d'une thèse qui présente une étude sur l'application des outils d'alignement multiple pour des données de séquençage de troisième génération. Cette introduction vise à expliquer le sujet de cette thèse. La section 1.1 présente le contexte. La section 1.2 explicite la problématique de ces travaux. Pour terminer, la section 1.3 dévoile le plan du manuscrit de thèse.

1.1 Contexte

L'*ADN (Acide DésoxyriboNucléique)* est un élément fondamental du vivant puisqu'il porte une grande partie des informations permettant le développement, le fonctionnement et la reproduction de tous les êtres vivants. Depuis l'invention du séquençage de Sanger en 1977, qui permet d'extraire l'information de la molécule d'ADN, l'analyse de ce que l'on nomme des séquences d'ADN a pris de plus en plus d'importance. En effet, réussir à analyser l'information contenu dans l'ADN, c'est comprendre comment le vivant fonctionne.

Plusieurs méthodes de séquençage sont nées après celle de Sanger [SNC77]. On parle principalement des méthodes dites de deuxième génération et de celles dites de troisième génération. Aucune d'entre elle ne permet d'obtenir l'information contenue dans l'ADN de façon claire. En effet, ces méthodes n'offrent que des parties de l'information qu'il faut ensuite assembler entre elles pour tout reconstituer. Si l'ADN était un livre, ces méthodes ne nous donneraient qu'un ensemble de citations du livre, sans aucune indication sur l'endroit où se situent ces citations dans le livre. Ces "citations", qu'on appelle des *reads*, peuvent en plus comporter des erreurs.

Les reads issus du séquençage de troisième génération sont appelés *long reads* car ils sont considérablement plus longs que ceux de la génération précédente.

1.2 Objectif

En bioinformatique, l'analyse de séquences d'ADN est un large domaine qui recoupe plusieurs sous-domaines. Durant mes travaux, je me suis surtout intéressée au sous-domaine de l'alignement de séquences et plus particulièrement à celui de l'alignement multiple. Aligner des séquences d'ADN entre elles permet de pouvoir les comparer [Not07]. Utiliser l'alignement multiple sur des reads a de nombreuses utilités, notamment de pouvoir identifier et supprimer les erreurs de séquençage en comparant les différents reads entre eux [SS11 ; Mor+21]. L'une des autres applications est de pouvoir détecter les différences entre plusieurs ADN différents. Mon but premier était d'essayer de détecter ces différences en utilisant l'alignement multiple et les long reads.

Au cours de mes recherches, j'ai découvert qu'il existait de nombreux outils d'alignement multiples et que certains étaient déjà utilisés sur les long reads [Vas+17a]. Cependant, je n'ai trouvé aucune étude qui testait clairement ces outils avec ces données. En effet, ces même outils ont été développés à l'époque pour des données très éloignées des long reads. La question était donc de savoir comment les outils d'alignement multiple se comportaient sur des reads issus de séquençage de troisième génération. Arrivaient-ils à donner des alignements de bonne qualité avec ces données ? Quels étaient les problèmes et les limites qu'ils pouvaient rencontrer sur ce type de données ? Le coût en terme de mémoire et de temps était-il important ? Une étude s'imposait pour répondre à ces problématiques.

1.3 Organisation du manuscrit

Ce manuscrit est organisé en huit chapitres. Le chapitre 1 que vous êtes en train de lire est l'introduction qui présente la thèse. Le chapitre 2 est consacré à la présentation du contexte biologique. Il présentera en détail ce qu'est l'ADN, à quoi il sert et comment il est organisé. Je parlerai aussi du rôle que joue l'informatique dans l'analyse de l'ADN et des données de séquençage d'ADN. Le chapitre 3 présentera en détail l'alignement de séquences. Je définirai d'abord ce terme avant de présenter les différentes méthodes qui permettent de le réaliser. Je distinguerai l'alignement de séquences deux à deux de l'alignement multiple et présenterai les outils d'alignement multiple étudiés. Le chapitre 4 présente le pipeline d'analyse développé durant ma thèse pour tester les différents outils d'alignement multiple. J'exposerai l'intégralité du protocole mis en œuvre par le pipeline, ainsi que tous

les aspects techniques qui l'accompagnent. Le chapitre 5 présentera les différents jeux de données utilisés pour la construction du benchmark. J'expliquerai les raisons de ces choix de données ainsi que les méthodes déployées pour leur génération. Le chapitre 6 présente l'intégralité des résultats obtenus. Je parlerai du comportement des outils et de l'influence de la taille des données, des erreurs de séquençage sur la qualité des alignements multiples ainsi que sur la performance en temps et mémoire. Le chapitre 7 est consacré à une discussion autour de l'ensemble de la thèse qui concerne certains points de l'analyse ainsi que d'autres travaux en cours. Le chapitre 8 présentera pour finir la conclusion et les perspectives de l'ensemble de mes travaux.

Chapitre 2

Bioinformatique des séquences

Cette thèse porte sur l'analyse de séquences d'ADN. Avant d'aborder le travail de recherche accompli, ce chapitre donne les éléments de contexte indispensables à la compréhension. La section 2.1 explique ce qu'est l'ADN, son utilité ainsi que la manière dont il est agencé. Une fois la notion de séquence d'ADN pleinement définie, j'expliquerai le rôle de l'informatique dans le traitement et l'exploitation de ces séquences dans la section 2.2 Je terminerai ce chapitre sur l'obtention, les caractéristiques et l'utilisation de ces séquences en section 2.3. C'est aussi dans cette dernière section que seront présentés les long reads, les données au centre de cette thèse.

2.1 Séquence d'ADN

2.1.1 Qu'est-ce que c'est ?

L'ADN (Acide désoxyribonucléique) est une molécule fondamentale au sein de la cellule, car elle contient l'ensemble de l'information génétique permettant le développement et le fonctionnement d'un individu. Cette molécule est composée de deux brins, eux-mêmes constitués d'une succession de *nucléotides* liés entre eux. Ces nucléotides sont composés de trois éléments : un *sucre* (le désoxyribose), un *groupement phosphate* et une *base azotée*. Les bases azotées sont au nombre de quatre : *adénine*, *thymine*, *guanine* et *cytosine*, notées respectivement A, T, G et C. Par simplification de langage, les nucléotides sont également appelés *bases*, en raison de leurs bases azotées, et donc notés A, T, G et C.

Les bases azotées se lient par *paires*. L'adénine et la thymine s'apparient entre elles grâce à deux liai-

sons hydrogène, tandis que la guanine et la cytosine s'apparient grâce à trois liaisons hydrogène. Les deux brins de l'ADN sont liés grâce à ces *paires de bases*, conférant à l'ADN sa structure particulière en double hélice, décrite en figure 2.1 et figure 2.2.

On appelle *séquence d'ADN* la succession des bases azotées des nucléotides qui constituent un brin d'ADN. La taille de ces séquences est notée en *paires de bases*, abrégée *pb*.

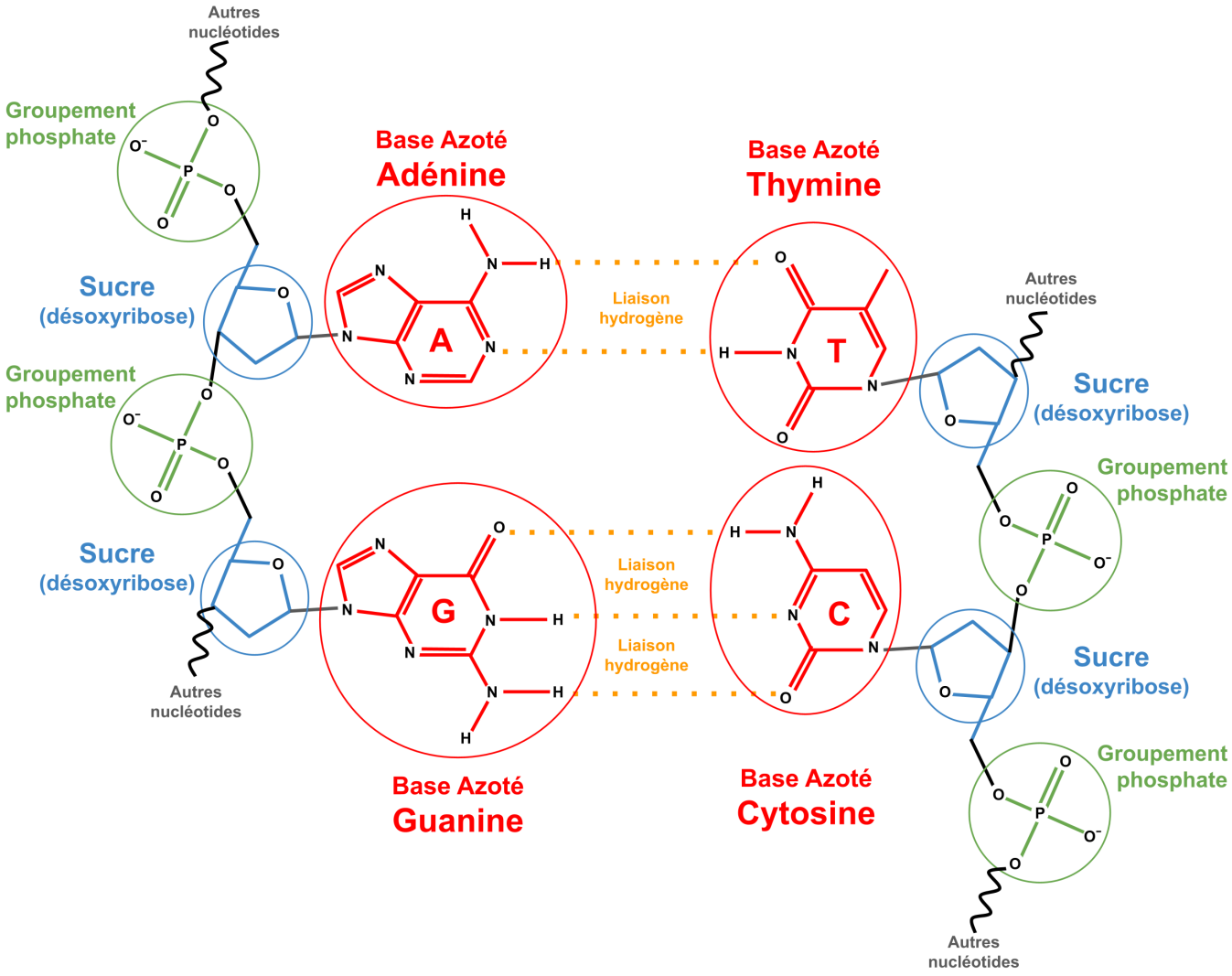


FIGURE 2.1 – Structure moléculaire de l'ADN. Un nucléotide est composé d'un sucre, d'un phosphate et d'une base azotée (Adénine, Thymine, Guanine ou Cytosine). Un brin d'ADN est composé d'une succession de nucléotides. Les deux brins de l'ADN sont liés grâce aux liaisons hydrogène entre les bases azotées.

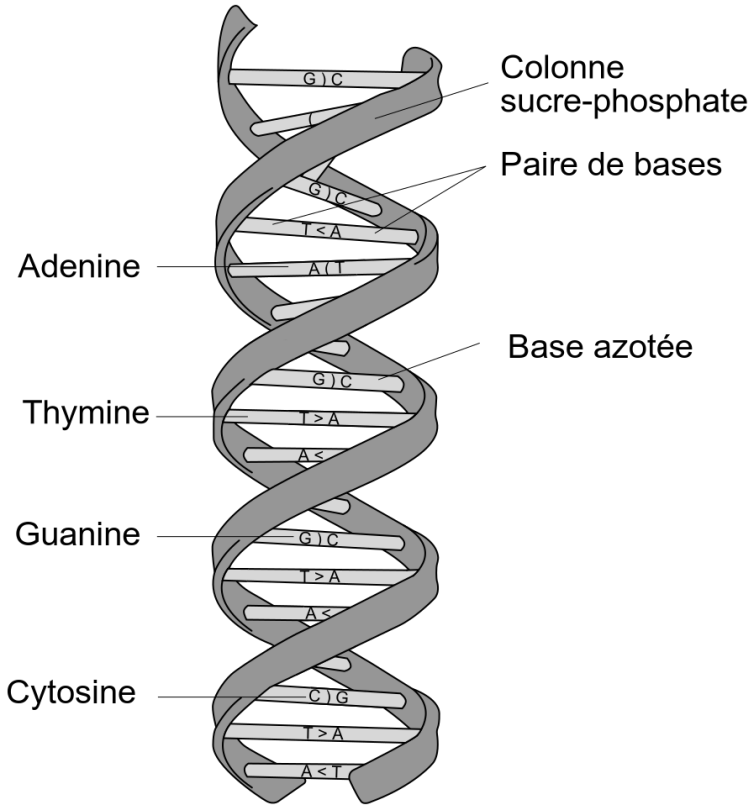


FIGURE 2.2 – Structure en double hélice de l'ADN. Les deux brins de l'ADN sont liés grâce aux paires formées par les bases azotées et forment cette structure particulière. Source : https ://fr.wikipedia.org/wiki/Acide_désoxyribonucléique.

2.1.2 Pourquoi est-ce important ?

L'ordre des nucléotides, ou bases, dans une séquence d'ADN permet de coder les acides aminés qui composent les protéines, des molécules essentielles au fonctionnement d'une cellule et donc d'un organisme. La correspondance entre bases et acides aminés se fait grâce au *code génétique* où une succession de trois bases code un acide aminé. La correspondance est détaillée en figure 2.3 et figure 2.4. Une séquence de nucléotides qui code une protéine est appelée un *gène*.

Pour comprendre le vivant, il est essentiel de connaître les séquences d'ADN et de savoir les analyser, car tous les domaines de la biologie reposent directement ou indirectement sur l'ADN. Connaître une séquence d'ADN, c'est connaître une protéine qui a été, est ou sera utilisée au cours du cycle de vie d'un organisme. Cela permet également de comprendre les mécanismes qui régulent l'expression de ces différents gènes au cours du temps et qui expliquent le fonctionnement de toute la machinerie cellulaire d'un être vivant. Une thèse entière serait nécessaire pour expliquer l'importance de l'ADN dans la science du vivant.

	T		C		A		G		
T	TTT TTC	Phénylalanine	TCT TCC	Sérine	TAT TAC	Tyrosine	TGT TGC	Cystéïne	T C
	TTA TTG	Leucine	TCA TCG		TAA TAG	STOP	TGA	STOP	A
							TGG	Tryptophane	G
C	CTT CTC CTA CTG	Leucine	CCT CCC CCA CCG	Proline	CAT CAC	Histidine	CGT CGC CGA CGG	Arginine	T C A G
					CAA CAG	Glutamine			
A	ATT ATC ATA	Isoleucine	ACT ACC ACA ACG	Thréonine	AAT AAC	Asparagine	AGT AGC	Sérine	T C
	ATG	Methionine			AAA AAG	Lysine	AGA AGG	Arginine	A G
G	GTT GTC GTA GTG	Valine	GCT GCC GCA GCG	Alanine	GAT GAC	Aspartate	GGT GGC GGA GGG	Glycine	T C A G
					GAA GAG	Glutamate			

FIGURE 2.3 – Correspondance entre les triplets de nucléotides et les acides aminés du code génétique. Par exemple, le triplet de nucléotides TTT code l'acide aminé phénylalanine. Les triplets TAA, TAG et TGA ne codent pas un acide aminé mais la fin de la protéine. Le codon ATG code la méthionine, mais aussi le début de la séquence protéique, qui commence systématiquement par cet acide aminé.

ADN	A T G	C C T	C T T	C G T	T C T	G T T	T G A
Proteine	Met	Pro	Leu	Arg	Ser	Val	STOP

FIGURE 2.4 – Exemple d'une séquence d'ADN traduite en protéine.

2.1.3 Chromosomes

Dans une cellule, on trouve un ou plusieurs *chromosomes*. Chacun d'entre eux est composé d'une seule molécule d'ADN. L'ensemble des chromosomes d'une cellule contient l'information génétique nécessaire au cycle de vie de l'organisme. Dans certains organismes, les chromosomes sont présents en deux exemplaires. On parle alors de *paires de chromosomes*. Les deux chromosomes d'une paire contiennent les mêmes gènes mais dans des versions différentes, sauf exception des chromosomes sexuels qui peuvent contenir des gènes différents, à l'image des chromosomes X et Y chez l'humain. Lorsque les chromosomes sont présents en un seul exemplaire, on dit que l'organisme est *haploïde*. S'ils sont présents en deux exemplaires, on dit que l'organisme est *diploïde*. Il est possible d'avoir d'autres ploïdies. On parle alors de *polyploïdie*. On peut citer comme exemple le rat-viscache roux d'Argentine qui est un rongeur naturellement tétraploïde [EB03], c'est-à-dire que tous ses chromosomes sont présents en quatre exemplaires, à l'exception de ses chromosomes sexuels.

La figure 2.5 montre un caryotype d'un être humain. Un caryotype est une façon de visualiser

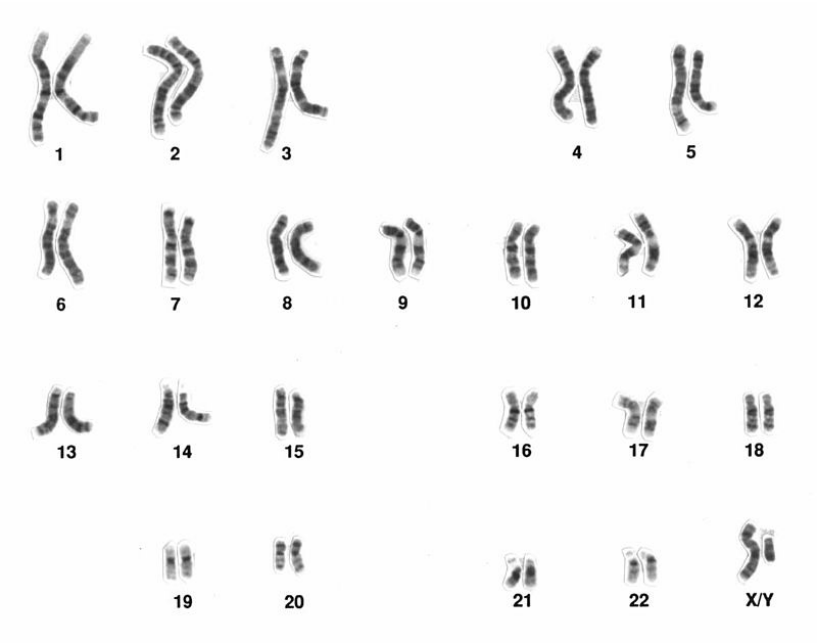


FIGURE 2.5 – Caryotype d'un individu de sexe masculin (XY) avec coloration au Giemsa. On observe des paires de chromosomes en métaphase. Source : https ://fr.wikipedia.org/wiki/Caryotype.

l'ensemble des chromosomes d'une cellule à partir d'une prise de vue microscopique. On peut voir sur cette figure les 46 chromosomes d'un être humain, divisés en 23 paires de chromosomes. L'être humain est un organisme diploïde. Les bactéries sont un exemple d'organisme haploïde. Elles possèdent un unique chromosome circulaire qui est présent en un seul exemplaire. Un autre exemple d'organisme haploïde est la levure *Saccharomyces cerevisiae* qui contient 16 chromosomes uniques.

2.1.4 Génomes

On parle de *génome* pour désigner l'intégralité de l'information génétique d'un organisme contenue dans ses chromosomes.

Tailles et gènes Le génome d'un organisme contient beaucoup d'informations et il n'est pas facile à analyser en raison de nombreux facteurs. Premièrement, la taille d'un génome varie énormément d'un organisme à l'autre. Par exemple, celui de la bactérie *Escherichia coli* fait 4,7 mégapaires de bases [Goo+18], celui du génome humain 3 gigapaires de bases [Nur+22a] et celui du blé fait 17 gigapaires de bases (ou le gui de plus de 90 gigapaires [Sch+22]). Le nombre de gènes est lui aussi variable, environ 4 200 gènes pour *Escherichia coli*, 20 000 gènes pour l'humain et 107 000 gènes pour le blé.

Une autre raison est que l'intégralité du génome n'est pas codante. On parle de *régions codantes* pour les sections du génome qui contiennent des informations portant un gène codant pour une protéine. Ces régions codantes peuvent être minoritaires : chez l'humain, on estime qu'environ 1,1% du génome (35 Mb) seulement est codant [Pio+19]. Cela ne signifie pas que les sections non codantes ne contiennent pas aussi des informations intéressantes. Elles peuvent par exemple contenir des gènes non-codants ou jouer un rôle dans la régulation de l'expression des gènes.

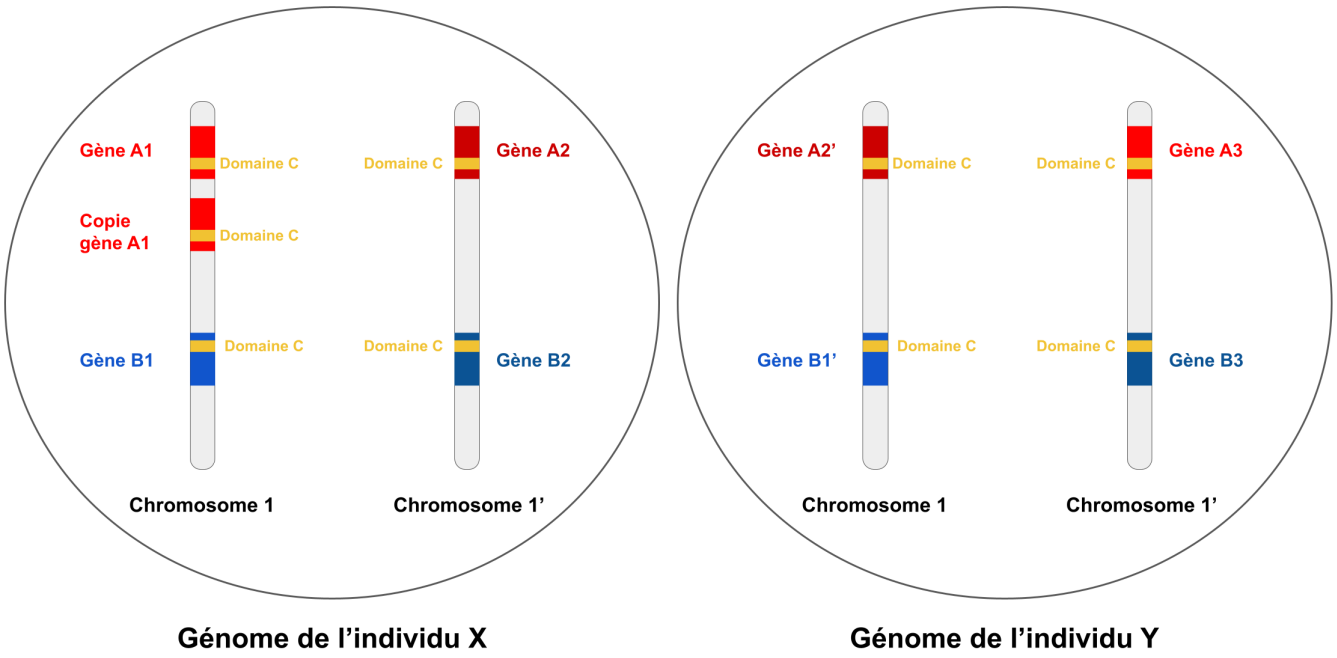


FIGURE 2.6 – Schéma d'une paire de chromosomes du génome de deux individus diploïdes.

Répétitions Pour ajouter encore de la complexité, les génomes contiennent souvent des *répétitions*. Chez l'humain, on estime qu'environ 75% du génome est constitué de répétitions [Lie21]. La figure 2.6 illustre les cas possibles de répétition. On peut trouver plusieurs copies d'un même gène au sein d'un même chromosome, comme c'est le cas du gène A1 chez l'individu X. Dans le cas diploïde, on a deux versions d'un même gène sur deux chromosomes d'une même paire, comme c'est le cas des gènes A1 et A2 chez l'individu X. Ces différentes versions sont assez similaires pour rendre complexe l'analyse, mais assez différentes pour avoir un intérêt biologique. De plus, beaucoup de gènes différents peuvent aussi avoir des séquences de nucléotides communes, comme c'est le cas de la séquence du domaine C que l'on retrouve à la fois dans le gène A et dans le gène B.

Génomes de référence et variants Lorsqu'on séquence une molécule d'ADN, on séquence toujours un individu en particulier. De ce fait, on accède au génome d'un individu d'une espèce et non au génome de toute l'espèce dans sa globalité. Chaque génome séquencé contient donc des variations

qui sont propres à l'individu. On parle de *génome de référence* pour désigner un génome séquencé de bonne qualité.

S'il existe différentes versions d'un gène au sein d'un individu d'une espèce (exemple du gène A1 et A2 chez l'individu X), il existe aussi bien sûr des versions différentes de ces mêmes gènes entre différents individus d'une même espèce. Sur la figure 2.6, on observe que l'individu Y est porteur du gène A2', qui est une variation légère du gène A2, mais aussi du gène A3, une autre variation du gène A. On parle de *variant* pour désigner des sections du génome qui diffèrent entre un individu et un génome de référence.

On parle aussi de variants pour désigner des variations intra-individu dans les cas polyploïdes. Aujourd'hui, il est encore difficile de séparer les différents haplotypes d'un individu [Wan+23]. Les génomes de référence contiennent donc des variants de plusieurs haploïdes dans le cas polyploïde.

On définit trois types de *variant ponctuel* : les *substitutions*, les *délétions* et les *insertions*, représentées en figure 2.7 :

— Les substitutions sont des variants où un ou plusieurs nucléotides sont remplacés par d'autres nucléotides, en nombre égal. La longueur de la séquence ne varie pas, ce qui rend plus facile la comparaison entre deux séquences.
— Les insertions sont des variants où un ou plusieurs nucléotides sont ajoutés par rapport au génome de référence. La longueur de la séquence varie, ce qui rend plus difficile la comparaison entre deux séquences à cause du décalage.
— Les délétions sont des variants où un ou plusieurs nucléotides sont supprimés par rapport au génome de référence. Comme pour l'insertion, la longueur de la séquence change.

FIGURE 2.7 – Schéma des variants ponctuels existants : substitution, délétion et insertion.

Il existe aussi ce qu'on appelle des *variants structuraux* qui modifient grandement la structure des chromosomes d'un individu à l'autre. Ces variants sont dus à des réarrangements génomiques. On peut les définir en trois catégories :

— Les *duplications* sont des portions de séquences de l'ADN qui ont été dupliquées sur le même

chromosome ou sur un autre chromosome. Quand cela concerne un gène entier, on parle alors de copie du gène. Cela peut aussi concerner juste un domaine mais aussi des portions non codantes. Ce phénomène est extrêmement courant et responsable des répétitions dans le génome.

— Les *inversions* sont des séquences d'ADN qui correspondent à l'autre brin. Dans la section 2.1, l'ADN est présenté comme étant composé de deux brins. Cela correspond donc à un échange de l'information entre ces deux brins.

— Les *translocations* sont des sections de l'ADN qui se sont déplacées par rapport à la référence. Elles peuvent avoir lieu sur le même chromosome ou sur un autre chromosome.

Je reviendrai sur les variants structuraux et leur identification en section 2.3.3.

2.2 Informatique et ADN

Au fil de l'évolution des sciences du vivant, la quantité de données à traiter et à analyser n'a cessé de croitre, notamment en ce qui concerne les données d'ADN disponibles [Kat+21]. L'informatique a permis d'automatiser l'analyse de ces données.

2.2.1 Problématique

Je donne ici quelques exemples qui illustrent le type d'analyse à mettre en œuvre pour exploiter des séquences d'ADN.

En premier lieu, pour extraire l'ADN, une cellule doit être broyée, ce qui engendre des cassures et donne des fragments d'ADN. Il est de ce fait impossible de lire un génome dans sa globalité. De plus, les méthodes de séquençage (voir section 2.3, page 33) ne permettent pas de lire un fragment d'ADN dans sa totalité. Il est donc nécessaire de passer par une étape d'*assemblage* de ces séquences d'ADN pour reconstituer un génome complet [Ban+12]. Aujourd'hui, les données de séquençage sont telles qu'un ordinateur de bureau standard ne suffit pas pour assembler un génome complexe.

Ensuite, une fois la séquence d'ADN assemblée, il faut réussir à identifier les gènes présents dans ces séquences [See14]. Bien que des motifs permettent d'identifier le début et la fin d'un gène, trouver ces motifs dans la séquence revient à chercher une aiguille dans une botte de foin. À titre d'exemple, l'ADN humain contient environ 3,2 milliards de paires de nucléotides. Le motif permettant de trouver le début d'un gène ne comporte que 3 nucléotides. L'ADN humain contenant environ 20 000 gènes,

ce motif représente 0,000019% du génome humain et sans compter ceux qui sont présents mais non fonctionnels.

Une fois obtenus les génomes et l'emplacement des gènes dans ces derniers, on détecte les variants entre les différents génomes, car ceux-ci sont essentiels pour comprendre le vivant. Par exemple, si on détecte une substitution dans le gène A chez un individu atteint d'une maladie, le gène A devient un candidat pour expliquer cette maladie [Wel+13]. Or, la comparaison entre deux génomes qui peuvent comporter des milliards de bases avec des décalages entre les nucléotides dus aux insertions et aux délétions n'est pas une tâche simple d'un point de vue humain.

2.2.2 Bioinformatique

Face à ces problématiques et bien d'autres, il est devenu évident que l'informatique pourrait jouer un grand rôle dans l'analyse des données biologiques en permettant l'automatisation de nombreuses tâches. On a appelé *bioinformatique* la discipline permettant de répondre à ces problèmes scientifiques concernant la biologie en utilisant des compétences informatiques et statistiques. Cette discipline se situe au carrefour de plusieurs domaines, telles que la biologie, l'informatique, les mathématiques, la physique et la médecine.

2.2.3 Analyse de séquences

Bien qu'il existe de nombreux autres domaines d'application, le sous-domaine de l'analyse de séquences d'ADN est l'un des plus anciens et des plus développés en bioinformatique [Ros59]. Pour traiter l'ADN informatiquement, une séquence d'ADN est considérée comme un mot utilisant un alphabet de quatre lettres $\{A, T, G, C\}$. Ainsi, on peut chercher des sous-mots spécifiques dans l'ADN en le traitant comme un simple texte et extraire des informations de ces séquences. Cela permet également de stocker les séquences dans des fichiers. Le format le plus couramment utilisé est le format FASTA, qui permet de stocker plusieurs séquences. Le symbole ">" sert à identifier le nom de la séquence. Un exemple de fichier FASTA est disponible sur la figure 2.8.

2.3 Séquençage

Le *séquençage* permet d'obtenir la succession des nucléotides d'une séquence d'ADN. On l'utilise pour passer d'une molécule biologique à une séquence composée des caractères A, T, G et C, qui

```
>ERR4352154.1
GCGCAACCGAACCACCCCCGACAAACAATGAGTCGTCCTAGGACTTCAGGAATTC
CTCGGGGATGCAATCCAATTCTTCCACAGTGGGTGAGAAAACCATTTTCCCGAAG
AGTTTGCGGCTAGAAGTGGTGAGAAAAAGCAGCACTTGTGATATATACGACAAAA
>ERR4352154.2
CTCGGGGATGCAATCCAATTCTTCCACAGTGGGTGAGAAAACCATTTTCCCGAAG
AGTTTGCGGCTAGAAGTGGTGAGAAAAAGCAGCACTTGTGATATATACGACAAAA
GCGCAACCGAACCACCCCCGACAAACAATGAGTCGTCCTAGGACTTCAGGAATTC
```

FIGURE 2.8 – Exemple de deux séquences d'ADN au format FASTA.

permettra de décortiquer l'information génétique d'un organisme. Plusieurs méthodes de séquençage existent, chacune produisant différents types de données et ayant diverses applications. Je les détaille ci-dessous.

2.3.1 Méthodes de séquençage

Les méthodes de séquençage sont utilisées pour déterminer l'ordre dans lequel les nucléotides sont agencés. Elles génèrent des fragments de taille variable du génome, appelés *reads* en anglais et *lectures* en français. Dans ce texte, j'utiliserai le terme anglais. Toutefois, le séquençage n'est pas parfait et les reads présentent un certain taux d'erreur. Pour pallier ce problème, on séquence plusieurs fois le même fragment. La *profondeur de séquençage* désigne ainsi le nombre de reads couvrant un nucléotide. Par exemple, une profondeur de séquençage de 60x signifie qu'en moyenne chaque nucléotide a été séquencé 60 fois, permettant ainsi de corriger les erreurs de séquençage.

Générations de séquençage et séquençage à haut-débit Les méthodes de séquençage sont classées en trois générations. La méthode de Sanger [SNC77], issue de la première génération, est basée sur la réplication de l'ADN. Bien qu'elle soit l'une des plus anciennes, cette méthode est toujours utilisée aujourd'hui en raison de sa grande fiabilité. Cependant, elle est coûteuse et avec un faible débit. Avec l'évolution des sciences du vivant, le besoin d'obtenir davantage d'informations à partir des échantillons d'ADN s'est accru. Les méthodes de première génération n'étant pas adaptées, les méthodes de séquençage de deuxième et troisième générations, appelées *séquençage à haut débit* ou Next Generation Sequencing (NGS), ont été développées à partir des années 2000. Elles sont moins coûteuses en termes de prix et permettent d'obtenir un plus grand nombre de nucléotides à partir d'un échantillon d'ADN [Sch08].

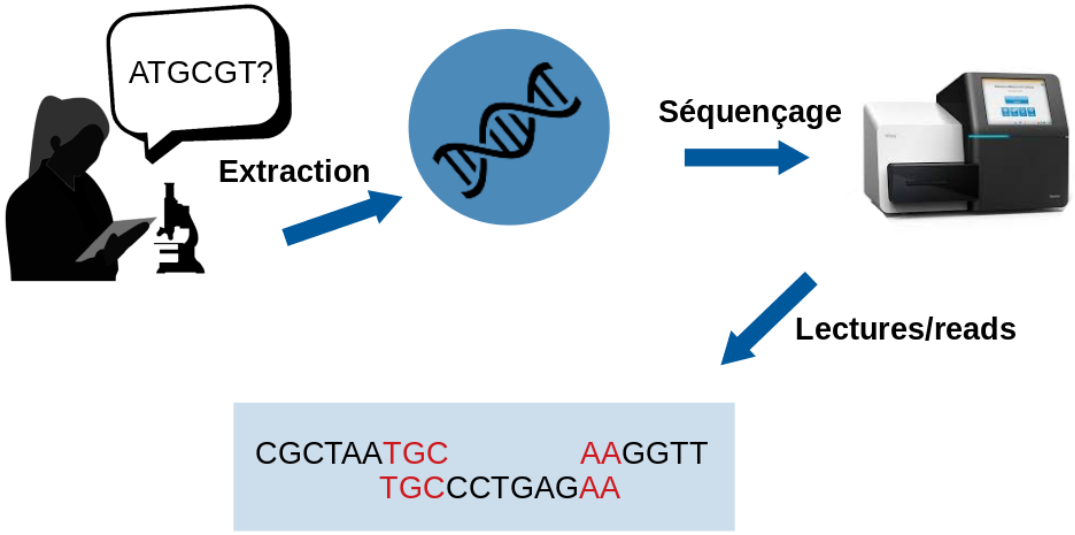


FIGURE 2.9 – Déroulement d'un séquençage.

Séquençage de deuxième génération : l'exemple d'Illumina Les séquenceurs majoritairement utilisés aujourd'hui sont dits de *deuxième génération*, notamment ceux de la marque Illumina, qui sont actuellement les plus courants. Ces méthodes utilisent d'abord un processus appelé PCR (Polymerase Chain Reaction) [Zhu+20] pour amplifier les fragments d'ADN. Ensuite, on réalise une synthèse de ces fragments, un processus qui permet d'obtenir une copie du fragment. Au cours de la synthèse, on détermine quels nucléotides sont incorporés via différentes techniques.

Ces méthodes sont capables de produire plusieurs millions de reads d'une longueur de quelques centaines de nucléotides (environ 150 bases), appelés *lectures courtes* ou *short reads*. En moyenne, on obtient des profondeurs de séquençage de 60x avec un faible taux d'erreur (<1%), les erreurs étant presque exclusivement des erreurs de substitution [SN21].

L'amplification par PCR induit également un biais au niveau des bases GC, c'est-à-dire qu'elle a tendance à amplifier davantage les sections du génome qui sont riches en nucléotides GC que les autres [Sat+19]. On risque donc d'avoir une profondeur de séquençage non homogène sur l'ensemble du génome.

Séquençage de troisième génération : Nanopore et PacBio De nouveaux séquenceurs utilisant des méthodes de séquençage de *troisième génération* sont disponibles sur le marché depuis quelques années. Les plus utilisés sont les séquenceurs proposés par PacBio [Eid+09] et Oxford Nanopore Technologies (ONT) [Cla+09].

Dans le cas d'ONT, les fragments d'ADN passent à travers un nanopore, un trou de 1 à 100 nm de

diamètre qui lisent les nucléotides lors de leur passage à travers le pore. Dans le cas de Pacbio, on transforme les fragments en ADN circulaire avant d'en réaliser une synthèse qui se fera en boucle grâce à ce procédé.

Ces techniques ne nécessitent pas d'étapes d'amplification par PCR, ce qui les rend plus rapides et moins coûteuses que la deuxième génération. Elles permettent également de produire des lectures de séquençage plus longues, appelées *longues lectures* ou *long reads* qui sont justement le sujet principale de cette thèse.

2.3.2 Caractéristiques des long reads

Tailles. Les long reads sont appelés ainsi car ils mesurent quelques milliers de bases ou dizaines de milliers de bases, contrairement aux short reads qui font environ 150pb.Il existe aussi des ultra-long reads qui peuvent atteindre jusqu'à 100 000 bases, voire même dépasser le million. Disposer de long reads ou ultra-long reads est particulièrement intéressant car ils permettent de couvrir plusieurs régions du génome [Jai+18].

Taux et types d'erreurs. En contrepartie de leur longueur accrue, les long reads présentent un taux d'erreur important, entre 5 et 15% [WJH19], ce qui les rend plus difficile à exploiter. De plus, ces erreurs peuvent être des insertions, des délétions ou encore des substitutions. Cela complique leur traitement, car lors d'une insertion ou d'une délétion, un décalage se crée dans le cadre de lecture des nucléotides. Or, le cadre de lecture est essentiel pour comprendre et comparer les séquences.

Un autre problème concerne les reads chimériques, qui sont des reads composés de plusieurs fragments d'ADN qui ne se suivent pas dans le génome. Ils sont dus à un problème lors du passage successif de plusieurs reads dans le nanopore. Cependant, les reads chimériques peuvent être facilement identifiés.

Contrairement à la deuxième génération, il n'y a pas de problème d'amplification des bases GC dû à l'absence d'étape d'amplification par PCR.

2.3.3 Utilisations des long reads

Les caractéristiques des long reads permettent de résoudre certains problèmes auxquels les données précédentes, avec les short reads, ne pouvaient pas s'attaquer. Leurs utilisations sont nombreuses.

Assemblage de génomes. J'ai expliqué en section 2.2 qu'une fois le séquençage réalisé, la première étape vers l'obtention du génome était celle de l'assemblage : on exploite le fait que les reads se chevauchent pour reconstituer les séquences d'ADN. Ces séquences reconstruites à partir de plusieurs reads sont appelées des *contigs*. Ils peuvent ne pas couvrir l'intégralité du génome, car certaines régions sont plus complexes à résoudre que d'autres.

L'assemblage est une étape délicate qui peut induire des erreurs, notamment si les reads eux-mêmes comportent des erreurs. La taille des reads peut également jouer un rôle important sur la qualité de l'assemblage, d'où l'utilité des long reads. Le génome comportant de nombreuses répétitions (voir section 2.1.4), les short reads peuvent induire des erreurs d'assemblage si elles ne couvrent pas l'intégralité des zones répétées, comme l'illustre la figure 2.10. Les long reads peuvent résoudre les zones difficiles à assembler même s'il ne faut pas négliger leur taux d'erreur plus élevé. Aujourd'hui, on utilise souvent des short reads et des long reads pour bénéficier des atouts des deux méthodes [Mai+19 ; Wic+17b].

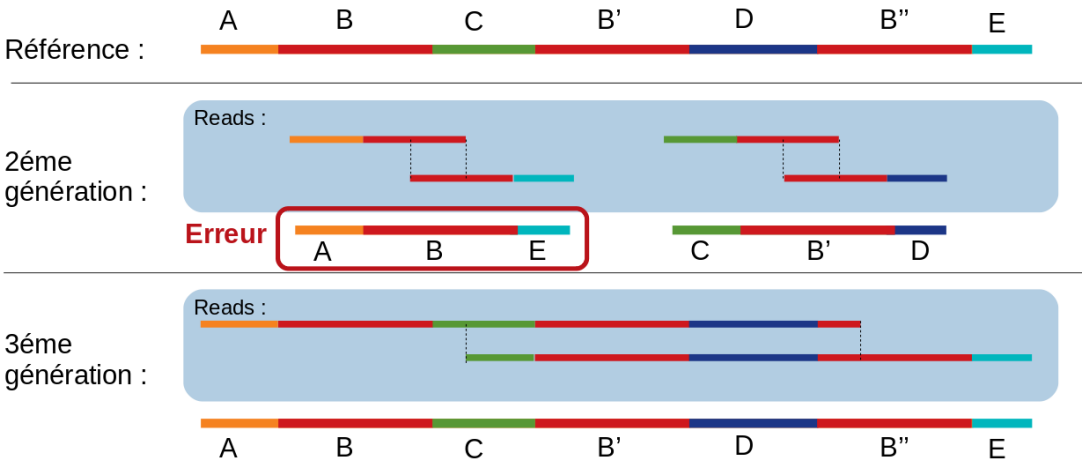


FIGURE 2.10 – Assemblage short reads et assemblage long reads. Les régions B, B' et B" ont des séquences identiques, car c'est une région répétée. Les reads de deuxième génération ne couvrant pas toute la région B, l'assemblage est compliqué et on peut obtenir un mauvais contig, comme avec l'assemblage des régions A, B et E. Les reads de troisième génération couvrant ces régions, le problème ne se pose pas.

Réarrangements génomiques. Le réarrangement génomique est un processus qui modifie l'ordre des régions d'ADN dans le génome d'un organisme. Ces réarrangements peuvent être causés par plusieurs mécanismes. Une section du génome peut être dupliquée ou déplacée sur le même chromosome, voire même sur un autre chromosome. On parle alors de variants structuraux (section 2.1.4).

Ces réarrangements génomiques peuvent avoir des conséquences importantes sur la structure et la fonction du génome, notamment en modifiant la régulation de l'expression des gènes, en créant de

nouvelles combinaisons de gènes ou en générant des variantes génétiques causant des maladies [CL16].

Pour les identifier, on aligne les reads issus d'un organisme sur une référence [HV19 ; Lec+20]. Les long reads permettent d'identifier plus facilement certains réarrangements, que les short reads. Dans certains cas, ils sont même les seuls qui permettent de les identifier. Les figures 2.11 et 2.12 permettent de visualiser ces cas de figure.

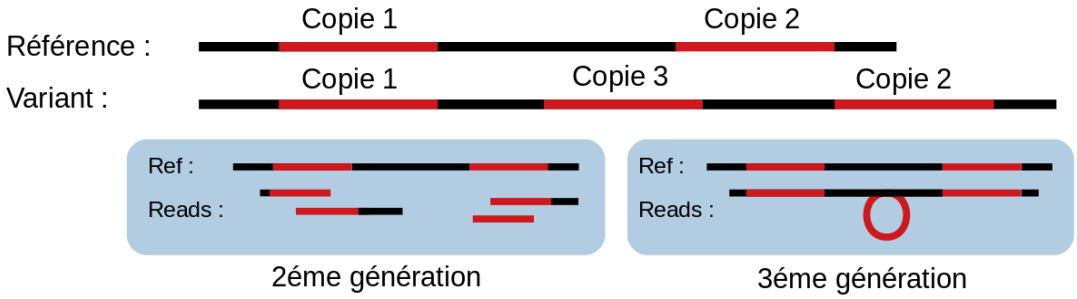


FIGURE 2.11 – Réarrangement génomique avec copie supplémentaire. Le génome variant contient une copie supplémentaire de la région en rouge. Dans le cas des short reads, il est plus difficile d'identifier cette troisième copie car les reads ne couvrent pas forcément toute la région répétée. Dans le cas des long reads, il est plus facile d'identifier ce réarrangement.

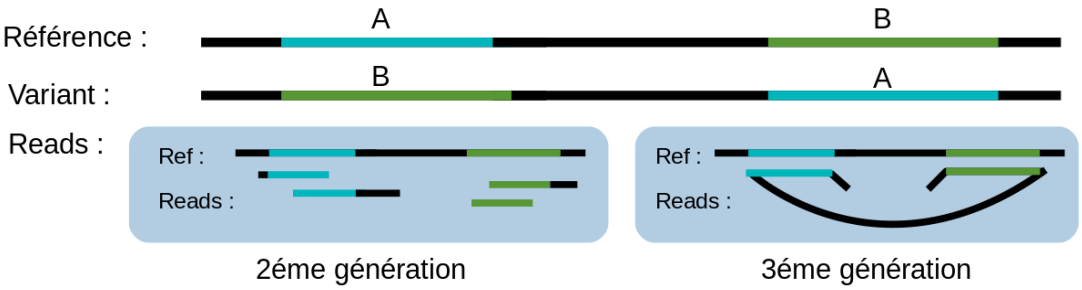


FIGURE 2.12 – Réarrangement génomique avec inversion structurale. Le génome variant contient une inversion entre la région A et la région B. Dans le cas des short reads, il est plus difficile d'identifier cette inversion en raison de sa longueur. Dans le cas des long reads, il est plus facile d'identifier ce réarrangement.

Phasage diploïde. Un organisme diploïde contient son génome en deux exemplaires (voir section 2.1.3). Lors du séquençage, il n'est pas possible de séparer les deux versions des chromosomes, que l'on nomme des haplotypes. Les reads obtenus proviennent donc d'un mélange de ces deux haplotypes. On peut identifier les variants ponctuels entre les deux haplotypes [DeP+11], mais il est souvent difficile d'associer les variants entre eux, c'est-à-dire de les lier à un haplotype en particulier. Comme l'illustre la figure 2.13, on observe une variation entre un G et un C et une autre variation entre un A et un T. Il est cependant plus difficile de savoir si le premier haplotype contient G et A ou G et T. Dans le cas des long reads, c'est plus facile à identifier car on peut avoir des reads qui contiennent plusieurs variations et permettent de phaser les variants [Gar+18].

Ce problème est décuplé dans les cas de polyploïdie.

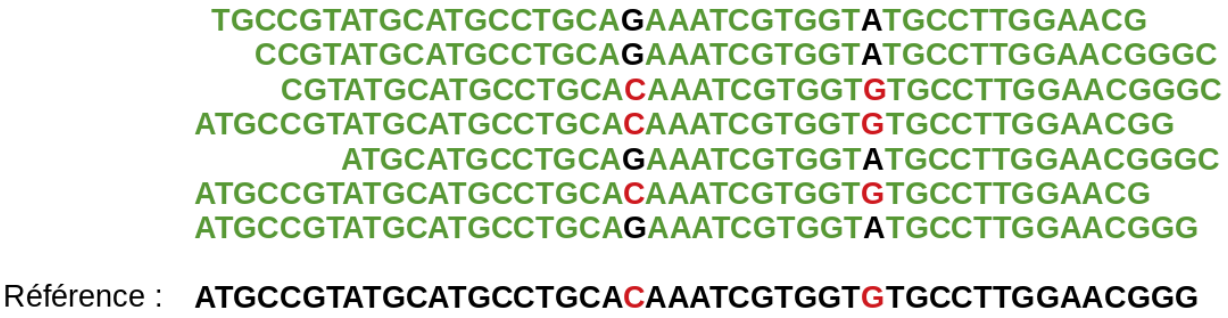


FIGURE 2.13 – Reads diploïdes alignés sur une référence haploïde. On observe en vert les nucléotides identiques à la référence, en rouge les variants qui correspondent à la référence et en noir les variants portés par l'autre chromosome.

Transcriptomique. Lorsqu'un gène s'exprime, la séquence d'ADN qui correspond à ce gène est transcrit en ARN (acide ribonucléique). L'ARN est composé d'une succession de nucléotides, comme l'ADN, mais diffère par sa structure et son rôle dans la cellule. Contrairement à l'ADN, il ne comporte qu'un seul brin, des sucres ribose à la place des sucres désoxyribose, et est beaucoup plus petit que l'ADN en nombre de bases. Il contient une copie d'une partie de l'information contenue dans l'ADN qui sera ensuite utilisée pour synthétiser une protéine à l'aide du code génétique (cf. section 2.1).

Si l'ADN contient tous les gènes qui peuvent s'exprimer au cours de la vie d'un organisme, l'ARN contient les gènes qui s'expriment à un instant t de la vie d'un organisme ou d'une cellule. Il est donc très intéressant d'un point de vue biologique d'étudier l'ARN pour comprendre l'expression des gènes. En réalité, les fonctions de l'ARN sont beaucoup plus vastes que la simple expression des gènes, mais elles ne seront pas abordées ici.

On appelle *transcriptomique* la branche qui étudie l'ARN. Comme l'ADN, l'ARN peut être analysé grâce au séquençage. Cela se fait via une étape de rétrotranscription, qui convertit l'ARN en séquence d'ADN, ou en séquençant la molécule d'ARN directement. On obtient ainsi l'information contenue dans l'ARN. Cependant, comme pour le séquençage d'un génome, la taille des reads est limitée et dépend de la génération de séquençage.

Les long reads ont ici un avantage par rapport aux shorts reads, car ils permettent de couvrir l'intégralité, ou en tout cas une grande portion, de la molécule d'ARN. Cela permet par exemple d'étudier plus facilement l'*épissage* de l'ARN. Les gènes sont composés de sections codantes, appelées exons, et de sections non codantes, appelées introns. L'épissage est une étape importante de la régulation de l'expression génique. Il permet à une cellule de produire différents types d'ARN à

partir d'un seul gène, en variant le choix des exons et des introns. Certaines séquences ne peuvent être identifiées qu'après un épissage. Les long reads permettent d'identifier plus facilement que les short reads les successions des introns et des exons, ainsi que les différentes variations d'ARN qui peuvent provenir d'un même gène [Gli+22]. Ils peuvent aussi permettre de détecter des variants transcrits, des variants issus de l'ADN qui seront présents dans l'ARN et impacteront les protéines issues de ces ARN.

Chapitre 3

Alignement de Séquences d'ADN

L'alignement de séquences est primordial dans le domaine de la bioinformatique, car c'est une méthode qui permet de comparer des séquences d'ADN. Cela permet de faire des recoupements entre des séquences et de simplifier leurs analyses. Une définition formelle de l'alignement est donnée en section 3.1. On utilise des méthodes d'alignement par paire, présentées en section 3.2, qui permettent d'aligner les séquences deux par deux. Il est aussi possible d'aligner un nombre plus élevé de séquences en même temps. On parlera alors d'*alignement multiple*. Ces alignements nécessitent des méthodes plus complexes et sont détaillés dans la section 3.3. Pour clore ce chapitre, la section 3.4 aborde plus précisément la problématique des long reads et de l'alignement multiple appliqué aux longs reads, puisque c'est le cœur de cette thèse.

3.1 Définition de l'Alignement de Séquences

L'alignement de séquences est une stratégie efficace pour la comparaison de séquences d'ADN. Il permet d'identifier et de mettre en évidence les régions similaires et les régions divergentes. C'est donc un outil de base en bioinformatique [Alt+90a]. Différentes méthodes sont employées pour réaliser des alignements, toutes visant à maximiser un score qui dépend du type d'alignement effectué. Un alignement est dit *optimal* s'il possède le score le plus élevé parmi tous les alignements possibles pour les séquences considérées.

L'alignement de séquences est habituellement représenté par un tableau. Chaque ligne correspond à une séquence spécifique, tandis que chaque colonne représente une position donnée dans l'alignement. Les cellules de ce tableau peuvent contenir soit un nucléotide, soit un "trou", aussi connu sous le nom

de *gap*, comme illustré dans la Figure 3.1. Lorsque les nucléotides dans une colonne donnée sont identiques, cela constitue un *match*. En revanche, s'ils divergent, il s'agit d'un *mismatch*. Cette correspondance entre les séquences permet d'identifier les similitudes et les divergences entre elles.

Sequence 1	A	-	-	-	A	T	G	C	A	T	C	G	A	T	C	G	A	T
Sequence 2	A	T	G	A	A	T	G	T	A	T	C	G	-	T	C	C	A	T

FIGURE 3.1 – Alignement de deux séquences avec en vert les nucléotides identiques appelés *matchs*, en rouge les nucléotides différents appelés *mismatchs* et en noir les colonnes où il manque des nucléotides à cause des *insertions* ou des *délétions*, appelées *gaps*.

3.1.1 Justification de l'Alignement de Séquences

Depuis le début du séquençage et la génération des premières séquences, l'alignement de séquences est une étape cruciale dans l'analyse, car cela permet de quantifier la ressemblance entre différentes séquences. On parle d'*homologie* pour désigner des attributs communs à deux espèces et hérité d'un même ancêtre. Lorsque deux séquences sont identifiées comme homologues grâce à l'alignement de séquences qui permet de mettre en lumière leurs ressemblances, cela facilite l'analyse puisque qu'on peut réaliser des recoupements d'information entre elles [Kim80].

En génomique, l'alignement de séquences peut servir à identifier des gènes homologues. Par exemple, si un gène A inconnu présente une grande similitude avec un gène B connu, on peut déduire que le gène A et B sont homologues et que A joue probablement le même rôle que B dans l'organisme. De même, en identifiant une région dont la séquence d'ADN est très conservée entre deux gènes homologues, on peut en déduire que cette région subit une pression de sélection et joue sûrement une importance majeure dans le rôle que partagent ces deux gènes.

L'alignement de séquences est également crucial en biologie structurale des protéines. Si la structure d'une protéine X issue du gène A est connue et qu'une grande similitude est observée avec le gène B grâce à l'alignement de séquences, on peut prédire que la protéine issue du gène B a une structure similaire à celle de la protéine X.

En phylogénie, l'alignement de séquences peut servir à étudier l'évolution et la parenté génomique des espèces en comparant des séquences provenant de différents organismes [Edg22].

3.1.2 Méthodologies d'Alignement et Score

Un large panel de méthodes existe pour effectuer l'alignement de séquences, avec le même objectif : obtenir un alignement de qualité, voire optimal. La détermination de la qualité d'un alignement est basée sur une *fonction de score*, qui évalue l'alignement. Le système de score est un élément crucial à considérer, car ces méthodes visent à obtenir un score élevé.

De manière basique, le score d'un alignement est défini comme la somme des scores de toutes les paires de bases alignées. On reprend l'alignement de la figure 3.1 sur la figure 3.2. Pour obtenir le score de cet alignement, on additionne les scores de chaque colonne qui correspond à une paire de base.

Sequence 1	A	-	-	-	A	T	G	C	A	T	C	G	A	T	C	G	A	T
Sequence 2	A	T	G	A	A	T	G	T	A	T	C	G	-	T	C	C	A	T
Final score : 8	+1	-1	-1	-1	+1	+1	+1	+0	+1	+1	+1	+1	-1	+1	+1	+0	+1	+1

FIGURE 3.2 – Score d'un alignement de deux séquences. Dans cet exemple, les gaps valent -1, les matchs +1 et les mismatchs 0. En additionnant le score individuel de chaque colonne, on obtient le score final, 8. Dans ce système de score, deux séquences parfaitement identiques de longueur n obtiennent un alignement de score n.

Matrices de substitutions. Pour déterminer le score d'une paire de bases, on utilise des *matrices de substitution* [Alt+05] qui attribuent un score à chaque substitution possible entre deux nucléotides, ainsi que pour les identités. Sur des séquences d'ADN, on utilise deux types de matrices : les matrices identités et les matrices transition-transversions. Une matrice identité part du principe que toutes les substitutions sont équiprobables et elle leurs attribue le même score. C'est par exemple le cas dans l'exemple sur la figure 3.2 où toutes les substitutions valent 0 et les identités 1. La matrice transition-transversion favorise les substitutions entre deux purines ou deux pyrimidines. Par exemple, il est plus fréquent de voir une adénine (A) remplacée par une guanine (G) que par une cytosine (C). Dans ce cas, la paire A-G obtient un meilleur score que la paire A-C.

En pratique, ce sont les matrices identités qui sont les plus utilisées. La matrice DNAfull est l'une des plus courantes. Elle fixe les mismatchs à -4 et les matchs à +5.

Pénalités de gap. Les gaps sont également pris en compte dans le calcul du score [Alt89]. Généralement, on cherche à minimiser le nombre de gaps dans un alignement. Par conséquent, l'ajout d'un gap entraîne une pénalité forte dans le score. Contrairement à la figure 3.2, on considère rarement que le coût d'un gap est fixe car cela ne correspond pas à la réalité. En effet, les gaps isolés sont plus

rares et il est plus probable d'avoir un unique gap de longueur k que d'avoir k gaps de longueur 1. Pour traduire ce modèle, on préfère mettre une pénalité plus forte sur le premier gap, appelée *pénalité d'ouverture* g_{op}, et une pénalité moins forte sur les suivants, appelée *pénalité d'extension* g_{ext}. L'idée est de rendre l'ouverture d'un nouveau gap plus coûteux que d'étendre un gap déjà existant. La totalité du coût du gap de longueur k est une fonction affine en k.

$$g(k) = g_{op} + g_{ext} \times k$$

.

Sequence 1	A	-	-	-	A	T	G	C	A	T	C	G	A	T	C	G	A	T
Sequence 2	A	T	G	A	A	T	G	T	A	T	C	G	-	T	C	C	A	T
Score final : 24	+5	-8	-6	-6	+5	+5	+5	-4	+5	+5	+5	+5	-8	+5	+5	-4	+5	+5

FIGURE 3.3 – Score d'un alignement de deux séquences avec fonction de gap affine. Dans cet exemple, la pénalité d'ouverture de gap est de -8 et la pénalité d'extension de gap est de -6. Les matchs valent 5 et les mismatchs -4. On obtient le score final de 24.

Le modèle de matrice de substitution, le modèle pour les pénalités de gap ainsi que l'ensemble de leurs paramètres respectifs sont autant de facteurs qui influencent grandement le score d'un alignement. Les figures 3.2 et 3.3 montrent bien l'écart possible entre différents scores. Il faut donc choisir sa fonction de score de manière judicieuse.

3.2 Alignement de Séquences par Paires

L'alignement de séquences par paires, ou alignement deux à deux, permet de comparer deux séquences de nucléotides l'une par rapport à l'autre. C'est le cas le plus simple d'alignement de séquences, et le seul pour lequel il est possible en pratique de déterminer l'alignement optimal.

On parle d'alignement *global* lorsque l'intégralité des deux séquences est alignée, et d'alignement *local* lorsque seules certaines sous-séquences sont alignées.

Les méthodes d'alignement de séquences par paires peuvent être classées en deux catégories : les méthodes *exactes* et les méthodes *heuristiques*.

3.2.1 Méthodes exactes

Les méthodes exactes visent à obtenir le meilleur alignement possible entre deux séquences, c'est-à-dire l'alignement de score maximal. L'algorithme de Needleman-Wunsch [NW70] est la première méthode à avoir été proposée. Il permet de résoudre le problème de l'alignement global et est encore majoritairement utilisé aujourd'hui

Principe de l'algorithme Needleman-Wunsh. L'algorithme repose sur un paradigme bien connu, celui de la *programmation dynamique* qui consiste à résoudre un problème en le décomposant en plusieurs sous-problèmes plus simples et moins coûteux à résoudre. On trouve une solution optimale à partir de ces sous-problèmes en les combinant ensemble.

Dans le cadre de cet algorithme, on décompose l'alignement de deux séquences en un alignement de sous-séquences, qui sont des préfixes des séquences initiales. Cela donne une matrice (figure 3.4) où l'on stocke le score optimal de ces sous-problèmes dans les cases. Le score optimal de l'alignement est trouvé dans la dernière case calculée, celle en bas à droite. L'alignement correspondant est retracé en remontant ensuite la matrice pour trouver un chemin qui réalise ce score. Les matchs et les mismatchs correspondent à des déplacements en diagonale. Se déplacer horizontalement ou verticalement dans la matrice implique l'ajout d'un gap dans l'alignement. La figure 3.4 représente une matrice et l'alignement qui en découle.

Pour résoudre chaque sous-problème, on utilise la formule suivante :

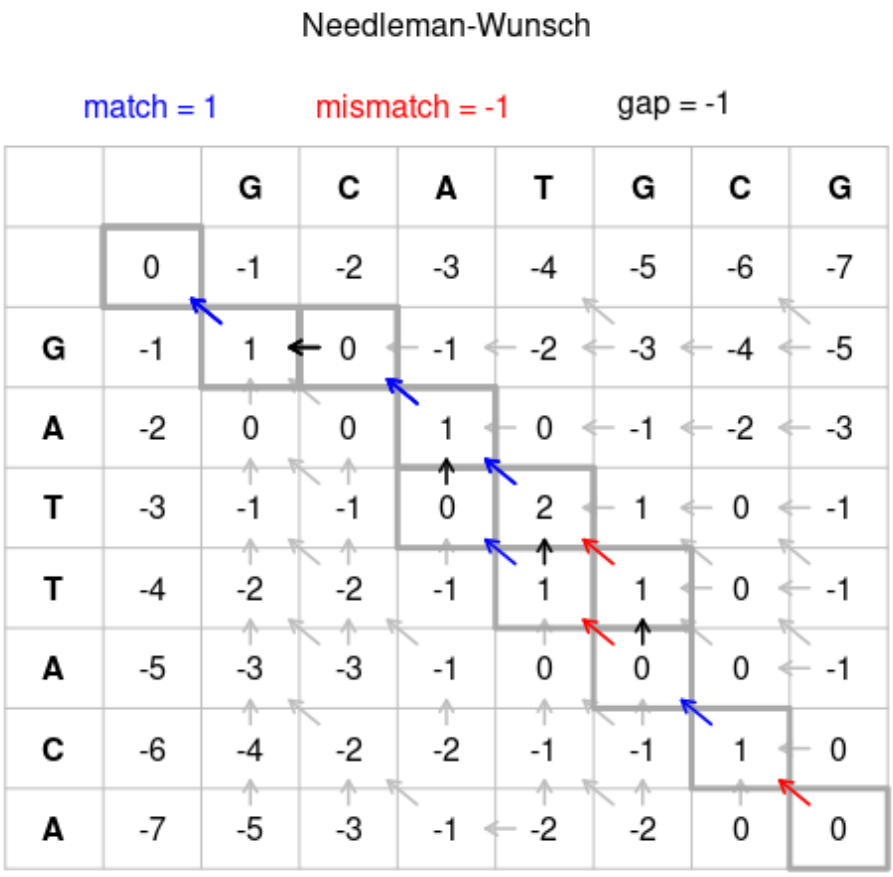
$$M_{i,j} = max \begin{cases} M_{i-1,j} + S(-, b_j) & \uparrow \\ M_{i-1,j-1} + S(a_i, b_j) & \nwarrow \\ M_{i,j-1} + S(a_i, -) & \longleftarrow \end{cases}$$

où $M_{i,j}$ représente la case (i, j) de la matrice, S la fonction de score et a_i et b_j les nucléotides des positions i et j de la première et de la seconde séquence respectivement.

Une autre façon d'écrire cette formule est la suivante :

$$M_{i,j} = max \begin{cases} M_{i-1,j} + gap & \uparrow \\ M_{i-1,j-1} + (match\ or\ mismatch) & \nwarrow \\ M_{i,j-1} + gap & \longleftarrow \end{cases}$$

La figure 3.5 est un exemple de remplissage de cette matrice. On observe donc que chaque case



Sequence 1	G	C	A	T	-	G	C	G
Sequence 2	G	-	A	T	T	A	C	A
Valeurs intermédiaires	1	0	1	2	1	0	1	0
Score d'alignement : 0	1	-1	1	1	-1	-1	1	-1

FIGURE 3.4 – Matrice de programmation dynamique pour l'alignement par l'algorithme de Needleman-Wunsch (Source : `https://en.m.wikipedia.org/wiki/Needleman-Wunsch_algorithm`). Le système de score est de 1 pour un match et -1 pour un mismatch ou un gap. Le tableau en dessous représente un alignement optimal issu de cette matrice, les valeurs intermédiaires pour les couples de préfixes lus dans la matrice et le score de chaque opération d'édition (match en rouge, mismatch en vert et gap en noir) constituant cet alignement. Le score total est égal à 0, correspondant à la valeur de la case en bas à droite de la matrice.

dépend de trois cases adjacentes. Les trois chemins possibles sont calculés et on conserve le meilleur des trois, au sens du score, pour chaque case.

Les pénalités de gap utilisées jusqu'ici étaient un cas simple où la pénalité était constante. Il est possible d'utiliser aussi la fonction affine définie dans la section 3.1.2 avec cet algorithme. Dans ce cas, l'algorithme est modifié pour utiliser plusieurs matrices, au lieu d'une.

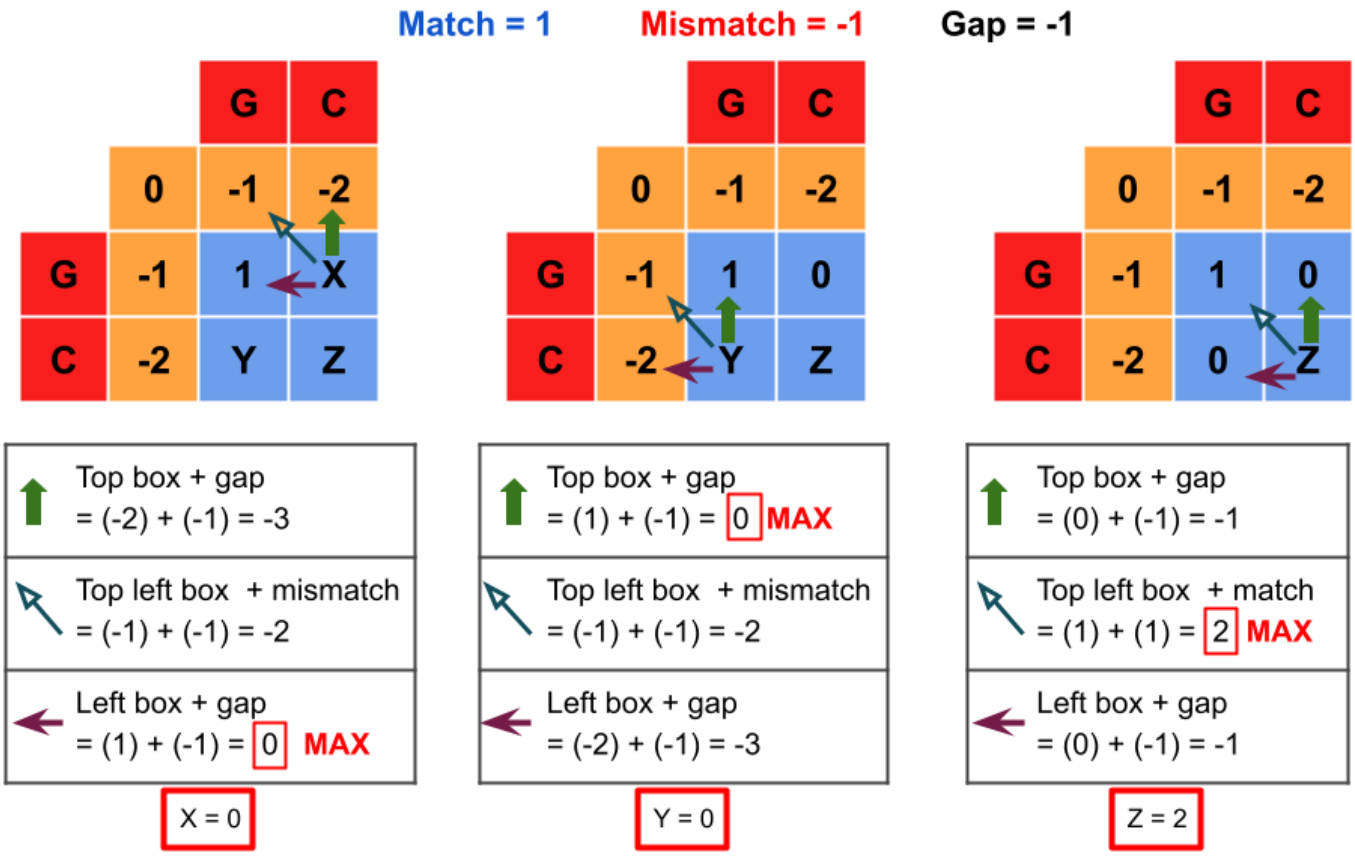


FIGURE 3.5 – Exemple du remplissage d'une matrice d'alignement Needleman-Wunsch. Les matchs valent 1, les mismatchs et les gaps -1, comme dans la figure 3.4.

Résolutions des sous-problèmes.

L'algorithme de Smith-Waterman. L'algorithme de Smith-Waterman pour l'alignement local [SW81] découle de celui de Needleman-Wunsch. Les deux principales différences sont que toutes les valeurs négatives dans la matrice sont remplacées par des zéros et la création de l'alignement s'arrête lorsqu'on rencontre un zéro. Ces deux règles permettent de faire ressortir les alignements locaux, et non plus les alignements globaux. Ces alignements sont utilisés pour plusieurs raisons notamment l'identification de régions similaires au sein des séquences.

Complexité. La complexité est une manière de comparer le coût de deux algorithmes. Il en existe deux types : la complexité en temps et la complexité en mémoire. Cela consiste généralement à compter le nombre d'opérations élémentaires (addition, soustraction, multiplication, affectation etc.) effectuées par un algorithme. On ne donne presque jamais la complexité exacte, mais plutôt un ordre de grandeur, noté $\mathcal{O}$.

Les deux algorithmes d'alignement global et local ont une complexité en termes de temps et de mémoire de $\mathcal{O}(l_1 \times l_2)$, où l_1 et l_2 sont les longueurs des deux séquences à aligner. Même si cela peut sembler peu coûteux en première analyse, il est extrêmement courant de vouloir aligner des séquences de plusieurs milliers, dizaines ou centaines milliers de bases. Dans de tels cas, une telle complexité n'est pas envisageable. Pour pallier ce problème, d'autres méthodes ont été développées, notamment les méthodes heuristiques qui permettent de réduire le coût en temps et en mémoire de la programmation dynamique.

3.2.2 Méthodes Heuristiques

En complément des algorithmes de Needleman-Wunsch et Smith-Waterman, il existe des méthodes heuristiques qui offrent des solutions approximatives mais sont nettement plus rapides. Ces méthodes sont utilisées pour rechercher des séquences similaires dans de grandes bases de données.

Ces algorithmes heuristiques visent à réduire le coût de calcul en effectuant l'alignement sur une région plus petite de la séquence, au lieu de l'intégralité de la séquence. Sans garantir de trouver l'alignement optimal, ils permettent souvent d'obtenir de bons alignements malgré tout.

Un principe fréquent dans la définition de ces méthodes est l'utilisation de mots de taille k, également appelés k-mers, qui sont de courtes sous-séquences de taille k dans la séquence requête et qui vont servir de point d'ancrage. Ces k-mers sont comparés à un index d'autres k-mers provenant des séquences que l'on cherche à aligner. Une fois que plusieurs k-mers ont été identifiés dans une séquence, on compare l'ordre des k-mers et la distance qui les sépare dans les deux séquences. Si l'ordre et la distance sont compatibles, cela peut indiquer que les séquences s'alignent dans cette région.

Une fois les points d'ancrage identifiés, on peut aligner les régions entre les k-mers en utilisant des méthodes exactes. Cela permet de réduire considérablement la taille des séquences à aligner, ce qui diminue le coût computationnel [Mar+18].

Ces méthodes sont surtout connues pour leur utilisation dans l'outil de recherche BLAST [Alt+90b], qui a été en partie créé pour identifier la famille à laquelle appartient un gène. Mais il existe de nombreux autres outils qui utilisent ce principe d'ancrage et d'extension. On utilise par exemple ces approches lorsqu'on cherche à aligner des reads de séquençage avec un génome de référence, car l'algorithme de Needleman-Wunsch n'est pas adapté lorsque la différence de taille entre les deux séquences est trop importante. On utilise donc l'approche k-mer pour identifier l'emplacement des reads sur la séquence de référence puis on aligne le reste du reads grâce aux méthodes exactes. Les algorithmes de Bowtie2 et BWA [LD09 ; LS12] en sont un exemple.

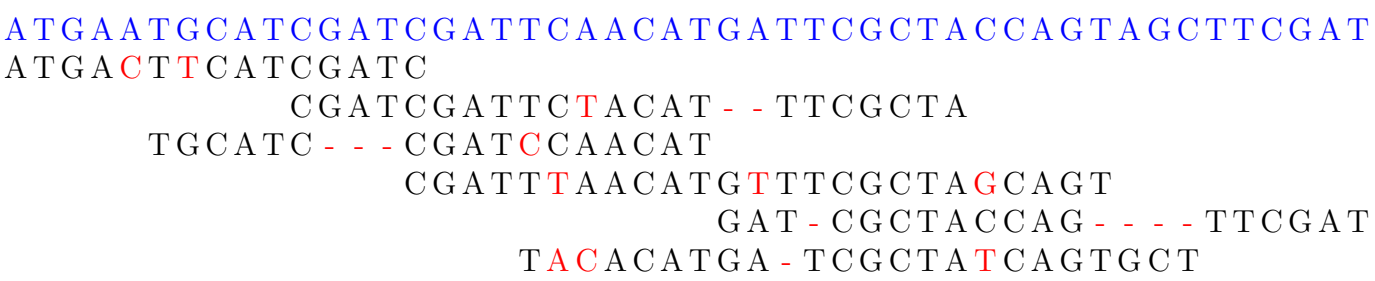


FIGURE 3.6 – Alignement de plusieurs reads, ici en noir, sur une séquence de référence, ici en bleu. Les variants entre les reads et la référence sont indiqués en rouge. Pour réaliser ce genre d'alignement, on utilise le plus souvent des méthodes heuristiques à base de k-mers.

3.3 Alignement multiple

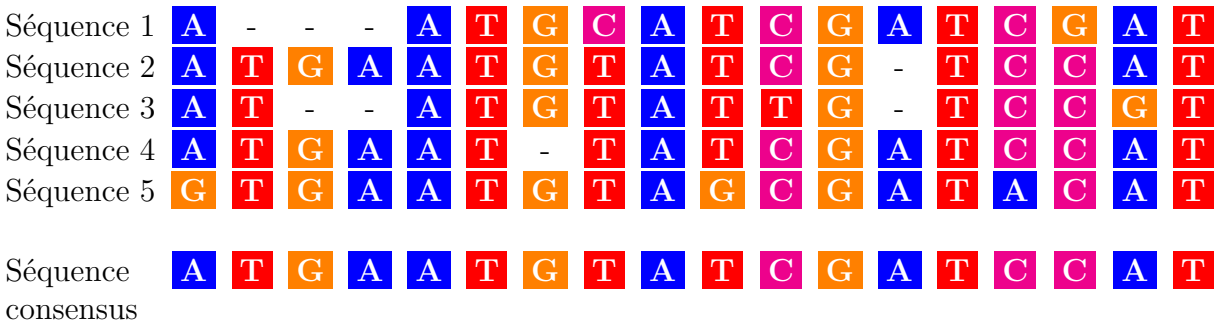


FIGURE 3.7 – Exemple d'un alignement multiple et de la séquence consensus associée. Pour faciliter la lisibilité, les outils de visualisation attribuent souvent une couleur par nucléotide.

Lorsque l'on procède à l'alignement de trois séquences d'ADN ou plus, on parle d'*alignement multiple*, ou en anglais, "multiple sequence alignment" (MSA). Ce type d'alignement a été principalement mis au point pour analyser les gènes d'une même famille. Cela sert en phylogénie [Der+08], pour retracer l'histoire évolutive des séquences, pour identifier et étudier des domaines conservés de protéines, pour chercher des éléments de structure secondaire dans des séquences d'ARN et ou encore pour identifier des variants. Le principe de cet alignement demeure similaire à celui de l'alignement par paire : on dispose les séquences dans un tableau, une séquence par ligne, afin de mettre en relation les positions correspondantes entre les séquences.

Une particularité de l'alignement multiple par rapport à l'alignement deux à deux réside dans la possibilité de générer une *séquence consensus*. Cette séquence est une séquence créée en conservant le ou les nucléotides les plus fréquents à chaque position de l'alignement. C'est une sorte de résumé de l'alignement, qui peut servir à identifier les positions bien conservées. Dans ma thèse, j'utiliserai la faculté d'un alignement multiple d'être associé à une séquence consensus pour évaluer les alignements multiples des long reads (voir chapitre 4). Il est également possible de préserver dans cette séquence consensus les différents nucléotides possibles pour chaque position en utilisant le *code IUPAC*, qui

sera détaillé plus loin.

La construction d'un alignement multiple est une question algorithmique plus poussée que celle d'un alignement deux à deux. En première instance, il est possible d'étendre l'algorithme de Needleman-Wunsch à un nombre arbitraire de séquences grâce à une matrice de programmation dynamique multidimensionnelle pour obtenir des alignements multiples optimaux. Cependant, cela soulève plusieurs problèmes. Le premier est celui de la définition de score d'alignement. Le score *sum of pairs* (SP) est fréquemment utilisé, qui estime la qualité d'un alignement en additionnant les scores de toutes les paires d'alignements de toutes les séquences. Le second problème, et le plus grand défi, est celui du coût en temps et en mémoire qui augmente exponentiellement avec le nombre de séquences pour ce type de méthode [WJ94 ; Eli06]. En pratique, cette complexité rend une approche exacte inutilisable sur des ensembles de données même de taille modeste.

Afin de résoudre le problème du coût, on utilise des méthodes heuristiques. Il existe de nombreuses méthodes différentes [Cha+15]. Je présente ici une sélection, organisée en quatre catégories : les méthodes en étoile, les méthodes progressives, les méthodes itératives et les méthodes à base de graphes.

3.3.1 Méthodes en étoile

L'*alignement en étoile* [Gus93] est fondé sur l'hypothèse qu'avec trois séquences A, B et C, si A et B s'alignent en paire et que B et C s'alignent en paire, alors A et C peuvent également s'aligner en paire. En se basant sur cette hypothèse, cette méthode choisit une séquence centrale qu'elle aligne avec toutes les autres séquences avant de créer l'alignement final. Dans l'exemple cité, la séquence centrale est B.

Le choix de la séquence centrale a un impact significatif sur la qualité de l'alignement. Elle peut être générée ou choisie parmi les séquences. Le meilleur choix serait de sélectionner la séquence qui a la distance moyenne la plus faible par rapport à toutes les autres, mais ce calcul est coûteux. Par conséquent, il est difficile d'appliquer ce choix de séquence centrale à des ensembles de données comportant un grand nombre de séquences dans la pratique. Pour contourner ce problème, le choix récurrent est de sélectionner comme séquence centrale la séquence la plus longue. En effet, plus la séquence est longue, moins elle a perdu d'informations par rapport aux autres séquences.

La limitation de cette méthode est qu'elle est moins efficace dans le cas des données très hétérogènes. En reprenant l'exemple précédent, on prend B comme séquence centrale et on aligne B avec A et B avec C. Si A est très différent de C, l'alignement final sera de moindre qualité puisque qu'on n'aligne

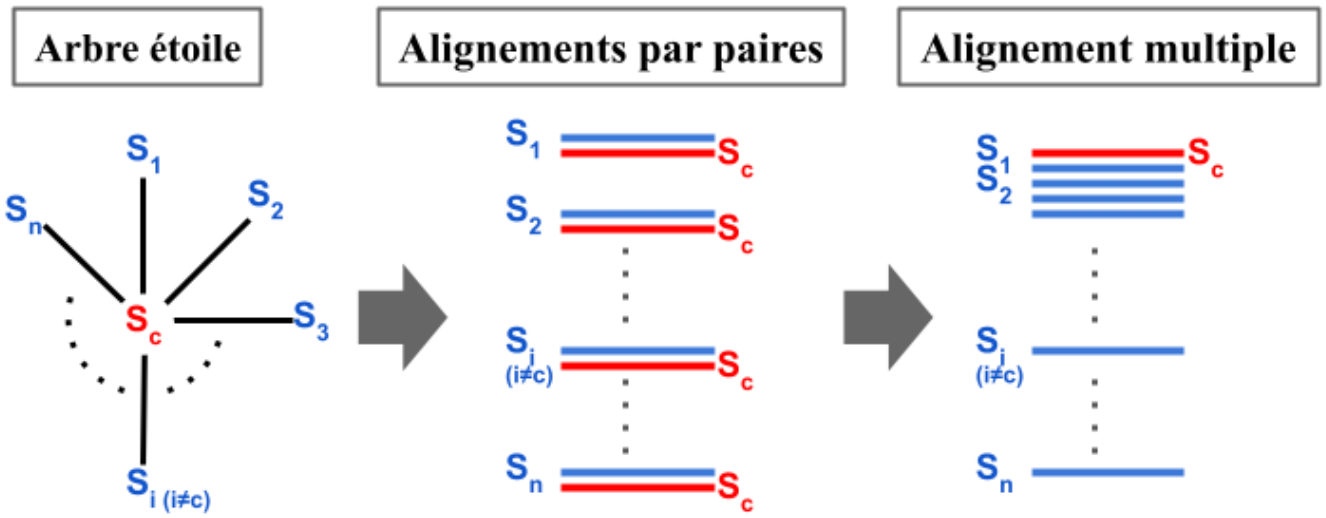


FIGURE 3.8 – Schéma de la méthodologie de l'alignement en étoile. S_c est la séquence centrale.

pas directement A avec C. Cependant, dans le cas de données avec des séquences plus homogènes, on obtient de très bons résultats.

L'un des meilleurs avantages de cette méthode est sa simplicité, qui s'accompagne d'un coût en temps qui est extrêmement faible, bien meilleur que les autres algorithmes. Mais l'alignement final est très sensible au choix de la séquence centrale, et la méthode d'alignement ne convient pas à des séquences mal conservées, peu similaires.

3.3.2 Méthodes progressives

L'alignement progressif [FD87] peut être vu comme une amélioration de la méthode en étoile, dans le sens où son point de départ reste des alignements deux à deux, tout en s'affranchissant du choix d'une séquence centrale. Le principe est d'établir un alignement multiple en combinant des alignements par paires, en commençant par les paires les plus similaires. Pour déterminer quelles paires sont les plus similaires, on utilise un *arbre guide*. C'est un concept fondamental, au cœur de toutes les méthodes progressives.

Chaque feuille de l'arbre guide représente une séquence, et chaque nœud représente un alignement multiples des séquences correspondant aux feuilles situées en dessous du nœud. La méthode consiste à aligner les séquences en remontant l'arbre, des feuilles jusqu'à la racine. On commence par aligner deux séquences (feuilles) pour obtenir un alignement par paire (nœud), puis on aligne cet alignement avec la séquence la plus proche (feuille) ou avec d'autres alignements (nœuds) jusqu'à avoir incorporé

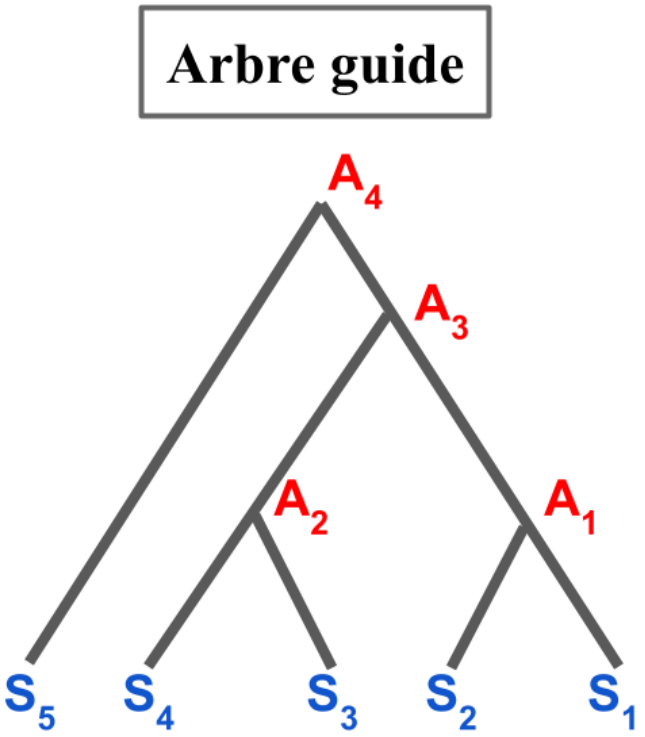


FIGURE 3.9 – Schéma d'un arbre guide. Les séquences notées S sont regroupées par similarité et sont ensuite alignées par paire. On aligne S_1 et S_2 pour obtenir l'alignement A_1 puis S_3 et S_4 pour obtenir l'alignement A_2. A partir de A_1 et A_2, on obtient l'alignement A_3 qu'on réaligne avec S_5 pour obtenir l'alignement final A_4.

toutes les séquences. On est alors à la racine de l'arbre. La figure 3.9 est un exemple d'utilisation d'arbre guide.

La qualité de l'alignement dépend de la qualité de l'arbre guide. Il existe plusieurs algorithmes pour construire cet arbre. L'algorithme UPGMA (Unweighted Pair Group Method with Arithmetic Mean), qui construit l'arbre à partir d'une matrice de distances entre les séquences, est le plus utilisé. C'est le cas des premières versions de Clustal par exemple [Lar+07]. Cependant, le coût en temps et en mémoire peut encore poser problème puisqu'il faut stocker cette matrice mais surtout aligner toutes les séquences deux à deux pour calculer les distances. Cela donne une complexité de $\mathcal{O}(N^2)$ où N est le nombre de séquences. Pour pallier ce problème, les méthodes plus modernes utilisent des heuristiques pour accélérer la construction de l'arbre guide.

Clustal Omega Afin de réduire le coût de calcul, Clustal Omega [Sie+11] emploie une méthode appelée mBed [Bla+10] (Modified BED) pour créer un arbre guide. Cette technique permet d'éviter l'alignement de toutes les séquences deux à deux pour générer la matrice de distance de l'algorithme

UPGMA. Pour ce faire, elle représente chaque séquence par un vecteur dans un espace de dimension inférieure, vecteurs qui sont ensuite regroupés par des techniques de clustering. Cela aboutit à un arbre guide précis tout en réduisant significativement les coûts en temps et en mémoire. La complexité est en $\mathcal{O}(NlogN)$).

Une fois l'arbre guide créé, Clustal Omega a recourt à des modèles statistiques (modèles de Markov cachés) pour calculer les alignements, conduisant à des alignements de haute qualité. Il est connu pour obtenir d'excellents résultats avec les séquences d'ADN d'une même familles de gènes ou de protéines qui partagent une relation évolutive.

T-Coffee T-Coffee [Mag+14] génère un ensemble de données comprenant tous les alignements par paires entre les séquences en amont. Cela permet de construire une bibliothèque d'informations d'alignement locaux et globaux qui guident l'alignement progressif. Les alignements intermédiaires s'appuient non seulement sur les séquences à aligner par la suite, mais aussi sur la manière dont toutes les séquences précédentes s'alignent entre elles. Ces informations d'alignement proviennent de différentes sources, comme un mélange de programmes d'alignement ou des superpositions de structures.

L'algorithme de T-Coffee utilise une stratégie proche de celle de Clustal Omega. Toutefois, il se sert des informations de la bibliothèque pour réaliser l'alignement progressif en tenant compte des alignements entre toutes les paires de séquences à chaque étape - une caractéristique qui le distingue des autres outils. Cela permet d'obtenir un alignement progressif plus précis, tout en préservant la simplicité de l'approche progressive.

Kalign et Kalign3 Le principe de Kalign [LS05] est de comparé les séquences deux par deux, puis de construire un arbre guide, comme le font les autres méthodes progressives, grâce à la méthode UPGMA. La différence réside dans la comparaison deux à deux, qui se fait sur un mode local avec l'algorithme de Wu-Manber [WM92] de correspondance de chaînes approximatives.

Il existe deux nouvelles versions de Kalign : Kalign2 [LFS09] et Kalign3 [Las20]. Kalign2, que j'appellerai simplement Kalign par la suite car il est très proche de la première version de Kalign tout en l'ayant supplanté, utilise l'algorithme de correspondance de chaînes approximative de Muth et Manber [MM96] au lieu de celui de Wu-Manber, ce dernier étant plus rapide mais moins flexible. Kalign utilise également l'algorithme de Myers et Miller pour gérer l'alignement de séquences, pour une meilleure efficacité en termes de mémoire. Kalign3 remplace l'algorithme Muth et Manber par une nouvelle implémentation de l'algorithme de Gene Myers [Mye99], qui calcule la distance d'édition exacte entre deux chaînes tout en étant plus rapide et en traitant de plus longues séquences.

3.3.3 Méthodes itératives

Les méthodes progressives ont comme inconvénient d'être fortement dépendantes de l'arbre guide initial, qui conditionne grandement l'alignement multiple final. Les *méthodes itératives* visent à limiter cet inconvénient, en ajoutant des étapes de correction de l'alignement, de manière itérative [Hir+95].

Les méthodes itératives commencent donc comme les méthodes progressives, avec un arbre guide à partir de comparaisons deux à deux. Elles créent ainsi un premier alignement multiple. Ce qui les différencie des méthodes progressives, c'est que l'alignement obtenu sert ensuite à créer un nouvel arbre guide pour recréer un nouvel alignement avec un meilleur score. Ce processus se répète jusqu'à une convergence du score.

Cette amélioration de l'alignement de manière itérative permet de réduire le problème de dépendance aux alignement initiaux. Les outils d'alignement multiples suivants utilisent des méthodes itératives.

MAFFT MAFFT [KAT09] utilise l'algorithme de la transformée de Fourier rapide pour identifier les régions homologues entre chaque paire de séquences. Les paires de séquences sont ensuite alignées en se basant sur ces régions homologues. L'arbre guide est créé à partir de ces alignements. Les séquences sont finalement alignées en suivant l'ordre indiqué par l'arbre. MAFFT utilise également un système de notation simplifié qui réduit le temps de calcul et améliore la précision des alignements.

Il utilise ensuite une méthode itérative pour peaufiner son alignement. L'association de ces différentes méthodes permet à MAFFT d'être rapide tout en conservant une excellente qualité d'alignement. Il a une complexité en temps de $\mathcal{O}(N^2L^2)$, où N est le nombre de séquences et L la longueur maximale des séquences.

MUSCLE MUSCLE [Edg04] commence par construire un premier arbre guide à l'aide de la méthode UPGMA, avec une distance rapide à calculer : le nombre de k-mers partagés par les deux séquences. C'est un alignement progressif qui privilégie la rapidité à la qualité. Cet alignement est ensuite utilisé pour améliorer l'arbre guide en utilisant cette fois la distance de Kimura entre les alignements par paires. Un nouvel alignement est construit à partir de ce second arbre. La dernière étape divise l'arbre en sous-arbres et produit un nouvel alignement multiple en réalignant les sous-alignements issus des sous-arbres. Si le score est amélioré, le nouvel alignement est conservé, sinon il est écarté. Cette étape est répétée jusqu'à la convergence. Les deux premières étapes ont une complexité de $\mathcal{O}(N^2.L + N.L^2)$. La dernière étape ajoute une complexité de $\mathcal{O}(N^3.L)$.

3.3.4 Méthodes à base de graphe

POA La dernière catégorie de méthodes d'alignements multiples que je présente a été initiée avec POA [LGS02] (Partial Order Alignment). Le principe est de construire un graphe d'ordre partiel, dont les nœuds sont des nucléotides (pour les séquences d'ADN ou d'ARN) ou des acides aminés (pour les protéines), et dont les arcs correspondent à la succession des caractères dans la séquence. Un exemple est donné en figure 3.10. Une séquence n'est pas alignée avec une autre séquence mais incorporée directement dans le graphe d'ordre partiel, en étendant l'algorithme d'alignement de Needleman-Wunsh à l'aide d'une copie des embranchements du graphe sur la matrice. Chaque itération ajoute de nouveaux embranchements possibles sur le graphe. De cette manière, le graphe contient l'intégralité des séquences et représente une version compressée de l'alignement. Le graphe est ensuite parcouru pour chercher les nœuds les plus partagés, et c'est à partir de ces nœuds que l'alignement final est créé. On obtient une complexité en temps qui est linéaire au nombre de séquences. Plus précisément, la complexité est $\mathcal{O}(2p+1)n|V|)$ avec V le nombre de noeuds dans le graphe et p étant la moyenne du nombre de prédécesseurs des noeuds du graphe. Intuitivement plus les séquences sont similaires plus p et V seront faibles, si toutes les séquences sont identiques $p = 1$ et $|V| = l$ avec l la longueurs de séquences.

sPOA et abPOA sPOA [Vas+17a] et abPOA [**abPOA**] sont des outils inspirés tous les deux de la méthode de graphe d'ordre partiel de POA. sPOA utilise des vecteurs SIMD (Single Instruction on Multiple Data) pour paralléliser la méthode, à l'image de l'approche de parallélisation de l'algorithme de Smith-Waterman de Rognes et Seeberg [RS00]. Cela réduit le coût en temps permettant de passer la complexité à $\mathcal{O}((2np/k+1)n|V|)$, où k est le nombre de variables qui rentrent dans un vecteur SIMD. abPOA utilise une stratégie de parallélisation similaire à celle de sPOA qu'il améliore encore en parcourant mieux la matrice en collectant des informations sur l'alignement optimal.

3.3.5 Inconvénients des heuristiques

Les méthodes heuristiques d'alignement multiples comportent certains défauts majeurs en fonction des méthodes. Pour certaines, une fois qu'un gap est inséré, il ne peut jamais être retiré et affectera tout le reste des alignements notamment pour l'alignement en étoiles et l'alignement progressif. La qualité de l'arbre et de la séquence centrale est donc primordiale dans ce cas. Un autre problème est que ces méthodes peuvent favoriser les alignements locaux au détriment des alignements globaux. En effet, elles favorisent les optimums locaux qui sont plus faciles à identifier et qui permettent de relier plus facilement les séquences entre elles. Cela peut affecter négativement l'alignement final, car

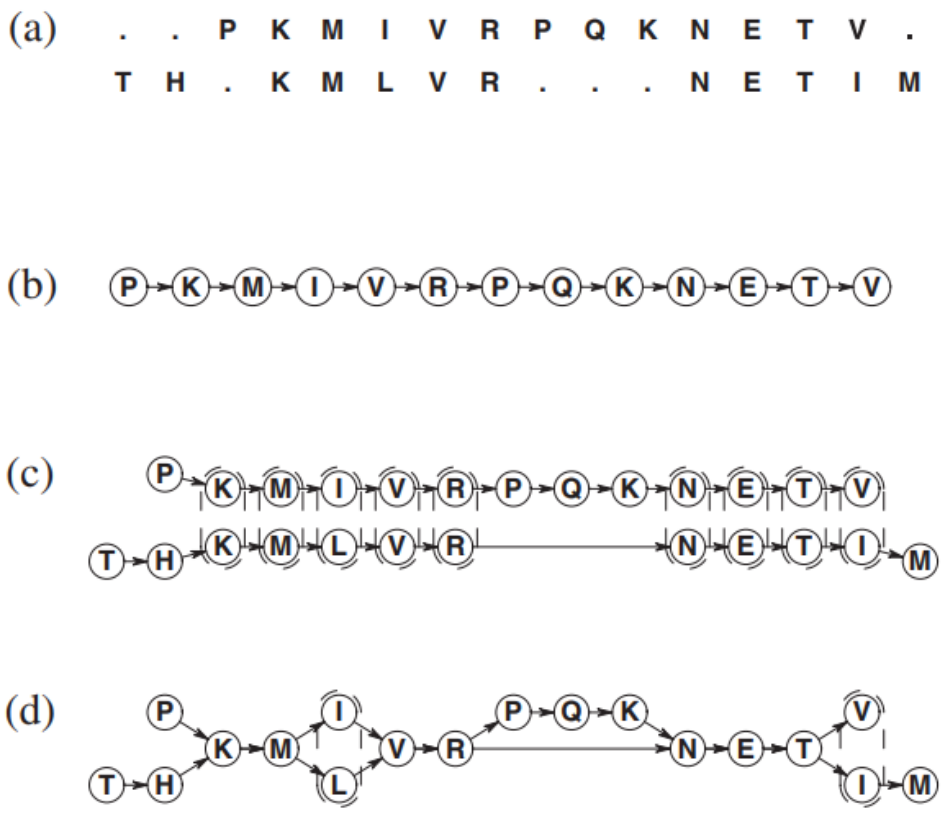


Fig. 1. MSA in the POA representation. (a) RC-MSA representation of a pairwise protein sequence alignment. (b) A single sequence in PO-MSA format. (c) Two protein sequences in PO-MSA format aligned to each other. Dashed circles indicate that two nodes are aligned. (d) PO-MSA representation of a pairwise protein sequence alignment. Dashed circles indicate that two nodes are aligned.

FIGURE 3.10 – Schéma de la méthodologie de POA

l'optimum sélectionné pour un sous-problème particulier n'est pas nécessairement optimal pour le problème global. Bien sûr dans le cas où les séquence présentent une similarité globale, ce problème est diminué.

Il existe cependant des méthodes pour remédier à ces problèmes. En vérifiant systématiquement la cohérence d'un nouvel alignement avec le reste des alignements déjà réalisés, on peut réduire les erreurs des étapes d'alignement. On peut aussi procéder à un affinement itératif après avoir terminé l'alignement pour essayer d'éliminer les erreurs qui auraient pu se glisser dedans.

Ces corrections sont cependant coûteuses en termes de temps. Reste à savoir si on préfère un résultat

le plus rapide possible ou le plus qualitatif possible. En réalité, les deux sont appréciables. Cela dépend des données que l'on traite et de ce que l'on cherche à obtenir.

3.4 Alignement multiple et Long Reads

Malgré ces difficultés, les alignements multiples sont déjà appliqués aux long reads, présentés dans la section 2.3.2. Cela permet de corriger certaines erreurs de séquençage en superposant les séquences et en identifiant les erreurs qui n'apparaissent généralement que dans un seul read. De plus, la création de séquences consensus peut considérablement réduire le bruit dans les données. Une des principales applications de l'alignement multiple sur les long reads est donc la suppression des nombreuses erreurs de séquençage afin de faciliter l'exploitation subséquente des données. Cette approche est couramment utilisée dans les logiciels d'assemblage des long reads pour obtenir un génome.

Les long reads ont des caractéristiques spécifiques. Historiquement, les outils et méthodes d'alignement multiples discutés dans ce chapitre n'ont pas été conçus pour traiter les long reads. Ces logiciels ont principalement été développés pour comparer les gènes et analyser les familles génétiques, représentées par des séquences protéiques ou nucléiques. Dans ce contexte, les disparités sont le fruit de l'évolution et les insertions, délétions et substitutions obéissent à une pression de sélection.

Les long reads diffèrent sensiblement de ce type de données. En effet, les séquences obtenues à partir des séquenceurs de troisième génération comportent un taux d'erreur non négligeable (5 à 15%) et, contrairement aux attentes basées sur les séquences apparentées, une grande majorité de ces erreurs sont des insertions et des délétions distribuées aléatoirement dans les séquences. Ces erreurs présentent un défi pour les outils d'alignement traditionnels, qui sont généralement conçus pour manipuler des séquences avec peu de gaps. De plus, la taille des long reads est une source de complication supplémentaire.

La question se pose donc de l'applicabilité des approches d'alignement aux long reads. Les approches aujourd'hui les plus couramment utilisées dans ce contexte sont celles issues de la famille POA, avec les méthodes à bases de graphe. On peut citer PBDAGCON, le module de correction d'erreurs de HGAP [Chi+13], ou Nanocorrect [LQS15], [Kor+17 ; Chi+16 ; Xia+17] pour l'assemblage et [KCS19 ; RL20 ; Bao+19 ; Miy+14 ; YM16 ; Mor+21] pour la correction/polissage. sPOA a été développé spécifiquement dans le cadre de RACON, pour améliorer la correction et le polissage [Vas+17b]. RACON a été largement utilisé pour améliorer de nombreux génomes publiés et est intégré dans d'autres outils d'assemblage, tels que Unicycler [Wic+17a] et Raven [V21].

La communauté des long reads a donc majoritairement adopté les outils d'alignement à bases de graphes. Mais, il n'existe pas de véritables études justifiant un tel choix. Les travaux existants pour l'évaluation des logiciels d'alignement sont nombreux, mais ont été menés avec d'autres applications en tête : familles de protéines homologues avec par exemple Balibase3 [Tho+05], OXbench [03] ou Quantest2 [SH19]), séquences d'ARN structurés [GWW05 ; Wri20], éléments transposables [HWS22]. Aucune de ces applications ne s'approche des long reads en termes de profil d'erreur ou de volumétrie. C'est ce constat qui a motivé le travail de cette thèse, et m'a poussé à développer un cadre complet pour l'évaluation des logiciels d'alignement multiple spécifiquement dédié aux long reads.

Chapitre 4

Pipeline d'analyse des alignements multiples entre long reads

Le but de cette thèse est d'étudier la capacité des outils d'alignement multiple présentés dans le chapitre 3 à s'appliquer aux données spécifiques que sont les long reads, avec un focus sur les reads ONT dont j'ai décrit le profil d'erreurs et la volumétrie en chapitre 2. Pour faire cela, j'ai développé le pipeline d'analyse `MSA_Limit` présenté dans ce chapitre.

La section 4.1 présente les différentes étapes de traitements jusqu'à l'obtention des alignements ainsi que la manière d'évaluer ces alignements. Elle présente aussi le *méta-consensus*, une méthode qui combine les différents outils entre eux pour essayer d'améliorer encore la qualité de l'alignement. Une sous-section est aussi consacrée au cas particulier des génomes diploïdes. La section 4.2 quant à elle présente les aspects techniques liés au développement du pipeline et sa manière d'être utilisé.

4.1 Déroulé du protocole `MSA_Limit`

Le pipeline `MSA_Limit` peut à partir d'un jeu de données de long reads pour lesquels on dispose d'une séquence de référence, telle qu'un génome, générer des alignements avec divers outils d'alignement, tailles de reads, profondeurs et différentes sections du génome et fournir des métriques d'évaluation pour les alignements obtenus. L'objectif est de tester les limites et les performances des différents outils d'alignement dans le plus grand nombre de cas possible, et d'établir si les outils étaient capables de gérer ces données en termes d'échelle, de taux d'erreurs et de types d'erreurs, mais aussi d'identifier si une méthode se démarque des autres sur ces reads.

Outils	Méthodes	Complexités
Clustal Omega	Progressive	$\mathcal{O}(NlogN)$
T-Coffee	Progressive	NA
Kalign	Progressive	$\mathcal{O}(NlogN)$
Kalign3	Progressive	$\mathcal{O}(NlogN)$
MAFFT	Itérative	$\mathcal{O}(N^2L^2)$
MUSCLE	Itérative	$\mathcal{O}(N^2L + N.L^2 + N^3.L)$
POA	Graphe	$\mathcal{O}(2p+1)n\|V\|$
sPOA	Graphe	$\mathcal{O}(2p/k+1)n\|V\|$
abPOA	Graphe	Similaire à POA

TABLE 4.1 – Résumé des outils d'alignements multiples utilisés. N étant le nombre de séquences, L leurs longueurs, V le nombre de noeud dans le graphe de POA qui peut être vu comme une mesure de la diversité totale des séquences présente et p le nombre moyen de prédécesseur dans les noeuds du graphe qui peut être vu comme une mesure de la divergence interséquences.

4.1.1 Les grandes étapes du pipeline

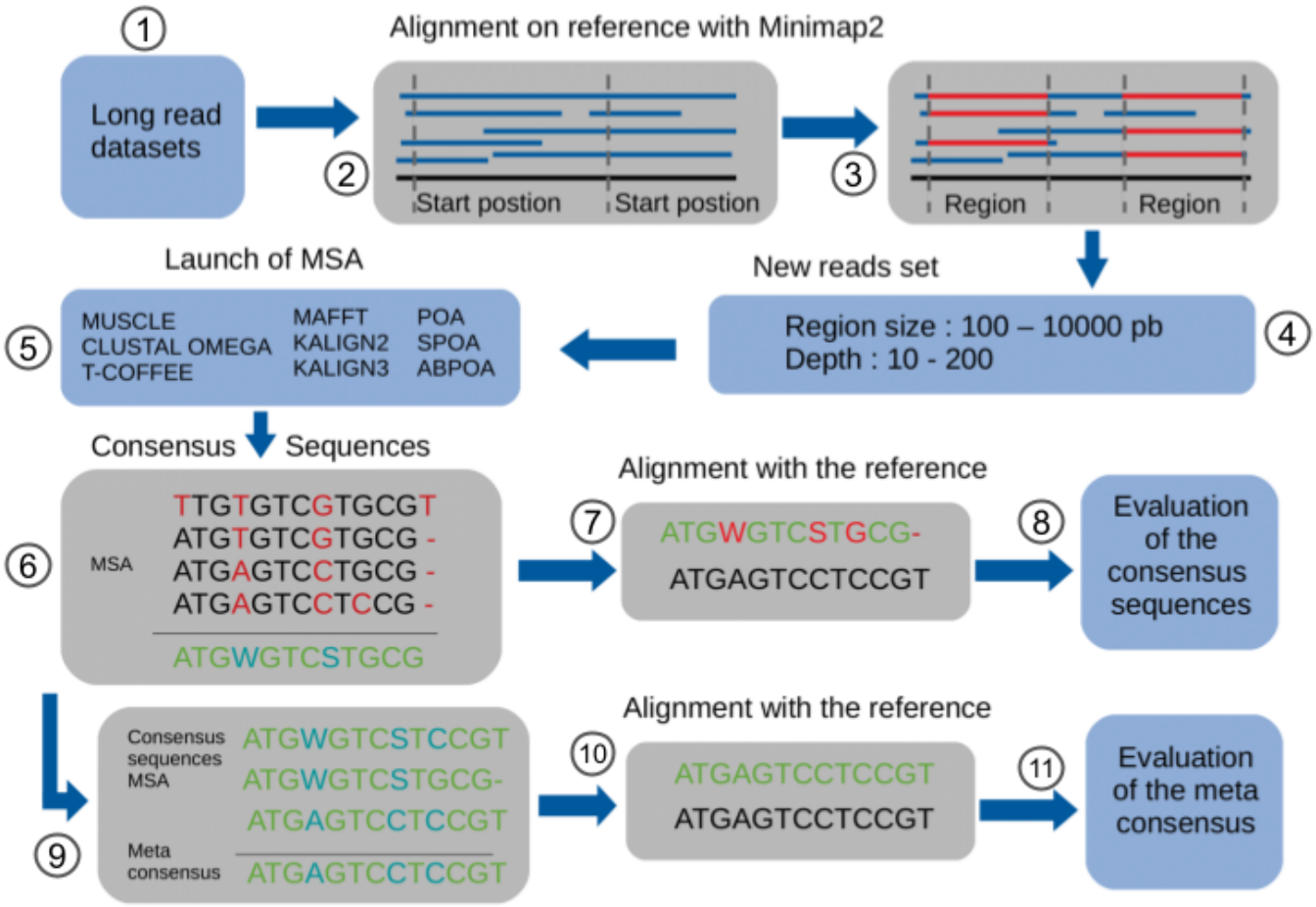


FIGURE 4.1 – Schéma global des grandes étapes du pipeline `MSA_Limit`

La figure 4.1 montre les différentes grandes étapes du pipeline, que l'on peut regrouper en trois

grandes tâches.

Étapes 1 à 5, traitement du jeu de données de long reads. Ces étapes consistent à créer plusieurs sous-jeux de données appelés *piles de reads* qui correspondent chacun à une portion de la référence et permettent de considérer des longueurs et des profondeurs de séquençage variables. Cela vise à normaliser les sous-jeux de données pour que les outils puissent tourner dans des conditions adéquates, c'est-à-dire que les reads qu'ils ont à analyser par sous-jeu de données, ont été sélectionnés pour être similaires en terme d'informations et de volume. Ces étapes sont détaillées dans la section 4.1.2.

Étapes 6 à 8, évaluation de la qualité de l'alignement multiple. Elles consistent à créer une séquence consensus pour chaque alignement multiple, qui servira à l'évaluation par comparaison à la séquence de référence. Meilleur sera l'alignement multiple, plus proche sera la séquence consensus de la séquence de référence. La proximité entre la séquence consensus et la séquence de référence est évaluée en alignant ces deux séquences, puis en calculant une série de métriques liées au taux d'identité. Ces étapes sont détaillées dans la section 4.1.3.

Étape 9 à 11, construction d'un méta-consensus. Ces dernières étapes consistent à combiner les différentes séquences consensus en les réalignant entre elles pour créer un consensus des séquences consensus issus des différents outils, appelé *méta-consensus*. Le but de cette réalisation est d'observer si la combinaison des différents outils permet d'améliorer la qualité de l'alignement. Cela est discuté en section 6.5.

4.1.2 Du jeu de données aux alignements multiples

La première étape consiste à préparer des *piles de reads* qui seront utilisées pour créer les alignements. Une pile de reads est ici définie par un nombre fini de reads qui s'alignent entièrement sur une région génomique donnée, et dont on ne conserve que la partie du read qui s'est réellement alignée (voir figure 4.2 page 62). L'avantage de ces piles est d'obtenir des ensembles de reads issus d'une même région et de même longueur, afin d'être dans un cas idéal d'utilisation des outils d'alignement multiple. Cela permet également d'identifier une référence pour chaque pile, qui permettra par la suite de valider l'alignement.

Pour créer ces piles, on part d'un jeu de données de long reads et d'un génome de référence corres-

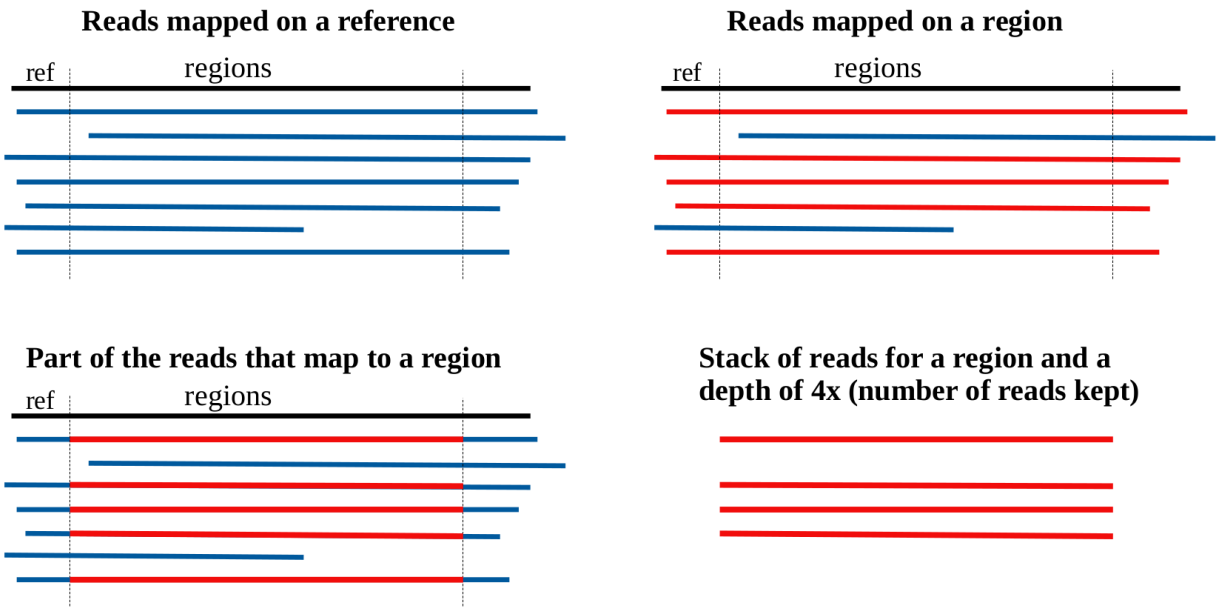


FIGURE 4.2 – Étapes pour obtenir une pile de reads

pondant à ce jeu. On aligne l'ensemble des reads sur la référence grâce à l'outil de mapping de long reads Minimap2 [Li17]. On définit ensuite sur la référence plusieurs positions de départ, qui sont identifiées par la position du nucléotide sur la référence (voir étape 1 de la figure 4.3 page 63). Ces différentes positions de départ permettent de tester des zones différentes du génome pour répéter l'expérience. À partir de chaque position de départ, on définit plusieurs régions de tailles différentes, qui démarrent toutes à la même position (voir étape 2 de la figure 4.3 page 63). À l'aide d'un script et de l'alignement issu de Minimap2, on récupère les reads qui s'alignent sur ces régions, dont on ne conserve que la partie alignée (voir étape 3 de la figure 4.3 page 63). Parmi cet ensemble de reads, on sélectionne ensuite des sous-ensembles aléatoires de cardinal variable, afin de simuler des profondeurs de séquençage différentes. Ainsi, pour chaque position de départ, taille de région et profondeur de séquençage données, on obtient une pile de reads pour laquelle la séquence de référence est connue.

Chacune de ces piles est ensuite donnée en entrée à tous les outils d'alignement multiple que l'on souhaite tester (voir étape 4 de la figure 4.3 page 63). On obtient ainsi un alignement par outil et par pile.

4.1.3 Séquences consensus et métriques

Séquences consensus Une fois les piles et les alignements obtenus, on souhaite évaluer la qualité de ces alignements. C'est une question classique quand on créé un benchmarks d'outils d'alignements

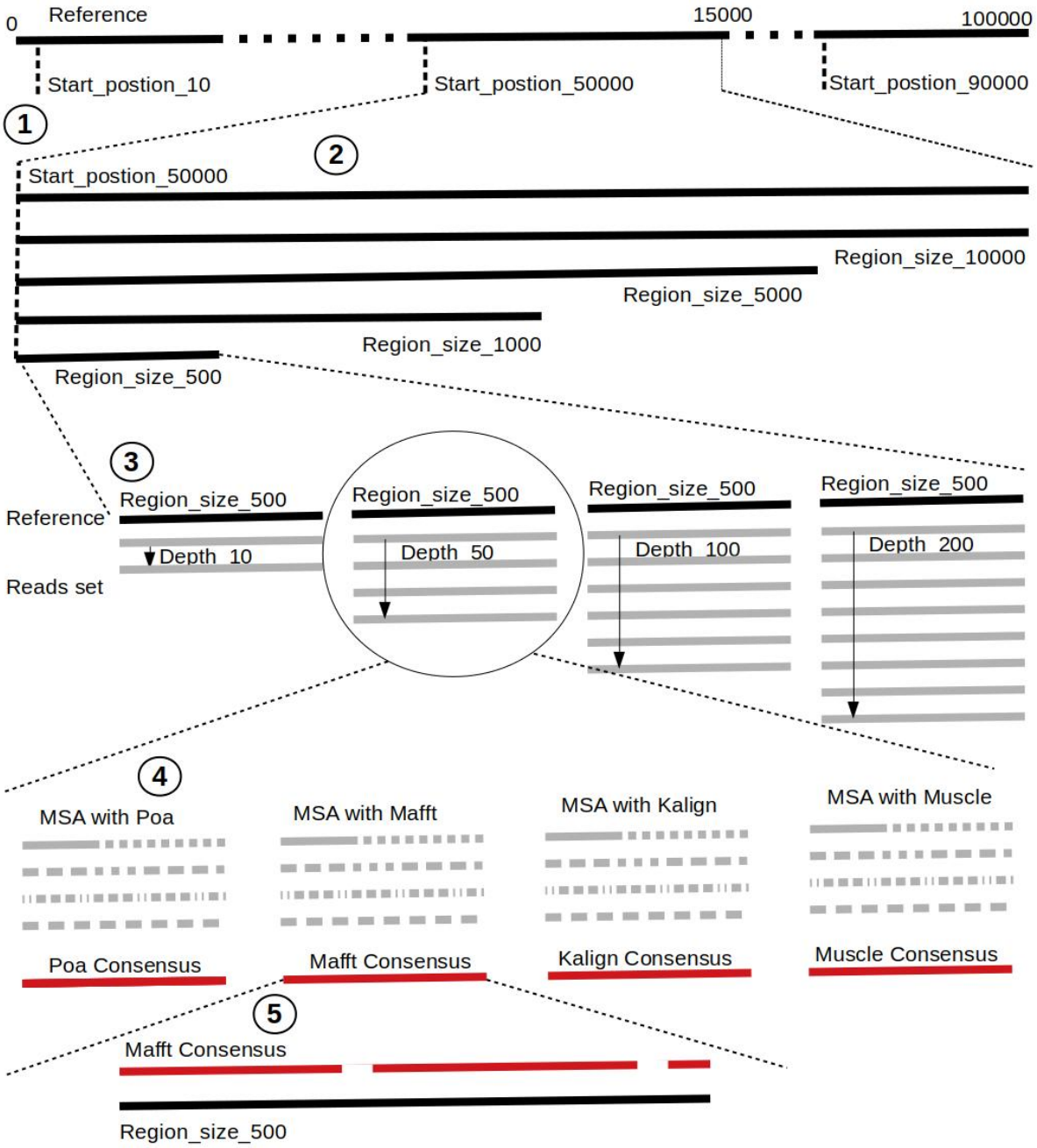


FIGURE 4.3 – Schéma des différentes étapes du pipeline pour passer des long reads bruts aux piles de reads, puis aux séquences consensus.

multiples [War21], que j'ai abordé dans le contexte spécifique des long reads. Contrairement aux benchmarks cités en section 3.4, les séquences à aligner ne sont pas sous pression de sélection. Elles ne dérivent pas d'un ancêtre commun et les différences ne sont pas dues à un processus évolutif. Dans mon cas, toutes les séquences sont issues d'une même séquence de référence, et les différences sont le fruit d'artéfacts techniques, les erreurs de séquençage. J'ai ainsi décidé de créer une séquence consensus à partir de chaque alignement, puis de comparer cette séquence obtenue avec la référence associée à la pile.

La séquence consensus utilise le code IUPAC pour l'ADN, décrit en figure 4.4 page 64. Ce code est un standard qui permet en un seul caractère de définir un ensemble de nucléotides possibles. Pour simplifier la suite, on appellera caractère IUPAC tous les caractères de ce code autres que les nucléotides A, T, G, C et les gaps, notés "-". Pour obtenir cette séquence consensus, l'algorithme

CODE	BASE
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
-	gap

FIGURE 4.4 – Code IUPAC pour l'ADN

est le suivant. On traite l'alignement colonne par colonne. Pour chacune d'entre elles, on calcule la fréquence pour chaque nucléotide ou gap rencontré. On ordonne ensuite par fréquence décroissante ces caractères. Si le caractère le plus fréquent est un gap, le gap l'emporte. En cas d'égalité de fréquences avec un nucléotide, on estimera que le nucléotide l'emporte sur le gap. Si c'est un nucléotide qui l'emporte, on ajoutera les autres nucléotides par ordre décroissant de fréquence tant que la totalité des fréquences ne dépasse pas un seuil fixé (voir exemple sur la figure 4.5 page 65). Les gaps ne sont pas pris en compte dans le calcul de la fréquence totale. Si on a une égalité de fréquences entre deux nucléotides, ces derniers seront tous les deux conservés. On utilise ensuite le code IUPAC associé aux nucléotides conservés pour représenter le consensus. On construit ainsi la séquence consensus petit à petit, colonne par colonne.

Une fois la séquence consensus obtenue, on l'aligne avec la référence de la pile grâce à l'outil Exonerate [SB05], qui permet d'aligner deux séquences. L'alignement permet de calculer différentes métriques :

— Nombre et pourcentage d'identité : nombre de caractères IUPAC et de nucléotides strictement identiques dans la séquence consensus et dans la référence. La référence peut contenir des

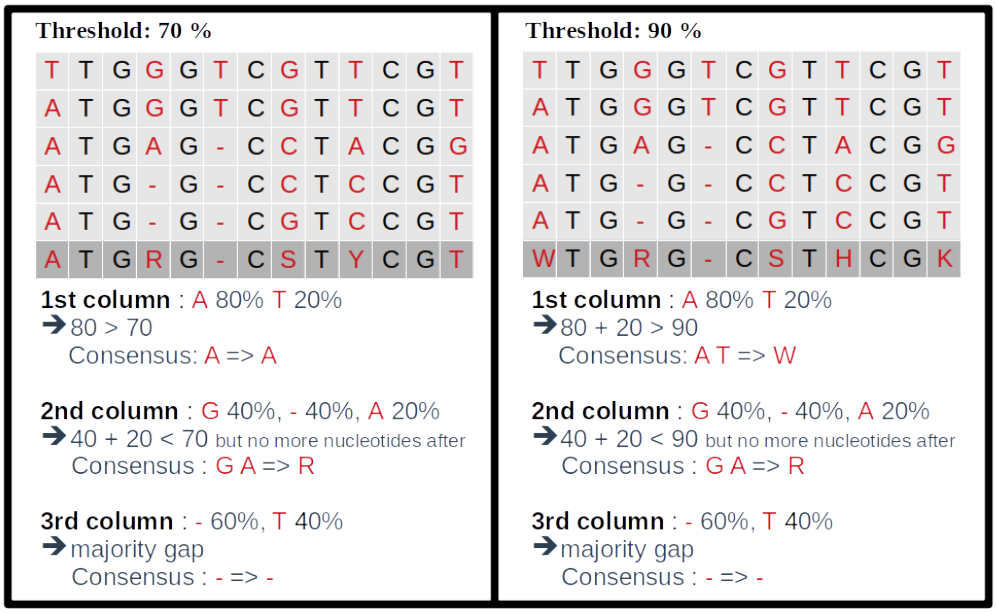


FIGURE 4.5 – Exemples de construction de séquence consensus avec un seuil de 70% et un seuil de 90%.

caractères IUPAC dans le cas diploïde.

- Nombre et pourcentage d'erreur : nombre de caractères IUPAC et de nucléotides différents dans la séquence consensus et dans la référence (inverse de l'identité).
- Score et pourcentage de match : les caractères IUPAC et les nucléotides identiques entre la séquence consensus et la référence comptent pour 1. Pour tous les autres cas, on applique l'indice de Jaccard, où pour deux ensembles A et B, on a : $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$.
  Exemple : W vs V => (A,T) vs (A,C,G) => $J(W,V) = \frac{|(A,T) \cap (A,C,G)|}{|(A,T) \cup (A,C,G)|} = \frac{1}{4}$ Ce score permet de prendre en compte la similarité entre les ensembles, contrairement à l'identité et l'erreur.
- Nombre et pourcentage de substitution : nombre de substitutions dans les erreurs entre la séquence consensus et la référence.
- Nombre et pourcentage de délétion : nombre de délétions dans les erreurs entre la séquence consensus et la référence.
- Nombre et pourcentage d'insertion : nombre d'insertions dans les erreurs entre la séquence consensus et la référence.
- Nombre et pourcentage d'ambiguïté : nombre de caractères IUPAC.
- Taille de la séquence consensus, sans compter les gaps.

On conserve également les données sur les performances des outils en termes de temps et de mémoire pour chaque pile de reads.

size	depth	msa_tool	1st	2nd	3rd	4th	5th	6th	7th
500	10	abPOA	72	4	9	5	5	4	1
500	10	kalign	57	3	8	7	10	10	5
500	10	kalign3	32	1	1	3	2	9	52
500	10	MAFFT	59	2	7	8	9	14	1
500	10	MUSCLE	70	5	6	12	6	1	0
500	10	POA	72	6	4	5	8	4	1
500	10	sPOA	81	5	6	3	5	0	0

FIGURE 4.6 – Exemple de classements pour 7 outils d'alignements multiples avec une taille de région de 500pb et une profondeur de 10x suivant le nombre d'identités. Ces classement ont été réalisés sur 100 positions de départ différentes. Attention, la somme des lignes ne fait pas 100 à cause de nombreux cas d'égalité pour certains rangs.

Affichage des résultats Pour faciliter l'analyse des résultats, le pipeline calcule automatiquement la moyenne et l'écart-type entre toutes les positions de départ pour chaque métrique de chaque expérience. Il génère également un graphique par métrique et par outil, représentant la métrique en fonction de la profondeur avec une courbe pour chaque taille de région. Ces graphiques sont réalisés pour les moyennes, mais aussi pour chaque position de départ. Enfin, afin de permettre la comparaison entre les différentes expériences si nécessaire, il est possible de demander un format où plusieurs expériences sont représentées pour une métrique donnée sur le même fichier. Ce format est plus facilement lisible par un être humain, mais n'est pas recommandé pour une utilisation informatique.

Le pipeline propose également la possibilité de créer un classement des différents outils, de manière automatique. Selon une métrique choisie, on classe les outils du meilleur au moins bon pour chaque position de départ, taille de région et profondeur fixées. On stocke ensuite dans un fichier le nombre de fois où chaque outil a été classé premier, deuxième, troisième, etc., pour chaque taille de région et profondeur fixées. Cela permet de résumer toutes les positions de départ dans un même fichier. Un exemple de ce classement est disponible sur la figure 4.6 page 66.

4.1.4 Cas des organismes diploïdes

Le cas des organismes diploïdes est particulier dans le sens où le génome de référence n'est pas résumé par une séquence, mais par deux séquences avec quelques variations entre ces deux séquences. Pour représenter ces variations, la référence peut ainsi comporter des caractères IUPAC du fait de la présence de deux haplotypes différents en son sein. Pour gérer cela, j'ai décidé de créer des métriques supplémentaires spécifiques à ce cas afin d'avoir un meilleur détail des différentes possibilités :

— Nombre de caractères IUPAC dans la référence.

— Nombre d'IUPAC identiques : le même caractère IUPAC entre la référence et la séquence consensus.
— Nombre d'IUPAC vs IUPAC : deux caractères IUPAC différents entre la référence et la séquence consensus.
— Nombre de nucléotides vs IUPAC : un nucléotide dans la séquence consensus et un caractère IUPAC dans la référence. L'inverse est aussi calculé.
— Nombre de gaps vs IUPAC : un gap dans la séquence consensus et un caractère IUPAC dans la référence. L'inverse est aussi calculé.

4.2 Aspects techniques

Le pipeline `MSA_Limit` a été développé en Python et en Perl. Je détaille dans cette section les différents choix techniques qui ont été faits.

4.2.1 Portabilité et reproductibilité

Dans le but de rendre le pipeline robuste, modulaire et facile d'utilisation, il a été développé avec Snakemake [KR12]. Cet outil permet de créer des workflows, des processus séquentiels accomplissant une tâche, en définissant un ensemble de règles qui permettent de générer des fichiers à partir d'autres fichiers et de scripts. Le fichier A est créé grâce à la règle 1 via le script S1, qui génère le fichier B utilisé par la règle 2 via le script S2, etc. Chaque règle constitue une étape du pipeline d'analyse.

La figure 4.7 page 68 représente le graphe des règles pour `MSA_Limit`. Il est important de noter que l'étape shuffle fasta, qui permet de mélanger les reads avant leur sélection pour créer les piles, est réalisée avec une graine aléatoire fixe. Ceci permet de reproduire l'aléatoire et donc l'expérience sans changement.

Snakemake permet naturellement de garantir la reproductibilité des résultats. Dans cette optique, j'ai aussi utilisé l'outil Conda, un système de gestion de paquets et d'environnements informatiques. Compatible avec Snakemake, il permet d'associer un environnement et ses paquets à une règle du pipeline. Cela évite les conflits, fixe les versions des paquets utilisés et facilite la portabilité de l'outil.

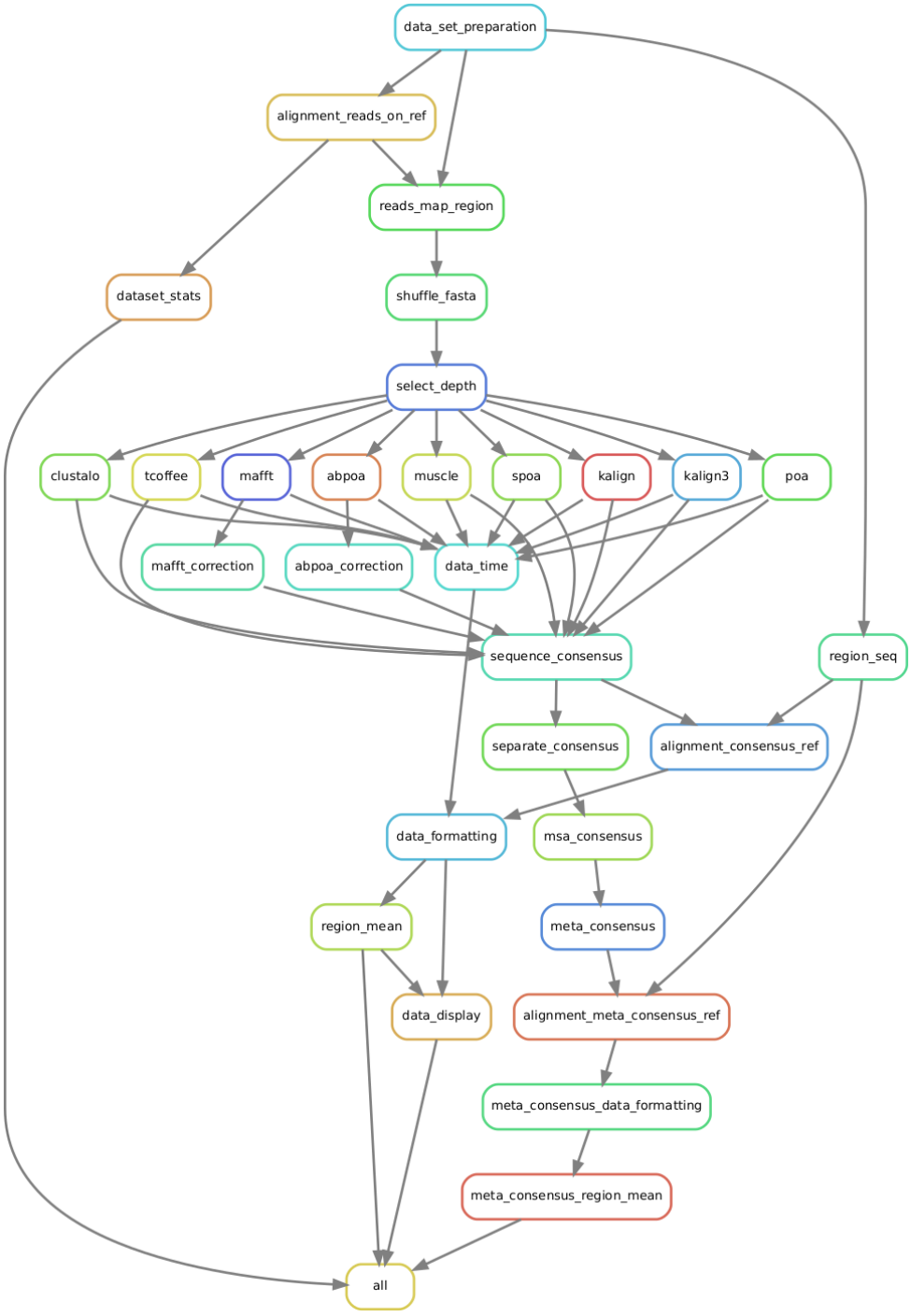


FIGURE 4.7 – Graphe des règles snakemake du pipeline `MSA_Limit`

4.2.2 Maintenance et optimisation

Afin de faciliter la maintenance et la modification du pipeline si nécessaire, les langages de scripts utilisés sont principalement en Python et en Bash. Certaines règles utilisent également des scripts en Perl.

Dans cette perspective, j'ai privilégié l'utilisation de formats standardisés couramment utilisés en bioinformatique. J'ai principalement utilisé le format FASTA, qui permet de stocker des séquences de nucléotides. Ce format peut également stocker des alignements si l'on représente les gaps dans les séquences. Néanmoins, il a fallu composer avec le format CLUSTAL, un format permettant de stocker un alignement de séquences, puisque certains outils ne fournissent que ce format. L'alignement avec minimap2 est quant à lui au format SAM, dédié au stockage des séquences alignées sur une séquence de référence. Le reste des données, essentiellement les résultats, est stocké dans des fichiers au format CSV.

Dans le but d'améliorer les performances du pipeline et des outils, j'ai implémenté la possibilité de rendre le pipeline multithread, c'est-à-dire capable d'exécuter plusieurs tâches sur plusieurs cœurs en simultané, ce qui permet de réduire le temps de calcul sur un ordinateur ou un serveur.

4.2.3 Paramètres

Le pipeline nécessite en entrée un jeu de données ONT et une référence associée au format FASTA ou FASTQ (formats de stockage de séquences de nucléotides). Plusieurs paramètres du pipeline peuvent être modifiés :

- *Le nombre de positions de départ.* Par défaut, le pipeline en définit 10, réparties uniformément le long de la référence grâce à un algorithme qui détermine automatiquement les numéros des nucléotides sur la référence. Il est possible d'en définir un nombre différent. Mais si les positions sont trop nombreuses ou si la référence est trop petite, certaines régions de différentes positions de départ se chevaucheront. Il est également possible de définir manuellement les positions de départ en saisissant directement les numéros des nucléotides.
- *Les tailles de régions testées.* par défaut, le pipeline propose deux tailles de régions, une de 100pb et l'autre de 200pb. Il est possible de tester d'autres tailles de régions.
- *La profondeur de séquençage.* Par défaut, le pipeline teste les profondeurs 10x, 20x et 50x, mais il est possible d'en définir d'autres.
- *Les outils d'alignement multiples.* Actuellement, neuf outils sont incorporés, abPOA, clustal omega, kalign (version 2 et version 3), MAFFT, MUSCLE, POA, sPOA et T-coffee. Par défaut, le pipeline lance abPOA, kalign, kalign3, MAFFT, MUSCLE, POA et sPOA. Clustal omega et T-coffee ne sont pas utilisés par défaut pour des raisons de performance. Je reviendrai sur ce point, qui sera discuté dans le chapitre 6. Il est possible de sélectionner les outils souhaités. Une documentation est disponible pour ajouter un outil non présent dans le pipeline via une nouvelle règle Snakemake.

— *Le seuil de fréquence pour l'obtention de la séquence consensus* (décrit en section 4.1.3 page 62). Par défaut, le pipeline utilise un seuil de fréquence pour la séquence consensus à 50%. Ce seuil est adapté pour le cas haploïde mais pas pour le cas diploïde. Il est possible de le modifier et même d'en tester plusieurs simultanément.

Grâce à des fichiers de configuration au format YAML pris en charge par Snakemake, il est possible de conserver une trace de l'expérience réalisée avec tous ses paramètres et ses données d'entrée.

Chapitre 5

Plan d'expérience

Le pipeline `MSA_Limit` présenté au chapitre précédent permet de tester des outils d'alignement multiple sur des données ONT avec des caractéristiques variées. Une réflexion a été faite pour savoir quel benchmark constituer pour mettre en œuvre ce pipeline. Ce chapitre est consacré à cette question. Dans la section 5.1 est présentée la sélection de jeux de données long read que j'ai constituée et dans la section 5.2 je détaille le choix des paramètres avec ces mêmes jeux. Les résultats associés sont présentés dans le chapitre suivant.

5.1 Jeux de données

5.1.1 Problématique du choix des jeux de données

Pour évaluer les différents outils d'alignement, on a besoin de jeux de données de séquençage ONT. Quatre critères principaux ont guidé mon choix : l'organisme, le taux et type d'erreur présents dans les reads, la profondeur de séquençage et la possibilité d'avoir une référence fiable. Le choix a aussi été fait de disposer à la fois de jeux de données réels et de jeux de données simulés.

Choix de l'organisme Tous les organismes n'ont pas le même degré de complexité. Certains possèdent plus de chromosomes, plus de répétitions ou plus de gènes que d'autres. Pour pouvoir tester au mieux les outils, il faut avoir un panel d'organismes différents les uns des autres, pour observer comment réagissent les différents outils sur des problèmes plus ou moins complexes.

Un autre critère à prendre en compte lors du choix de l'organisme est la diploïdie de celui-ci. Lors du séquençage, il est impossible de séparer les deux versions, appelées haplotypes, d'un diploïde et on obtient des reads où les deux sont mélangés de manière ambiguë. C'est un cas intéressant à observer dans l'étude des outils, afin de savoir si l'on peut récupérer les variations des deux haplotypes ou si celles-ci vont être confondues avec les erreurs de séquençage. Cependant, ce cas pose souci pour l'obtention d'une référence fiable. Cette problématique est détaillée plus en profondeur dans la suite de cette section.

Le choix, qui a été fait, est de privilégier des organismes haploïdes, avec une sélection de génomes procaryotes et eucaryotes et de consacrer un jeu de données à l'étude du cas diploïde.

Taux et types d'erreur Les données ONT peuvent avoir des taux et des types d'erreurs différents en fonction de l'avancée de la technologie et de l'organisme étudié. Ces deux paramètres ont un fort impact sur la qualité des alignements multiples. Afin de pouvoir tester un panel le plus large possible, j'ai essayé de prendre des jeux de données avec des taux d'erreurs variables. C'est une donnée que l'on peut facilement déterminer en alignant les reads sur le génome de référence avec Minimap2 et en estimant le taux d'erreur avec l'outil Samtools stats. Les types d'erreurs, gaps ou mismatchs, sont par contre plus difficiles à contrôler sur des données réelles, ce qui m'a incité à considérer également des données simulées.

Profondeur de séquençage J'ai veillé à sélectionner des jeux de données offrant des profondeurs de séquençage conséquentes, supérieures ou égales à 100x, pour pouvoir tester les outils sur leurs capacité à gérer des quantités plus ou moins importantes de reads. Les profondeurs de séquençage plus faibles sont simulées par échantillonage de l'ensemble des reads.

Avoir une référence fiable Le pipeline `MSA_Limit` a besoin d'un génome de référence pour créer les piles de reads et pour calculer les différentes métriques. Ceci implique qu'il est essentiel d'avoir une référence correspondant aux données, faute de quoi les résultats seraient difficiles à exploiter, car biaisés par la variation entre les reads et la référence. Ce point est loin d'être anecdotique quand on travaille sur des données réelles, car on ne dispose souvent que de la séquence de référence représentative d'une espèce, ou d'une souche, et les reads proviennent d'un individu particulier récemment séquencé qui présente d'inévitables variations.

J'ai ainsi fait le choix de reconstruire la séquence de référence par assemblage de reads, tel que décrit en section 2.3.3. Toutefois, utiliser les long reads de l'étude pour cet assemblage est proscrit. En effet, les logiciels permettant d'assembler des long reads comportent souvent une étape d'alignement

multiple avant l'assemblage à proprement parlé. Il y a donc un problème méthodologique à utiliser des assembleurs qui exploitent l'alignement multiple sur des long reads lorsque l'on veut justement évaluer les outils d'alignement multiple sur les long reads.

La solution retenue est de recourir à des jeux de données pour lesquels on dispose de deux types de reads obtenus indépendamment : des long reads d'une part, et des reads générés avec une autre technologie de séquençage d'autre part. Les long reads seront donnés en entrée du pipeline et les autres reads serviront à la construction de la séquence de référence par assemblage. Pour ces autres reads, j'ai considéré deux types de technologie de séquençage : le séquençage de deuxième génération Illumina et le séquençage HiFi. Pour les reads Illumina, j'ai utilisé le logiciel d'assemblage SPAdes [Ban+12] et pour les reads HiFi le logiciel d'assemblage hifiasm [Che+21] sans l'option d'intégration des long reads.

Il faut toutefois noter que les assembleurs ne sont pas capables de pleinement gérer le cas diploïde. Ils donnent une référence haploïde qui est un amalgame des deux haplotypes. Dans ce cas, il y aura toujours un léger biais entre la séquence consensus du pipeline et la référence puisque celle-ci a perdu de l'information. C'est pour cela qu'il a été choisi de privilégier les organismes haploïdes quand cela est possible.

Données simulées ou données réelles. De manière complémentaire, j'ai considéré deux types de jeux de données : des jeux de données réels et des jeux de données simulés à partir d'une séquence génomique de référence. Les données simulées présentent l'avantage d'être simples à obtenir, de pallier le problème de la coïncidence entre les reads et la référence et de permettre de maîtriser parfaitement le taux et le type d'erreur. Mais, bien qu'il existe de nombreux simulateurs qui tentent de se rapprocher au plus près des reads réels, il peut exister des comportements différents entre des données simulées et des données réelles. Si c'est une bonne option, ce n'est pas une solution suffisante pour une étude. Le cœur du benchmark est donc constitué de données réelles.

5.1.2 Données réelles

J'ai constitué cinq jeux de données réels, accompagnés de 32 jeux de données simulés. Le récapitulatif est visible dans le tableau 5.1.

Escherichia coli. *Escherichia coli* (abrégée *E. coli*) est une bactérie intestinale présente chez les mammifères, notamment chez l'être humain. C'est l'un des organismes les plus étudiés à ce jour grâce

Jeux réels	Référence	Taux d'erreur	Profondeur
E.coli illumina	assemblage de reads Illumina	16.36%	650x
E.coli HiFi	assemblage de reads HiFi	17.28%	200x
BMB yeast	assemblage de reads Illumina	10.8%	110x
Human	T2T-CHM13v2.0 (chromosome X)	6.6%	120X
Diploid	consensus d'assemblages de reads Illumina	11.6%	110x
Jeux simulés	Types d'erreur	Taux d'erreur	Profondeur
DEL	100% délétion	1, 2, 5, 10, 15, 20, 25, 30%	400x
INS	100% insertion	1, 2, 5, 10, 15, 20, 25, 30%	400x
SUB	100% substitution	1, 2, 5, 10, 15, 20, 25, 30%	400x
MIXTE	23% sub, 31% ins et 46% del	1, 2, 5, 10, 15, 20, 25, 30%	400x

TABLE 5.1 – Résumé des jeux de données retenus pour le benchmark. Les assemblages de reads Ilumina ont été réalisés avec SPAdes et les assemblages de reads HiFi avec hifiasm.

à son cycle de vie court. Il sert d'organisme modèle dans le séquençage et a été très souvent séquencé. Il est haploïde et son génome, composé d'un seul chromosome, est extrêmement bien connu. Pour toutes ces raisons, j'ai décidé de l'utiliser dans cette étude. J'ai utilisé des données issues de deux souches différentes.

Pour le premier jeu de données, noté *E. coli Illumina* par la suite, la souche est *E. coli* CFSAN027350, provenant de l'échantillon SAMN10604456. Pour ces données, on possède à la fois des reads ONT (SRR8335315) et des reads Illumina (SRR8333590). J'ai construit la référence à partir des reads Illumina avec l'assembleur SPAdes. Les reads ONT ont une profondeur estimée à 650X et une longueur moyenne de read de 7 192 pb. Le taux d'erreur est estimé à 16,36%.

Pour le second jeu de données, noté *E. coli HiFi* par la suite, la souche utilisée provient de l'échantillon SAMN13901561 d'*E. coli* O127 :H6 str E2348/69. Pour ces données, on a des reads ONT (SRR12801740) et des reads HiFi (SRR11434954). La référence a été générée avec l'assembleur hifiasm en utilisant uniquement les lectures HiFi. Les reads ONT ont une profondeur estimée à 200X et une longueur moyenne de read de 10 409 pb. Le taux d'erreur est estimé à 17,28%.

***Saccharomyces cerevisiae* haploïde.** *Saccharomyces cerevisiae* est une levure notamment utilisée pour la fabrication des boissons fermentées. C'est un organisme très étudié qui sert souvent de modèle pour les organismes eucaryotes. Son génome est composé de 16 paires de chromosomes. Comme il est très connu et plus complexe que *E. coli*, j'ai décidé de l'utiliser dans cette étude. Cependant, cette levure est diploïde. J'ai donc sélectionné les données d'une levure homozygote, issus de l'article nPhase [Abo+21], qui possède des gènes identiques sur chaque chromosome d'une même paire. Bien qu'elle soit diploïde, on peut traiter cette levure comme un organisme haploïde. La levure considérée est la souche BMB, pour laquelle on dispose de reads Illumina (ERR1308675) et des

reads ONT (ERR4352154). Comme pour *E. coli Illumina*, j'ai assemblé les reads Illumina à l'aide de SPAdes pour obtenir la référence. Les reads ONT ont une profondeur estimée à 110x et une longueur moyenne de read de 17 kb. Le taux d'erreur est estimé à 10,8%.

Homo sapiens. Le génome humain est plus complexe et difficile à séquencer que les deux organismes précédents, avec 23 paires de chromosomes. Il comporte de nombreuses séquences répétées, ce qui rend son assemblage difficile. Ce n'est qu'en 2022 que le consortium Telomere-to-telomere (T2T) a assemblé le premier génome humain complet [Nur+22b]. J'ai décidé d'utiliser les reads ONT générés par le consortium (disponibles sur leur site web [1]) pour ce projet, afin d'avoir des données de séquençage récentes et surtout un génome humain complet comme référence. Ces reads ONT ont une profondeur estimée à 120x et un taux d'erreur de 6,6%.

Comme il s'agit d'un jeu de données volumineux, j'ai décidé de le réduire en utilisant uniquement le chromosome X issu du génome complet (T2T-CHM13v2.0) et les reads ONT qui s'alignent dessus. Pour identifier ces reads, il suffit d'aligner l'intégralité des reads sur l'ensemble du génome avec Minimap2. On obtient ainsi un jeu de données réduit et le chromosome X comme référence. Il faut toutefois noter que ces données proviennent d'une femme, ce qui signifie qu'il faut considérer qu'il s'agit d'un jeu de données diploïde et que, par conséquent, les résultats pourront être biaisés par l'hétérozygotie.

***Saccharomyces cerevisiae* diploïde.** Afin d'étudier le cas diploïde de la meilleure manière possible, j'ai décidé de créer artificiellement une levure diploïde en fusionnant deux jeux de données réels provenant de deux levures haploïdes. Le but étant ensuite de pouvoir crée une référence diploïde à partir des deux références haploïdes et ainsi d'avoir une meilleur référence. Ce protocole s'inspire de celui introduit dans nPhase [Abo+21]. J'ai repris les données de la souche BMB de *Saccharomyces cerevisiae* et l'assemblage créé à partir des reads Illumina. J'ai appliqué la même méthode sur la souche CCN, issus du même article que BMB et le protocole, dont on possède des reads Illumina (ERR1308732) et des reads ONT (ERR4352155). Cette souche présente les mêmes caractéristiques que celles de la souche BMB (taux d'erreur de 10,9%, levure homozygote, reads ONT et reads Illumina disponibles). Une fois obtenu l'assemblage issu du séquençage Illumina de CCN, je l'ai aligné avec l'assemblage de BMB, ce qui m'a permis de créer une séquence consensus en utilisant le même algorithme que celui employé dans le pipeline `MSA_Limit` (voir figure 4.5) et un seuil fixé à 100% pour conserver toutes les variations entre les deux assemblages. Ainsi, j'obtiens une séquence utilisable comme référence, qui contient des caractères IUPAC aux emplacements où se situent les variations entre les deux levures.

1. `github.com/marbl/CHM13`

Pour obtenir le jeux de reads ONT correspondant, j'ai fusionné les deux jeux de données ONT issus des deux souches BMB et CCN, formant ainsi un jeu de séquençage ONT correspondant à cette nouvelle référence diploïde. Ce jeu a une profondeur estimée à 110x, un taux d'erreur de 11,6% et une hétérozygotie de 1,18%.

5.1.3 Données simulées

Afin de tester plus facilement différents cas de figure, j'ai décidé d'utiliser des long reads simulés à partir d'une séquence de référence. Cela permet d'avoir un contrôle sur le taux et le type d'erreur, mais aussi d'être sûr d'avoir une correspondance exacte entre la référence et les reads. Pour cela, j'ai utilisé PBSIM2 [OAH21], un simulateur de long reads et la souche d'*E. coli* K-12 (Code GenBank : GCA_000005845.2) comme séquence de référence.

J'ai défini quatre profils d'erreurs, notés DEL, INS, SUB et MIXTE, qui contiennent respectivement uniquement des insertions, uniquement des délétions, uniquement des substitutions et, pour MIXTE une proportion de 23% de substitutions, 31% d'insertions et 46% de délétions (ce qui correspond au profil d'erreur par défaut des reads ONT proposé par PBSIM2). Pour chacun de ces quatre profils, j'ai généré huit jeux de données avec des taux d'erreur de 1, 2, 5, 10, 15, 20, 25 et 30%, qui ont tous une profondeur de séquençage de 400x. Cela donne un total de 32 jeux de données simulés.

5.1.4 Workflow et reproductibilité

Les paragraphes précédents montrent que la génération des jeux de données pour le benchmark passe par plusieurs étapes computationnelles. Dans le but d'assurer la reproductibilité de ce protocole, j'ai créé un workflow avec Snakemake et Conda (à l'image de ce que j'avais fait pour `MSA_Limit`, en section 4.2.1). Ce workflow génère intégralement les données réelles et simulées. Il télécharge les données nécessaires, puis réalise toutes les étapes pour créer les différents jeux de reads ONT et leur référence associée. Il permet également de générer des fichiers de configuration pour le pipeline `MSA_Limit` afin de faciliter le fait de relancer l'intégralité des expériences avec les bons paramètres et les jeux de données adéquats. De plus, il est multithread pour optimiser le temps d'exécution.

5.2 Paramètres des expériences

Après avoir choisi les jeux de données, la seconde partie du plan d'expérience est le choix des paramètres et des outils avec lesquels lancer le pipeline `MSA_Limit`.

Les outils d'alignement multiple à évaluer. J'ai retenu l'intégralité des outils intégrés, c'est-à-dire MUSCLE, MAFFT, POA, sPOA, abPOA, kalign, kalign3 et clustal omega, qui ont tous été présentés en chapitre 3. Cette collection d'outils permet à la fois d'avoir une bonne couverture en terme de représentativité et complémentarité au sein de l'état de l'art, et de mettre un accent sur la famille des logiciels de type POA qui sont davantage utilisés dans la communauté long reads.

Profondeur de séquençage. J'ai décidé de tester des profondeurs de séquençage allant de 10x à 200x, en mettant l'accent sur les profondeurs entre 30x et 60x. Cet intervalle correspond aux recommandations de bonnes pratique et on y retrouve les profondeurs de séquençage les plus courantes.

Taille des régions. J'ai choisi de les faire varier entre 100pb et 10 000pb, puisque les long reads peuvent atteindre de telles tailles.

Seuil des séquences consensus. J'en ai choisi deux : 50% et 70%, le premier étant plus adapté aux haploïdes et le second plus adapté aux diploïdes.

Nombre de régions échantillonnées. Deux valeurs ont été considérées : 10 positions de départ pour obtenir des résultats préliminaires de manière rapide et quand cela a été possible 100 positions de départ pour obtenir des statistiques d'évaluation plus fiables. Nous reprenons le détail dans le chapitre suivant.

Paramétrage des outils. Pour toutes les expériences, j'ai utilisé minimap2 avec l'option prévue pour les reads ONT et tous les outils d'alignement multiple ont été utilisés avec les options par défaut.

Chapitre 6

Résultats

J'ai analysé les jeux de données présentés dans le chapitre 5 avec le pipeline d'analyse du chapitre 4 et les outils d'alignement multiple du chapitre 3.

Dans ce chapitre sont présentés les différents résultats : le comportement général des outils dans la section 6.1, l'évaluation de la qualité des alignements multiples dans la section 6.2 et l'évaluation en termes de mémoire et de temps dans la section 6.3. Toutes ces observations se placent dans le cadre de génomes haploïdes. Le cas de génomes diploïdes est abordé dans la section 6.4.

6.1 Comportement général des outils

La première expérience a été menée sur un nombre restreint de régions échantillonnées, 10 régions, afin d'obtenir des résultats préliminaires de manière rapide et guider le reste de l'analyse. Les caractéristiques sont les suivantes :

- 10 positions de départ
- profondeur de séquençage : 10x, 20x, 30x, 45x, 50x, 60x, 100x, 150x, 200x
- outils : MUSCLE, MAFFT, POA, sPOA, abPOA, kalign, kalign3, clustal omega
- tailles des régions : 100pb, 200pb, 500pb, 1000pb, 2000pb, 5000pb, 10000pb
- seuil pour la séquence consensus : 50%
- jeux de données : tous les jeux réels

La seule exception est T-coffee, pour lequel j'ai du limiter la taille des données testées. Étant donné que celui-ci consomme énormément de mémoire, il n'a pas pu être testé sur des profondeurs dépassant 100x et des tailles de reads dépassant 1 000pb. Les positions de départ sont identiques que pour les

	Ecoli-Hifi	Ecoli-Illumina	BMB yeast	Human
Taux d'identité				
abPOA	95.2 ▷ 97.9	95.8 ▷ 97.4	98.8 ▷ 99.7	99.6 ▷ 99.9
clustal o	83.9 ▷ 89.7	84.8 ▷ 91.2	92.4 ▷ 96.2	95.1 ▷ 98.6
kalign	93.6 ▷ 98.7	92.9 ▷ 97.8	97.8 ▷ 99.7	99.6 ▷ 100
kalign3	90.1 ▷ 98.6	90.7 ▷ 97.4	96.8 ▷ 99.7	99.4 ▷ 99.9
mafft	93.1 ▷ 98.9	94.2 ▷ 98.4	98.2 ▷ 99.8	99.5 ▷ 100
muscle	95.5 ▷ 98.6	95.3 ▷ 97.8	98.7 ▷ 99.6	99.7 ▷ 99.9
POA	93.7 ▷ 97.0	96.1 ▷ 97.1	98.9 ▷ 99.6	99.7 ▷ 99.9
sPOA	95.2 ▷ 98.6	95.9 ▷ 98.0	98.9 ▷ 99.8	99.6 ▷ 99.9
T-Coffee	96.2 ▷ 99.3	95.7 ▷ 98.3	99.1 ▷ 99.9	99.7 ▷ 100.0
Temps				
abPOA	0.0 ▷ 0.7	0.0 ▷ 0.6	0.0 ▷ 0.5	0.0 ▷ 0.6
clustal o	0.4 ▷ 89.7	0.4 ▷ 82	0.4 ▷ 129.2	0.3 ▷ 100.9
kalign	0.0 ▷ 3.4	0.0 ▷ 3.1	0.0 ▷ 3.5	0.0 ▷ 3.3
kalign3	0.1 ▷ 5.4	0.1 ▷ 5.3	0.1 ▷ 6	0.1 ▷ 5.9
mafft	0.3 ▷ 8.7	0.2 ▷ 8.5	0.2 ▷ 9.2	0.2 ▷ 8.5
muscle	0.4 ▷ 173.3	0.3 ▷ 171.1	0.3 ▷ 122.6	0.3 ▷ 70.9
POA	0.1 ▷ 44.7	0.1 ▷ 24.3	0.1 ▷ 19.8	0.1 ▷ 15.4
sPOA	0.0 ▷ 2.1	0.0 ▷ 1.8	0.0 ▷ 1.7	0.0 ▷ 1.6
T-Coffee	1.3 ▷ 8102.9	12.9 ▷ 7479.5	13.5 ▷ 8114.3	13.5 ▷ 8114.3
Mémoire				
abPOA	3421 ▷ 42286	3295 ▷ 16240	3126 ▷ 35368	3668 ▷ 35388
clustal o	6108 ▷ 79812	6084 ▷ 77292	6092 ▷ 77816	5932 ▷ 76412
kalign	1889 ▷ 3564	1884 ▷ 3561	1876 ▷ 3507	2300 ▷ 3884
kalign3	4294 ▷ 8992	4296 ▷ 7938	4242 ▷ 7815	4524 ▷ 8412
mafft	22400 ▷ 37982	22324 ▷ 37792	22377 ▷ 36442	23460 ▷ 38404
muscle	78834 ▷ 71446	7794 ▷ 72808	7915 ▷ 73539	17976 ▷ 73212
POA	2497 ▷ 12533	2461 ▷ 11608	2431 ▷ 11512	3852 ▷ 13784
sPOA	8463 ▷ 43323	8972 ▷ 39088	8199 ▷ 38194	8832 ▷ 38844
T-Coffee	47640 ▷ 355802	47419 ▷ 452623	115860 ▷ 434440	110488 ▷ 363888

TABLE 6.1 – Résultats pour plusieurs longueurs de régions et plusieurs profondeurs de séquençage, pour tous les outils de MSA et tous les jeux de données réels. Pour le taux d'identité, sont considérées toutes les combinaisons possibles de longueurs de régions (100, 200, 500, 1000, 2000, 5000 et 10 000pb) et de profondeur de séquençage (10x, 20x, 30x, 45x, 50x, 60x, 100x, 150x, 200x). Dix régions sont sélectionnées pour chaque combinaison, permettant le calcul de la moyenne du taux d'identité de la séquence consensus sur ces 10 régions. Pour chaque cellule du tableau, dans l'expression min ▷ max, min fait référence au taux d'identité moyen le plus faible et max au taux d'identité le plus élevé observés sur l'ensemble de ces combinaisons. Le temps d'exécution (en secondes) et l'utilisation de la mémoire (en Mo) sont reportés pour 10 régions d'une longueur de 200pb et d'une profondeur de séquençage de 10x (min) et 10 régions d'une longueur de 1000pb et d'une profondeur de séquençage de 100x (max).

autres outils, et de ce fait, les piles de reads sont identiques, c'est-à-dire que T-coffee a été testé sur exactement les mêmes données que les autres outils dans les cas où il a été possible de l'exécuter.

Sur les neuf outils d'alignement multiple testés, sept ont été capables de traiter tous les jeux de données en fournissant de bons ou très bons résultats : abPOA, Kalign, Kalign3, MAFFT, MUSCLE, POA et sPOA. La tableau 6.1 illustre ces résultats. Il existe cependant deux outils qui n'ont pas donné des résultats exploitables : Clustal Omega et T-Coffee.

Les expériences montrent que Clustal Omega n'est pas à la hauteur des autres outils pour ce qui est de l'alignement des long reads. Dans le tableau 6.1, on observe qu'il est largement en deça des autres outils en terme de taux d'identité. L'algorithme de Clustal Omega a du mal à gérer les insertions et les suppressions, ce qui engendre de nombreux gaps erronés dans les alignements multiples obtenus. De ce fait, Clustal Omega semble particulièrement inadapté pour gérer les erreurs de séquençage des long reads.

T-Coffee, quant à lui, produit des résultats de haute qualité. Mais il est gourmand en ressources. Il est considérablement plus lent et son utilisation de la mémoire est prohibitive. Par conséquent, pour les grandes régions ou le séquençage profond, T-Coffee n'est pas utilisable en pratique. Compte-tenu de l'amélioration mineure de la précision par rapport aux autres outils, son utilité est limitée.

J'ai donc décidé d'écarter Clustal Omega et T-coffee pour la suite de l'étude et de me concentrer sur les autres outils.

6.2 Évaluation de la qualité des alignements multiples

Afin d'obtenir des résultats plus fiables, la suite des expériences a été conduite sur 100 régions échantillonnées, au lieu de 10. Le premier critère évalué est l'influence de la longueur des régions.

6.2.1 Influence de la taille de la région génomique

Je présente ici les résultats pour une profondeur de séquençage fixée à 100x, avec une taille de région variable :

- 100 positions de départ
- profondeur de séquençage : 100x
- outils : MUSCLE, MAFFT, POA, sPOA, abPOA, Kalign, Kalign3

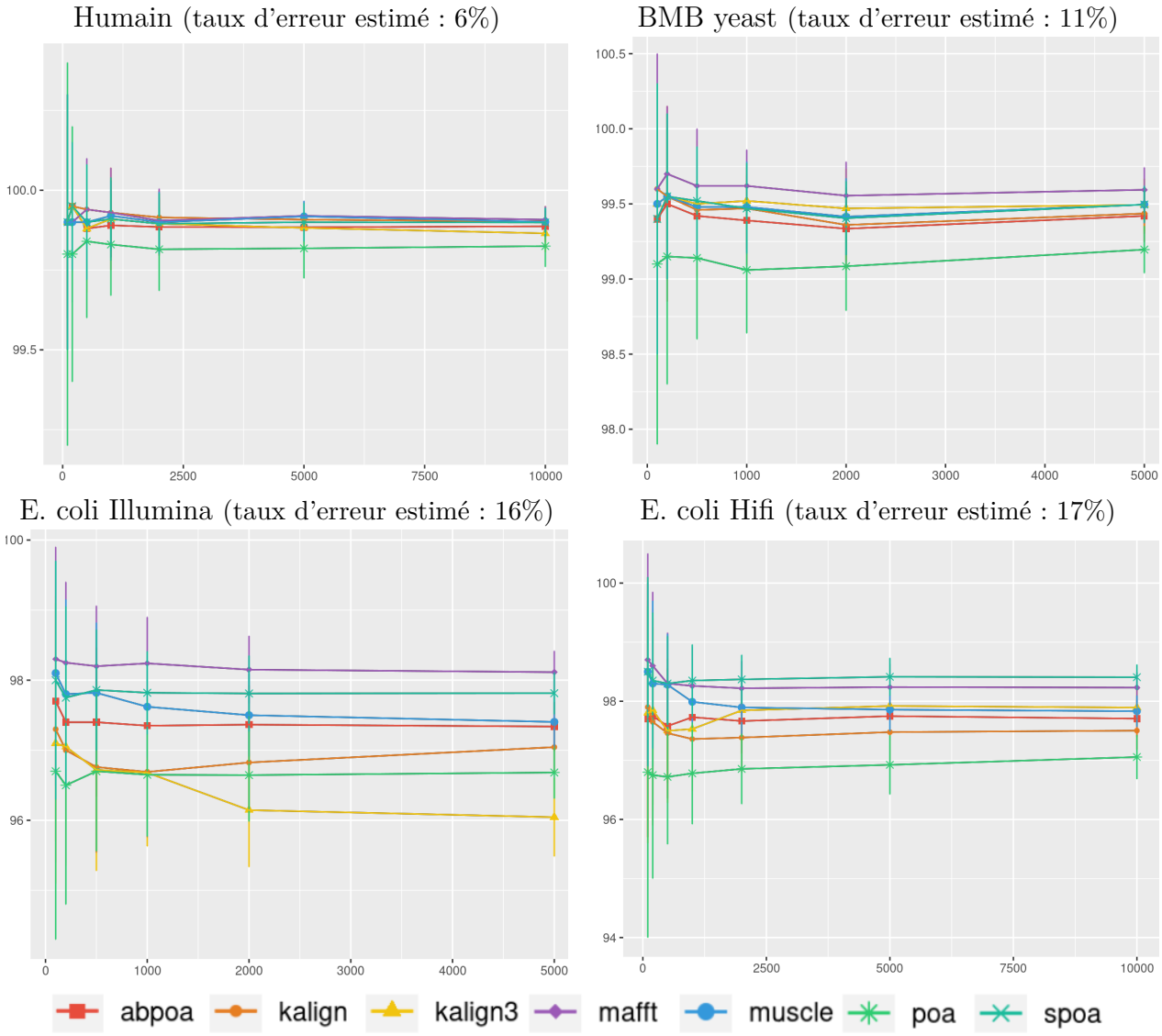


FIGURE 6.1 – Influence de la taille de la région sur le taux d'identité avec une profondeur de 100x sur 100 régions pour chaque jeu de données. L'axe des abscisses représente la longueur de la région (en bases) tandis que l'axe des ordonnées indique le pourcentage d'identité entre les séquences consensus et la référence. Les pourcentages d'identité moyens et les écarts types sont représentés.

— tailles des régions : 100pb, 200pb, 500pb, 1000pb, 2000pb, 5000pb, 10000pb
— seuil pour la séquence consensus : 50%
— jeux de données : tous les jeux réels

La figure 6.1 montre les taux d'identité obtenus. La conclusion est que la taille de la région a peu d'impact sur le taux d'identité. Dans la plupart des expériences, la différence des taux d'identité entre les différentes tailles pour un outil donné est inférieure à 1%. J'ai également testé les autres profondeurs de séquençage, pour une observation analogue.

Sur la base de ces résultats, j'ai décidé de fixer la taille de la région à 500pb pour la suite des

expériences.

— 100 positions de départ
— profondeur de séquençage : 10x, 20x, 30x, 45x, 50x, 60x, 100x, 150x, 200x
— outils : MUSCLE, MAFFT, POA, sPOA, abPOA, Kalign, Kalign3
— tailles des régions : 500pb
— seuil pour la séquence consensus : 50%
— jeux de données : tous les jeux réels

6.2.2 Influence de la profondeur de séquençage

La figure 6.2 montre l'évolution du taux d'identité pour les séquences consensus en fonction de la profondeur de séquençage. Dans le cas de reads avec un faible taux d'erreur, tels que ceux issus du jeu Human, l'influence est négligeable. Les trois autres jeux de données, qui ont un taux d'erreur compris entre 10% et 17%, font apparaître que la profondeur de séquençage est un facteur qui pèse de manière sensible sur la qualité de la séquence consensus. Dans l'ensemble, on observe que les outils fournissent de meilleurs résultats avec une profondeur plus élevée. Cependant, pour certains outils (famille POA et MUSCLE), le meilleur consensus n'est pas obtenu avec la profondeur la plus élevée. Au contraire, d'autres outils semblent constamment améliorer leur qualité avec une profondeur plus élevée (Kalign, Kalign3, et MAFFT). Certains outils sont très efficaces avec une faible profondeur (POA, sPOA), tandis que d'autres nécessitent systématiquement une plus grande profondeur pour obtenir une grande précision, comme Kalign, Kalign3 et MAFFT. Cela implique qu'aucun outil n'est universellement optimal. Le meilleur outil dépend de la profondeur disponible. L'écart-type relativement élevé (environ 2 % dans la plupart des cas) souligne une variance importante de la précision entre les différentes régions testées.

6.2.3 Influence du type d'erreur de séquençage

Étant donné que le séquençage de troisième génération produit des reads avec un taux d'erreur variable, il est important d'évaluer l'impact du taux d'erreur sur la qualité de l'alignement. Les résultats des sous-sections précédentes montrent, sans surprise, que le taux d'erreur impacte négativement le pourcentage d'identité de la séquence consensus : un taux d'erreur de séquençage plus faible conduit à un consensus plus précis.

Pour aller plus loin dans cette analyse, avec un intervalle de taux d'erreurs plus étendu et en contrôlant

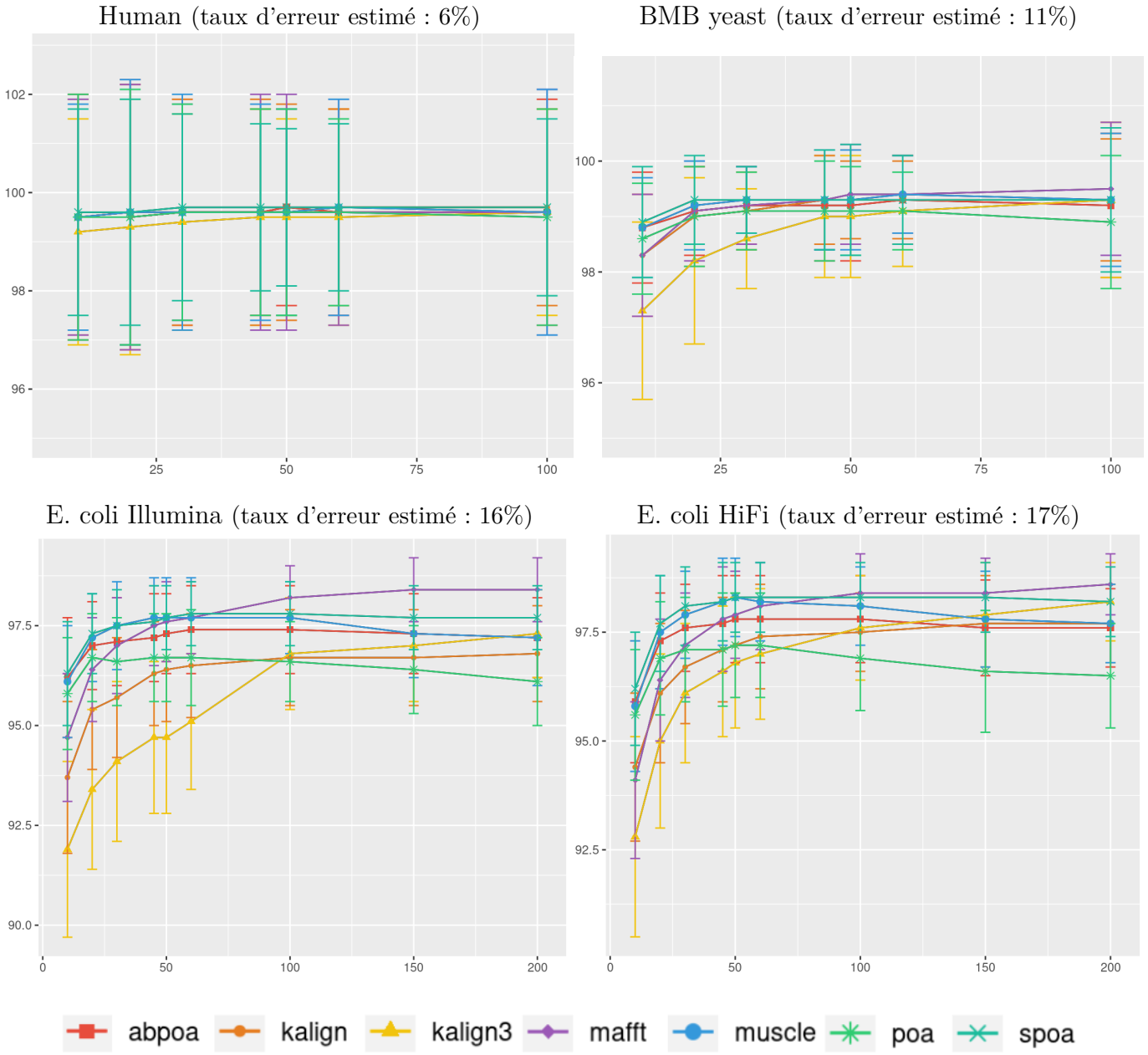


FIGURE 6.2 – Influence de la profondeur de séquençage sur le taux d'identité sur 100 régions de longueur 500pb pour chaque jeu de données. L'axe des abscisses représente la profondeur de séquençage, tandis que l'axe des ordonnées indique le pourcentage d'identité entre les séquences de consensus et la référence. Les figures montrent le taux d'identité moyen du consensus ainsi que l'écart-type.

le type des erreurs (insertion, délétion ou substitution), j'ai eu recours aux données simulées décrites en section 5.1.3. Les premiers résultats concernent les jeux simulés de type MIX, qui reprennent un profil d'erreur fidèle au séquençage ONT, et sont visibles en figure 6.3. Sur ces données simulées, tous les outils atteignent un consensus extrêmement précis quand le taux d'erreur est inférieur à 10%. Avec des taux d'erreur plus élevés, la précision de tous les outils chute très rapidement, à l'exception de POA qui résiste mieux à des taux d'erreur élevés. Ce résultat peut expliquer que le choix de POA

était pertinent pour les taux d'erreur très élevés, propres aux premiers séquençages ONT.

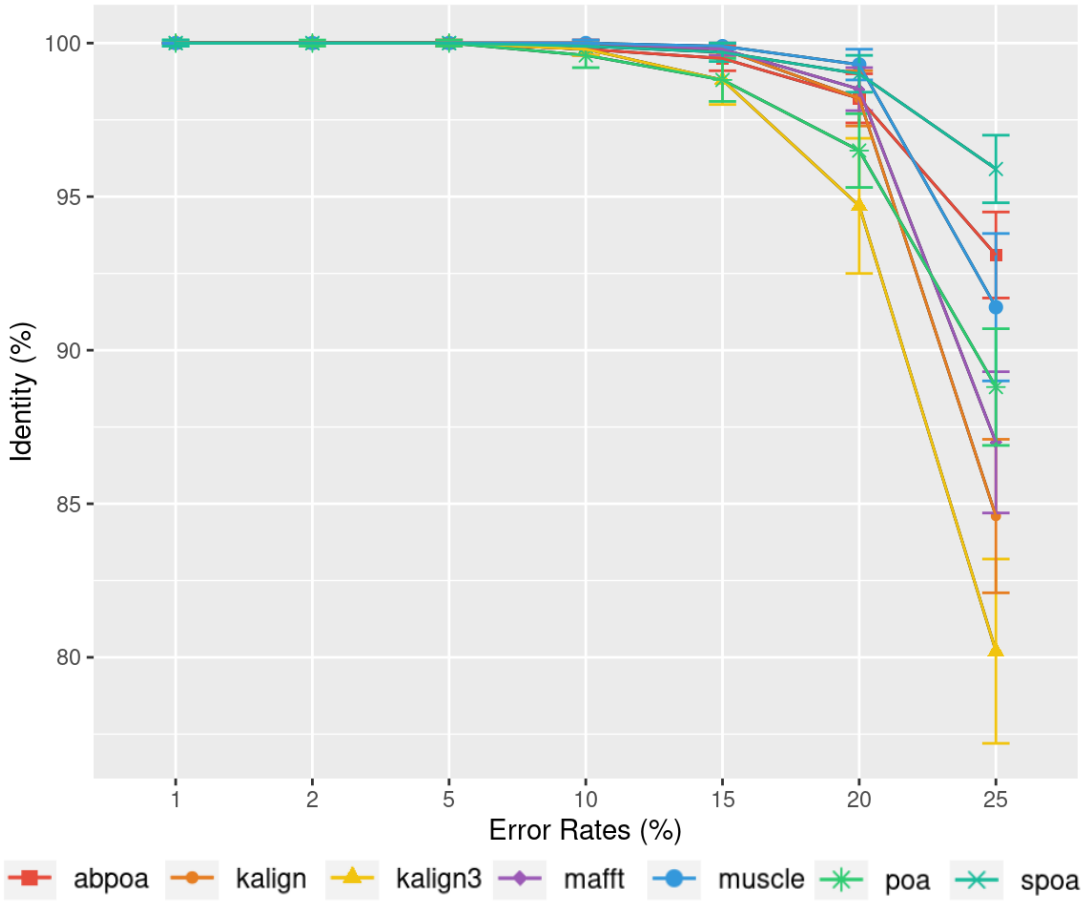


FIGURE 6.3 – Influence du taux d'erreur sur le taux d'identité des consensus sur un jeu de données *E.coli* simulé de type MIX pour 100 régions distinctes de taille 500 et d'une profondeur de 45x. Le taux d'identité moyen et l'écart-type sont affichés en fonction du taux d'erreur utilisé.

Étant donné que les différentes techniques de séquençage de troisième génération présentent également différents types d'erreurs (principalement des insertions pour les reads PacBio et principalement des délétions pour les reads ONT), j'ai voulu évaluer la manière dont les différents outils gèrent les différents types d'erreurs. La figure 6.4 montre l'évolution du taux d'identité des consensus en fonction du taux d'erreur en isolant chaque type d'erreur :

— SUB, toutes les erreurs sont des substitutions,
— INS, toutes les erreurs sont des insertions,
— DEL, toutes les erreurs sont des délétions.

On observe que le type d'erreur a un impact important sur la précision du consensus. Les substitutions sont faciles à corriger et la plupart des outils atteignent un consensus presque parfait sur ce type d'erreur. C'est un résultat attendu puisque la plupart de ces méthodes ont était conçues pour ce type de configuration. Une exception notable est la famille POA, qui a du mal à gérer les jeux SUB avec un

taux d'erreur élevé. Les erreurs d'insertion/délétion sont plus difficiles à corriger puisqu'elles créent un décalage entre les séquences. Les résultats obtenus sont assez similaires à ceux de la figure 6.3, à l'exception de sPOA qui est moins performant sur les jeux de données de délétion. Sans surprise, le jeu de données DEL est plus difficile à traiter que le jeu de données INS, et produit une qualité globale inférieure.

6.3 Évaluation de l'utilisation de la mémoire et du temps d'exécution

6.3.1 Influence de la taille de la région

Comme observé dans la sous-section 6.2, l'influence de la taille des séquences sur la qualité de l'alignement multiple est négligeable. Cependant, cette taille a un impact significatif sur la mémoire et le temps d'exécution pour la plupart des algorithmes. La figure 6.5 représente l'évolution de la consommation de la mémoire et du temps d'exécution en fonction de la taille de la région. Cela montre que l'utilisation de la mémoire de certains outils croît linéairement avec la taille des séquences (kalign3, kalign, MUSCLE), tandis que pour d'autres outils elle semble croître de manière super-linéaire (POA, MAFFT) ou même quadratique (abPOA, sPOA). Le temps d'exécution présente également des motifs différents selon les outils, certains semblant presque linéaires (sPOA, abPOA, kalign, kalign3) tandis que d'autres affichent clairement une croissance super-linéaire (MUSCLE, POA, MAFFT). En pratique, tant pour l'utilisation de la mémoire que pour le temps d'exécution, on observe une divergence de performance de plusieurs ordres de grandeur entre les outils testés.

Ces observations tendent à valider le choix des travaux précédents [Mor+21 ; Mar+20] de s'appuyer sur une stratégie de partition pour construire des alignement multiples à partir de multiples séquences courtes plutôt que longues, car la taille de la région a un impact important sur l'utilisation des ressources et n'a guère d'impact sur la précision du consensus.

6.3.2 Influence de la profondeur de séquençage

La figure 6.6 montre l'impact de la profondeur sur le temps d'exécution et la consommation de mémoire. On observe que la profondeur a un faible impact sur l'utilisation de la mémoire ou le temps d'exécution pour la plupart des outils, à l'exception de MUSCLE et POA.

6.3.3 Influence du taux d'erreur de séquençage

La figure 6.7 montre que le taux d'erreur a un faible impact sur les performances en termes de temps et de mémoire sur l'ensemble des outils, à l'exception de POA et Muscle, qui voit leur temps d'exécution augmenter avec le taux d'erreur.

6.4 Les cas des génomes diploïdes

Dans les sections précédentes, l'analyse a porté sur des données issues de génomes haploïdes, ou avec un taux d'hétérozygotie faible, en ce qui concerne le génome humain.

Pour traiter le cas de génomes polyploïdes, et en particulier ici le cas d'un génome diploïde, il faut disposer d'une référence contrôlée, où les positions des SNP hétérozygotes sont connues. Pour ce faire, j'ai construit une référence diploïde synthétique en combinant deux références haplotypes fiables et proches, grâce au code IUPAC (voir figure 4.4 dans le chapitre 4 pour le code IUPAC). Cela permet de faire la distinction entre les erreurs d'alignements et les variants du génome diploïde.

6.4.1 Construction d'un génome de référence synthétique hétérozygote

Inspirée par les expériences de l'article nPhase [Abo+21], j'ai combiné des données provenant de deux souches diploïdes homozygotes de *Saccharomyces cerevisiae* pour simuler un génome de levure hétérozygote. Ce nouveau jeu de données a déja été présenté dans la section 5.1.2. Les séquences de référence pour le génome diploïde ont été créées en alignant les contigs des deux souches à l'aide de Minimap2. Les alignements ont été affinés avec Exonerate, conduisant à une séquence consensus où les symboles IUPAC indiquent les polymorphismes hétérozygotes entre les deux allèles.

6.4.2 Plan expérimental

J'ai sélectionné 100 régions d'une longueur de 500pb, qui contiennent collectivement un total de 290 variants entre les deux allèles. Pour chaque régions, j'ai exécuté le pipeline `MSA_Limit` avec un seuil de 70% pour le calcul de la séquence consensus. Cette valeur de seuil est adaptée au cas diploïde. Le seuil par défaut utilisé pour les cas haploïdes, 50%, ne permettrait de récupérer qu'un seul des haplotypes, puisqu'il est attendu qu'un variant soit présent dans 50% des reads dans le cas diploïde.

Dans ce contexte, les symboles IUPAC de la séquences consensus devraient représenter les variations entre les deux allèles. L'expérience a été menée à trois profondeurs de séquençage différentes : 20x, 50x et 100x.

6.4.3 Résultats

L'objectif était d'évaluer la capacité des outils à identifier les variants à partir des jeux de données. J'ai regardé si les 290 variants du génome de référence étaient présents dans les séquences consensus générées par chaque outil, en étant représentés par le caractère IUPAC correspondant. J'ai également identifié les caractères IUPAC ne correspondant pas à des variations alléliques. Les résultats sont présentés dans le tableau 6.2 et la figure 6.8. Le taux de faux positif n'est clairement pas négligeable ce qui indique qu'aucun outil n'arrive à gérer le cas diploïde.

6.5 Discussion

Les expériences ont permis de dégager plusieurs enseignements. Tout d'abord, Clustal Omega et T-Coffee semblent être les moins adaptés parmi les outils testés. Comme prévu, les performances des autres outils, à savoir MAFFT, MUSCLE, KALIGN, KALIGN3, abPOA, sPOA et POA, s'améliorent à mesure que le taux d'erreur de séquençage diminue. Ces outils affichent de très bonnes performances avec les données de séquençage récente, avec un taux d'erreur qui se situe autour de 5%, comme observé dans le jeu de données humain.

Une analyse plus approfondie révèle que, dans tous les scénarios, POA peut être remplacé avantageusement par ses dérivés, abPOA ou sPOA. KALIGN3, en comparaison avec son prédécesseur KALIGN, semble moins convaincant. MAFFT apparaît comme un choix équilibré. MUSCLE donne de bons résultats en terme de qualité, mais ses performances sont contrebalancées par ses exigences en matière de coût de calcul.

Il est intéressant de noter que l'influence de la profondeur de séquençage n'est pas uniforme d'un outil à l'autre. Les outils de la lignée POA sont recommandés pour les ensembles de données dont la profondeur est faible (10x ou 20x), tandis que MAFFT, Kalign et Kalign3 excellent avec les ensembles de données dont la profondeur est supérieure à 50x.

Concernant le génome diploïde de la section 6.4, les résultats montrent qu'aucun outil ne fournit une solution complète, ce qui souligne la nécessité de développer des outils spécialisés.

Rappel								
profondeur	abPOA	kalign	kalign3	mafft	muscle	POA	sPOA	T-Coffee
20x	0.72	0.76	0.76	0.80	0.81	0.73	0.69	0.78
50x	0.87	0.92	0.92	0.93	0.94	0.88	0.85	0.93
100x	0.90	0.96	0.95	0.96	0.93	0.92	0.86	0.97

Précision								
	abPOA	kalign	kalign3	mafft	muscle	POA	sPOA	T-coffee
20x	0.34	0.22	0.10	0.18	0.28	0.26	0.35	0.43
50x	0.44	0.35	0.19	0.29	0.42	0.30	0.46	0.59
100x	0.43	0.43	0.33	0.40	0.45	0.26	0.44	0.64

TABLE 6.2 – Rappel et précision des variants pour la levure diploïde pour les profondeurs de séquençage 20x, 50x et 100x. Pour chaque outil, le rappel est calculé comme le nombre de symboles IUPAC dans la séquence consensus correspondant à des variants, divisé par le nombre total de variants (290). La précision est le nombre de symboles IUPAC dans la séquence consensus correspondant à des variants par le nombre total de symboles IUPAC dans la séquence consensus.

Une question intrigante est de savoir si les différents outils testés sont complémentaires, c'est-à-dire s'ils se trompent à des positions distinctes des long reads, ou s'ils échouent aux mêmes endroits. Si les outils se trompent à différentes positions, une approche combinée pourrait améliorer la précision. Pour explorer cette question, j'ai introduit un "méta-consensus" construit à partir des séquences de consensus de divers outils dans la section 4.1. La précision du "méta-consensus" a ensuite été comparée aux séquences de consensus spécifiques à chaque outil. Les résultats, détaillés dans le tableau 6.3, révèlent que si le méta-consensus est plus performant dans le cas de faibles profondeurs de séquençage, telles que 10x. Pour des profondeurs égales à 20x ou 50x, le gain est faible, voire inexistant. Par contre, le méta-consensus faire toujours mieux que le moins bon des outils, quelque soit la profondeur. J'ai également comparé le méta-consensus aux résultats produits par T-Coffee. Nous avons dit en section 6.1, à partir d'un petit échantillonnage, que T-Coffee donnait des résultats de très bonne qualité, mais souffrait d'une consommation mémoire et d'un temps d'exécution excessifs. Faisant abstraction de ces limites, il apparaît que T-Coffee est meilleur que l'approche combinée.

E.coli HiFi (taux d'erreur de séquençage estimé : 17.28%)

Profondeur	Meilleur outil				Pire outil	T-coffee			
	>	=	<	Δ moyenne	Δ moyenne	>	=	<	Δ moyenne
10	100	0	0	1.76	5.80	0	1	99	-1.32
20	72	11	17	0.52	3.88	0	0	100	-1.29
50	50	32	18	0.14	2.57	0	0	100	-1.08
100	59	13	28	0.16	2.43	0	0	100	-0.93

E.coli Illumina (taux d'erreur de séquençage estimé : 16.38%)

Profondeur	Meilleur outil				Pire outil	T-coffee			
	>	=	<	Δ moyenne	Δ moyenne	>	=	<	Δ moyenne
10	95	4	1	1.17	6.07	0	0	100	-2.10
20	73	14	13	0.44	4.77	0	0	100	-1.87
50	28	28	44	-0.10	3.34	0	0	100	-2.00
100	18	17	65	-0.23	2.08	0	0	100	-1.83

BMB yeast (taux d'erreur de séquençage estimé : 10.8%)

Profondeur	Meilleur outil				Pire outil	T-coffee			
	>	=	<	Δ moyenne	Δ moyenne	>	=	<	Δ moyenne
10	51	29	20	0.17	2.06	0	16	84	-0.64
20	8	39	53	-0.13	1.12	0	8	92	-0.68
50	2	39	59	-0.15	0.57	0	14	86	-0.59
100	0	31	55	-0.16	0.62	0	13	73	-0.58

Human (taux d'erreur de séquençage estimé : 6.6%)

Profondeur	Meilleur outil				Pire outil	T-coffee			
	>	=	<	Δ moyenne	Δ moyenne	>	=	<	Δ moyenne
10	15	72	12	0	0.55	0	63	36	-0.32
20	0	73	26	-0.07	0.39	0	57	42	-0.38
50	0	72	27	-0.09	0.26	0	58	41	-0.32
100	0	72	24	-0.08	0.22	0	61	35	-0.31

Levure diploïde

Profondeur	Meilleur outil				Pire outil	T-coffee			
	>	=	<	Δ moyenne	Δ moyenne	>	=	<	Δ moyenne
10	54	24	22	0.18	1.99	0	1	99	-1.08
20	4	26	70	-0.18	1.02	0	0	100	-1.27
50	3	30	67	-0.27	0.57	0	0	100	-1.29
100	4	20	76	-0.30	0.57	0	0	100	-1.19

TABLE 6.3 – Comparaison du méta-consensus avec les meilleures et les pires séquences consensus obtenues avec différents outils et avec T-Coffee à différentes profondeurs. Est indiquée la fréquence à laquelle le méta-consensus fournit une identité supérieure (>), égale (=) ou inférieure (<) par rapport à l'outil le plus performant pour une expérience donnée, accompagnée de la différence moyenne observée. De même, je présente les performances du méta-consensus par rapport à l'outil le moins performant et à T-Coffee.

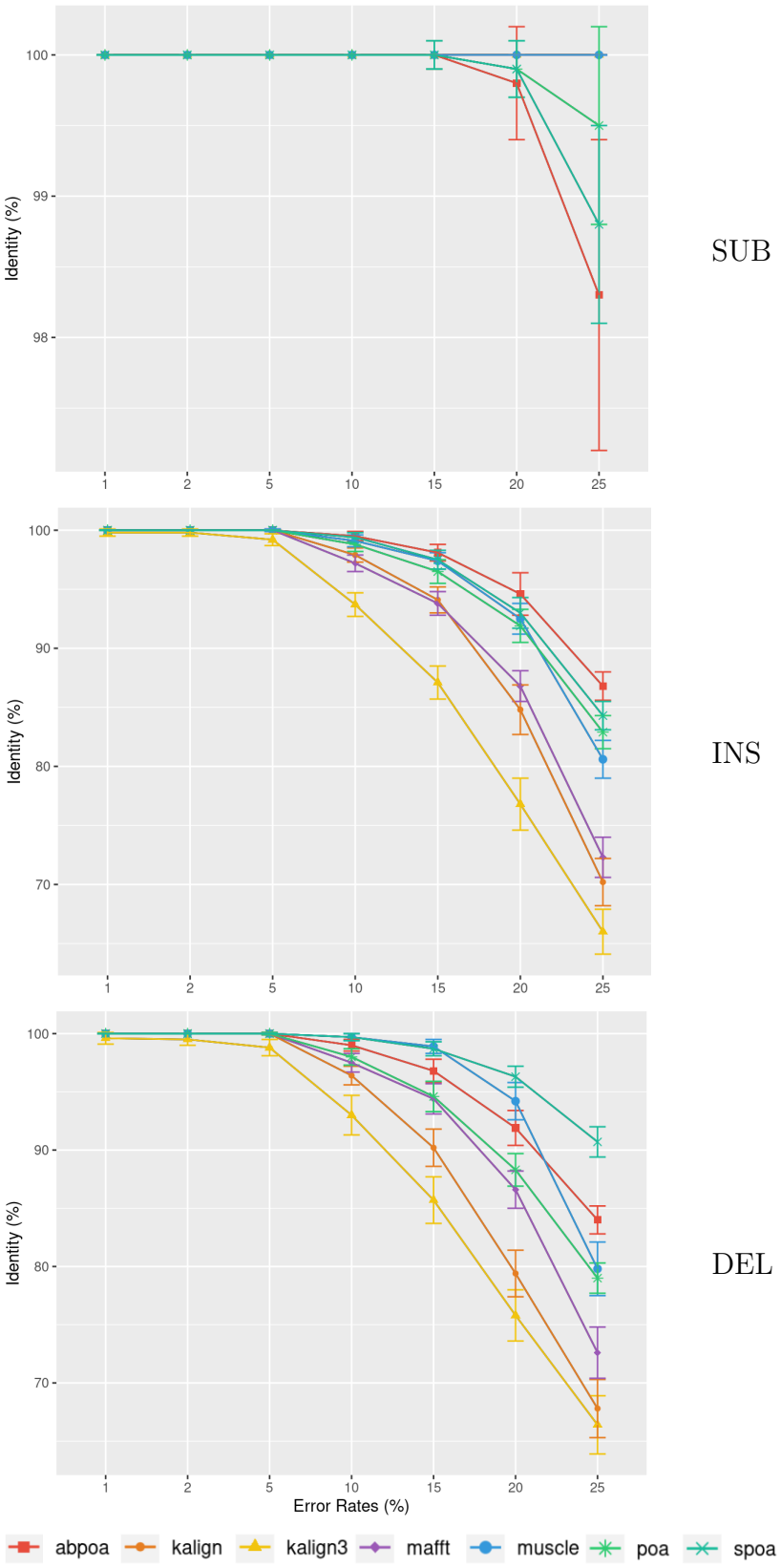


FIGURE 6.4 – Influence du taux d'erreur sur le taux d'identité du consensus sur un ensemble de données E.coli simulé sur 100 régions distinctes.de taille 500 et d'une profondeur de 45x avec un seul type d'erreur, respectivement la substitution (SUB), l'insertion (INS) et la délétion (DEL). Le taux d'identité moyen et l'écart-type sont affichés en fonction du taux d'erreur utilisé.

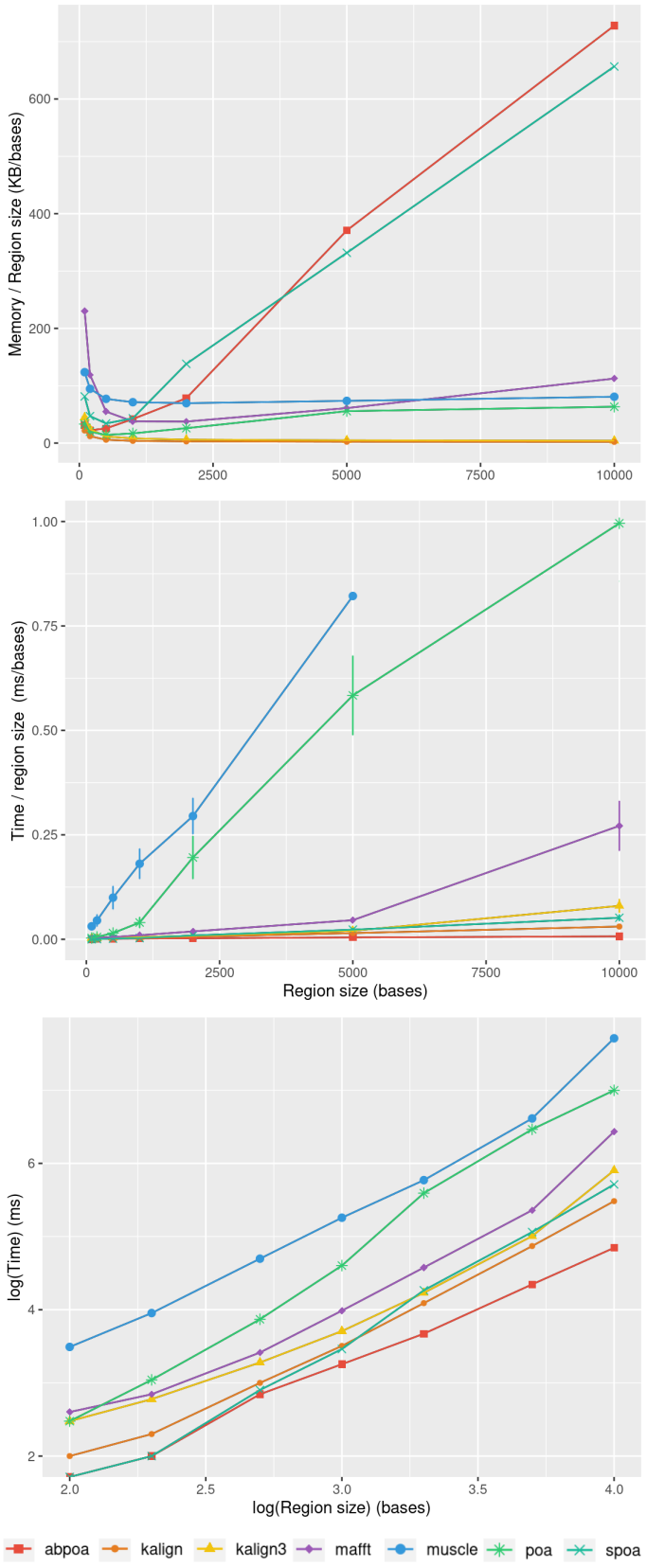


FIGURE 6.5 – Influence de la taille de la région sur l'utilisation de la mémoire et du temps CPU pour le jeu de données E.coli HiFi avec une profondeur de 100x sur 100 régions distinctes. La moyenne de l'utilisation de la mémoire divisée par la taille de la région (en haut) et le temps CPU moyen (au milieu) sont affichés en fonction de la taille de la région. On affiche aussi le temps avec une échelle logarithmique (en bas) pour mieux discerner les courbes. L'écart-type est affiché en noir.

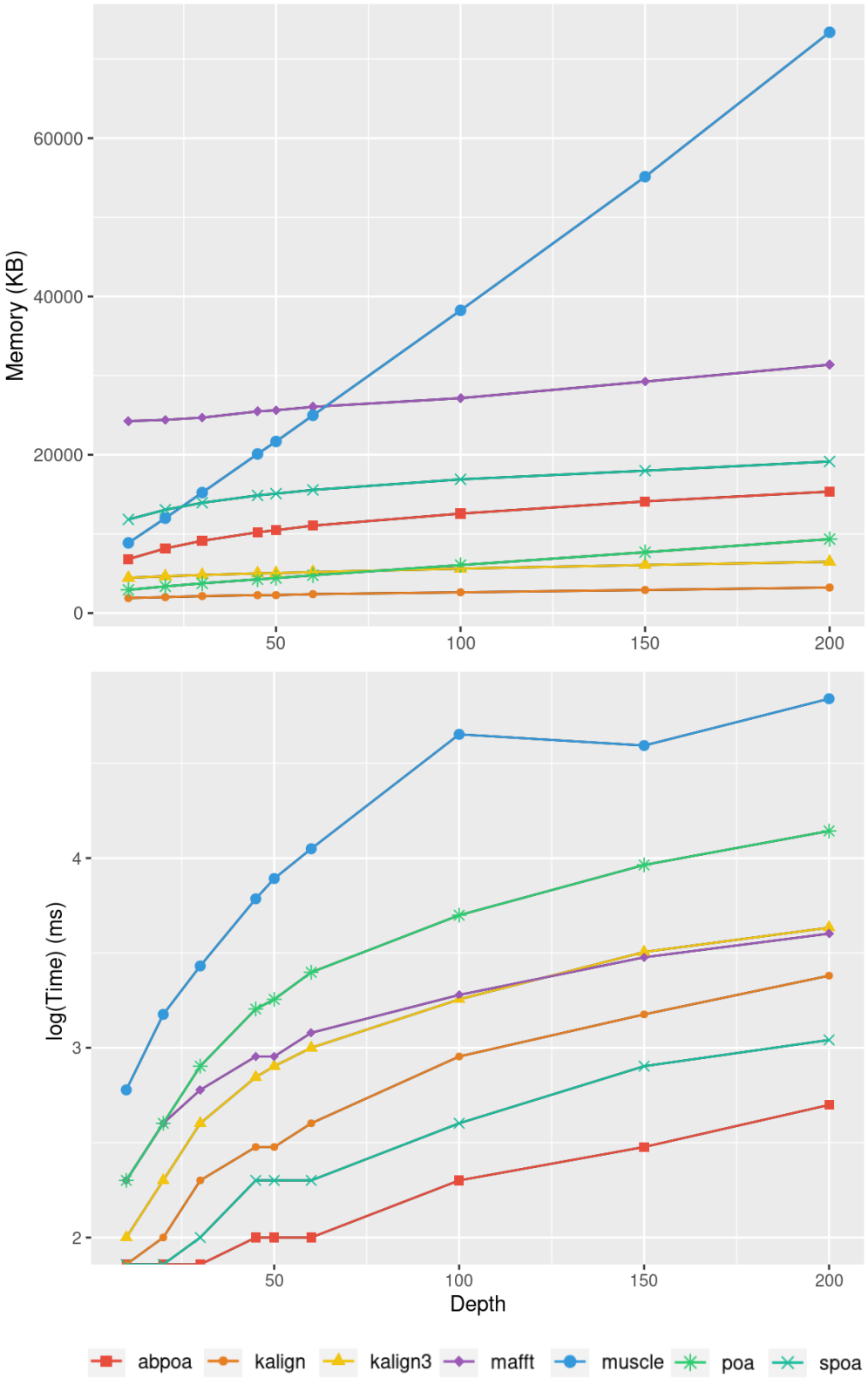


FIGURE 6.6 – Influence de la profondeur de séquençage sur l'utilisation de la mémoire et le temps d'exécution pour le jeu de données *E.coli* HiFi sur 100 régions distinctes de taille 500. Le temps d'exécution moyen est affiché pour chaque profondeur de séquençage.

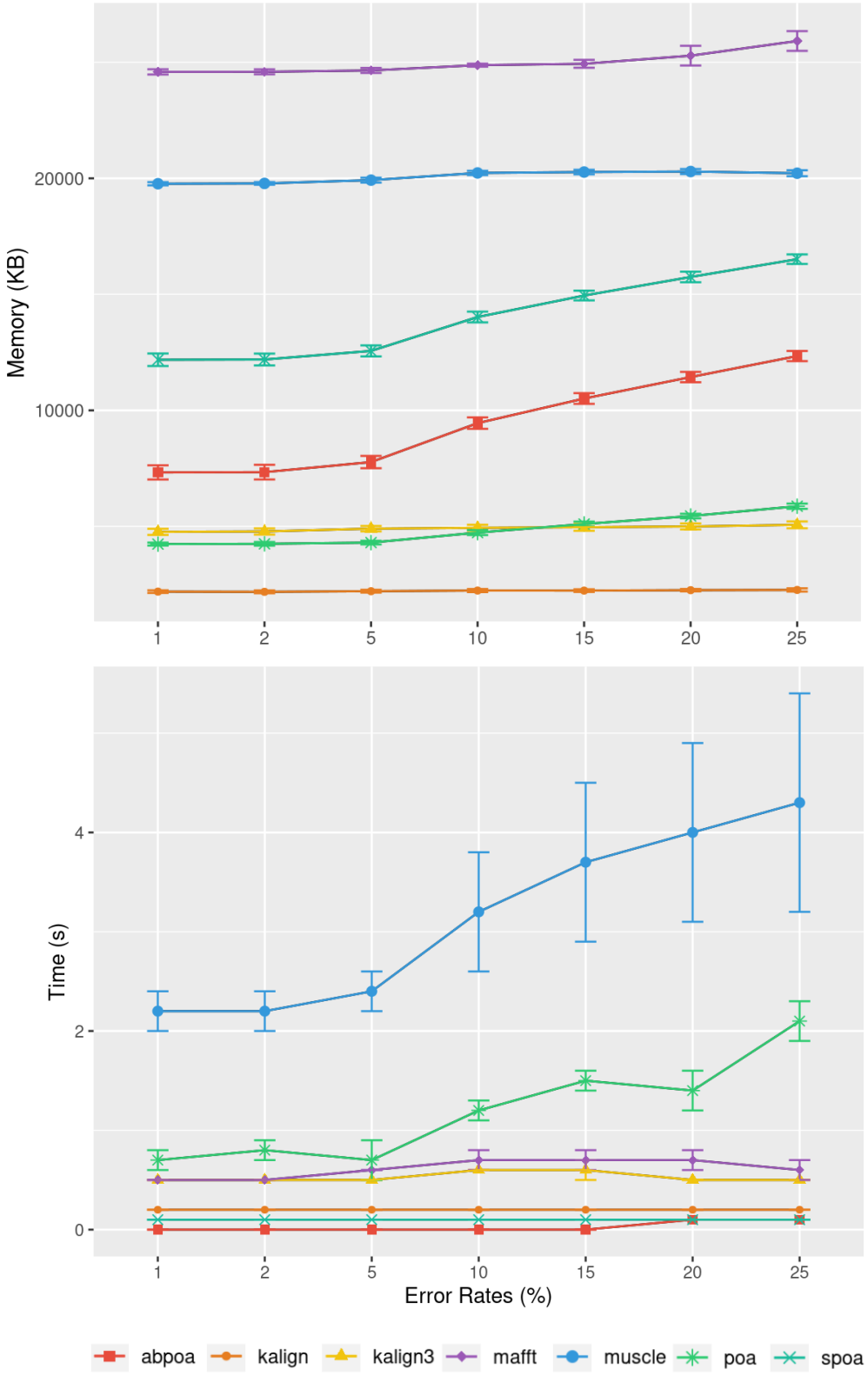


FIGURE 6.7 – Influence du taux d'erreur de séquençage sur l'utilisation de la mémoire (en haut) et le temps d'exécution (en bas) sur un ensemble de jeux de données *E.coli* de type MIX simulés sur 100 régions distinctes de taille 500 et d'une profondeur de 45x. Le temps moyen, l'utilisation moyenne de la mémoire et leurs écart-types respectifs sont affichés en fonction du taux d'erreur utilisé.

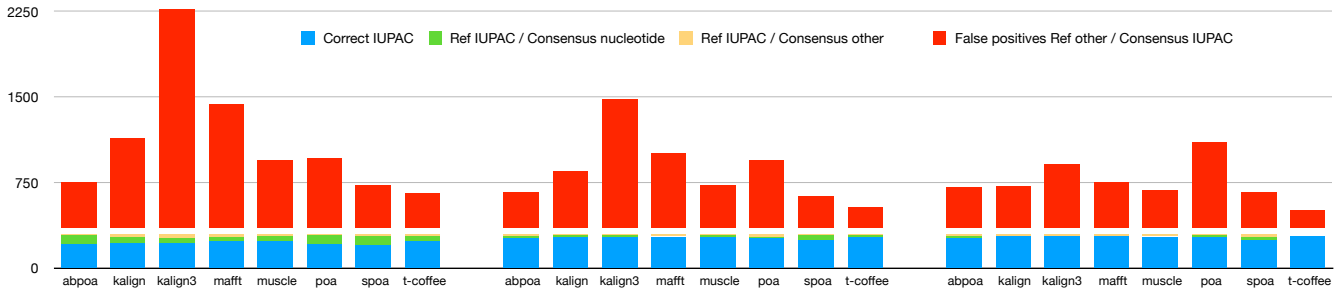


FIGURE 6.8 – Performances qualitatives des différents outils dans le cas diploïde avec des profondeurs de séquençage de 20x (à gauche), 50x (au milieu) et 100x (à droite).

Chapitre 7

Discussion autour de la thèse

Ce chapitre présente divers sujets à débattre autour de la thèse. La section 7.1 détaille certains points de l'analyse. La section 7.2 revient sur le méta-consensus et son potentiel. La section 7.3 présente deux idées approfondies pour mieux gérer des génomes diploïdes. La dernière section est consacrée au travail effectué en dehors de l'activité de recherche.

7.1 Limites du pipeline d'analyse et pistes infructueuses

Cette section est consacrée à des points de discussion centrés sur le travail réalisé autour de l'évaluation des alignements multiples. J'aborderai d'abord le paramétrage des différents outils testés ainsi que la question de la reproductibilité, puis la difficulté à trouver de bonnes données à tester. Je mentionnerai pour finir une tentative avortée d'utiliser la phylogénie dans l'étude.

7.1.1 Paramétrage des outils et ajout de nouveaux outils

Lors de l'analyse présentée tout au long de cette thèse, les différents outils d'alignement multiple utilisés l'ont été dans leurs conditions par défaut. Hors, ces outils sont paramétrables sur de nombreux points et il est certain que leurs résultats sont dépendants de leur paramétrage. Cependant, le développement de toute cette analyse ayant déjà était conséquent, tester chaque outils avec plusieurs paramétrages différents n'était pas possible pour moi.

Afin que d'autres puissent poursuivre cette étude s'ils le désirent, j'ai laissé dans le fichier ReadMe

(disponible en annexe) de présentation du pipeline sur le dépôt les instructions nécessaires pour ajouter un nouvel outil d'alignement à analyser. Ces instructions peuvent aussi être utilisées pour tester les outils avec un autre paramétrage. De ce fait, l'une des utilisation de ce pipeline est aussi de permettre à l'utilisateur de chercher le meilleur paramétrage pour un jeu de données.

7.1.2 Reproductibilité

Durant cette thèse, j'ai mis l'accent sur la reproductibilité comme présenté dans le chapitre 4. C'est un travail qui est chronophage, mais d'une importance capitale dans le milieu de la recherche, et notamment dans le cadre d'une analyse comparative. Cela a motivé l'adoption de Snakemake associé à des environnements Conda de le rendre le plus reproductible possible, mais aussi le plus facile d'installation. Au cours de ma thèse, quelques personnes ont dû installer mon pipeline et ont rencontré peu de difficulté. Pourtant, malgré tous mes efforts, je me suis rendu compte que je n'étais pas reproductible entièrement. Deux problèmes se sont présentés, venant des outils testés. L'outil abPOA refuse sur certaine machine de fonctionner dans l'environnement Conda et nécessite d'être installé localement. L'outil T-coffee n'est quand à lui plus compatible avec les versions de linux supérieures à 18.

Avec Snakemake, j'ai garanti de pouvoir reproduire les expériences avec l'ensemble de leurs paramètres. Avec Conda, j'ai pu garantir de conserver les outils dans leurs bonnes versions pour ce projet. Afin de résoudre le problème de compatibilité entre les outils et le système d'exploitation, il faudrait conserver l'environnement de travail. Cela serait possible en utilisant des logiciels comme Docker ou Singularity, qui permettent d'encapsuler un projet dans un container qui contient l'environnement et toutes les dépendances. Malheureusement, par manque de temps, je n'ai pas pu me pencher sur cette solution.

7.1.3 Difficulté de trouver des jeux de données

Si le développement d'un pipeline d'analyse reproductible et l'analyse des résultats sont des taches qui ont déjà pris beaucoup de temps, ce qui a été le plus chronophage a été, sans conteste, de créer un benchmark pour ce projet. Trouver des jeux de données avec une référence la plus correcte possible s'est avéré une tâche ardue, voire impossible. Quand la décision a été prise de créer moi-même les génomes de référence par assemblage, il fallait encore trouver les jeux de données qui rendait cela possible et éviter de biaiser la référence en utilisant l'alignement multiple dans son processus de création. Réussir à obtenir une bonne références pour le diploïde s'est avéré tellement compliqué que

j'ai préféré simuler un diploïde à partir de deux jeux homozygotes. La création des jeux de données a aussi donné lieu à la création d'un workflow pour pouvoir reproduire le benchmmark ce qui a demandé beaucoup de travail.

7.1.4 Tentative d'utiliser la phylogénie

Lors de l'analyse des résultats des différents outils, je me suis demandée comment je pouvais les comparer entre eux, en plus des métriques calculés, afin de voir si certains outils avaient des comportements analogues. L'idée est venue d'essayer d'utiliser des algorithmes issus de la phylogénie sur les séquences consensus issues des différents outils. Ce domaine permet normalement d'étudier les liens de parentés entre différentes espèces. La phylogénétique moléculaire, en particulier, permet de travailler avec des séquences d'ADN en regroupant les espèces à partir de leur similitude en séquence. L'idée était d'utiliser des outils dédiés à la phylogénie moléculaire pour créer un arbre phylogénique avec les séquences consensus et la référence. Cela aurait pu nous permettre de visualiser la distance entre les séquences et la référence et si certains outils avaient tendance à se regrouper ou non. Malheureusement, si l'idée était intéressante, la trop grande similitude entre les séquences ne permettent pas d'avoir un arbre phylogénétique correct. Il aurait peut-être été possible d'améliorer cela en affinant les paramètres, mais j'ai finalement dû abandonner cette idée.

7.2 Méta-consensus

Dans le chapitre 4 est présenté le méta-consensus qui est un consensus réalisé à partir des séquences consensus, elles-même obtenues à partir des alignements multiples produits par les différents outils. Les résultats présentés dans le chapitre 6 montrent que le méta-consensus n'est pas forcément meilleur que les consensus qui ont permis de le construire. Cependant, cette idée de méta-consensus pourrait encore être approfondie. Le point le plus important porte sur le choix des outils utilisés. J'avais déjà fait le choix d'exclure Clustal Omega et Tcoffee, le premier pour la qualité, le second pour le temps. On pourrait étudié si on pourrait améliorer la qualité du méta-consensus en changeant les outils sélectionnés, ou en donnant des poids différents à certains outils. C'est un projet qui a était développé pendant un stage de plusieurs mois visant à tester différentes combinaison d'outils. (dépôt : https ://gitlab.univ-lille.fr/flavien.lihouck.etu/mc_msa)

7.3 Gérer le cas diploïde

Comme on peut le constater dans les chapitres précédents, la diploïdie, et par extension les autres polyploïdies, sont plus compliquées à traiter que le cas haploïde. En réfléchissant à la manière de traiter au mieux ces cas, j'ai développé deux pistes de réflexion que je vous présente dans cette section : une extension du code IUPAC et une méthode de phasing de reads diploïdes.

7.3.1 Étendre le code IUPAC

Le code IUPAC présenté dans la section 4.1.3 et la figure 4.4 permet de condenser en un seul caractère un ensemble de nucléotides possibles. Par exemple, on note W l'ensemble des nucléotides A et T. C'est un code particulièrement utile pour représenter la diploïdie, puisqu'il permet de conserver les deux chromosomes en une seule séquence. Si on a un A dans le premièr haplotype et un T dans le second haplotype, la séquences consensus aura donc le code W.

Cependant ce code est limité aux nucléotides. Il ne prend pas en compte le fait qu'il puisse exister un gap. Hors entre deux haplotypes, il est possible qu'il y ai eu des insertions et des délétions et non pas uniquement des substitutions. Dans cette situation, on traite souvent le gap différemment d'un nucléotide. Dans mon propre cas, dans la section 4.1.3, j'ai différencié le gap et les nucléotides. Si le gap est le plus fréquent, alors j'ai un gap dans ma séquence consensus, sinon je m'intéresse aux nucléotides et utilise le code IUPAC correspondant. En cas d'égalité entre le gap et le nucléotide majoritaire, les nucléotides l'emportent puisque j'ai préféré conserver une information plutôt qu'un manque d'informations. Si on prend un cas où on a un A dans le première l'haplotype et un gap dans le second haplotype à cause d'une délétion, on se rend vite compte que cette information ne pourra pas être conservée dans la séquence consensus. On obtiendra soit un A, soit un gap, en fonction de la répartition des reads.

Afin de résoudre ce problème, je propose d'étendre le code IUPAC pour qu'il puisse prendre en compte le gap en plus des nucléotides. Ce code, que j'ai baptisé le code IG (IUPAC + Gap), reprend la base du code IUPAC mais en ajoutant des lettres minuscules pour les cas où l'ensemble contient aussi un gap. L'ensemble du code IG est visible dans la figure 7.1. Grâce à ce code, on pourra conserver la présence d'un gap dans le consensus.

CODE	BASE
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
R	A or G
Y	C or T
S	G or C
W	A or T
K	G or T
M	A or C
B	C or G or T
D	A or G or T
H	A or C or T
V	A or C or G
N	any base
a	Adenine or -
c	Cytosine or -
g	Guanine or -
t	Thymine or -
r	A or G or -
y	C or T or -
s	G or C or -
w	A or T or -
k	G or T or -
m	A or C or -
b	C or G or T or -
d	A or G or T or -
h	A or C or T or -
v	A or C or G or -
n	any base or -
-	gap

FIGURE 7.1 – Le code IG est une extension du code IUPAC pour pouvoir prendre en compte le gap en plus des nucléotides.

7.3.2 Phasage d'un diploïde

Pour évaluer l'alignement multiple, je réalise des séquences consensus à partir de ces alignements. Or, dans le cas diploïde, ces séquences consensus contiennent pêle-mêle les deux haplotypes. Dans le but d'améliorer l'analyse avec des reads diploïdes, je me suis demandé s'il n'était pas possible de réaliser un phasing, c'est-à-dire de séparer les deux haplotypes. Pour cela, j'ai conceptualisé une méthode à partir d'un alignement multiple dont j'ai développé un prototype en C qui est disponible sur le dépôt

https ://github.com/CoralieRohmer/MSADiff.

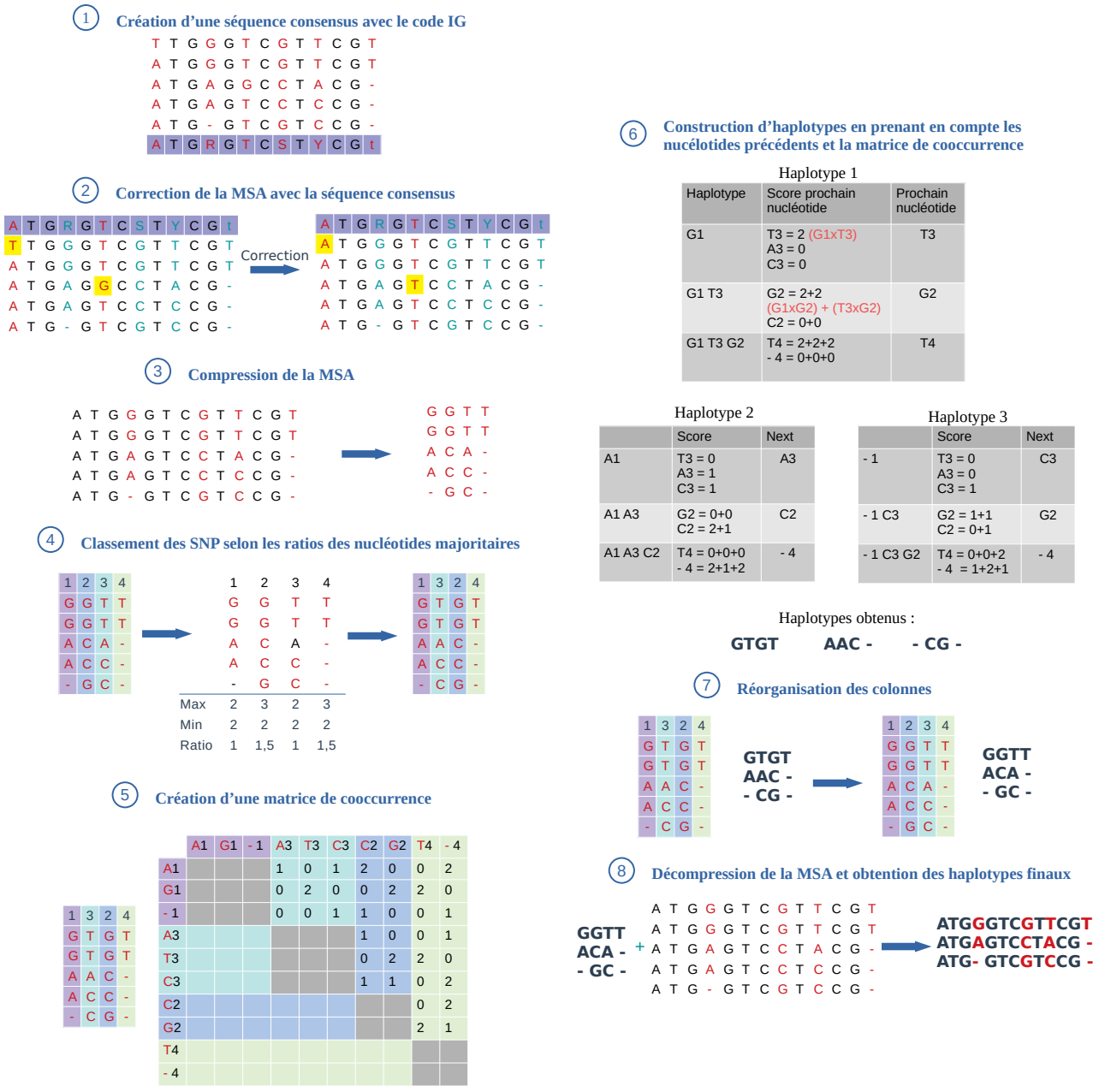


FIGURE 7.2 – Méthode de phasage d'un diploïde.

Descriptif de la méthode La méthode est présentée dans la figure 7.2. J'ai pris en entrée un alignement multiple issus d'une pile de reads, comme présenté dans le chapitre 4. L'étape 1 créée une séquence consensus à partir de l'alignement avec la même méthode que celle du chapitre 4, à l'exception du gap qui est traité comme un nucléotide normal. On obtient ainsi une séquence consensus avec le code IG présenté dans la section précédente. L'étape 2 consiste à corriger les

positions où la séquence consensus ne contient aucune ambiguïté pour éliminer les erreurs des reads. L'étape 3 compresse l'alignement multiple en éliminant les colonnes où l'information est identique pour chaque read. A partir de cette étape, je considère que chaque colonne est un SNP (Single Nucleotide Polymorphism), c'est-à-dire, un variant isolé.

L'étape 4 vise à réorganiser les colonnes pour que celles-ci soient rangées du meilleur SNP au pire SNP. Je réalise ce classement à l'aide d'un score qui est le ratio entre les fréquences des deux nucléotides les plus majoritaires. J'ai choisi ce score en partant du principe que dans un cas parfait et diploïde, on obtiendrai pour un SNP deux nucléotides avec deux fréquences à 50%. Avec un SNP parfait, on obtiendrai donc un ratio de 1. Plus les fréquences des nucléotides sont déséquilibrées, plus le ratio augmente et plus la frontière est mince entre SNP et erreur de séquençage.

L'étape 5 consiste à créer une matrice de co-occurrence entre chaque nucléotide de chaque SNP. L'étape 6 recrée les haplotypes grâce à la matrice de co-occurrence. On part des nucléotides du premier SNP (Dans l'exemple : G1, A1 et - 1). On regarde ensuite la matrice de co-occurrence pour sélectionner le nucléotide du deuxième SNP par rapport à celui d'avant (Pour G1, on prend T3 car on les trouve deux fois ensembles). On recommence cette opération en additionnant à chaque fois les co-occurrences de chaque nucléotide. Cette étape permet de crée un haplotype SNP après SNP, tout en validant chaque nouveau nucléotide avec ceux ajoutés précédemment. Cette méthode a un biais important si les premier nucléotides ajoutés sont des erreurs. En triant les SNP en amont, on réduit ces erreurs et donc le biais.

Les étapes 7 et 8 visent à réorganiser les SNP et décompresser l'alignement afin d'obtenir les haplotypes dans leur globalité.

Discussion autour de cette méthode Ce projet n'étant qu'à l'étape de prototype, il reste encore de nombreux points à améliorer mais aussi des tests avec des données réelles à effectuer. Le point d'amélioration le plus notable est celui du score pour classer les SNP. Il n'est actuellement adapté qu'au cas diploïde. J'aimerai le modifier pour qu'il puissent prendre en compte toutes les ploïdies possibles. De plus, même pour le cas diploïde, il est encore biaisé car il ne prend en compte que les deux nucléotides les plus fréquents et occulte complètement les autres nucléotides. Dans un cas où on aurait un SNP avec 40% de A, 40% de C et 20% de T et un autre SNP avec 50% de G et 50% de C, les deux SNP auraient le même score alors que ce sont deux cas différents.

Un autre problème est le fait que le nombre d'haplotypes dépend du premier SNP. On l'observe très bien sur la figure 7.2 qui est un cas pensé pour le diploïde mais où l'on obtient pourtant trois haplotypes.

Une solution au cas polyploïde pour ces deux problèmes serait de développer une méthode pour déterminer la ploïdie de l'organisme à partir de l'alignement, et d'adapter le score et le nombre d'haplotypes à obtenir en fonction de cette information.

Ce projet dans son ensemble est intéressant même s'il n'est pas achevé. D'autres membres de mon équipe de recherche se penchent déjà dessus dans le but de réutiliser cette méthode pour séparer des souches de virus à partir de données d'amplicons. De plus, des outils ont été développés dans ce cadre, notamment la compression de l'alignement qui pourrait être utile pour d'autres projets.

7.4 Activités annexes

Au cours de la thèse, l'activité de recherche n'est pas le seul travail d'un doctorant. J'ai eu l'occasion de participer à d'autres expériences en dehors du travail présenté jusqu'ici.

Comme mentionné en préambule, j'ai à coeur de transmettre mes connaissances scientifiques aux autres. Je suis donc ravie d'avoir eu l'occasion d'enseigner l'informatique pendant presque 300 heures à Polytech Lille, d'abord comme vacataire puis comme attaché temporaire d'enseignement et de recherche. Le contact avec les étudiants et le monde de l'enseignement en général à été une expérience enrichissante. J'ai même eu l'occasion de faire un cours magistral et de réaliser mon propre projet pour les étudiants. Enseigner est aussi la meilleure façon d'apprendre selon moi.

J'ai aussi eu l'occasion à trois reprises de travailler avec le programme *Informatique au féminin* en témoignant et en encadrant des collégiennes dans le but de leurs faire découvrir l'informatique lors d'un stage d'initiation, pour les convaincre que ce n'est pas un domaine réservé aux hommes. D'ailleurs dans cette même optique, je me suis engagée auprès de la commission Parité-égalité femmes-hommes de mon laboratoire pour essayer activement d'améliorer l'inclusion de toutes et tous en son sein.

J'ai aussi eu l'occasion de présenter la bioinformatique à des élèves de lycée lors des journées dédiées NSI (Numérique et Sciences Informatiques).

Enfin, j'ai eu l'occasion d'assister à des conférences notamment à plusieurs repris à JOBIM (Journées Ouvertes en Biologie, Informatique et Mathématiques) et même d'y présenter un poster en 2022 (disponible en annexe) et de présenter mes travaux lors de journées de l'ANR ASTER.

Chapitre 8

Conclusion et perspectives

Ce chapitre présente la conclusion générale de l'intégralité de cette thèse ainsi que les perspectives qui en découlent.

8.1 Conclusion

8.1.1 Remise en contexte

L'objectif de cette thèse était d'étudier les outils d'alignement multiple avec des données long reads pour vérifier que les méthodes fonctionnent bien sur ce type de données.

Après avoir présenté l'ADN et son séquençage dans le chapitre 2, j'ai présenté les caractéristiques des long reads qui sont des reads avec une taille conséquente mais qui comportent aussi un taux d'erreur plus élevé, et surtout un nouveau profil d'erreur. En effet, les erreurs sont majoritairement des insertions et des délétions, ce qui pose plus de souci que les substitutions car ces erreurs causent un décalage entre les séquences lors de l'alignement.

J'ai ensuite présenté les différents outils qui permettent de réaliser un alignement multiple de séquences d'ADN dans le chapitre 3. J'ai testé neuf outils utilisant différentes approches algorithmiques. Clustal Omega, T-coffee, Kalign et Kalign3 utilisent des méthodes progressives. Mafft et Muscle utilisent des méthodes itératives et pour finir, la famille de POA avec sPOA et abPOA qui utilisent une méthode à base de graphe.

8.1.2 Résultats

A l'aide du pipeline d'analyse présenté dans le chapitre 4 et des données de séquençage présentées dans le chapitre 5, j'ai pu tester les différents outils dans plusieurs cas de figure.

T-Coffee et Clustal Omega sont à mettre de côté, le premier pour un souci de performance en terme de temps mais surtout en terme de mémoire et le second pour la qualité de ses alignements bien en dessous de tous les autres dû à une mauvaise gestion des insertions et des délétions.

Pour le reste des outils, tous permettent d'avoir un bon alignement multiple. Aucun ne se démarque réellement des autres tant les résultats peuvent varier d'un jeu de données à l'autre. On peut tout de même noter quelques tendances de fond. En premier lieu, POA peut toujours être avantageusement remplacé par sPOA ou abPOA, tous les deux plus performants que leurs prédécesseurs. En deuxième lieu, la taille des reads n'a pas d'influence sur la qualité des alignements obtenus. Enfin, la profondeur de séquençage joue, quant à elle, un rôle majeur. Les outils issus de la famille de POA sont particulièrement recommandés dans le cas d'une profondeur faible, tandis que MAFFT, KALIGN et KALIGN3 sont les plus recommandés pour une profondeur élevée.

Nous avons également exploré le cas de génomes diploïdes. Dans ce contexte, on peut noter qu'aucun outil n'est convaincant. En effet, si on essaye de détecter les variants diploïdes, on observe un nombre très élevé de faux positifs, c'est-à-dire des positions considérées comme variantes qui n'en sont pas.

8.2 Perspectives

8.2.1 Le cas diploïde

L'une des perspectives évidente de ma thèse est justement le cas diploïde. Les outils actuels n'arrivant pas à traiter ce cas, il faut développer de nouvelles méthodes pour pouvoir identifier les variants diploïdes sans conserver des erreurs de séquençage ou sans créer de nouvelles erreurs dues à un mauvais alignement. J'ai cependant remarqué que T-Coffee était l'outil qui donnait le moins de faux positifs. Cela pourrait donner une idée du style d'approche qu'il faudrait utiliser pour ces nouvelles méthodes. Une autre observation est que le taux de faux positifs diminue avec l'augmentation de la profondeur de séquençage, avec T-Coffee ou avec d'autres outils. Cela indique que la recherche de variants diploïdes doit être prise en compte dès le plan expérimental, afin d'avoir une profondeur de séquençage suffisante. Enfin, l'étude des génomes diploïdes ouvre la voie à des degrés de ploïdie plus élevés, où les variants spécifiques à une allèle seront encore plus difficiles à détecter.

Pour finir, je propose dans la section 7.3 un élargissement du code IUPAC et une méthode de phasing qui pourrait être développé.

8.2.2 Détection de variants

Dans le cas haploïde, les résultats des outils sont plutôt bons, même si cela dépend des jeux de données et des profondeurs de séquençage qu'on utilise. Avec ce résultat en tête, on peut envisager d'essayer de détecter les variants entre deux organismes différents directement avec des long reads, sans utiliser des short reads comme c'est le cas actuellement. C'est une piste intéressante à explorer surtout dans le cas des jeux de données récents qui ont des taux d'erreurs de plus en plus faibles.

Bibliographie

[Ros59] Robert Rosen. « The DNA-protein coding problem ». In : *The Bulletin of Mathematical Biophysics* 21.1 (mars 1959), p. 71-95. doi : 10.1007/bf02476459. url : https://doi.org/10.1007%2Fbf02476459.

[NW70] Saul B. Needleman et Christian D. Wunsch. « A general method applicable to the search for similarities in the amino acid sequence of two proteins ». In : *Journal of Molecular Biology* 48.3 (28 mars 1970), p. 443-453. issn : 0022-2836. doi : 10.1016/0022-2836(70)90057-4. url : http://www.sciencedirect.com/science/article/pii/0022283670900574 (visité le 22/06/2020).

[SNC77] Frederick Sanger, Steven Nicklen et Alan R Coulson. « DNA sequencing with chain-terminating inhibitors ». In : *Proceedings of the national academy of sciences* 74.12 (1977), p. 5463-5467.

[Kim80] Motoo Kimura. « A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences ». In : *Journal of Molecular Evolution* 16.2 (juin 1980), p. 111-120. doi : 10.1007/bf01731581. url : https://doi.org/10.1007%2Fbf01731581.

[SW81] T. F. Smith et M. S. Waterman. « Identification of common molecular subsequences ». In : *Journal of Molecular Biology* 147.1 (25 mars 1981), p. 195-197. issn : 0022-2836. doi : 10.1016/0022-2836(81)90087-5. url : http://www.sciencedirect.com/science/article/pii/0022283681900875 (visité le 22/06/2020).

[FD87] Da-Fei Feng et Russell F Doolittle. « Progressive sequence alignment as a prerequisitetto correct phylogenetic trees ». In : *Journal of molecular evolution* 25 (1987), p. 351-360.

[Alt89] Stephen F Altschul. « Gap costs for multiple sequence alignment ». In : *Journal of theoretical biology* 138.3 (1989), p. 297-309.

[Alt+90a] Stephen F. Altschul et al. « Basic local alignment search tool ». In : *Journal of Molecular Biology* 215.3 (oct. 1990), p. 403-410. doi : 10.1016/s0022-2836(05)80360-2. url : https://doi.org/10.1016%2Fs0022-2836%2805%2980360-2.

[Alt+90b] Stephen F. Altschul et al. « Basic local alignment search tool ». In : *Journal of Molecular Biology* 215.3 (5 oct. 1990), p. 403-410. issn : 0022-2836. doi : 10.1016/S0022-2836(05)80360-2. url : http://www.sciencedirect.com/science/article/pii/S0022283605803602 (visité le 24/06/2020).

[WM92] Sun Wu et Udi Manber. « Fast text searching : allowing errors ». In : *Communications of the ACM* 35.10 (1992), p. 83-91.

[Gus93] Dan Gusfield. « Efficient methods for multiple sequence alignment with guaranteed error bounds ». In : *Bulletin of mathematical biology* 55.1 (1993), p. 141-154.

[WJ94] Lusheng Wang et Tao Jiang. « On the complexity of multiple sequence alignment ». In : *Journal of computational biology* 1.4 (1994), p. 337-348.

[Hir+95] Makoto Hirosawa et al. « Comprehensive study on iterative algorithms of multiple sequence alignment ». In : *Bioinformatics* 11.1 (1995), p. 13-18.

[MM96] Robert Muth et Udi Manber. « Approximate multiple string search ». In : *Annual Symposium on Combinatorial Pattern Matching*. Springer. 1996, p. 75-86.

[Mye99] Gene Myers. « A fast bit-vector algorithm for approximate string matching based on dynamic programming ». In : *Journal of the ACM (JACM)* 46.3 (1999), p. 395-415.

[RS00] Torbjørn Rognes et Erling Seeberg. « Six-fold speed-up of SmithWaterman sequence database searches using parallel processing on common microprocessors ». In : *Bioinformatics* 16.8 (août 2000), p. 699-706. issn : 1367-4803. doi : 10.1093/bioinformatics/16.8.699. eprint : https://academic.oup.com/bioinformatics/article-pdf/16/8/699/48842248/bioinformatics_16_8_699.pdf. url : https://doi.org/10.1093/bioinformatics/16.8.699.

[LGS02] Christopher Lee, Catherine Grasso et Mark F. Sharlow. « Multiple sequence alignment using partial order graphs ». In : *Bioinformatics* 18.3 (1er mars 2002), p. 452-464. issn : 1367-4803. doi : 10.1093/bioinformatics/18.3.452. url : https://academic.oup.com/bioinformatics/article/18/3/452/236691 (visité le 22/06/2020).

[EB03] Guy S Eakin et Richard R Behringer. « Tetraploid development in the mouse ». In : *Developmental dynamics : an official publication of the American Association of Anatomists* 228.4 (2003), p. 751-766.

[03] « OXBench : A benchmark for evaluation of protein multiple sequence alignment accuracy. » In : *BMC Bioinformatics* 47.4 (2003). DOI : https://doi.org/10.1186/1471-2105-4-47.

[Edg04] Robert C. EDGAR. « MUSCLE : a multiple sequence alignment method with reduced time and space complexity ». In : *BMC Bioinformatics* 5.1 (19 août 2004), p. 113. ISSN : 1471-2105. DOI : 10.1186/1471-2105-5-113. URL : https://doi.org/10.1186/1471-2105-5-113 (visité le 14/06/2020).

[Alt+05] Stephen F. ALTSCHUL et al. « Protein database searches using compositionally adjusted substitution matrices ». In : *FEBS Journal* 272.20 (oct. 2005), p. 5101-5109. DOI : 10.1111/j.1742-4658.2005.04945.x. URL : https://doi.org/10.1111%2Fj.1742-4658.2005.04945.x.

[GWW05] Paul P. GARDNER, Andreas WILM et Stefan WASHIETL. « A benchmark of multiple sequence alignment programs upon structural RNAs ». In : *Nucleic Acids Research* 33.8 (jan. 2005), p. 2433-2439. ISSN : 0305-1048. DOI : 10.1093/nar/gki541. URL : https://doi.org/10.1093/nar/gki541.

[LS05] Timo LASSMANN et Erik LL SONNHAMMER. « Kalign an accurate and fast multiple sequence alignment algorithm ». In : *BMC Bioinformatics* 6.1 (12 déc. 2005), p. 298. ISSN : 1471-2105. DOI : 10.1186/1471-2105-6-298. URL : https://doi.org/10.1186/1471-2105-6-298 (visité le 15/06/2020).

[SB05] Guy St C. SLATER et Ewan BIRNEY. « Automated generation of heuristics for biological sequence comparison ». In : *BMC Bioinformatics* 6.1 (15 fév. 2005), p. 31. ISSN : 1471-2105. DOI : 10.1186/1471-2105-6-31. URL : https://doi.org/10.1186/1471-2105-6-31 (visité le 24/06/2020).

[Tho+05] Julie D. THOMPSON et al. « BAliBASE 3.0 : Latest developments of the multiple sequence alignment benchmark ». In : *Proteins : Structure, Function, and Bioinformatics* 61.1 (2005), p. 127-136. DOI : https://doi.org/10.1002/prot.20527. URL : https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.20527.

[Eli06] Isaac ELIAS. « Settling the intractability of multiple alignment ». In : *Journal of Computational Biology* 13.7 (2006), p. 1323-1339.

[Lar+07] M.A. LARKIN et al. « Clustal W and Clustal X version 2.0 ». In : *Bioinformatics* 23.21 (sept. 2007), p. 2947-2948. ISSN : 1367-4803. DOI : 10.1093/bioinformatics/btm404. eprint : https://academic.oup.com/bioinformatics/article-pdf/23/21/2947/49823064/bioinformatics_23_21_2947.pdf. URL : https://doi.org/10.1093/bioinformatics/btm404.

[Not07] Cédric NOTREDAME. « Recent evolutions of multiple sequence alignment algorithms ». In : *PLoS computational biology* 3.8 (2007), e123.

[Der+08] A. DEREEPER et al. « Phylogeny.fr : robust phylogenetic analysis for the non-specialist ». In : *Nucleic Acids Research* 36.Web Server (mai 2008), W465-W469. DOI : 10.1093/nar/gkn180. URL : https://doi.org/10.1093%2Fnar%2Fgkn180.

[Sch08] Stephan C SCHUSTER. « Next-generation sequencing transforms today's biology ». In : *Nature methods* 5.1 (2008), p. 16-18.

[Cla+09] James CLARKE et al. « Continuous base identification for single-molecule nanopore DNA sequencing ». In : *Nature Nanotechnology* 4.4 (fév. 2009), p. 265-270. DOI : 10.1038/nnano.2009.12. URL : https://doi.org/10.1038%2Fnnano.2009.12.

[Eid+09] John EID et al. « Real-time DNA sequencing from single polymerase molecules ». In : *Science* 323.5910 (2009), p. 133-138.

[KAT09] Kazutaka KATOH, George ASIMENOS et Hiroyuki TOH. « Multiple Alignment of DNA Sequences with MAFFT ». In : *Bioinformatics for DNA Sequence Analysis*. Sous la dir. de David POSADA. Methods in Molecular Biology. Totowa, NJ : Humana Press, 2009, p. 39-64. ISBN : 9781597452519. DOI : 10.1007/978-1-59745-251-9_3. URL : https://doi.org/10.1007/978-1-59745-251-9_3 (visité le 14/06/2020).

[LFS09] Timo LASSMANN, Oliver FRINGS et Erik LL SONNHAMMER. « Kalign2 : high-performance multiple alignment of protein and nucleotide sequences allowing external features ». In : *Nucleic acids research* 37.3 (2009), p. 858-865.

[LD09] Heng LI et Richard DURBIN. « Fast and accurate short read alignment with BurrowsWheeler transform ». In : *Bioinformatics* 25.14 (mai 2009), p. 1754-1760. DOI : 10.1093/bioinformatics/btp324. URL : https://doi.org/10.1093%2Fbioinformatics%2Fbtp324.

[Bla+10] Gordon BLACKSHIELDS et al. « Sequence embedding for fast construction of guide trees for multiple sequence alignment ». In : *Algorithms for Molecular Biology* 5 (2010), p. 1-11.

[DeP+11] Mark A DEPRISTO et al. « A framework for variation discovery and genotyping using next-generation DNA sequencing data ». In : *Nature Genetics* 43.5 (avr. 2011), p. 491-498. DOI : 10.1038/ng.806. URL : https://doi.org/10.1038%2Fng.806.

[SS11] Leena SALMELA et Jan SCHRÖDER. « Correcting errors in short reads by multiple alignments ». In : *Bioinformatics* 27.11 (2011), p. 1455-1461.

[Sie+11] Fabian Sievers et al. « Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega ». In : *Molecular Systems Biology* 7.1 (1er jan. 2011), p. 539. ISSN : 1744-4292. DOI : 10.1038/msb.2011.75. URL : https://www.embopress.org/doi/full/10.1038/msb.2011.75 (visité le 14/06/2020).

[Ban+12] Anton Bankevich et al. « SPAdes : A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing ». In : *Journal of Computational Biology* 19.5 (mai 2012), p. 455-477. DOI : 10.1089/cmb.2012.0021. URL : https://doi.org/10.1089%2Fcmb.2012.0021.

[KR12] J. Koster et S. Rahmann. « Snakemake–a scalable bioinformatics workflow engine ». In : *Bioinformatics* 28.19 (1er oct. 2012), p. 2520-2522. ISSN : 1367-4803, 1460-2059. DOI : 10.1093/bioinformatics/bts480. URL : https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bts480 (visité le 14/06/2020).

[LS12] Ben Langmead et Steven L Salzberg. « Fast gapped-read alignment with Bowtie 2 ». In : *Nature Methods* 9.4 (mars 2012), p. 357-359. DOI : 10.1038/nmeth.1923. URL : https://doi.org/10.1038%2Fnmeth.1923.

[Chi+13] Chen-Shan Chin et al. « Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data ». In : *Nature methods* 10.6 (2013), p. 563-569.

[Wel+13] Danielle Welter et al. « The NHGRI GWAS Catalog, a curated resource of SNP-trait associations ». In : *Nucleic Acids Research* 42.D1 (déc. 2013), p. D1001-D1006. DOI : 10.1093/nar/gkt1229. URL : https://doi.org/10.1093%2Fnar%2Fgkt1229.

[Mag+14] Cedrik Magis et al. « T-Coffee : Tree-Based Consistency Objective Function for Alignment Evaluation ». In : *Multiple Sequence Alignment Methods*. Sous la dir. de David J Russell. Methods in Molecular Biology. Totowa, NJ : Humana Press, 2014, p. 117-129. ISBN : 9781627036467. DOI : 10.1007/978-1-62703-646-7_7. URL : https://doi.org/10.1007/978-1-62703-646-7_7 (visité le 14/06/2020).

[Miy+14] Mari Miyamoto et al. « Performance comparison of second-and third-generation sequencers using a bacterial genome with two chromosomes ». In : *BMC genomics* 15.1 (2014), p. 1-9.

[See14] Torsten Seemann. « Prokka : rapid prokaryotic genome annotation ». In : *Bioinformatics* 30.14 (mars 2014), p. 2068-2069. DOI : 10.1093/bioinformatics/btu153. URL : https://doi.org/10.1093%2Fbioinformatics%2Fbtu153.

[Cha+15] Maria Chatzou et al. « Multiple sequence alignment modeling : methods and applications ». In : *Briefings in Bioinformatics* 17.6 (nov. 2015), p. 1009-1023. ISSN : 1467-5463. DOI : 10.1093/bib/bbv099. eprint : https://academic.oup.com/bib/article-pdf/17/6/1009/8280322/bbv099.pdf. URL : https://doi.org/10.1093/bib/bbv099.

[LQS15] Nicholas J Loman, Joshua Quick et Jared T Simpson. « A complete bacterial genome assembled de novo using only nanopore sequencing data ». In : *Nature methods* 12.8 (2015), p. 733-735.

[CL16] Claudia M. B. Carvalho et James R. Lupski. « Mechanisms underlying structural variant formation in genomic disorders ». In : *Nature Reviews Genetics* 17.4 (fév. 2016), p. 224-238. DOI : 10.1038/nrg.2015.25. URL : https://doi.org/10.1038%2Fnrg.2015.25.

[Chi+16] Chen-Shan Chin et al. « Phased diploid genome assembly with single-molecule real-time sequencing ». In : *Nature methods* 13.12 (2016), p. 1050-1054.

[YM16] Chengxi Ye et Zhanshan Sam Ma. « Sparc : a sparsity-based consensus algorithm for long erroneous sequencing reads ». In : *PeerJ* 4 (2016), e2016.

[Kor+17] Sergey Koren et al. « Canu : scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation ». In : *Genome research* 27.5 (2017), p. 722-736.

[Li17] Heng Li. « Minimap2 : fast pairwise alignment for long nucleotide sequences ». In : *arXiv :1708.01492 [q-bio]* (août 2017). arXiv : 1708.01492. URL : http://arxiv.org/abs/1708.01492 (visité le 22/11/2017).

[Vas+17a] Robert Vaser et al. « Fast and accurate de novo genome assembly from long uncorrected reads ». In : *Genome Research* 27.5 (1er mai 2017), p. 737-746. ISSN : 1088-9051, 1549-5469. DOI : 10.1101/gr.214270.116. URL : http://genome.cshlp.org/content/27/5/737 (visité le 22/06/2020).

[Vas+17b] Robert Vaser et al. « Fast and accurate de novo genome assembly from long uncorrected reads ». In : *Genome research* 27.5 (2017), p. 737-746.

[Wic+17a] Ryan R Wick et al. « Unicycler : resolving bacterial genome assemblies from short and long sequencing reads ». In : *PLoS computational biology* 13.6 (2017), e1005595.

[Wic+17b] Ryan R. Wick et al. « Unicycler : Resolving bacterial genome assemblies from short and long sequencing reads ». In : *PLOS Computational Biology* 13.6 (juin 2017). Sous la dir. d'Adam M. Phillippy, e1005595. DOI : 10.1371/journal.pcbi.1005595. URL : https://doi.org/10.1371%2Fjournal.pcbi.1005595.

[Xia+17] Chuan-Le Xiao et al. « MECAT : fast mapping, error correction, and de novo assembly for single-molecule sequencing reads ». In : *nature methods* 14.11 (2017), p. 1072-1074.

[Gar+18] Shilpa Garg et al. « A graph-based approach to diploid genome assembly ». In : *Bioinformatics* 34.13 (juin 2018), p. i105-i114. doi : 10.1093/bioinformatics/bty279. url : https://doi.org/10.1093%2Fbioinformatics%2Fbty279.

[Goo+18] Emily C. A. Goodall et al. « The Essential Genome of iEscherichia coli/i K-12 ». In : *mBio* 9.1 (mars 2018). Sous la dir. de Swaine L. Chen et Kimberly A. Kline. doi : 10.1128/mbio.02096-17. url : https://doi.org/10.1128%2Fmbio.02096-17.

[Jai+18] Miten Jain et al. « Nanopore sequencing and assembly of a human genome with ultra-long reads ». In : *Nature Biotechnology* 36.4 (jan. 2018), p. 338-345. doi : 10.1038/nbt.4060. url : https://doi.org/10.1038%2Fnbt.4060.

[Mar+18] Guillaume Marçais et al. « MUMmer4 : A fast and versatile genome alignment system ». In : *PLOS Computational Biology* 14.1 (jan. 2018). Sous la dir. d'Aaron E. Darling, e1005944. doi : 10.1371/journal.pcbi.1005944. url : https://doi.org/10.1371%2Fjournal.pcbi.1005944.

[Bao+19] Ergude Bao et al. « FLAS : fast and high-throughput algorithm for PacBio long-read self-correction ». In : *Bioinformatics* 35.20 (2019), p. 3953-3960.

[HV19] David Heller et Martin Vingron. « SVIM : structural variant identification using mapped long reads ». In : *Bioinformatics* 35.17 (jan. 2019). Sous la dir. d'Inanc Birol, p. 2907-2915. doi : 10.1093/bioinformatics/btz041. url : https://doi.org/10.1093%2Fbioinformatics%2Fbtz041.

[KCS19] Ritu Kundu, Joshua Casey et Wing-Kin Sung. « HyPo : super fast & accurate polisher for long read genome assemblies ». In : *bioRxiv* (2019).

[Mai+19] Nicola De Maio et al. « Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes ». In : *Microbial Genomics* 5.9 (sept. 2019). doi : 10.1099/mgen.0.000294. url : https://doi.org/10.1099%2Fmgen.0.000294.

[Pio+19] Allison Piovesan et al. « Human protein-coding genes and gene feature statistics in 2019 ». In : *BMC research notes* 12.1 (2019), p. 1-5.

[Sat+19] Mitsuhiko P Sato et al. « Comparison of the sequencing bias of currently available library preparation kits for Illumina sequencing of bacterial genomes and metagenomes ». In : *DNA Research* 26.5 (juill. 2019), p. 391-398. doi : 10.1093/dnares/dsz017. url : https://doi.org/10.1093%2Fdnares%2Fdsz017.

[SH19] Fabian Sievers et Desmond G Higgins. « QuanTest2 : benchmarking multiple sequence alignments using secondary structure prediction ». In : *Bioinformatics* 36.1 (juill. 2019), p. 90-95. issn : 1367-4803. doi : 10.1093/bioinformatics/btz552. eprint : https://academic.oup.com/bioinformatics/article-pdf/36/1/90/

48981291/bioinformatics_36_1_90.pdf. URL : https://doi.org/10.1093/bioinformatics/btz552.

[WJH19] Ryan R. WICK, Louise M. JUDD et Kathryn E. HOLT. « Performance of neural network basecalling tools for Oxford Nanopore sequencing ». In : *Genome Biology* 20.1 (juin 2019). DOI : 10.1186/s13059-019-1727-y. URL : https://doi.org/10.1186%2Fs13059-019-1727-y.

[Las20] Timo LASSMANN. *Kalign 3 : multiple sequence alignment of large datasets*. 2020.

[Lec+20] Lolita LECOMPTE et al. « SVJedi : genotyping structural variations with long reads ». In : *Bioinformatics* 36.17 (mai 2020). Sous la dir. de Peter ROBINSON, p. 4568-4575. DOI : 10.1093/bioinformatics/btaa527. URL : https://doi.org/10.1093%2Fbioinformatics%2Fbtaa527.

[Mar+20] Camille MARCHET et al. « ELECTOR : evaluator for long reads correction methods ». In : *NAR Genomics and Bioinformatics* 2.1 (2020), lqz015.

[RL20] Jue RUAN et Heng LI. « Fast and accurate long-read assembly with wtdbg2 ». In : *Nature methods* 17.2 (2020), p. 155-158.

[Wri20] Erik S. WRIGHT. « RNAconTest : comparing tools for noncoding RNA multiple sequence alignment based on structural consistency ». In : *RNA* 26.5 (2020), p. 531-540. eprint : http://rnajournal.cshlp.org/content/26/5/531.full.pdf+html. URL : http://rnajournal.cshlp.org/content/26/5/531.abstract.

[Zhu+20] Hanliang ZHU et al. « PCR past, present and future ». In : *BioTechniques* 69.4 (oct. 2020), p. 317-325. DOI : 10.2144/btn-2020-0057. URL : https://doi.org/10.2144%2Fbtn-2020-0057.

[Abo+21] Omar ABOU SAADA et al. « nPhase : an accurate and contiguous phasing method for polyploids ». In : *Genome biology* 22.1 (2021), p. 1-27.

[Che+21] Haoyu CHENG et al. « Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm ». In : *Nature methods* 18.2 (2021), p. 170-175.

[Kat+21] Kenneth KATZ et al. « The Sequence Read Archive : a decade more of explosive growth ». In : *Nucleic Acids Research* 50.D1 (nov. 2021), p. D387-D390. DOI : 10.1093/nar/gkab1053. URL : https://doi.org/10.1093%2Fnar%2Fgkab1053.

[Lie21] Thomas LIEHR. « Repetitive elements in humans ». In : *International Journal of Molecular Sciences* 22.4 (2021), p. 2072.

[Mor+21] Pierre MORISSE et al. « Scalable long read self-correction and assembly polishing with multiple sequence alignment ». In : *Scientific Reports* 11.1 (2021), p. 1-13.

[OAH21] Yukiteru Ono, Kiyoshi Asai et Michiaki Hamada. « PBSIM2 : a simulator for long-read sequencers with a novel generative model of quality scores ». In : *Bioinformatics* 37.5 (2021), p. 589-595.

[SN21] Nicholas Stoler et Anton Nekrutenko. « Sequencing error profiles of Illumina sequencing instruments ». In : *NAR Genomics and Bioinformatics* 3.1 (jan. 2021). doi : 10.1093/nargab/lqab019. url : https://doi.org/10.1093%2Fnargab%2Flqab019.

[V21] Robert Vaser et Mile iki. « Time-and memory-efficient genome assembly with Raven ». In : *Nature Computational Science* 1.5 (2021), p. 332-336.

[War21] Tandy Warnow. « Revisiting Evaluation of Multiple Sequence Alignment Methods ». In : *Multiple Sequence Alignment : Methods and Protocols*. Sous la dir. de Kazutaka Katoh. New York, NY : Springer US, 2021, p. 299-317. isbn : 978-1-0716-1036-7. doi : 10.1007/978-1-0716-1036-7_17. url : https://doi.org/10.1007/978-1-0716-1036-7_17.

[Edg22] Robert C. Edgar. « Muscle5 : High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny ». In : *Nature Communications* 13.1 (nov. 2022). doi : 10.1038/s41467-022-34630-w. url : https://doi.org/10.1038%2Fs41467-022-34630-w.

[Gli+22] Dafni A. Glinos et al. « Transcriptome variation in human tissues revealed by long-read sequencing ». In : *Nature* 608.7922 (août 2022), p. 353-359. doi : 10.1038/s41586-022-05035-y. url : https://doi.org/10.1038%2Fs41586-022-05035-y.

[HWS22] Robert Hubley, Travis J Wheeler et Arian F A Smit. « Accuracy of multiple sequence alignment methods in the reconstruction of transposable element families ». In : *NAR Genomics and Bioinformatics* 4.2 (mai 2022), lqac040. issn : 2631-9268. doi : 10.1093/nargab/lqac040. eprint : https://academic.oup.com/nargab/article-pdf/4/2/lqac040/43733809/lqac040.pdf. url : https://doi.org/10.1093/nargab/lqac040.

[Nur+22a] Sergey Nurk et al. « The complete sequence of a human genome ». In : *Science* 376.6588 (avr. 2022), p. 44-53. doi : 10.1126/science.abj6987. url : https://doi.org/10.1126%2Fscience.abj6987.

[Nur+22b] Sergey Nurk et al. « The complete sequence of a human genome ». In : *Science* 376.6588 (2022), p. 44-53.

[Sch+22] Lucie Schröder et al. « The gene space of European mistletoe (Viscum album) ». In : *The Plant Journal* 109.1 (2022), p. 278-294.

[Wan+23] Yibin Wang et al. « Sequencing and Assembly of Polyploid Genomes ». In : *Polyploidy : Methods and Protocols*. Springer, 2023, p. 429-458.

Annexe A

Read me du pipeline d'analyse

msa-limit

Msa-limit is an analysis pipeline to test the efficiency of different multiple alignment software (MSA) on long reads. Using nanopore reads and a reference, it generates consensus sequences from the different MSA software to compare to the reference and see if the alignment is correct. (See the schematic in the doc file for more details)

Usable MSA software: muscle,mafft,poa,kalign,spoa,kalign3,clustalo,abpoa,tcoffee

Usage

Conda (>4.10) must be installed (see https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html)

To install msa-limit:

```
git clone https://gitlab.cristal.univ-lille.fr/crohmer/msa-limit.git
cd msa-limit
```

Run a test to verify proper operation:

```
./msa-limit.py test
```

To start the analysis pipeline:

```
Usage:
msa-limit.py -i <file_reads> -r <file_ref> [-options]

Arguments:
  required:
    -i <string>
       nanopore long reads file (fasta or fastq)
    -r <string>
       reference sequence file (fasta, a single sequence)
       IUPAC consensus sequence in the diploid case

  optional:
    -n <string>
       default: date and time of execution
       name of the experiment
    -o <int>
       default: 10
       number of regions to be tested
    -b <int>,<int>,...
       beginning(s) position of region(s) (replacing -o)
    -d <int>,<int>,...
       default: 10,20,50
       sequencing depth(s) (number of reads)
    -s <int>,<int>,...
       default: 100,200
       size(s) of region(s)
    -t <int>,<int>,...
```

```
    -h
       help

Ex: ./msa-limit.py -i reads.fastq -r ref.fasta -b 1,150 -n exp -d 10,100 -s 100,200 -t 50,75 -m mafft,p
```

Others modes There are other features than the basic one for msa-limit:

```
Usage:
msa-limit.py -i <file_reads> -r <file_ref> [-options]

Other modes:
  test
      Launches a pipeline test
  list
      List of existing experiments
  summary
      More readable summary of experiments for a human
        optional:
         -n <string> <string> <string> ...
            default: all the names of the experiments
            names of the experiments you want to display in the summary.
  run_config <string> <string> ...
       Launches the pipeline from configuration file(s)
         required: path to the configuration file(s).
  rulegraph
       Displays a graph of the snakemake rules
```

Configuration file The basic mode of msa-limit creates a configuration file which is then used by the pipeline. It is possible with the run_config mode (msa-limit run_config) to directly launch the pipeline with its own configuration file which must respect the following format:

```
I: <reads_file>     #REQUIRED, absolute path of preference
I: <ref_file>       #REQUIRED, absolute path of preference
n: test             #OPTIONAL, -n
D: [10,20,50]       #OPTIONAL, -d
S: [100,200]        #OPTIONAL, -s
T: [50]             #OPTIONAL, -t
M: [muscle,mafft]   #OPTIONAL, -m
O: 10               #OPTIONAL, -o, can be replaced by -b (B: [1,150])
```

Only snakemake This pipeline is created from snakemake. If you are familiar with this tool, you can launch the pipeline directly from snakemake with a configuration file. You will need to install snakemake (6.10++) and set the option to use conda

```
snakemake --configfile <config_file> -c24 --use-conda
```

Dependencies

- conda 4.10.1+
- python 3.7.4+

Add a new msa software

If you want to add a new msa software in the pipeline, you will have to add a rule in the Snakefile. You will have to either install the software locally or create a conda environment file with the software. The output must be in fasta format In the following commands, replace with the name of the software.

Create a conda environment file:

```
conda create -n <new_msa>
conda install <new_msa>
conda env export >env_conda/<new_msa>.yaml
```

Add the rule below in the Snakefile. Replace with the name of the software. Replaces with the command to run the software. In your command, the input and output file must be replaced with {input} and {output.out}. (Ex: muscle -in {input} -out {output.out})

```
rule <new_msa> :
    input :
        os.path.join('{data_set}','selected_read','reads_r{region_size}_d{depth}.fasta')
    output :
        time = os.path.join('{data_set}','time','MSA_<new_msa>_r{region_size}_d{depth}'),
        out = os.path.join('{data_set}','msa','MSA_<new_msa>_r{region_size}_d{depth}.fasta')
    message:
        "<new_msa> for {wildcards.data_set} (Region size={wildcards.region_size} & Depth={wildcards.dep
    log:
        os.path.join('{data_set}','logs','6_<new_msa>_r{region_size}_d{depth}.log')
    conda:   #Only if you use conda
        "env_conda/<new_msa>.yaml"
    shell :
        './src/run_MSA.sh "<command_to_launch_the_software>" {input} {output.out} {output.time} {log} 1
```

Warning: If the output of the software is done by the terminal output stream, put only the command with the input and change the 6th parameter of the script run_msa.sh from 1 to 0 (see the rule for Spoa for this case)

Potential issue

Abpoa doesn't run

abpoa may not launch from conda on some machines. To solve this problem, you will have to install it locally (see https://github.com/yangao07/abPOA) and modify the abpoa rule.

Annexe B

Poster pour JOBIM 2022