DÉVELOPPEMENT D'UNE LIBRAIRIE DE CODE ET D'OUTILS BIO-INFORMATIQUES FACILITANT L'ANALYSE DE GRANDES QUANTITÉS DE DONNÉES GÉNOMIQUES

par

Alexei Nordell-Markovits

Thèse présentée au Département de biologie en vue
de l'obtention du grade de docteur ès sciences (Ph.D.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, Septembre 2016

Le 15 septembre 2016

Le jury a accepté la thèse de Monsieur Alexei Nordell-Markovits
dans sa version finale.

Membres du jury

Pierre-Étienne Jacques
Directeur de recherche
Département de biologie

Shengrui Wang
Codirecteur de recherche
Département d'informatique

Nicolas Gévry
Codirecteur de recherche
Département de biologie

Michelle Scott
Évaluatrice interne
Département de biochimie

Mathieu Blanchette
Évaluateur externe
Université McGill

Sébastien Rodrigue
Président-rapporteur
Département de biologie

SOMMAIRE

La présente thèse décrit des travaux effectués dans le domaine du développement d'outils en bio-informatique génomique. Plus spécifiquement, ces outils visent à faciliter le développement et l'analyse des données issues d'expériences impliquant le séquençage haut débit. Les percées au niveau des technologies de séquençage de l'ADN ont révolutionnées la recherche en génomique depuis les dernières années. En conséquence, la quantité massive de données générées nécessite de nouveaux outils et de nouvelles méthodes d'analyse afin de faciliter l'interprétation des expériences par les chercheurs.

Mes travaux de recherche viennent pallier une partie de ce problème en proposant des outils de développement et d'analyse adaptés à la réalité des données de séquençage haut débit. J'ai ainsi participé au développement d'une librairie de code en C++ spécialisé dans le prototypage d'outils d'analyse génomique; de pair avec cette librairie, j'ai développé un code de lecture en C++ moderne pour le format génomique BigWig.

En utilisant ces outils de développement, j'ai construit un outil d'analyse nommé *Genomic Efficient Correlator* (GeEC) visant à rendre accessible la génération de matrices de corrélations massives. Cet outil a été validé en analysant les résultats de deux ensembles de données résultant respectivement de quelques centaines et quelques milliers d'expériences biologiques. Le résultat de ces tests démontre qu'il est possible de générer des matrices de corrélations massives dans un temps acceptable et que les matrices produites permettent d'identifier des relations qui reflètent la réalité biologique fonctionnelle des données.

Mot-clés : Bio-informatique, génomique, librairie de code, séquençage haut débit, analyse de données, corrélation.

REMERCIEMENTS

J'aimerais tout d'abord remercier mon directeur de recherche, Pierre-Étienne Jacques pour sa contribution essentielle à ma formation et mes travaux de recherche durant les dernières années. Je remercie également mes co-directeurs : Nicolas Gévry avec qui j'ai commencé mes travaux dans le domaine de la génomique et Shengrui Wang qui m'a également supervisé durant ma maitrise.

Merci à tous les membres des laboratoires du P^{r} Jacques, du P^{r} Gévry et du P^{r} Rodrigue avec qui j'ai beaucoup partagé. J'aimerais également remercier les membres du laboratoire ProspectUS dirigé par Shengrui Wang, qui m'ont épaulé pour certains aspects informatiques et analytiques de mes travaux. Je remercie particulièrement Mylène Brunelle qui a défriché avec moi le domaine du séquençage haut débit et Jean-Pierre Glouzon avec qui j'ai eu de longues discussions sur le domaine de la bio-informatique.

J'aimerais remercier ma mère Veronica Nordell et mon père Henry Markovits pour leur support et leur amour durant le long processus des études.

Un remerciement général aux étudiants de mon association, le RECSUS. Nos discussions sur l'heure du midi et la possibilité de partager avec plusieurs autres étudiants gradués ont constitué de précieuses opportunités. Je remercie mes amis Marc Therrien, Luc Paquette et Heidi Larkins que j'ai côtoyé presque tout le long de mon séjour à Sherbrooke et qui m'ont offert plus d'aide qu'ils ne l'imaginent. Finalement, un remerciement immense à ma copine Carine Hamel qui m'a encouragé et écouté tout au long de mon doctorat et sans qui je ne serais pas venu à bout de l'épreuve.

TABLE DES MATIERES

LISTE DES ABREVIATIONS

ARN	Acide ribonucléique
ARNt	ARN de transfert
ARNm	ARN messager
ADN	Acide désoxyribonucléique
BAM	Binary Alignement/Map
BED	Browser Extensible Data
bp	Base pair
ChIP	Chromatin Immuno-precipitation
ChIA-PET	Chromatin Interaction Analysis by Paired-End Tag Sequencing
ChIP-Seq	*ChIP followed by sequencing*
CTCF	Transcriptional repressor CTCF
GeEC	Genomic Efficient Correlator
Go	Gigaoctets
IHEC	International Human Epigenome Consortium
Mo	Megaoctets
pb	Paire de base
SAM	Sequence Alignment/Map
UCSC	University of California in Santa Cruz
WIG	Wiggle Format

LISTE DES TABLEAUX

LISTE DES FIGURES

CHAPITRE 1

INTRODUCTION GENERALE

Les 40 dernières années ont été témoin d'une amélioration fulgurante des méthodes de séquençage. De la naissance du séquençage moderne avec l'invention de la méthode Sanger (Sanger, 1977) à aujourd'hui, les technologies de séquençages ont multiplié leur efficacité.

À titre d'exemple, « *The Human Genome Project* » débuté en 1990 était un ambitieux projet par lequel le code génétique complet de l'humain allait être séquencé et l'entièreté de ses gènes identifiés (Watson, 1990). Le projet prit 13 ans, coûta plus d'un milliard de dollars et dut contourner de nombreuses embûches sociales, éthiques et bio-informatiques. Bien qu'un franc succès, le séquençage du génome humain démontra les limitations des méthodes utilisées. Pour multiplier les projets de cette envergure, il fallait des procédés plus performants et surtout moins dispendieux, ce qui fut possible aux suites de récentes percées technologiques.

En effet, en 2001 le séquençage d'un million de paires de bases (pb) revenait à un peu moins de 10 000 $, alors qu'en 2015 il en coute moins de 1$ pour séquencer une mégabase (et ce malgré l'inflation!) (Figure 1.1). De plus, le débit d'une machine de séquençage a augmenté et une seule procédure de séquençage peut maintenant générer des milliards de pb et produire plusieurs centaines de millions de lectures. En plus de permettre une variété de nouvelles applications que nous couvrirons, cette révolution a décuplé l'accessibilité des technologies de séquençage.

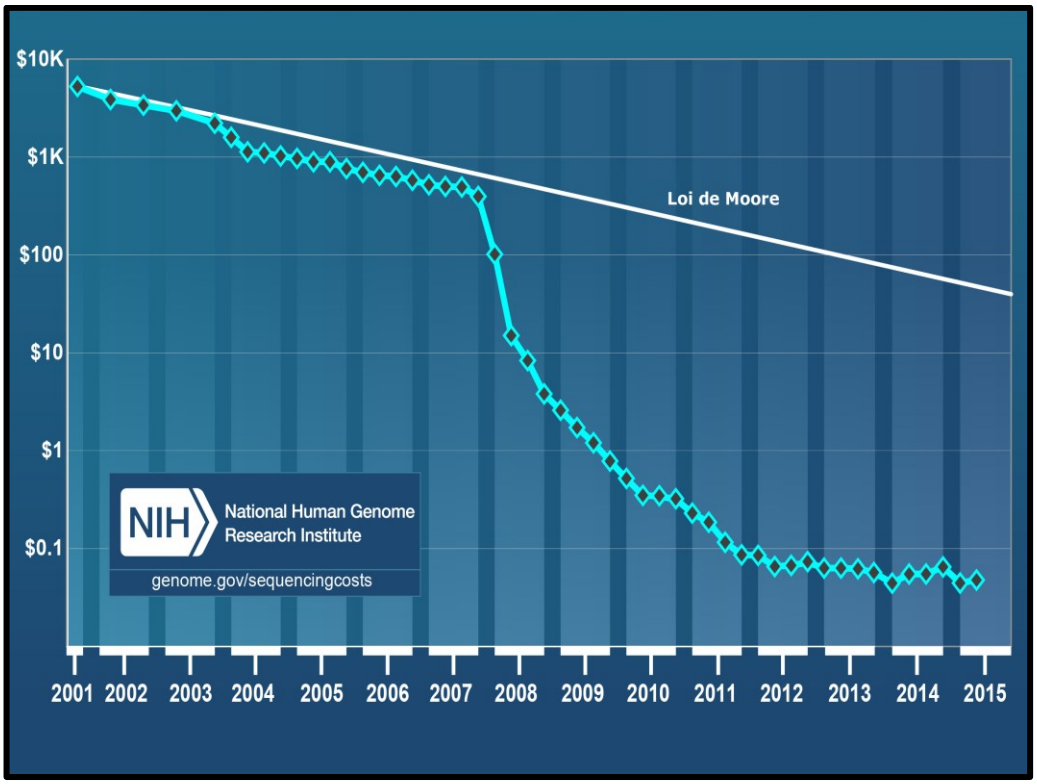

Figure 1.1 Estimation du coût de séquençage génomique de 2001 à 2015. Adapté de genome.gov/sequencingcosts.

Devenues accessibles à de petites équipes, de plus en plus de chercheurs s'intéressent à l'utilisation de ces méthodes pour supporter leurs efforts de recherche. Cependant, l'accessibilité des méthodes de séquençage modernes a modifié la problématique rattachée à celle-ci. Précédemment, le défi se situait au niveau de la production de données tandis que l'analyse était (en terme relatif) moins coûteuse. La situation initiale s'est ainsi inversée, le défi principal étant maintenant l'analyse (Mardis, 2010). De plus, la quantité de données qui peut être rapidement générée amène aussi son lot de défis au niveau de la gestion de l'équipement matériel et des infrastructures informatiques.

Le domaine de la bio-informatique a connu une croissance allant de pair avec le domaine du séquençage. Bien que les outils informatiques fussent utilisés en biologie avant l'apparition des données de séquençage haut débit (Hagen, 2000), le développement du séquençage fut

sans contredit un élément important dans la croissance du domaine. Plusieurs des premières grandes bases de données étaient faites pour stocker des données de séquençage où les outils de gestion de séquences étaient essentiels (Stevens, 2013). La croissance exponentielle des données disponibles ne fait qu'augmenter l'importance de la bio-informatique dans le traitement et l'analyse des données.

Le début de mes travaux de doctorat dans le laboratoire du P[r] Nicolas Gévry puis leur poursuite dans le laboratoire du P[r] Pierre-Étienne Jacques se situent dans ce contexte. Mes contributions se sont surtout concentrées sur le développement d'outils bio-informatiques. Pour bien comprendre les motivations reliées à ces contributions, il est important de pouvoir les mettre en contexte. J'aborderai ainsi certains concepts de la transcription eucaryote et de l'épigénétique afin de faciliter la compréhension des applications du séquençage haut débit, puis j'exposerai certains des nouveaux défis qui sont venus de pair avec la révolution des technologies de séquençage. Finalement, je discuterai de certains sujets spécifiques de la bio-informatique génomique tels que l'utilisation des formats de fichiers et le besoin de nouveaux outils de manipulation et d'analyse de grandes quantités de données. À noter que, pour alléger le texte, l'utilisation du terme séquençage dans cette introduction fait référence au séquençage haut débit.

1.1 Principes élémentaires de la transcription eucaryote

La transcription est un processus complexe qui, à partir d'une séquence d'ADN, produit un brin d'ARN qui est, dans certains cas, ensuite traduit en protéines. C'est un des processus fondamentaux du fonctionnement d'une cellule. Dans un génome eucaryote, elle est régit par une panoplie d'éléments qui assurent sa régulation. Approfondir notre compréhension du

système impliqué dans la régulation de la transcription est un champ de recherche important pour la compréhension de la biologie humaine.

Cependant, il existe une multitude d'éléments qui participent à la régulation. Des éléments amplificateurs augmentent le niveau de transcription du gène, des éléments dits répresseurs l'empêchent ou la diminuent. Certains éléments sont situés à de très grandes distances du gène régulé (éléments distaux) et peuvent être eux-mêmes régulés en partie par d'autres éléments, tels les insulateurs (Figure 1.2).

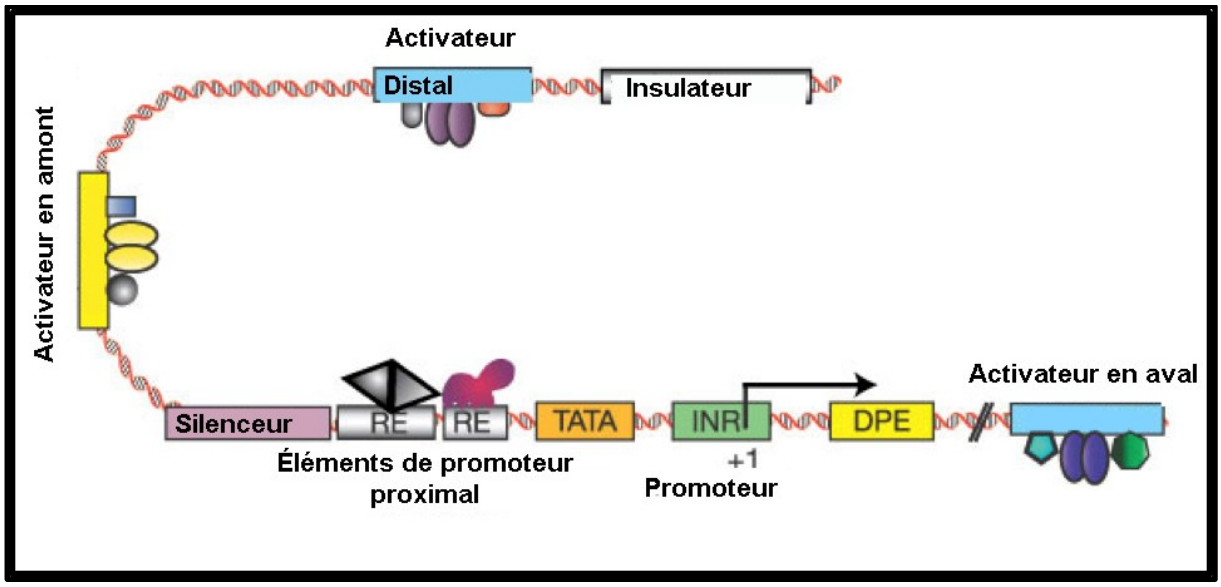

Figure 1.2 Certains éléments impliqués dans la régulation de la transcription eucaryote. Adapté de Levine et Tijan, 2003.

Un des éléments essentiels à sa régulation est la chromatine. Nos travaux s'intéressent en grande partie aux méthodes permettant de mieux comprendre la composition de la chromatine, d'où l'importance d'en connaître la fonction détaillée dans la section suivante.

1.1.1 Chromatine

Le génome haploïde de l'humain contient environ 3 milliards de paires de bases. Bien qu'une cellule humaine ne mesure qu'environ 10 micromètres, une fois déroulée, l'ADN de celle-ci mesure près de 2 mètres. Un compactage remarquable est donc nécessaire pour permettre à une telle quantité d'ADN de se retrouver dans la cellule (Dvir, Conaway, & Conaway, 2001).

La chromatine est le nom donné au complexe composé d'ADN enroulé autour de protéines impliquées dans ce système de compactage (Kornberg & Thomas, 1974). Pour réussir un tel exploit, le procédé passe par de multiples étapes de compactage (Figure 1.3). La chromatine peut sommairement être divisée en deux types : l'euchromatine, moins compacte et plus facile d'accès aux mécanismes biologiques, et l'hétérochromatine qui possède un repliement plus dense et donc particulièrement inaccessible (Talbert & Henikoff, 2006).

Cependant, même l'ADN de l'euchromatine peut être difficile d'accès ; en effet, l'enroulement de l'ADN autour des nucléosomes peut bloquer l'accès à la machinerie transcriptionnelle. Pour assurer que l'ADN soit accessible, les éléments de la chromatine possèdent un certain dynamisme qui permet d'ouvrir ou de refermer la structure à des endroits spécifiques. Ce dynamisme est possible grâce aux caractéristiques des éléments qui constituent la chromatine et leur étude est essentielle à une bonne compréhension de la transcription.

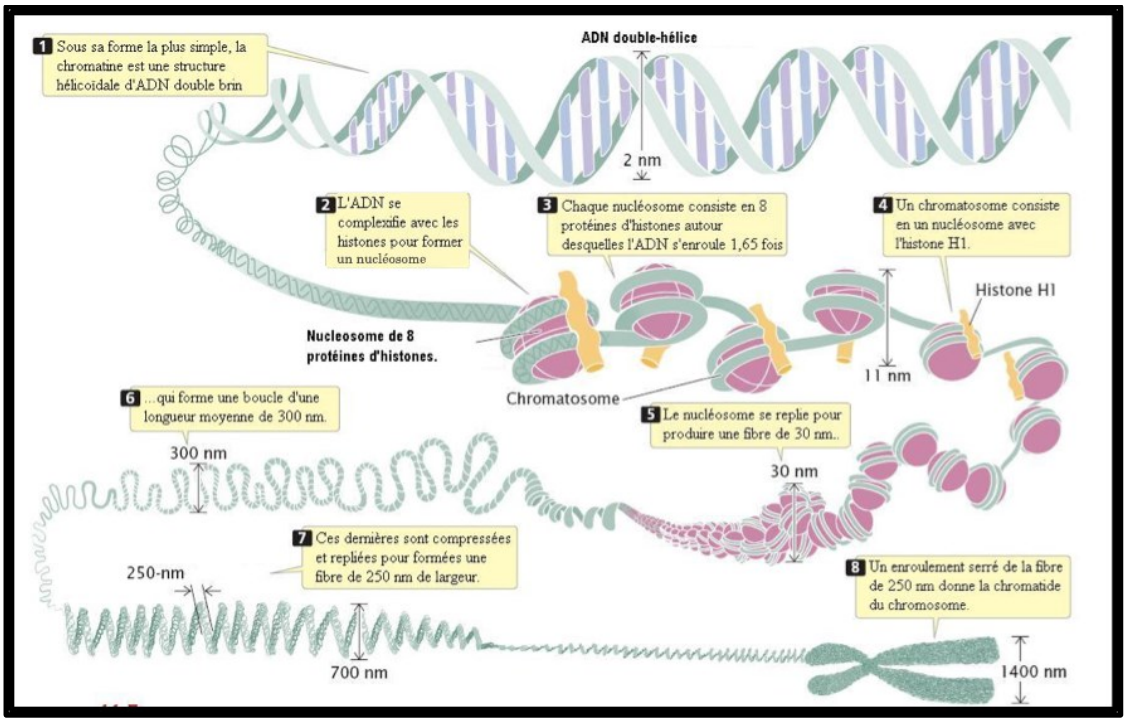

Figure 1.3 Composantes et étapes de repliement de la chromatine. Adapté de Annunziato, 2008.

1.1.2 Nucléosome

L'unité fonctionnelle de la structure chromatinienne est le nucléosome, une structure de huit protéines d'histones autour de laquelle s'enroule l'ADN. Cet octamère est composé de deux hétérodimères de H2A et H2B et d'un tétramère de H3 et H4. Ces protéines d'histones sont dites canoniques et sont présentes dans la majorité des nucléosomes.

Les nucléosomes sont présents de manière dense dans les génomes eucaryotes. La théorie initiale voulant que les nucléosomes aient un rôle quasi restreint au compactage s'est avérée limitée (Mueller-Planitz, Klinker et Becker, 2013). En fait, les nucléosomes jouent un rôle

essentiel dans le bon fonctionnement de la cellule (Jiang & Pugh, 2009). En bloquant l'accès à l'ADN, un nucléosome peut empêcher le recrutement d'éléments à ces positions et agir comme un frein à l'expression génique. Le positionnement des nucléosomes et les complexes de remodelage qui déplacent et organisent leurs positionnements sont ainsi des éléments essentiels de la régulation génique. La composition et les modifications post-traductionnelles des nucléosomes sont aussi, sinon plus, importantes.

1.1.3 Modifications et variantes d'histones

Les histones qui composent un nucléosome sont sujettes à de nombreux mécanismes qui les modifient. Spécifiquement, la modification post-traductionnelle et le remplacement des histones par des histones variantes sont deux mécanismes essentiels.

Le remplacement d'une histone canonique par une variante d'histone a lieu grâce à un processus de remodelage où l'histone canonique éjectée du nucléosome est remplacée par la variante. Cette nouvelle histone peut présenter des caractéristiques différentes telles que les modifications qu'elle peut recevoir ou affecter le recrutement potentiel de certaines protéines (Hake & Allis, 2006). Elle modifie directement les caractéristiques de la chromatine et certaines variantes jouent des rôles majeurs dans plusieurs mécanismes de la transcription. À titre d'exemple, la variante d'histone H2AZ (Zlatanova & Thakar, 2008 ; Gévry *et al*., 2009) est liée à plusieurs rôles essentiels dans le système humain, tels que le fonctionnement des éléments amplificateurs liés à l'estrogène et la stabilisation des régions hétérochromatiques (Meneghini *et al.*, 2003).

Chaque histone est sujette à plusieurs types de modification tels que l'acétylation, la méthylation et l'ubiquitination, en particulier sur leur extrémité N-terminale. Ces

modifications affectent des résidus spécifiques des histones (par exemple, H3K4me3 représente la tri-méthylation de la lysine en position quatre de l'histone H3) et la combinaison des types de modification avec le nombre de positions différentes que ces modifications peuvent affecter crée un très grand nombre de marqueurs pouvant moduler la régulation.

Ces modifications vont, entre autres, permettre le recrutement de régulateurs transcriptionnels et de complexes de remodelage de la chromatine. Souvent une modification donnée sera associée à l'activation (H3K27ac, H3K4me3) ou la répression (H3K27me3, H3K9me3) des gènes, mais il est possible de trouver des modifications qui peuvent jouer les deux rôles (Vakoc *et al.*, 2005). De plus, les modifications d'histones présentent des distributions distinctes dans le génome. Certaines ne sont présentes qu'à des endroits spécifiques, tandis que d'autres occupent de larges régions continues du génome et ces mêmes distributions peuvent être affectées par la présence de protéines qui bloquent leur étalement (par exemple, CTCF (Cuddapah *et al.*, 2009)

1.2 Applications et défis du séquençage haut débit de l'ADN

Dans la section précédente, nous avons survolé plusieurs éléments essentiels de la chromatine. Si nous en discutons dans une thèse dont le sujet est le développement d'outils bio-informatiques, c'est que ces éléments sont les cibles fréquentes de plusieurs applications devenues possibles grâce aux avancements des technologies de séquençage. L'évolution de ces technologies a permis de révolutionner nos connaissances de certains éléments de la chromatine en permettant aux chercheurs de les étudier dans l'entièreté du génome en une seule expérience.

Cependant, avec cette opportunité de nouveaux défis se sont ajoutés. Ces mêmes technologies produisent en effet des quantités gigantesques de données qu'il faut traiter à l'aide d'outils bio-informatiques. Pour bien saisir l'importance de ces développements en génomique, il faut comprendre les avancées visées par l'utilisation des nouvelles technologies de séquençage ainsi que les nouveaux défis survenus.

Dans cette section, nous commencerons par un bref résumé des étapes d'une expérience de séquençage haut débit. Ensuite, nous décrirons certaines applications que les chercheurs font de cette technologie avec un focus particulier sur les expériences en liens avec l'épigénomique. Puis, nous décrirons plusieurs des difficultés qui compliquent la réalisation de ces recherches, dont certaines sont des motivateurs pour le développement d'outils bio-informatiques adaptés à la génomique.

1.2.1 Résumé d'une expérience de séquençage haut débit

Les nouvelles applications découlant du développement des technologies de séquençage ont, malgré leurs objectifs différents, des traits généraux semblables tant au niveau de la production des données que des étapes préliminaires aux analyses. Dans cette section, nous détaillons brièvement certaines de ces étapes avec une attention particulière aux biais qu'elles peuvent introduire dans l'analyse de l'expérience. À noter que cette section n'est aucunement exhaustive et qu'il existe de nombreux articles qui traitent en détails des biais identifiés jusqu'à présent (Schirmer *et al.*, 2015 ; Meyer et Liu, 2014 ; Tyler *et al*., 2016 ; Taub *et al.*, 2010). La vitesse des avancées techniques du domaine nécessite une mise à jour constante de ses connaissances. Par ailleurs, l'existence de biais extrêmement ciblés (potentiellement associés à un équipement ou une méthode spécifique) rend difficile d'offrir plus d'un aperçu très global de la problématique. Finalement, notons que nous traitons des étapes les plus communes ; des

applications spécifiques (tel que ChIP-Seq discuté plus loin) auront des étapes supplémentaires ou modifiées.

1.2.1.1 Préparation de l'échantillon

L'étape de préparation de l'échantillon vise à fournir le matériel génétique que le chercheur veut étudier. Par définition, la préparation varie énormément selon le protocole utilisé et l'objectif de recherche. Faire une revue exhaustive des protocoles pour la préparation de l'échantillon est hors du cadre de ces travaux. Cependant, il demeure essentiel de mettre l'accent sur l'importance de choisir le bon protocole et de valider le succès de celui-ci. Si l'échantillon fourni est endommagé, contaminé ou qu'une autre erreur sérieuse a été commise, le processus de séquençage pourrait être compromis. Parfois, une telle erreur ne sera réalisée que suite à l'analyse de la fin du processus de séquençage, un évènement qui signifie la perte de plusieurs semaines de travail et d'une quantité non-négligeable d'argent de la part du chercheur.

1.2.1.2 Préparation de la librairie

Dans le présent contexte, une librairie est l'ensemble de fragments d'ADN fourni à l'appareil de séquençage et visant à représenter l'échantillon. Suite à la préparation de l'échantillon, l'objectif de l'étape de préparation de la librairie est de fournir une librairie de haute complexité (qui représente bien l'échantillon) tout en minimisant l'introduction de biais (duplication massive, sélection trop spécifiques de certaines régions d'ADN, etc.). La préparation passe, entre autres, par les étapes suivantes : fragmentation de l'ADN, ajout d'adaptateurs, sélection de la taille des fragments et multiplication des fragments par PCR.

Une fois l'ADN de l'échantillon fragmenté, il y a souvent une étape de sélection des fragments où seulement un pourcentage des fragments de chaque longueur est conservé. Les critères de sélection varient selon l'échantillon, les besoins du protocole et la machine de séquençage (Head *et al.*, 2015). Cependant, sachant que la distribution de la taille des fragments produits par la fragmentation varie selon le type de fragmentation utilisée et la structure de l'ADN, la sélection de la taille des fragments peut avoir une influence sur la représentativité des fragments qui sont conservés. Une mauvaise sélection peut biaiser la distribution en faveur d'une certaine caractéristique et possiblement diminuer la complexité de la librairie ou augmenter son biais (choisir d'éliminer les fragments longs pourrait par exemple enlever certaines régions où l'ADN est difficile à fragmenter) (Meyer & Liu, 2014).

L'étape de PCR permet de multiplier les fragments d'ADN sélectionnés avec comme objectif d'atteindre une quantité suffisante pour le séquençage. Cette étape introduit aussi des biais. Si la quantité de matériel fournie au PCR est faible, il est probable qu'un plus grand nombre de cycles dans l'amplifications soit nécessaire. Dans ce cas, les biais des étapes précédentes seront multipliés et la distribution des fragments va tendre vers ces biais. De plus, les fragments GC-neutres ont tendance à être multipliés à un rythme supérieur aux fragments GC-riches ou AT-riches, ce qui biaise la distribution des fragments produits par l'amplification (Meyer et Liu, 2014). Il existe également des biais introduits par la méthode spécifique (équipement de PCR, produit et approche) utilisée pour accomplir les phases d'amplification (Dabney & Meyer, 2012).

Les biais introduits sont multiplicatifs et affecteront directement le résultat de l'étape de séquençage. Il est donc crucial pour un chercheur de faire un travail méticuleux durant la préparation de la librairie et de noter les biais probables pour qu'ils soient contrôlés à l'étape d'analyse.

1.2.1.3 Séquençage

L'objectif de l'étape de séquençage est de fournir plusieurs millions de fragments appelés lectures (*reads*). Chaque lecture contient une séquence d'ADN qui a été lue par le séquenceur à partir d'un fragment de la librairie. Notez qu'en général, la longueur de la séquence lue est beaucoup plus courte que la longueur du fragment d'ADN (par exemple, le séquenceur lit les 50 premières pb d'un fragment de 200 pb). Si la librairie fournie au séquenceur est suffisamment complexe, l'étape de séquençage devrait bien représenter l'échantillon d'origine. Cependant, tout comme les étapes précédentes, le séquençage est sujet à ses propres sources de biais (Ross *et al.*, 2013).

La mauvaise lecture d'une base d'ADN est la source de biais la plus évidente dans le processus de séquençage. L'amélioration des technologies de séquençage a progressivement réduit ces erreurs, mais la quantité de lectures générées est si immense que même un faible taux d'erreurs implique un grand nombre absolu (par exemple, sur 40 millions de lectures de 40 bp, un taux d'erreur de 0.01% représente grossièrement 16 000 bases erronées). Des outils existent pour corriger spécifiquement ce type d'erreur (Kelley, Schatz, & Salzberg, 2010 ; Greenfield, Duesing, Papanicolaou, & Bauer, 2014).

Tout comme l'étape de PCR, le séquençage démontre un biais par rapport à la composition GC de l'ADN traité (Benjamini & Speed, 2012). De plus, des biais sont possibles selon le motif du fragment à séquencer, la présence de palindrome, la composition des extrémités et les particularités spécifiques de la technologie de séquençage utilisée (Ross *et al*., 2013).

L'étape de séquençage a la particularité d'être généralement hors des mains du chercheur qui performe l'expérience. Si la préparation de la librairie et l'analyse des résultats sont souvent accomplies par l'équipe de recherche, le séquençage lui-même est presque toujours effectué par un centre externe qui reçoit la librairie et fournit en retour des fichiers contenant les lectures.

Pour aider à contrôler ces biais, il est commun pour un chercheur de fournir des duplicatas de la librairie préparée ainsi qu'une librairie de contrôle (*input).* Les séquençages des duplicatas peuvent être comparés entre eux et comparés aux résultats du contrôle pour assister à la détection de problèmes ou de biais suite au séquençage. Il reste qu'un contrôle de qualité robuste suite à la réception des fichiers est de mise, par exemple en utilisant l'outil FastQC (Andrew, 2010). Notons finalement que si un biais significatif est détecté suite au séquençage, il peut être difficile de déterminer quelles étapes en sont responsables.

Les étapes de préparation précédentes sont résumées dans la Figure 1.4. Le résultat de ces étapes sera analysé par des outils bio-informatiques. Il forme la base essentielle de l'information obtenue par l'expérience. L'aspect du traitement de ces données sera abordé à la section 1.2.6.

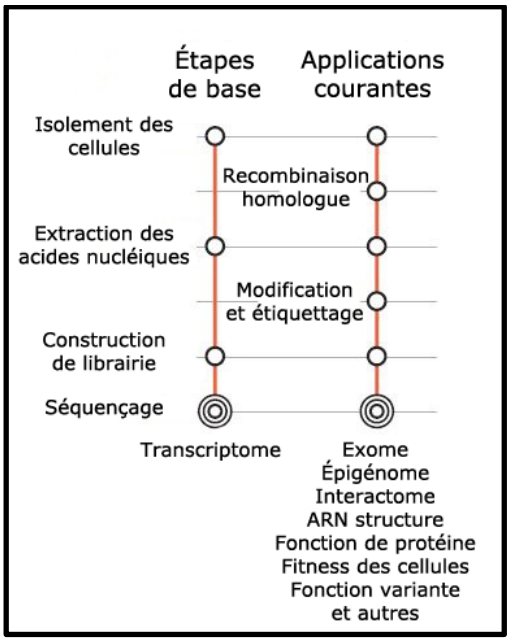

Figure 1.4 Sommaire des étapes pré-séquençage pour différentes applications du séquençage haut débit. Adapté de Shendure & Lieberman Aiden, 2012.

1.2.2 Étude de la localisation des protéines et des modifications d'histones

L'identification à l'échelle du génome de la localisation des protéines et des modifications d'histones est essentielle pour une compréhension accrue des systèmes de régulation de la transcription (Farnham, 2009).

Les modifications d'histones et la liaison des protéines dans le génome ne sont pas statiques. En plus de pouvoir étudier leur localisation, il est possible, en faisant plusieurs expériences, d'étudier le mouvement de ces éléments dans des conditions différentes. Ceci permet de tirer des conclusions sur le rôle dynamique que jouent ces éléments dans la régulation de la transcription.

La technique ChIP (*Chromatin immunoprecipitation*) (Gilmour & Lis, 1985) a été pendant plusieurs années la méthode de choix pour l'identification des interactions ADN-protéines. Sommairement, l'ADN et la protéine d'intérêt sont liés ensemble de manière réversible. Suite à la liaison, la chromatine est fragmentée par digestion ou sonication créant un ensemble de fragments de 200 à 1000 paires de bases. Les complexes ADN-protéine sont immunoprécipités en utilisant un anticorps spécifique à la protéine d'intérêt, le pontage est renversé, les protéines sont digérées et l'ADN est purifiée. Ensuite, une variété de techniques existe pour identifier les séquences d'ADN associées à la protéine précipitée dont entre autres, le qPCR, les puces à ADN (ChIP-chip) et la technique qui nous intéresse particulièrement, par séquençage (ChIP-Seq) (Park, 2009).

Le ChIP-Seq est une des premières applications issues du séquençage haut débit. Elle est utilisée pour identifier la localisation de ces éléments à l'échelle du génome à une précision théorique d'une paire de base. Ciblant une protéine ou modification d'histone spécifique, l'expérience produit des millions de lectures qui sont utilisées pour ensuite générer la carte de positionnement de la cible. Le processus général de la méthode est décrit à la Figure 1.5.

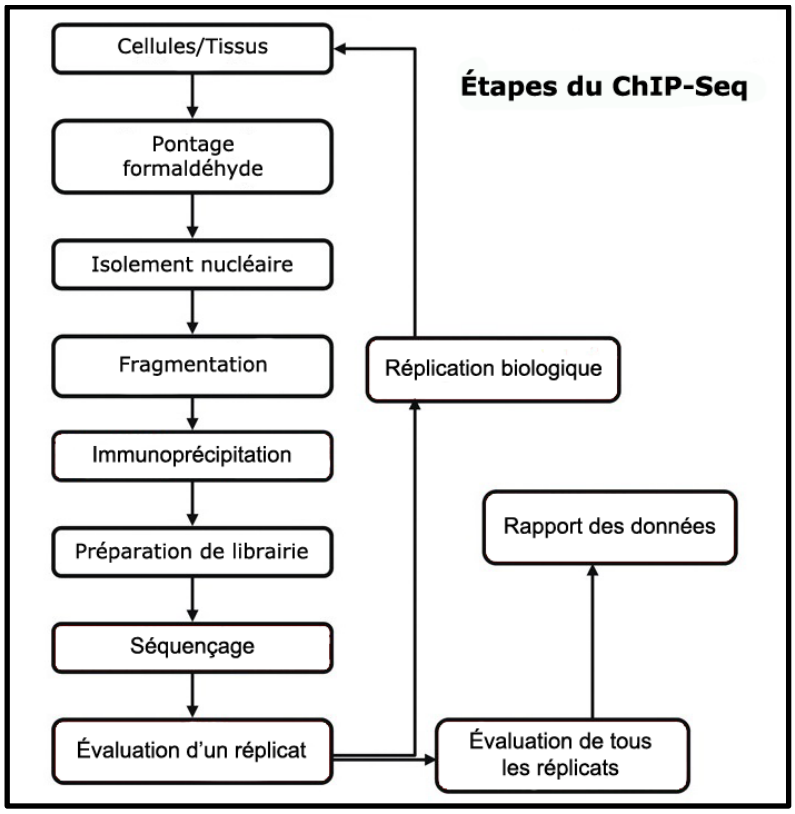

Figure 1.5 Flot d'une expérience ChIP-Seq. Adapté de Landt *et al.*, 2012.

Une des premières grandes expériences tirant parti du ChIP-Seq fut l'étude de plus d'une dizaine de modifications et variantes d'histones par le laboratoire de Zhao (Barski *et al.*, 2007). Depuis, la méthode a été largement utilisée. Par exemple, une recherche sur la base de données *Short Read Archive* du terme « ChIP-Seq » retourne plus de 62 000 résultats en date de la soumission de cette thèse. Il est incontournable pour un bio-informaticien œuvrant en génomique d'avoir à travailler régulièrement avec des données issues du ChIP-Seq. Les biais et les difficultés associés à l'analyse informatique de cette méthode ont déjà fait l'objet d'une étude poussée (Diaz, Park, Lim, & Song, 2012) que nous résumerons en partie dans les sections 1.2.6 et 1.3.

1.2.3 Étude du positionnement des nucléosomes

La présence ou l'absence d'un nucléosome à un endroit donné est un élément déterminant de l'accessibilité à l'ADN. Il est bien connu que la présence des nucléosomes dans le génome n'est pas uniforme et que ceux-ci sont moins présents aux niveaux de plusieurs éléments de régulation tels que les promoteurs et terminateurs (Struhl & Segal, 2014).

L'étude du positionnement des nucléosomes existait déjà avant le séquençage haut débit, mais la tâche était laborieuse. Considérant la présence d'un nucléosome potentiel à tous les 167 pbs et que, par exemple, le génome humain fait environ 3 milliards de paires de bases, le nombre potentiel de nucléosomes à positionner se situe dans les millions. La possibilité d'identifier l'ensemble des nucléosomes d'un coup a ouvert la possibilité d'étudier les déterminants du positionnement des nucléosomes de manière beaucoup plus précise (Figure 1.6) et de comparer les différents motifs possibles (Gaffney *et al*., 2012).

Notons que cette même densité est responsable d'une des particularités des expériences nécessaires pour les étudier. S'il est possible de cibler les nucléosomes à l'échelle d'un génome entier, il faut substantiellement plus de lectures que pour une expérience plus circonscrite ciblant par exemple un facteur de transcription. Plutôt que quelques millions de lectures pour un génome de la taille de l'humain, il faut minimalement des dizaines de millions de lectures pour offrir une résolution suffisante pour les analyses nécessaires (Chen *et al.*, 2013).

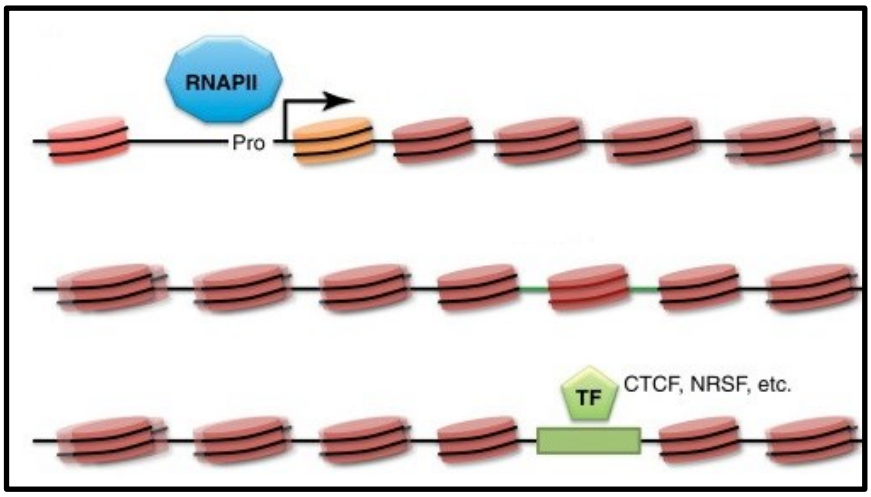

Figure 1.6 Exemple de positionnement de nucléosomes. Les cercles enroulés représentent des nucléosomes qui bloquent l'accès à l'ADN. Les nucléosomes flous représentent des nucléosomes mobiles et les figures sont des protéines ou complexes liés à l'ADN dans des zones sans nucléosome. Adapté de Iyer, 2012.

1.2.4 Étude du transcriptome

Le transcriptome est l'ensemble de toutes les molécules d'ARN (incluant les ARNt, ARNm, ARN non-codant et les autres) retrouvées dans une cellule. Le RNA-Seq est désormais une méthode couramment utilisée dans l'étude du transcriptome (Ozsolak & Milos, 2011). Il permet entre autres d'identifier la présence et la densité de tous les transcrits dans l'ensemble du génome pour une condition donnée. Ceci permet d'identifier précisément les régions transcrites et leur niveau d'activité, de délimiter plusieurs types d'éléments, tels que les introns/exons, et d'utiliser cette information pour inférer plusieurs détails fonctionnels sur le génome.

1.2.5 Étude du repliement chromatinien

La conformation de la chromatine dans l'espace joue également un rôle dans la régulation de la transcription. On observe que dans certain cas, la chromatine adopte une conformation qui permet à deux éléments distaux de se lier (Figure 1.7) et que certaines protéines, telle que la protéine CTCF, sont fortement associées à ce type de régulation distale (Ferraiuolo, Sanyal, Naumova, Dekker, & Dostie, 2012). C'est un mécanisme de plus par lequel la chromatine peut gérer la transcription. De plus, en adoptant une conformation plus ou moins refermée, la chromatine peut faciliter ou rendre plus difficile l'accès à des régions du génome.

L'étude du repliement chromatinien à l'échelle du génome peut se faire avec des méthodes telles que le ChIA-Pet (Fullwood *et al.*, 2009) et le Hi-C (Belton *et al.*, 2012). Ces méthodes permettent d'associer des groupes de lectures à des positions distales et d'inférer la liaison de ces régions. Cependant, ces méthodes sont particulièrement susceptibles au bruit expérimental du séquençage et nécessitent des analyses avancées ainsi que des outils bio-informatiques spécialisés. À titre d'exemple, dans une expérience de ChIA-PET visant les sites de liaisons distaux du récepteur d'estrogènes ER-α, les données initiales proposaient un peu plus de 19 000 sites de liaison, tandis que les résultats après nettoyage du bruit et analyse statistique chiffraient le nombre à un peu plus de 1000 (Fullwood *et al.*, 2009).

Figure 1.7 Exemple de repliement chromatinien. Adapté de Talbert et Henikoff, 2006.

1.2.6 Imprécision de la technique du ChIP-Seq

Dans un idéal, l'information extraite d'une expérience de séquençage serait certaine, c'est-à-dire qu'une expérience de ChIP-Seq (par exemple) nous renverrait la localisation exacte et précise de toutes les protéines visées par l'expérience pour la cellule utilisée. Le chercheur aurait donc une sorte de carte de localisation protéique qui lui dirait de manière absolue si, pour une position donnée du génome, la protéine est présente ou absente. Ensuite, il pourrait séquencer d'autres cellules pour faire une analyse différentielle dans le temps ou combiner sa carte avec d'autres cartes de différentes protéines.

La réalité est beaucoup plus complexe. Il est commun de représenter partiellement les résultats d'une expérience par une figure qui ressemble souvent à celle-ci (Figure 1.8).

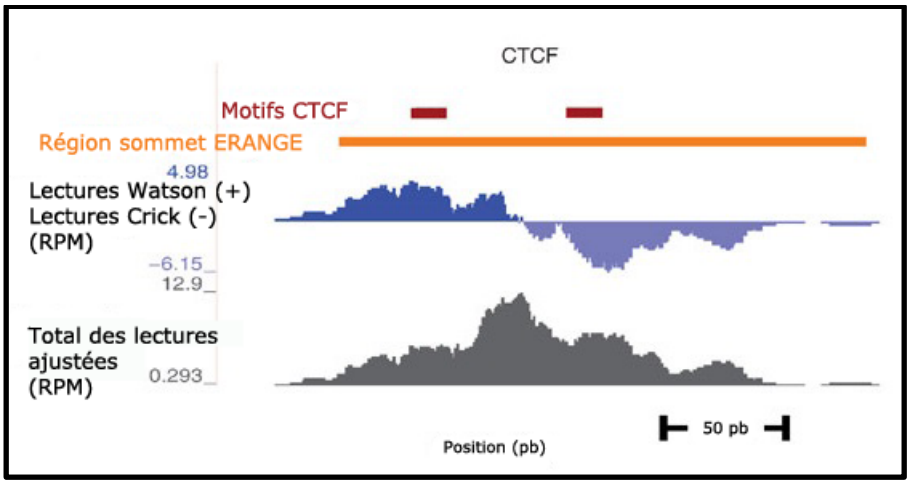

Figure 1.8 Représentation de résultat de séquençage. Les signaux représentent le nombre de lectures à ces positions génomiques. La barre jaune est un pic trouvé par l'outil d'analyse ERANGE et les carrés rouges sont des motifs dans la séquence d'ADN du génome connus pour être liés par la protéine CTCF. Adapté de Pepke, Wold & Mortazavi, 2009.

Plutôt qu'une carte qui nous indique la présence ou l'absence d'une protéine (CTCF dans cet exemple), nous avons un signal continu à valeur variable qui s'étend sur plusieurs centaines de paires de bases. Pourtant, par d'autres types d'expériences il est possible de prouver que CTCF ne lie que quelques dizaines de pb sur l'ADN. Alors, pourquoi les chercheurs doivent-ils présentement travailler avec un signal plutôt qu'une carte de localisation binaire?

1.2.6.1 Données provenant d'une population de cellules

En premier lieu, il faut rappeler que dans la majorité des expériences de séquençage (le séquençage à cellule unique étant une exception (Eberwine, Sul, Bartfai, & Kim, 2014), les données proviennent d'une population de cellules (Shendure & Lieberman Aiden, 2012). Nous avons déjà abordé la mobilité des protéines et des nucléosomes dans le génome. Il en découle que dans une expérience visant une protéine spécifique, cette protéine pourrait ne pas être exactement à la même position dans les millions de cellules utilisées pour l'expérience. Dans certain cas, le mouvement sera mineur (quelques paires de bases) et ceci contribue aux portions du signal qui ressemblent à un pic ou une distribution gaussienne. Dans d'autres situations, il y aura un mouvement important, souvent dû à un mécanisme de régulation. Ceci contribue à la situation où le signal présente deux pics suffisamment proches pour rendre improbable la présence de deux protéines distinctes mais suffisamment distincts pour avoir un creux entre les pics. Dans le cas d'une expérience ciblant certaines modifications d'histones, la densité des nucléosomes et la présence de ces modifications sur de larges espaces continus du génome (par exemple, H3K27me3) donne un signal qui ressemble plus à un flot continu qu'à des signaux en pics (Figure 1.9).

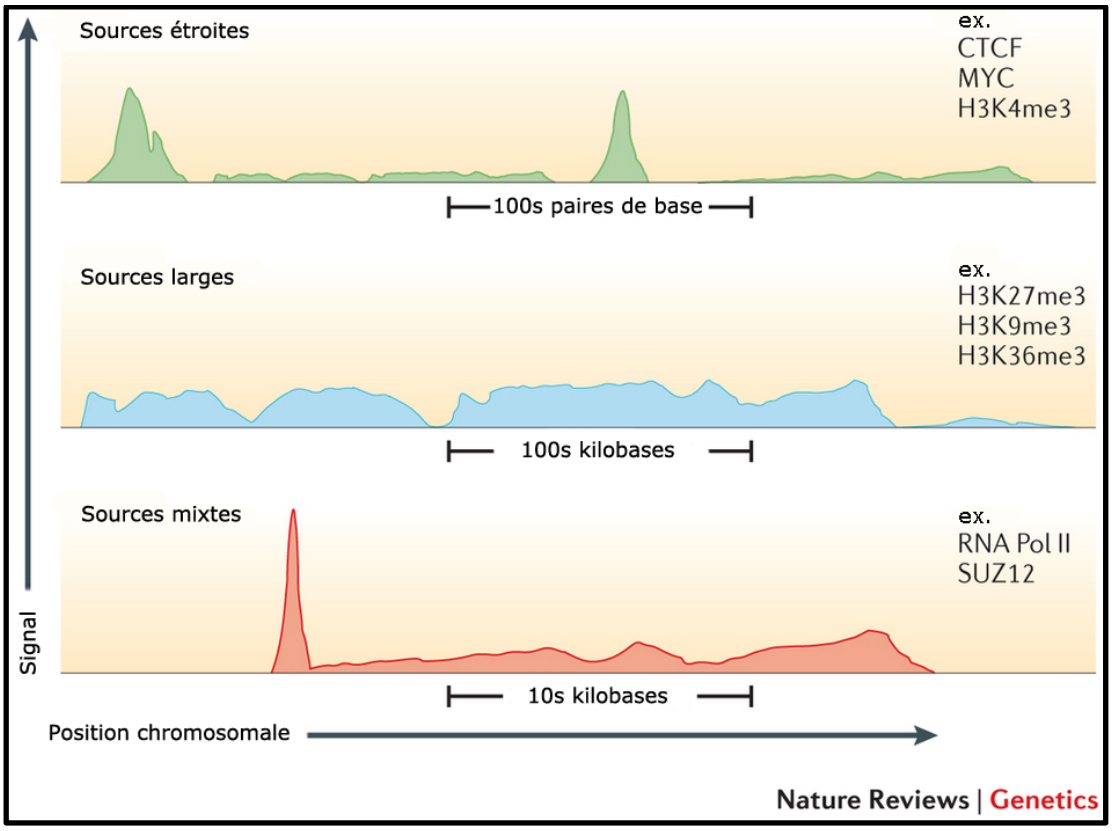

Figure 1.9 Exemple des multiples formes du signal génomique. Adapté de Sims, Sudbery, Ilott, Heger, & Ponting, 2014.

Malgré les complications causées par la forme de ce signal, il s'agit également d'une opportunité. Un chercheur qui travaillerait avec une carte de localisation provenant d'une seule cellule n'aurait aucune information sur la mobilité des éléments qu'il étudie. Étudier les données provenant d'une population de cellules permet de déterminer à quel point l'élément étudié est présent et/ou mobile, offrant des informations supplémentaires (Figure 1.10). Néanmoins, cette réalité ajoute une complexité de plus à l'interprétation des résultats de séquençage.

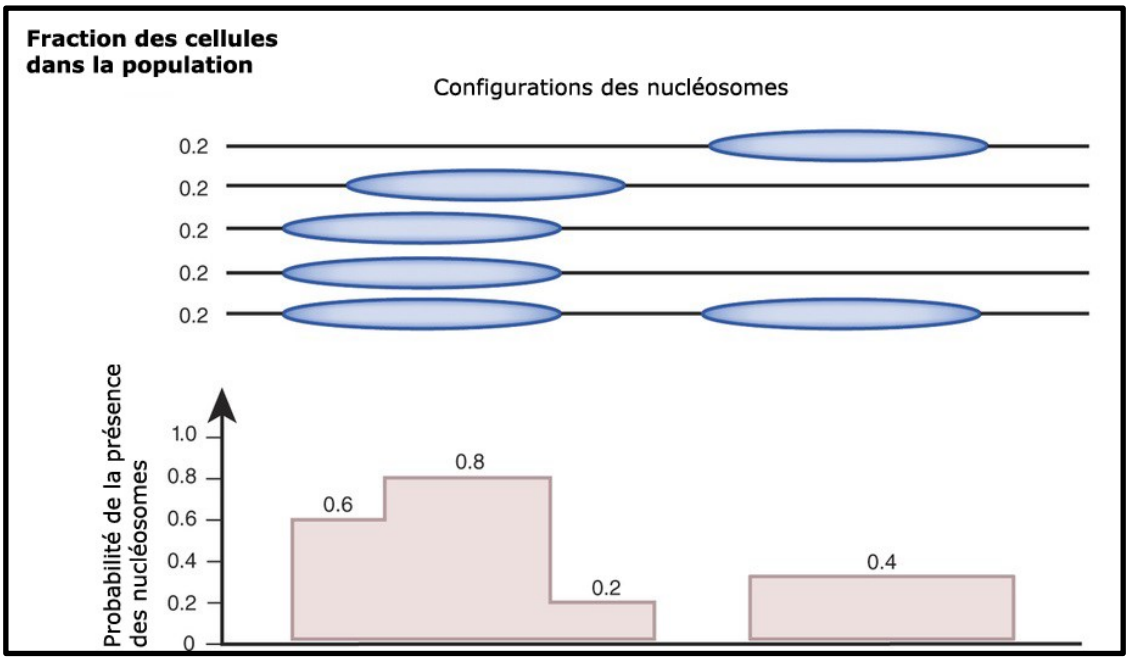

Figure 1.10 Exemple d'analyse possible ciblant la mobilité des nucléosomes. Adapté de Struhl & Segal, 2014.

1.2.6.2 Imprécision de la fragmentation

Tel que discuté précédemment, les données provenant d'expériences utilisant le séquençage sont actuellement intrinsèquement bruitées et ce bruit vient de multiples sources : le protocole de préparation de l'expérience, le type de machine utilisée, la taille des fragments, le type d'expérience, la structure de la chromatine aux endroits ciblés, la composition de l'ADN et d'autres (Benjamini & Speed, 2012 ; Zheng, Chung & Zhao, 2011). Certaines sources de bruits ont été fortement étudiées et plusieurs outils existent pour aider le chercheur à corriger, avec plus ou moins de succès, certains de ces bruits (Meyer & Liu, 2014).

En plus de l'imprécision causée par l'utilisation de population de cellules, il est important de discuter de l'imprécision due à la fragmentation et la taille inégale des brins d'ADN induits par la sonication. En effet, sans entrer dans les détails, il est nécessaire durant le processus du séquençage de casser l'ADN en petits segments, généralement autour de 200 paires de bases (Schmidt *et al*., 2009). Plusieurs expériences de séquençage utilisent ainsi la sonication (Knierim, Lucke, Schwarz, Schuelke, & Seelow, 2011) pour casser l'ADN, une méthode efficace mais imprécise en ce sens que les cassures peuvent se produire à presque n'importe quel endroit dans la chromatine non protégée par un nucléosome. En conséquence, une cassure produite dans la même région du génome dans deux cellules distinctes a très peu de chance d'être située au même endroit et peut ainsi générer des lectures dont les positions divergent de plusieurs dizaines de paires de bases. Cette imprécision contribue à la forme vague des pics de densité et assure que même pour une protéine liée de façon identique dans chaque cellule, le signal génomique l'identifiant resterait quand même imprécis. Naturellement, cette forme de bruit complique encore plus l'identification exacte des éléments et vient spécifiquement nuire à l'étape d'analyse informatique.

Outre l'amélioration des méthodes d'analyse, le raffinement des méthodes de séquençage est une avenue prometteuse pour réduire l'imprécision des méthodes actuelles. Par exemple, le Chip-Exo (Rhee et Pugh, 2011) est une méthode qui utilise une exonucléase pour offrir une résolution d'une paire de base. Bien que sujet à certains types de bruits spécifiques à la méthode, il s'agit d'une nette amélioration au niveau de la clarté tout en nécessitant moins de lectures que le Chip-Seq.

1.3 Étapes post-séquençage

Dans cette section, nous décrirons brièvement deux des étapes typiques du flot d'analyse d'une expérience de séquençage chez un organisme dont le génome de référence est disponible (comme l'humain), soit l'alignement des lectures et l'analyse des résultats tel que l'identification des sites enrichis (Kharchenko, Tolstorukov, & Park, 2008). Ces étapes sont présentées ici afin de promouvoir une compréhension globale du flot d'analyse d'une expérience de séquençage et de l'importance de la manipulation des lectures dans le développement d'outils bio-informatiques.

1.3.1 Alignement des lectures

Un séquençage typique fournit des lectures entre 35 et 150 nucléotides qu'il faut ensuite aligner sur le génome de référence lorsque ce dernier est disponible afin de trouver l'origine du fragment séquencé. Lorsqu'un génome de référence n'est pas disponible, il faut plutôt assembler les lectures pour former des segments continus (hors du contexte de cette thèse).

Intuitivement, cette tâche peut se résumer à la recherche d'une sous-séquence dans une séquence donnée, un problème classique en informatique qui a déjà été bien étudié (Karp & Rabin, 1987). Cependant, la tâche spécifique de recherche d'un relativement petit fragment séquencé dans un génome de référence amène son lot de complications propres (H. Li, Ruan, & Durbin, 2008).

Tel que discuté précédemment, le processus de séquençage des fragments introduit en effet un certain taux d'erreurs dans les lectures produites. Il est possible que la machine de séquençage attribue la mauvaise base à un nucléotide et cette probabilité augmente vers la fin du fragment.

Typiquement, les machines attribuent une probabilité d'erreur à chaque base du fragment séquencé et cette probabilité a avantage à être intégrée dans les stratégies d'alignement.

Plusieurs équipes se sont intéressées à ces difficultés en développant une variété d'outils qui effectuent l'alignement (H. Li & Homer, 2010). MAQ (H. Li, Ruan, & Durbin, 2008) utilise une technique de hachage pour accélérer l'alignement des séquences et permet des erreurs dans les fragments. BOWTIE (Langmead, Trapnell, Pop, & Salzberg, 2009) et BWA (H. Li & Durbin, 2009) utilisent des transformées de Burrows-Wheeler pour limiter l'empreinte mémoire nécessaire et accélérer l'alignement des fragments et SOAP2 (R. Li et al., 2009) utilise une stratégie similaire à BOWTIE, mais qui permet l'alignement de fragments d'expériences sens et anti sens. Ces outils permettent l'alignement de millions de fragments dans un temps acceptable et avec des résultats d'exactitudes similaires (H. Li & Homer, 2010).

Il existe de multiples types d'analyses possibles pour transformer le signal génomique produit par l'alignement des lectures en information permettant aux chercheurs de répondre aux questions biologiques à la base de l'expérience ayant générée ces données. Cependant, malgré la publication de dizaines d'outils spécialisés dans l'analyse de données de séquençage, les chercheurs doivent souvent écrire leurs propres programmes ou scripts d'analyse pour répondre à leurs besoins spécifiques.

Nous détaillons ici certaines des analyses communes en offrant un exemple d'outil pour chacune. Cette liste n'est qu'un bref aperçu des analyses possibles pour les types d'expériences.

1.3.2.1 Visualisation des données

L'analyse visuelle des données, bien que faible en rigueur statistique, est une tâche récurrente. La visualisation permet de s'assurer que les données n'offrent pas de biais évidents qui auraient échappés aux étapes de vérification et peut aider le chercheur à répondre à des questions sur des régions spécifiques du génome.

Le *UCSC Genome Browser* (Kent *et al.*, 2002) est un des premiers et encore un des plus populaires outils de visualisation de données génomiques disponibles. Accessible via une interface web, l'outil est intégré avec de nombreuses bases de données, permettant de faire une analyse et comparaison visuelle de nombreuses expériences différentes tout en permettant au chercheur de télécharger ses propres données dans l'outil. À noter que le format de donnée BigWig, sujet clé du chapitre 3, a été inventé spécifiquement pour faciliter l'interaction avec le *UCSC Genome Browser*.

Bien que critiqué pour son rendu visuel, la complexité de son interface et la lenteur relative de la visualisation, l'outil combine un excellent accès à des données avec des nombreuses options très prisées.

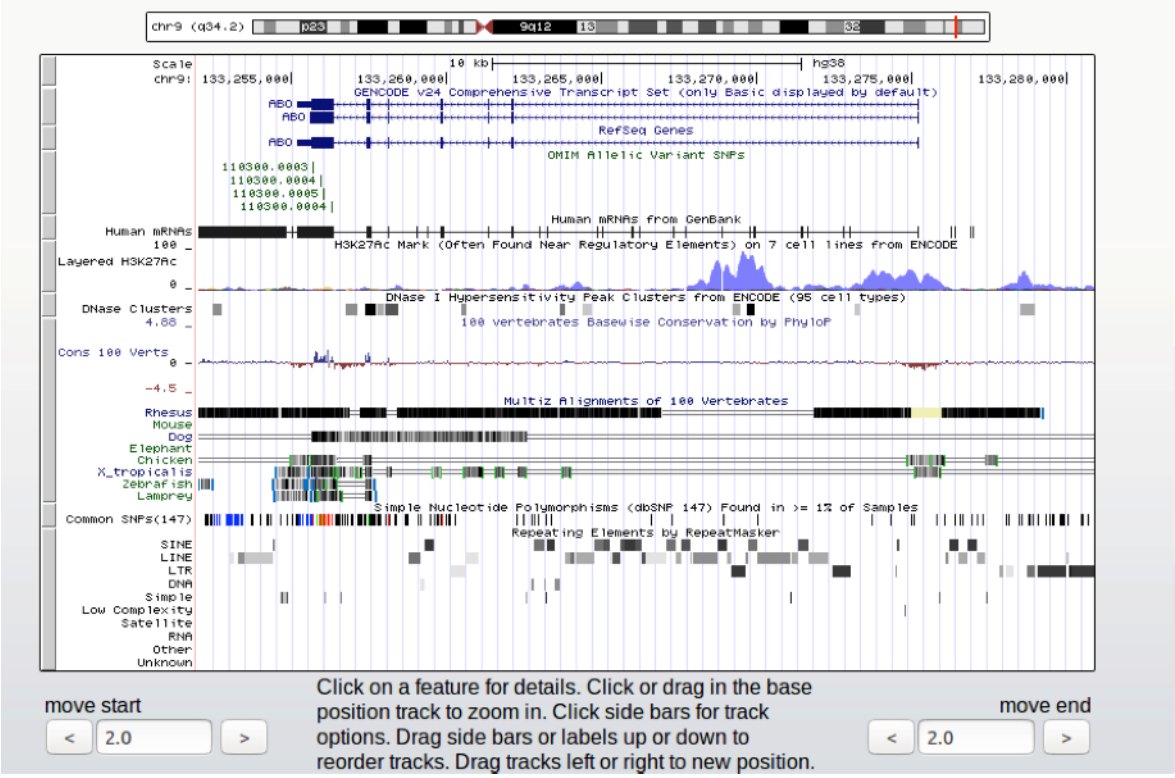

Figure 1.11 Exemple de la visualisation par le *UCSC Genome Browser*.

1.3.2.2 Détection des pics et zones enrichies

La détection des zones enrichies et une des analyses les plus communes utilisant des données de séquençage. Malgré qu'associée à plusieurs types d'expériences, elle est particulièrement populaire dans les cadres des données de ChIP-Seq où les sites de liaisons sont parfois relativement bien espacés les uns des autres.

L'outil *Model-based analyses of ChIP-Seq* (MACS) (Zhang *et al*., 2008) a été un des premiers outils de détection à utiliser des modèles statistiques pour identifier les sites enrichis.

Largement utilisé et intégré dans plusieurs plateformes, MACS est un des outils incontournables dans le domaine.

Pour identifier les sites enrichis, MACS évalue la quantité de fragments dans des fenêtres successives selon une loi de Poisson et identifie les fenêtres qui contiennent une quantité significative de fragments. L'outil combine les fenêtres adjacentes pour créer des régions enrichies et sélectionne la fenêtre avec le plus de fragments comme sommet pour la région. Pour éviter les biais d'enrichissements régionaux, MACS évalue plusieurs distributions de Poisson calculées selon des zones génomiques de tailles différentes. Une fois les distributions calculées, celle offrant la distribution la plus dense est utilisée pour mesurer l'enrichissement des sites de la région correspondante, permettant ainsi d'incorporer les biais régionaux dans la détection.

1.3.2.3 Détection des nucléosomes

Malgré un objectif similaire à la détection des zones enrichies, la détection des nucléosomes est compliquée par la proximité des sites d'intérêts. Avec un faible espace entre chaque nucléosome, le bruit du signal et la nécessité de situer les nucléosomes de manière très précise, les outils habituels de détections des pics offrent de faible résultat sur ces données.

L'outil nucleR (Flores et Orozco, 2001) est disponible via Bioconductor dans le langage R. L'outil base sa méthode d'analyse sur l'utilisation des transformations de Fourier pour nettoyer le signal et faciliter la détection des sites denses. Spécifiquement, l'outil utilise le fait que le signal d'une expérience de nucléosome est relativement régulier. Transformer un tel signal dans l'espace de Fourier permet de diviser les éléments du signal qui offre de l'information utile et d'enlever les éléments du signal qui contribue du bruit.

Une fois le bruit enlevé des données, nucleR cible le sommet des régions et renvoie une liste des sites associés. L'intégration avec le langage R permet de passer directement de cette étape d'analyse à des analyses subséquentes avec les données.

1.3.2.4 Corrélation de données génomiques

Vérifier si deux expériences ont des signaux qui corrèlent est une question très informative. La corrélation des expériences peut par exemple aider le chercheur à trouver des erreurs dans les étapes expérimentales (en comparant des duplicatas), des similarités entre des expériences différentes (en comparant des régions spécifiques) et potentiellement à trouver des relations inattendues dans les signaux génomiques.

Dans le cadre de la création du *UCSC Genome Browser*, un ensemble d'utilitaires, nommé *Kent source utilities* ont été crées pour offrir aux chercheurs plusieurs opérations fondamentales sur des données génomiques. Deux de ces outils, soit « wigCorrelate » et « bigWigCorrelate » offrent l'option de mesurer la corrélation de Pearson entre deux (ou plus) expériences. Facile d'utilisation, ces outils souffrent d'un faible nombre d'options et de limitations importantes sur le nombre d'expériences qu'ils peuvent traiter dans un contexte où l'on veut analyser des milliers de fichiers.

1.4 Impact des problématiques de représentation et des analyses de données sur le développement d'outils

Tel que mentionné précédemment, il existe une variété d'applications du séquençage. Nous avons explicité que les données de séquençage sont bruitées, que ce soit au niveau des lectures ou du signal génomique, et avons brièvement discuté de l'alignement des lectures et des analyses possibles suite à cet alignement.

Dans cette section nous allons décrire certains aspects informatiques des problématiques du domaine de la génomique haut débit. Il est à noter que les problématiques reliées à l'analyse et la gestion de données de séquençage sont vastes. Nous n'abordons que celles qui ont motivés nos travaux.

1.4.1 Stockage et représentation des données

Les premiers formats utilisés pour stocker et rendre accessible des données d'expériences génomiques étaient des formats *ad hoc* qui, à force d'être utilisés, sont devenus des standards *de facto* du domaine comme le format FastQ (Cock, Fields, Goto, Heuer, & Rice, 2010) . Des formats comme BED, BedGraph et WIG (https://genome.ucsc.edu/FAQ/FAQformat.html) développés par l'équipe de l'Université de Californie à Santa Cruz (UCSC) ont avant tout été conçus pour leur simplicité d'utilisation, ceux-ci pouvant en effet être modifiés par l'intermédiaire de simples éditeurs de texte. Avant d'aborder les difficultés reliées à l'utilisation de ces formats textes, nous souhaitons faire la distinction entre les deux types de format communs, soit les formats conservant l'information des lectures et les formats de densité de lectures.

1.4.1.1 Format conservant l'information des lectures

Tel que mentionné précédemment, le résultat d'une expérience génomique à haut débit est typiquement un ensemble de quelques millions de lectures accompagnées d'informations reliées à leur qualité. Il existe deux niveaux de stockage de ces informations. Les formats de stockage pré-alignement, tel que le format FastQ, conservent la séquence de base associée à chaque lecture ainsi que des informations sur la qualité de chaque base (Cock, Fields, Goto, Heuer, & Rice, 2010). Les formats de stockage post-alignement y rajoutent habituellement plusieurs informations dont bien entendu la position et la qualité assignées à la lecture par l'outil d'alignement. Les formats les plus populaires sont SAM/BAM (Li *et al.*, 2009) et plus récemment CRAM (Hsi-Yang Fritz, Leinonen, Cochrane, & Birney, 2011). Cependant, certaines des vielles expériences utilisaient des formats moins adaptés (ex : BED) qui évacuaient souvent l'information technique des lectures.

Notons que le format BAM a été un des premiers formats binaires avec indexage disponible dans le domaine de la génomique. Nous expliquerons l'importance de ses caractéristiques au chapitre 3.

1.4.1.2 Formats représentant la densité de lectures

Le deuxième type de format, que nous nommons format de densité de lectures (incluant les formats BedGraph, BED, WIG), permet d'associer une valeur à une région du génome. Essentiellement, un format de densité contient minimalement les quatres informations permettant de localiser la densité (Figure 1.13). Généralement, ces régions sont regroupées par chromosome dans un même fichier et ensuite, triées par ordre chromosomique et numérique. Il est à noter que certains formats (BedGraph et WIG) ne permettent pas le chevauchement des régions, tandis que le format de fichier BED n'a pas cette contrainte de régions non-

chevauchantes. Ces formats sont souvent utilisés pour représenter la densité de lectures, mais pourraient peuvent également être utilisés pour associer n'importe quel type de valeur à des positions génomiques. Aucun de ces formats n'offre la possibilité de rajouter des champs supplémentaires ou des métadonnées de manière souple, nécessitant souvent l'ajout d'un fichier secondaire pour expliciter le sens des valeurs contenues dans les fichiers.

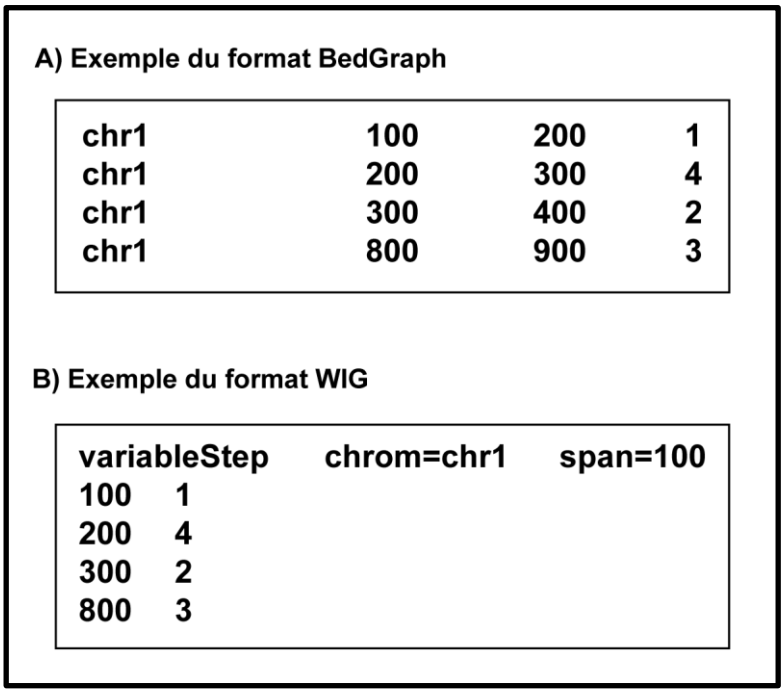

Figure 1.12 Exemples de fichiers de densité de lectures contenant des valeurs en format texte. Par exemple, « chr1 200 300 4 » indique que la région entre les positions 200 à 300 du chromosome 1 possède une valeur de densité de 4.

1.4.2 Utilisation des formats textes

Plusieurs des formats décrits (FastQ, SAM, BedGraph, BED et WIG) sont visualisables et éditables par n'importe quel éditeur texte. Bien que simple d'utilisation, ces formats de fichiers offrent de piètres performances lorsqu'il est question de l'accès et de la modification des données. De plus, leur usage perd sa simplicité quand ils dépassent une certaine taille et nécessitent des opérations coûteuses en temps pour retrouver des données. Bien que suffisant dans le cas où un laboratoire souhaite travailler avec tout au plus quelques dizaines de fichiers d'expériences de séquençage, ils deviennent rapidement limitatifs dans un cadre où un laboratoire souhaite utiliser des centaines voire des milliers de données d'expérience. Nous verrons comment des formats plus modernes pallient ces difficultés au chapitre 3.

1.4.3 Utilisation des paramètres par défaut

La plupart des outils d'analyse, incluant les outils de détections de pics mentionnés précédemment, peuvent être utilisés avec un ensemble de paramètres prédéterminés par les développeurs (paramètres par défaut) validés sur quelques ensembles de données lors du développement de l'outil. Par exemple, l'utilisateur fournit ses données à l'outil qui renvoie une liste de pics qui auraient été jugés valides avec les données utilisées lors du développement. Changer ces paramètres peut par contre parfois affecter grandement les résultats. Aussi, si le chercheur applique plusieurs outils, il pourrait être surpris de constater que le consensus des outils est souvent faible (Cairns *et al.*, 2011).

Ensuite le chercheur pourrait décider de comparer individuellement les scores de certains pics pour comparer les résultats de chaque outil, pour arriver à la conclusion que chaque outil utilise potentiellement son propre score, que ces scores sont calculés différemment et qu'il

existe peu de méthode pour comparer les résultats des différents outils, autres que des études laborieuses (Pepke, Wold, & Mortazavi, 2009).

La détection des pics est un exemple mais la même situation peut être retrouvée dans plusieurs types d'analyse génomique. Souvent les chercheurs se retrouvent à utiliser des outils tels que des boîtes noires, donc les résultats sont souvent fortement affectés par les paramètres fournis.

1.4.4 Taille importante des données

La manipulation de fichiers de données génomiques n'est pas un mince défi pour un laboratoire de recherche. Généralement, le nombre de lectures nécessaires pour couvrir certains génomes complexes à un niveau de profondeur acceptable, tel que le génome humain, se compte en dizaines de centaines de millions. Bien qu'une lecture prenne un espace infime individuellement (simple chaîne de caractères), plusieurs millions peuvent dépasser le gigaoctet (Go) d'espace disque et l'accumulation de ces fichiers dans le processus d'analyse d'expériences fait que l'espace nécessaire peut devenir prohibitif (Hsi-Yang Fritz, Leinonen, Cochrane, & Birney, 2011) .

Quelques outils d'analyse ne sont pas capables de travailler avec des données de cette taille en ayant été conçus pour utiliser la majorité des données en mémoire vive, une tâche impossible quand il faut travailler avec une multitude de fichiers de plusieurs Go chacun. Il n'est pas possible de gérer ces données avec des programmes de bureautique (par exemple, la suite Office) et des outils spécialisés en analyse de données tels que RapidMiner (https://rapidminer.com/) et Tableau (http://www.tableau.com/) sont mal adaptés à des données de cette taille. Même des modules d'analyse performants écrits dans des langages tel que R doivent être spécialement conçus pour des données génomiques (Huber *et al*., 2015).

De plus, l'augmentation constante de la taille des données de séquençage va rapidement rendre inutilisable plusieurs modules d'analyses existants. Il s'en suit que le domaine à besoin d'outils spécialisé non seulement pour les particularités des données de séquençage, mais pour la quantité de donnée. Ces outils devrait être capable de gérer des centaines, voir de milliers d'expériences, tout en s'exécutant de manière accessible aux chercheurs qui n'ont pas tous accès localement à une infrastructure informatique massive.

Un exemple d'un tel outil est Galaxy, (Goecks, Nekrutenko, Taylor, & Team, 2016) une plateforme de développement distribuée qui permet à des chercheurs d'analyser leurs données avec les nombreux outils intégrés dans la plateforme et d'exécuter ces calculs sur une infrastructure externe. Un autre exemple est le projet Genetics and genomics Analysis Platform (GenAP) (genap.ca) qui utilise entre autres la plateforme Galaxy, mais qui exécute les calculs sur l'infrastructure massive de Calcul Canada, permettant aux chercheurs d'exécuter leurs analyses sur des machines extrêmement puissantes.

1.4.5 Faciliter le développement d'outils bio-informatiques

Considérant la taille et l'importance des problèmes relevés, une toute nouvelle classe d'outils bio-informatiques capables de travailler avec les données de séquençage est donc nécessaire. En 2010-2011, la diversité d'outils était encore maigre et leur accessibilité limitée (Chilana, Palmer, & Ko, 2009).

Sachant que le domaine gagnerait à avoir des outils d'analyse réutilisables, facile à mettre à jour et qui offre des *workflows* transparents et reproductibles, il est souhaitable de faciliter le travail des développeurs d'outils. Si ces derniers ont accès à des fonctionnalités de développement qui leur permettent de réutiliser facilement des manipulation fondamentales

(par exemple, lire les données d'un fichier), le processus de développement sera plus sécurisé et plus rapide. De plus, la réutilisation de fonction commune permet un partage de la documentation entre les outils.

En l'absence de ces outils, le développeur risque d'introduire des erreurs à chaque manipulation qu'il permet à son outil de faire. Ceci le force à dépenser un effort indu dans la validation lui laissant moins de temps pour tester et implanter de nouveaux algorithmes.

Cette problématique n'est pas récente mais ne diminue pas en importance.

1.5 Motivations et objectifs des travaux

De 2010 à 2013, avant de rejoindre le laboratoire du P^{r} Jacques, mes efforts étaient concentrés à la création de méthodes d'analyses adaptées pour les besoins des expériences de séquençage que le laboratoire commençait à utiliser. Dans le cadre de ces travaux (dont certaines contributions se retrouvent dans (Brunelle *et al.*, 2015), j'ai eu à écrire de multiples outils pour réussir des tâches aussi simples que la transformation des données en format texte, jusqu'à des méthodes d'analyse de détection de régions enrichies dans les données de séquençage visant à détecter le positionnement des nucléosomes. La constante dans plusieurs de ces situations était la nécessité de recoder des fonctionnalités élémentaires de la manipulation de données génomiques, celles-ci étant rarement disponibles sous forme de librairie ou d'interface informatique.

Pourtant, pour éviter de réinventer la roue et pour rendre accessible les méthodes d'analyse utilisées par les laboratoires, il est souhaitable d'offrir des outils de développement qui visent la reproductibilité des méthodes. Des librairies offrant des opérateurs pour les données

génomiques permettant d'implémenter facilement des méthodes d'analyse sont un avantage net pour la reproductibilité. Plus précisément, plutôt que de devoir toujours réécrire les outils de manipulation, de telles librairies peuvent permettre aux développeurs de se concentrer sur la méthode et la construction de ces méthodes sur une base de code commun. Ce processus peut ainsi faciliter l'optimisation des opérations, le déploiement des outils et le partage des méthodes entre groupes de développement.

Ce constat m'a amené à créer notre propre librairie de fonctionnalités de manipulation de fichiers génomiques. À force d'utiliser cette librairie et suite à de nombreuses discussions avec des collaborateurs du laboratoire du P^{r} Droit de l'Université Laval, nous avons décidé de développer conjointement et de publier une librairie de codes qui englobe ces fonctionnalités. Permettant d'accéder et de manipuler des formats génomiques facilement, la librairie NGS++ a été un atout considérable dans le prototypage d'outils génomiques. L'article associé est présenté dans le Chapitre 2 de cette thèse.

En 2013, suite à cette publication, j'ai transité au laboratoire du P^{r} Jacques. À ce moment, mes efforts se sont déplacés vers le développement d'outils bio-informatiques pour faciliter et accélérer le développement de méthodes et d'analyses de grandes quantités de données génomiques. Ayant constaté l'absence d'interface de programmation moderne et accessible pour travailler avec le format BigWig dans le langage C++ nécessaire dans le cadre d'un autre projet du laboratoire, j'ai entrepris de créer le code de lecture qui est décrit au Chapitre 3. Il s'agissait d'une continuation logique avec mes précédents efforts.

Tel que mentionné précédemment, le nombre important et croissant d'expériences génomiques disponibles dans les bases de données publiques, tel que le portail du *International Human Epigenomic Portal* (IHEC), rend possible et souhaitable la comparaison de centaines voire des milliers d'ensembles de données. Ce type d'analyse nécessite de nouveaux outils s'inspirant de la programmation distribuée et utilisant des formats et méthodes s'inspirant du domaine du *Big Data*. Les travaux précédents ayant développés une expertise dans la manipulation efficiente de données génomiques, il était logique d'écrire un outil visant à rendre accessible

des manipulations massives. Nous avons donc ciblé une tâche qui en soit est simple, la corrélation de données génomiques, mais qui devient très difficile lorsqu'appliquée à des milliers de fichiers. L'outil *Genomic Efficient Correlator* (GeEC), que nous présentons au Chapitre 4, est le résultat de nos travaux dans le développement de librairies et d'outils génomiques. Cet outil permet aux chercheurs en génomique de créer des matrices de corrélations utilisant des milliers de fichiers sans devoir développer une expertise dans la manipulation de données massives. De plus, l'interface accessible de l'outil GeEC ainsi que son innovation au niveau de l'utilisation des formats de données scientifiques le rend un atout pour tout futur développeur qui souhaite travailler sur un outil similaire.

CHAPITRE 2

DEVELOPPEMENT DE LA LIBRAIRIE NGS++

Présentation

Ce chapitre décrit une librairie de code C++ développée en début de thèse et publiée en juin 2013 dans *Oxford Bioinformatics* (Nordell-Markovits *et al.*, 2013). Cette librairie avait comme objectif de rendre accessible la manipulation de données génomiques de manière transparente et ainsi faciliter la production et le prototypage d'outils compilés.

La faible disponibilité de librairies spécialisées dans le domaine de la génomique reste une réalité contrariante, mais NGS++ offre des fonctionnalités permettant de rapidement développer et tester des outils génomiques. L'utilitaire de traitement de données inclus dans la librairie permet de lire et d'écrire la majorité des formats standards du domaine de manière transparente, à l'exception du format BAM qui n'est supporté qu'en lecture. De plus, la librairie est fournie avec une interface Python spécifiquement adaptée pour permettre l'utilisation du module de traitement de données sans avoir à passer par un programme C++.

Depuis sa publication en 2013, nous avons continué à l'améliorer. Plusieurs petites erreurs ont été corrigées au niveau du code interne de la librairie. L'interface Python a également vu des améliorations au niveau de sa vitesse d'exécution et de ses fonctionnalités. Avec l'ajout d'un lecteur du format BigWig (Chapitre 3), le module de gestion de données de la librairie supporte désormais également ce format. La librairie continue de servir à prototyper plusieurs outils génomiques, incluant l'outil GeEC (Chapitre 4).

Finalement, il faut noter que la documentation principale de la librairie se retrouve sur son site web officiel (http://www.ngsplusplus.ca). Ce dernier contient des tutoriels complets sur l'utilisation des modules de la librairie ainsi qu'une documentation complète au format docXygen décrivant la structure interne du programme ; le tout fait plus de 40 pages. Cette documentation, ainsi que le code de la librairie, est disponible en ligne à l'adresse du compte GitHub de l'outil : https://github.com/NGS-lib/NGSplusplus.

Contribution

J'ai personnellement contribué à environ 70% du contenu de la librairie et des éléments associés. Plus précisément, j'ai écrit l'entièreté de la structure principale de manipulation de données, environ 50% des objets de lectures de données et environ 70% des tests unitaires. L'article lui-même a été rédigé conjointement par Charles Joly Beauparlant, étudiant au doctorat dans le laboratoire du Pr. Arnaud Droit à l'Université Laval, et moi-même dans des proportions semblables. La structure de la page web a été réalisée contractuellement par Philippe Boissonneault, alors que Charles et moi avons conjointement préparé les tutoriels et la documentation docXygen. Le site web est présentement hébergé par le laboratoire du Pr. Arnaud Droit.

NGS++: A LIBRARY FOR RAPID PROTOTYPING OF EPIGENOMICS SOFTWARE TOOLS

Alexei Nordell-Markovits[1], Charles Joly Beauparlant[2], Dominique Toupin[3], Shengrui Wang[3], Arnaud Droit[2] and Nicolas Gévry[1]

[1] Department of Biology, Université de Sherbrooke, Sherbrooke, Quebec J1K 2R1, Canada
[2] Department of Molecular Medicine, Centre de Recherche du CHU de Québec, Université Laval, Quebec, Quebec G1V 4G2
[3] Department of Computer Science, Université de Sherbrooke, Quebec J1K 2R1, Canada

Abstract

Motivation: The development of computational tools to enable testing and analysis of high throughput-sequencing data is essential to modern genomics research. However, while multiple frameworks have been developed to facilitate access to these tools, comparatively little effort has been made at implementing low level programming libraries to increase the speed and ease of their development.

Results: We propose NGS++, a programming library in C++11 specialized in manipulating both Next-Generation Sequencing (NGS) datasets and genomic information files. This library allows easy integration of new formats and rapid prototyping of new functionalities with a focus on the analysis of genomic regions and features. It offers a powerful, yet versatile and easily extensible interface to read, write and manipulate multiple genomic file formats. By standardizing the internal data structures and presenting a common interface to the data parser, NGS++ offers an effective framework for epigenomics tool development.

Implementation: NGS++ was written in C++ using the C++11 standard. It requires minimal efforts to build and is well-documented via a complete docXygen guide, online documentation

and tutorials. Source code, tests, code examples and documentation are available via the website at http://www.ngsplusplus.ca and the github repository at https://github.com/NGS-lib/NGSplusplus.

Contact: nicolas.Gévry@usherbrooke.ca and arnaud.droit@crchuq.ulaval.ca

1. Introduction

Previous years have witnessed an explosion in the amount of data produced using Next-Generation Sequencing (NGS) technologies, as exemplified by the ENCODE project (ENCODE Project Consortium *et al.*, 2012). However, analysis of these enormous datasets (easily over 100 GB) requires the use of a new generation of computational tools. As the quantity of data produced by NGS machines increases, so will the time spent on developing new tools. While substantial efforts have been made at integrating them into user-friendly frameworks such as Galaxy (Giardine *et al.*, 2005) or GeneSpace (Genome Space) relatively little effort has gone into providing the groundwork needed to increase the productivity of NGS developers, such as libraries and using standardized formats. Improvement in these areas would allow developers to greatly accelerate the speed at which they design and deploy new analysis software.

While certain tool suites such as BEDtools (Quinlan and Hall, 2010) and BAMtools (Barnett et al., 2011) offer a library or API to assist developers, these are generally aimed at giving access to the existing tool functionality rather than facilitating development of new ones. As such they are highly specialized. The SeqAn (Döring et al., 2008) library offers functionality for the developmental of future tools, but it specializes in sequence analysis rather than

genomic regions and features. Our proposed library, NGS++, aims to fill this gap by offering a powerful set of generic and flexible options to accelerate development and prototyping of epigenomics analysis tools.

2. Approach

It is impossible to predict the entirety of future needs for NGS data analysis. As such, NGS++ focuses on being a customizable and generic library that facilitates the prototyping and implementation of new functionalities via a transparent data interface. In this section, we summarize the three main components of NGS++:

1) file format management
2) data manipulation
3) functional operators.

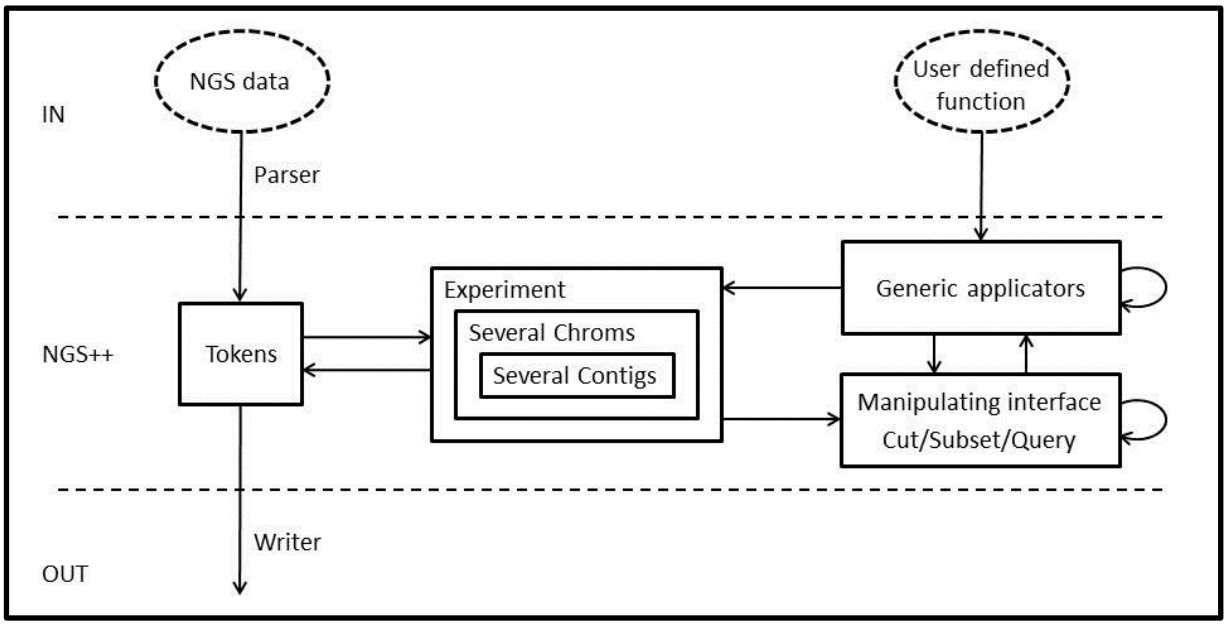

Figure 2.1 Typical workflow of the NGS++ library. Data are read by our Parsing interface then filtered as needed. User-defined functions are executed via our operators, and the transformed data are stored via our Writer interface.

Dealing with the wealth of existing file formats is a time consuming task. NGS++ offers a simple interface to parse and write in many frequently used genomics file formats (BED, GFF/GTF, SAM, WIG, BedGraph) using a generic data structure named Tokens that contain a number of standard features of genomic data entries (e.g. Start/End positions, position value, mapping quality). Additionally, the user can define "on-the-fly" custom formats to deal with the plethora of datasets that do not respect format specifications and BAM format is supported via integration of the BamTools API. The conversion between most supported formats is a trivial task:

```
uParser parser("filename.sam", "SAM");
uWriter writer("filename.bed", "BED");
while (!parser.eof())
        writer.writeToken(parser.getNextEntry());
```

The internal structure of the library is divided into a three tiers hierarchy separating a genomic dataset by unique scaffolds with each of them containing any number of contigs. This hierarchy is represented in (Figure 2.1), and each level offers a number of functions that can be extended via inheritance for specialized data manipulation. Loading datasets is done through integration with our Parser class and allows the user to easily load data:

```
TagExp.loadWithParser(inputStream,"SAM");
RegionExp.loadWithParser(inputStream,"BED");
```

NGS++ offers a powerful set of generic functions to compare, sort, merge and modify the previously loaded data. These include typical operations such as overlapping, merging, and comparing that allow the user to easily filter his data as needed. The following would return tags overlapping a BED file:

```
auto fExp=TagExp.getOverlapping(RegionExp)
```

Additionally, the majority of these operators can be used on any given feature of the data objects, including features added by the user via inheritance. This example sorts and counts a subset based on a new object feature:

```
RegionExp.sortSites([](uRegion item1, uRegion item2){return item1.Score < item2.Score},
&uRegion::getScore, &uRegion::getScore)
int count = RegionExp.getSubsetCount(0.5,0.8);
```

As shown above, this is greatly facilitated by the inclusion of anonymous lambda functions in the C++11 standard. Using these flexible operators allows the experienced developer to implement powerful modifications while the default genomic interval operators are easily usable by all.

Finally, to accommodate specific analysis needs, NGS++ offers an interface to run developer defined functions and transformations on the selected data. This allows the developer to concentrate on the function he wishes to implement without having to spend time on the underlying structure that will support it. Borrowing heavily from the functional programming paradigm, this allows for rapid prototyping and implementation. In the following we define and execute a function that allows us to trivially generate a histogram of contig sizes:

```
map<int,int> sizeOfContigs;
uTagsExp.applyOnSites([&](uTag Elem){sizeOfContigs[Elem.getSize()]++;});
```

Additional functional operators exist, allowing a variety of different operations on the dataset. This interface wraps many of the STL algorithms, enabling rapid parallelism via the OpenMP standard (Dagum and Menon, 1998).

3. Implementation

NGS++ is written in C++ using the C++11 standard. It offers a complete exception handling interface using the Boost exception class. A complete test suite is implemented using the Google test platform. It has been designed for a Linux environment using a C++11 compatible gcc compiler. Complete user guide, tutorial and discussion are available on the web page. Source code is hosted on GitHub.

4. Conclusion

Progress in the development of advanced bioinformatics analysis tools has undoubtedly been hindered by the lack of available programming frameworks. Our library aims to assist in filling this gap for the community of C++ epigenomic developers by giving them access to robust building blocks thus reducing significantly the time spent on development. Our efforts are now focused on including additional genomic formats and on increasing the breath of our tutorials. Future developments will include the integration of mid-level reusable functions such as similarity functions and normalization methods. The website provides a list of tutorials and commented working code examples, to assist developers in getting started with the library. Finally, the GitHub should facilitate the integration of suggestions and feedback from the community.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) for S.W, by the Canadian Institutes of Health Research (CIHR) for N.G, by Ministère du Développement Économique, Innovation et Exportation (MDEIE) for A.D. N.G. holds a Chercheur boursier (junior 1) award from the Fond de Recherche en Santé du Québec (FRSQ). A.D. holds a Réseau de médecine génétique appliquée (RMGA) salary award. C.J.B. holds a Centre de Recherche en Endocrinologie et Génomique Humaine (CREMOGH) award.

References

An integrated encyclopedia of DNA elements in the human genome. Nature. 489. 57-74.
Giardine *et al.*, 2005. Galaxy: A platform for interactive large-scale genome analysis. Genome Res. 15, pp. 1451-1455.

Genome Space : http://www.genomespace.org.

Quinlan AR and Hall IM, 2010. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 26, 6, pp. 841842.

Barnett DW *et al.*, BamTools: a C++ API and toolkit for analyzing and managing BAM files. Bioinformatics. 2011;27:16911692.

Andreas Dring, David Weese, Tobias Rausch and Knut Reinert, 2008. SeqAn an efficient, generic C++ library for sequence analysis. BMC Bioinformatics, 9:11.

L.Dagum and R.Menon, 1998. OpenMP: an industry standard API for shared-memory programming. IEEE Computational Science and Engineering, Vol. 5, No. 1. pp. 46-55.

ANNEXE A

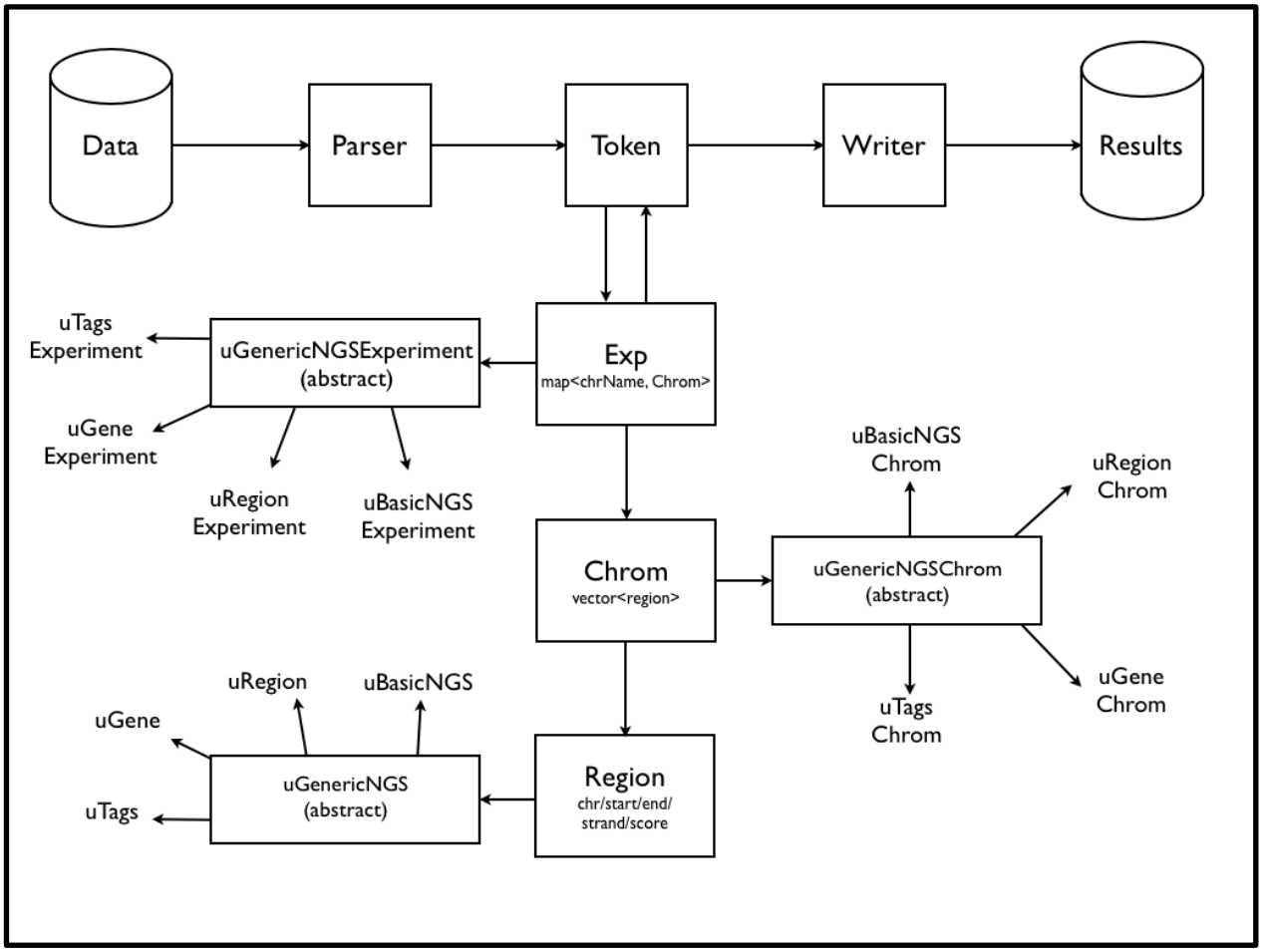

Figure 2.2 Représentation plus détaillée des classes et objets disponibles dans la librairie NGS++.

CHAPITRE 3

LECTEUR DE FORMAT BIGWIG EN C++ MODERNE

Introduction

Dans le chapitre précédent, nous avons présenté le module de lecture des fichiers génomiques qui est offert dans la librairie NGS++ (Nordell-Markovits *et al.*, 2013), l'importance de ce module, et brièvement les formats de données génomiques que celui-ci supporte. Le présent chapitre peut être vu comme une extension de cette section spécifique du chapitre précédent et se concentre sur un format en particulier, soit le format BigWig. Nous présenterons plus amplement ce dernier ainsi que les motivations qui ont mené à sa création.

Rappelons rapidement que les présents travaux se situent dans un contexte où la disponibilité de grandes quantités de données génomiques rend nécessaire l'utilisation d'outils et de formats adaptés aux méthodes *big data*. C'est dans ce contexte que nous discuterons du format de fichier BigWig et du code que nous avons écrit pour le lire. Le chapitre s'organise de la façon suivante. D'abord, nous ferons un bref rappel des différents formats de fichiers utilisés pour représenter les données générées par les techniques génomiques à haut débit. Nous situerons le format BigWig dans ce contexte, présenterons ses avantages techniques et expliquerons en détails la représentation interne du format. Ensuite, nous présenterons les quelques outils informatiques disponibles pour travailler avec ce format et les difficultés en découlant pour les développeurs d'outils. Finalement, nous présenterons notre code de lecture de fichier au format BigWig.

3.1 Problèmes de performance des plus vieux formats de densité de lectures

Tel qu'expliqué en introduction de cette thèse, le résultat d'une expérience impliquant le séquençage haut débit est habituellement un fichier de format FastQ contenant l'information des lectures générées. Ces lectures sont ensuite généralement alignées sur un génome de référence et représentées dans un fichier de format BAM. Cependant, plusieurs types d'analyses tels que la classification automatique (Ernst & Kellis, 2012), la segmentation par profondeur (Ye *et al.*, 2011) et la représentation de profils agrégés (Coulombe *et al.*, 2014) nécessitent des fichiers contenant la densité des lectures à toutes les positions du génome. Certains outils tels que BEDTools (Quinlan & Hall, 2010), MACS (Zhang *et al.*, 2008) et Findpeaks (Fejes *et al.*, 2008) permettent de calculer la densité des lectures à chaque position du génome, normalisée ou non par le nombre de lectures totales alignées. Il existe également des outils plus évolués tel que Wiggler (https://github.com/akundaje/align2rawsignal) permettant de faire des normalisations plus avancées en utilisant par exemple le taux d'alignement local. Les formats de fichiers de densité les plus populaires sont BedGraph, WIG et BigWig, ce dernier étant le plus récent.

Nous pouvons identifier deux inconvénients importants des formats de fichier BedGraph et WIG. Premièrement, leur représentation interne est linéaire et ne contient aucun indexage. Ceci signifie que si nous souhaitons accéder à une région spécifique du génome (ex : chr22 100-200), il est nécessaire de lire le fichier du début jusqu'à l'atteinte de cette région, une étape qui peut être parfois très longue. Pour une expérience sur un grand génome ceci peut être une opération onéreuse en temps. Deuxièmement, la représentation en texte brute occupe un espace disque substantiel ce qui, dans le contexte de multiplication des expériences et des besoins de transfert via le web, devient rapidement un problème.

3.2 Le format de fichier BigWig

Nous résumons ici les motivations qui ont mené à la création du format BigWig tel qu'expliqué dans l'article original (W. J. Kent, Zweig, Barber, Hinrichs, & Karolchik, 2010).

Tel que mentionné précédemment, l'équipe responsable du développement de ce format est la même qui a développé l'incontournable *UCSC Genome Browser* permettant entre autres aux usagers de visualiser leurs propres données (Kent *et al.*, 2002).

Cependant, la taille des fichiers d'expériences génomiques rend le téléversement (*upload*) dispendieux en temps et en bande passante. Quand l'usager souhaite visualiser quelques sections spécifiques des données de son expérience et que les données sont contenues dans un fichier en format linéaire, il doit charger l'entièreté du fichier sur le serveur. Selon l'expérience, la taille des fichiers peut être de l'ordre de centaines de Mo, voire quelques Go. Le temps de transaction se compte donc en minute, un processus lent qui répond mal au besoin d'une visualisation simple et rapide.

La motivation principale derrière la création du format BigBed (BB), duquel le format BigWig est une variante, était ainsi d'offrir un meilleur accès aux données personnelles de l'usager en proposant une alternative permettant d'éviter les problèmes énumérés plus haut. Par exemple, dans le cas où l'usager désire visualiser une seule section de son expérience, le transfert devrait cibler et n'envoyer que la région concernée par la requête, permettant ainsi à l'usager de rapidement visualiser les données spécifiques du génome dans un très court délai. Ceci étant impossible à accomplir avec les formats linéaires existants, il devenait nécessaire de créer un format qui permettrait de mieux répondre aux besoins des chercheurs. L'indexation de fichiers est une stratégie bien connue pour offrir ce type de fonctionnalité, c'est-à-dire, de permettre un accès direct à des données spécifiques d'un fichier (Peterson, 1957). Pour ce faire, le format BigWig adopte comme structure principale d'indexation un arbre R (Guttman,

1984). Pour expliquer de quelle manière le format utilise cette structure pour accélérer l'accès aux données, nous devons d'abord détailler ce qu'est un arbre de recherche binaire.

3.2.1 Introduction aux arbres de recherche binaire

Tel que discuté précédemment, si nous avons une liste linéaire d'items et nous souhaitons retrouver un ou plusieurs de ces items, il est *a priori* impossible de le faire sans parcourir la majorité de la liste. Une stratégie alternative pour accélérer la recherche est de stocker les items non pas dans une liste, mais dans un arbre de recherche binaire. Le concept général d'un arbre de recherche binaire est simple (Figure 3.1). Chaque nœud de l'arbre contient une valeur et le sous-arbre gauche de ce nœud ne contient que des éléments inférieurs aux nœuds parents, tandis que le sous-arbre droit ne contient que des éléments supérieurs. Pour vérifier si un élément spécifique est dans l'arbre, il suffit de parcourir cet arbre en visitant les sous-arbres gauches ou droits jusqu'à avoir retrouvé l'élément (ou pas si l'élément est absent de l'arbre).

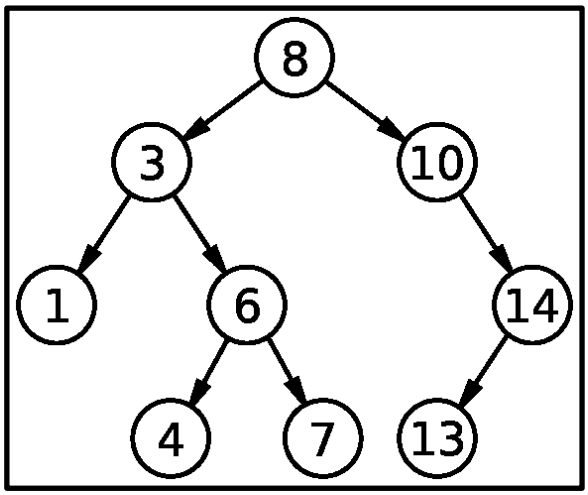

Figure 3.1 Exemple d'arbre de recherche binaire. Dans le cas présent il est possible de savoir en maximum quatre opérations si une valeur est présente dans l'arbre.

En utilisant une telle stratégie, il est possible de trouver un élément beaucoup plus rapidement que dans une liste linéaire. Cependant, une donnée génomique est rarement composée d'un seul élément. Dans la plupart des cas, il s'agit d'une combinaison de quatre éléments, soit le chromosome, la coordonnée de départ, celle de fin et la valeur de densité de signal.

3.2.2 Introduction aux arbres R

L'arbre R (*R-Tree*) est une structure de données optimisée pour l'accès et la recherche de données spatiales. Il s'agit d'une variante de l'arbre de recherche binaire permettant de gérer des coordonnées géographiques (composées minimalement de deux éléments, la longitude et la latitude). Par exemple, un arbre R peut être utilisé pour retrouver une étendue de coordonnées correspondant à tous les restaurants situés à 2 km d'une position. Comme nous le

verrons, il est facile de les adapter pour gérer des coordonnées génomiques. La structure générale d'un arbre R est assez simple (Figure 3.2). Chaque nœud contient un nombre de rectangles circonscrits[1] (Papadias & Theodoridis, 1997) et chaque rectangle pointe vers un nœud de la couche inférieure. Le nœud de la couche inférieure ne peut contenir que des rectangles qui sont situés dans les limites du rectangle circonscrit de la couche supérieure. Cette logique est répétée à chaque niveau jusqu'à l'atteinte des données terminales appelées feuilles. Les feuilles contiennent des éléments qui sont nécessairement contenus dans le rectangle du nœud supérieur.

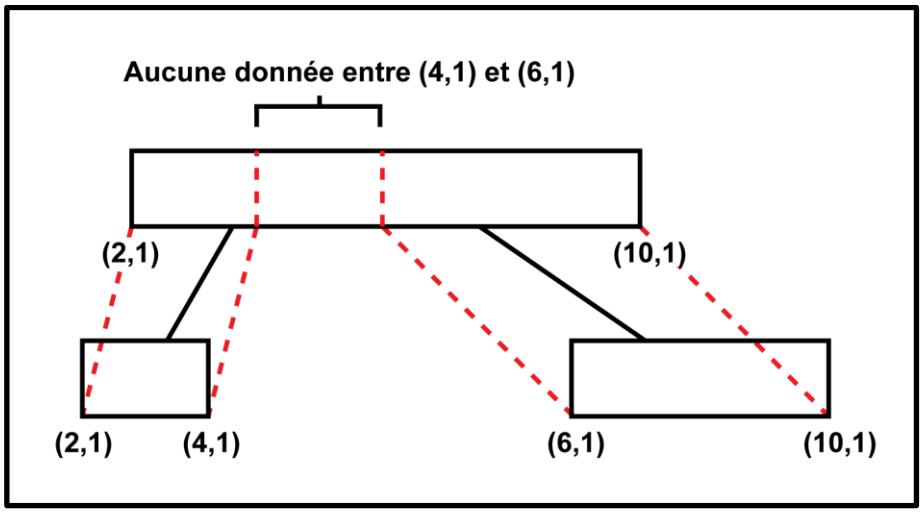

Figure 3.2 Exemple de structure d'un petit arbre R.

1 À noter qu'il n'existe pas de traduction largement acceptée du terme *"minimal bounding rectangle"*. Après consultation avec plusieurs collègues en mathématiques, le terme rectangle circonscrit semblait adéquat.

Utiliser cette structure pour vérifier si une position (latitude, longitude) est présente dans l'arbre est simple et se rapproche de la recherche dans un arbre binaire. En effet, dès que les coordonnées d'un nœud représenté par un rectangle ne contiennent pas le point que nous recherchons, nous pouvons l'écarter et par conséquent, écarter tous les éléments des couches inférieures reliés à ce rectangle. En parcourant les nœuds, il est ainsi possible de retrouver l'élément si notre position se retrouve dans le rectangle d'une feuille.

3.2.3 Utilisation d'un arbre R pour des données génomiques

Adapter un arbre R pour gérer des données génomiques nécessite très peu de modifications. Une région génomique étant décrite par un chromosome et une paire de coordonnées (par exemple, chr10 100-2500). Tel que mentionné précédemment, un élément d'un arbre R standard est représenté par une paire de coordonnées (X, Y) correspondant à la diagonale d'un rectangle dans un espace à deux dimensions (Figure 3.2). Une coordonnée génomique étant simplement une ligne en deux dimensions sur un chromosome, il est ainsi possible d'utiliser un arbre R standard à deux dimensions en définissant Y = 0. Dans l'exemple ci-dessus, l'étendue de recherche (chr10 100-2500) serait représentée par la paire de coordonnées (100,0) et (2500,0). Alternativement, il aurait pu être possible de modifier l'arbre R ainsi que ses algorithmes de recherche et de balancement pour traiter des données à une dimension.

L'élément chromosome (chr10) de la requête peut être gérée soit en générant un arbre R pour chaque chromosome, soit en incluant le chromosome comme une valeur supplémentaire contenant une valeur d'ordonnancement spécifique (alphanumérique ou autre). Dans ce deuxième cas, chaque rectangle représente au plus un chromosome.

L'utilisation d'un arbre R par chromosome est plus efficace pour des données denses avec un faible nombre de chromosomes (comme le génome humain), tandis que l'utilisation de chromosomes comme valeur de tri serait plus efficace pour des données avec un très grand nombre de chromosomes (comme lors d'assemblage *de novo* de génome complexe).

Dans le cas de l'implémentation du format BigWig, une version adaptée de l'arbre R a été utilisée; les algorithmes ainsi que la structure de l'arbre ont été modifiés pour gérer des données ne contenant que des coordonnées à une dimension. La structure de l'arbre R ainsi que la recherche dans celui-ci sont représentées visuellement à la Figure 3.1.

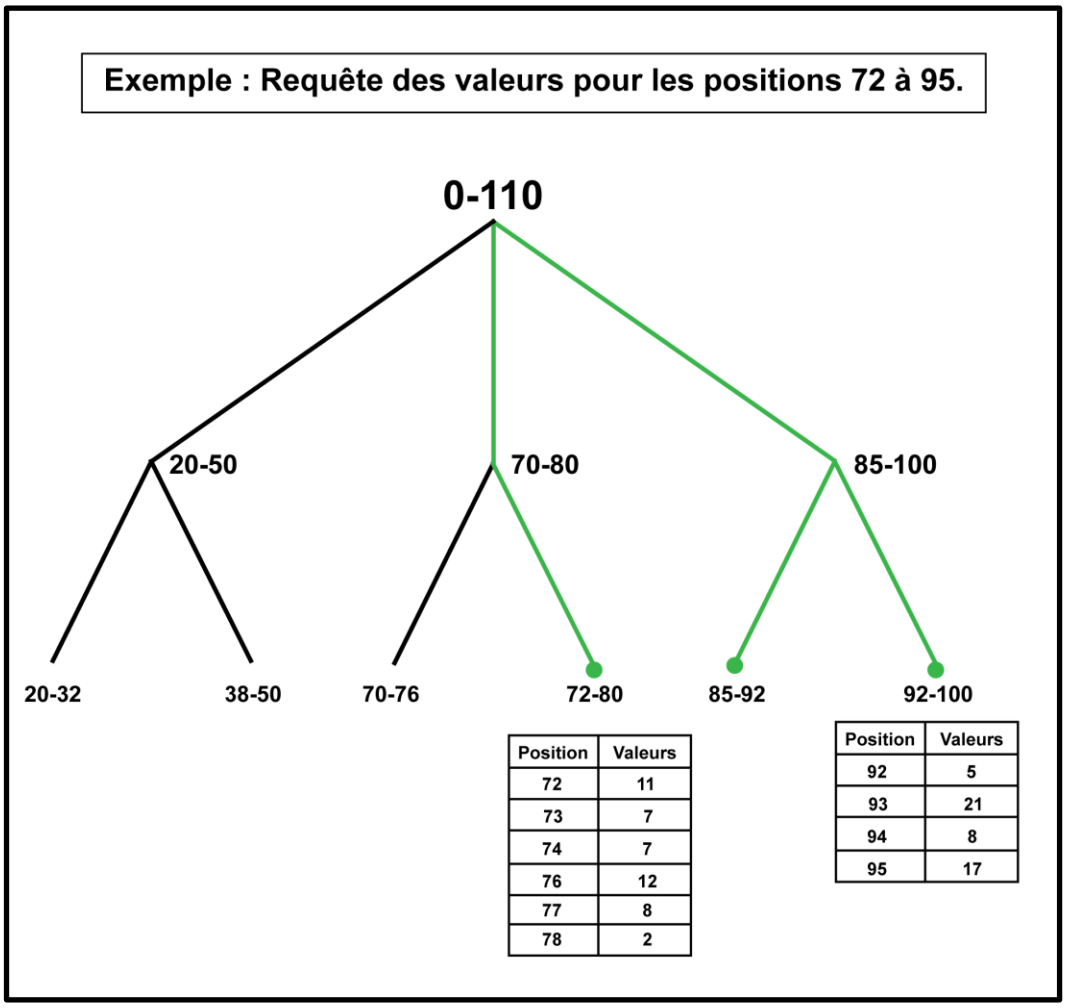

Figure 3.3 Exemple de structure de données en arbre R. Des requêtes qui cherchent à ressortir les valeurs à la position 74 et 93 pourraient consulter uniquement les branches en vert et éviter les autres nœuds. En contrepartie, une requête pour les positions 55-65 ne retournerait aucun résultat dès le deuxième nœud et donc serait presque instantanée. L'exemple présenté et la recherche de position génomique dans un fichier BigWig sont très similaires. Noter que pour un fichier épars, des positions sans couvertures (51-69) ne seront pas représentées dans l'arbre.

De plus, pour sauver de l'espace disque, seulement un élément sur 512 (paramètre par défaut) est utilisé dans l'indexation des feuilles de l'arbre R BigWig. Cette indexation est jumelée au fait que chaque feuille de 512 items est compressée en utilisant l'algorithme de compression

zlib utilisé entre autres dans l'outil UNIX gzip. Quand une feuille comprend une intersection valide avec la recherche présente, celle-ci est décompressée et tous ses éléments sont parcourus. Certains de ces éléments ne correspondront pas à la recherche et seront ignorés. Malgré le fait que ce choix d'indexation augmente le temps de recherche légèrement, elle minimise l'espace utilisé par l'index, celui-ci prenant en moyenne moins de 1% de l'espace des données.

3.3 Structure interne et avantages du format BigWig

L'arbre R adapté est l'élément principal du format BigWig, il contient les valeurs et l'index nécessaire pour accéder rapidement à ces valeurs. Le format est complété par un nombre de blocs de données qui contiennent des informations supplémentaires (métadonnées). Entre autres, l'entête contient les informations permettant de valider le format et les informations internes nécessaires pour lire le fichier. On retrouve également des informations sur le nombre de feuilles, le nombre de données dans le fichier et certaines informations techniques qui ne sont pas pertinentes dans le cadre présent. Les détails techniques complets de la structure de l'entête et des blocs de métadonnées sont disponibles directement dans le code source du programme de lecture.

Maintenant la structure interne des données du format BigWig expliquée, il est simple de constater comment ce format peut être utilisé pour un accès rapide à certaines données. Si un usager souhaite visionner une région spécifique d'un génome (ex : chr22 123214-132314), il envoie la requête d'accéder aux feuilles qui intersectent cette région. Déjà, l'usager aura évité de lire la majorité des données contenues dans le fichier. Une fois les feuilles accédées, elles sont décompressées et parcourues linéairement. En se rappelant que chaque feuille contient 512 données, mais qu'il n'est pas garantie que toutes ces données intersectent la région sélectionnée, chaque donnée est donc testée et ensuite sélectionnée en fonction de sa

pertinence par rapport à la requête de départ. Sachant que la recherche d'un bloc dans un arbre est d'un ordre de grandeur plus rapide que la recherche linéaire d'un fichier (log N vs N), il est évident qu'accomplir cette requête dans un fichier BigWig est beaucoup plus performant que dans un fichier linéaire équivalent.

À sa création, le format BigWig était un des seuls formats de données génomiques (en plus du format BAM introduit plus tôt) offrant une structure binaire bien définie et optimisée pour l'accès et l'espace, et il était le seul des formats à être spécialisé pour des valeurs de densité plutôt que pour des lectures. Il est à noter que conjointement à la création du format BigWig, le groupe responsable a également écrit une couche réseau, gérer par le fureteur UCSC. Cette couche est optimisée pour le transfert d'informations d'un fichier BigWig et se trouve détaillée dans l'article du format. Bien qu'en dehors du champ d'intérêt de ce chapitre, la structure de cette couche peut servir pour tout développeur souhaitant incorporer l'accès distal d'un fichier BigWig à partir d'un service Web.

3.4 Inconvénients du format BigWig

Jusqu'à tout récemment, le désavantage principal du format BigWig était sa difficulté d'utilisation. En effet, les utilisateurs travaillaient principalement avec des données en format texte et avaient donc l'habitude des données lisibles via des utilitaires de lecture standard. Puisque ce n'est pas le cas des formats binaires comme BigWig, il devient plus difficile de manipuler un tel fichier sans la disponibilité de librairies de codes et d'outils adéquats. À la création du format, le seul code disponible était celui produit par ses créateurs, soit la librairie de Kent écrite en C (W. J. Kent, Zweig, Barber, Hinrichs, & Karolchik, 2010). Plusieurs des outils de manipulation ont donc été construits sur la base de cette librairie. Cependant, le code

de celle-ci est écrit dans un style procédural contenant un très grand nombre de dépendances rendant difficile sa modification.

Dans le cadre de la publication de l'outil de visualisation IGV (Robinson *et al.*, 2011), l'équipe responsable de son développement a écrit en Java un code avec une structure orientée objet capable de lire des fichiers BigWig. Contrairement à la librairie de Kent, la structure du code de lecture d'IGV utilise des méthodes de génie logiciel moderne. En comparaison avec le code de Kent, le code de l'outil IGV permet un gain net de productivité, son utilisation étant distinctement plus simple.

3.5 Programmer un lecteur BigWig en C++ moderne

Notre motivation principale était de supporter le format BigWig pour l'outil VAP (Coulombe, *et al.*, 2014) développé dans notre laboratoire et codé en C++. Tel que mentionné, l'utilitaire de Kent, quoi que fonctionnel, est écrit dans un langage et dans une méthode de programmation qui nuit à son extensibilité. L'absence de code de lecture du format BigWig écrit en C++ moderne ainsi que la facilité que nous avons eu à travailler avec le code du lecteur d'IGV nous ont amené à vouloir implémenter une version moderne et orientée objet en C++ d'un outil de lecture BigWig. Ceci répondait également à trois besoins internes et externes que la librairie de Kent ne satisfait pas :

- Offrir une interface distincte pour lire le format BigWig ;
- Écrire cet interface en code orienté objet moderne pour permettre son extension par des sources externes advenant des nécessités futures ;
- Et rendre l'interface compatible avec des itérateurs C++ moderne pour permettre de travailler avec les algorithmes standards de la STL.

Pour expliquer le troisième besoin, nous suggérons (Nelson, 1995) qui explique bien les avantages d'offrir un itérateur standardisé en C++.

3.5.1 Code de lecture moderne

Dans cette section, nous allons présenter les éléments qui composent notre lecteur de code C++. Nous commencerons par présenter les outils externes que nous avons intégrés à notre code pour remplir des fonctionnalités spécifiques. Ensuite, nous discuterons de sa structure interne et de son interface externe suivie de quelques exemples d'utilisation de nos itérateurs. Finalement, nous donnerons des exemples de programmes qui intègrent notre lecteur et nous expliquerons comment intégrer le code dans de futurs outils.

3.5.2 Utilisation de Boost

La librairie de code Boost est une librairie libre accès revue par les pairs qui est un standard dans le domaine du développement C++ (Schling, 2011). Les données dans les feuilles de l'arbre R sont compressées avec l'algorithme d'encryption de l'outil gzip qui est accessible par la librairie de Boost Iostreams. Pour accéder au contenu d'une feuille et garantir le bon fonctionnement de notre routine de décompression, nous avons décidé d'utiliser cette implémentation dans notre librairie.

3.5.3 Utilisation et structure générale du code de lecture

Deux objets sont essentiels pour utiliser correctement notre code de lecture : le `BWReader` et le `BWIterator`. Le `BWReader` est l'objet qui encapsule l'ouverture d'un fichier BigWig. Ouvrir un fichier avec un objet `BWReader` est assez simple :

```
BWReader myBWFile(PathtoMyFile);
```

Utiliser un `BWReader` passe par l'utilisation d'un objet `BWIterator`. En soit, le `BWReader` ne fait que répondre à des requêtes en fournissant un `BWIterator` qui possède une liste des feuilles qui répondent à une requête spécifique. Par exemple, voici une requête dans le cas où l'usager désire un `BWIterator` qui va parcourir l'ensemble des données :

```
BWIterator myBigIterator = myBWFile.getBigWigIterator();
```

Ou si l'usager désire un `BWIterator` qui parcourt seulement les données du chr1 :

```
BWIterator myBigIterator = myBWFile.getBigWigIterator('chr1', 0, 0);
```

Et ainsi de suite pour toutes les régions souhaitées. Après avoir obtenu l'accès à un `BWIterator`, la syntaxe pour accéder aux données est soit celle d'un itérateur Java, soit celle d'un itérateur standard C++.

Dans l'exemple ci-dessous, l'utilitaire déclare deux itérateurs qui ciblent respectivement les régions 1400-1700 et 2400-4700 du chr21 de notre fichier. Ensuite il itère à travers tous les éléments du premier itérateur et écrit les données à l'écran :

```
std ::ifstream fis.open("monfichier.bigwig", ios::in | ios::binary);
BBFileReader monLecteurBW("monfichier.bigwig",fis);
BWIterator monIterateur =
monLecteurBW.getBigWigIterator("chr21",1400,"chr21",1700);
BWIterator autreIterateur =
monLecteurBW.getBigWigIterator("chr21",2400,"chr21",4700);
while (monIterateur.isEnd()==false){
  std::cout <<myIterator->getStartBase()<<" ";
```

```
        std::cout <<myIterator->getEndBase()<<" ";
        std::cout <<myIterator->getChromosome()<<" ";
        std::cout <<myIterator->getWigValue()<<"\n";  }
```

Cet exemple illustre bien la facilité d'utilisation de notre code.

De plus, dû à la structure orientée objet du code, il serait facile de spécialiser l'itérateur pour des besoins spécifiques d'un programme. Nous pourrions, par exemple, hériter de l'objet `BWIterator` et écrire un `FilteringBWIterator` qui pourrait ignorer automatiquement toutes les valeurs qui ne correspondent pas à une condition spécifique. Alternativement, il serait facile d'implémenter un `HttpBWIterator` permettant d'écrire le résultat d'une requête dans un format compatible à un service http transmis via l'internet.

Aussi, dans l'optique de développer des utilitaires de lecture efficaces et polyvalents, la structure de la classe `BWReader` permettrait son intégration dans un module de lecture générale de formats génomiques indexés, ce qui permettrait à l'usager de lire de manière transparente les formats BigWig/BAM/CRAM et tout autre futur format indexé. Ceci aurait été difficile à implémenter avec une structure de code procédurale telle que celle de l'utilitaire de Kent.

Finalement, l'encapsulation des fonctionnalités dans un objet bien défini rend trivial son intégration dans des utilitaires externes. Quand nous avons rajouté la fonctionnalité de lecture BigWig au programme VAP, le procédé d'intégration lui-même n'a pris que quelques heures.

3.5.4 Intégration dans des outils existants et futurs

Présentement, notre code de lecture est utilisé dans trois projets différents de notre laboratoire. Tel que mentionné précédemment, il a d'abord été intégré dans l'outil VAP. Le code a ensuite

été rajouté directement dans le module de lecture de la librairie NGS++ présenté au chapitre 2 afin de permettre l'accès à des fichiers BigWig de manière transparente et d'utiliser l'interface Python de cette librairie. Finalement, le module de conversion HDF5 de l'outil GeEC présenté au chapitre 4 utilise également notre code de lecture.

Dans les cas présentés, les outils ont intégré directement le code de lecture. Celui-ci inclut la librairie Boost nécessaire. Leur seul dépendance est ainsi un compilateur C++ supportant le standard de C++11. Cependant, il est également possible d'utiliser le code de lecture sous forme de librairie statique.

3.6 Améliorations à venir

Les améliorations prévues au code de lecture se divisent en trois éléments : la compatibilité avec la section algorithme de la *Standard Template Library* (STL), l'intégration de fonctionnalités de C++11 et l'ajout d'un objet d'écriture.

Le code de lecture offre présentement des itérateurs qui remplissent théoriquement les spécifications des *Forward Iterators* de la STL. Pourtant, certains de ses algorithmes échouent en utilisant nos itérateurs, et ce malgré l'atteinte des spécifications de la STL. Il est nécessaire de trouver et régler ce problème afin que notre itérateur puisse être utilisé pleinement avec les fonctions de la section algorithmes de la STL.

L'intégration de fonctionnalité C++11/14, surtout au niveau des boucles implicites et de l'utilisation potentielle de fonction anonyme (lambda), serait la suite logique de notre motivation de fournir un outil écrit dans un style moderne et pratique. Continuer à moderniser

le code et à l'adapter à la réalité « fonctionnelle » du développement C++ moderne pourra garantir que celui-ci reste à jour.

Finalement, le code s'occupe présentement uniquement de la lecture d'un fichier BigWig. Rajouter un objet responsable de l'écriture pour BigWig, écrit en C++ moderne, serait d'un grand atout pour la transformation des formats et l'intégration dans des programmes futurs.

Conclusion

Nous avons écrit un code de lecture C++ moderne qui permet de facilement lire les données d'un fichier en format BigWig afin de rapidement cibler les régions génomiques voulues et d'en extraire l'information pertinente. Cet outil peut aisément être intégré à des outils externes, soit sous forme de code, soit sous forme de librairie. La structure orientée objet moderne ainsi que les concepts de programmation déployés pour écrire cet outil faciliteront son expansion future et devrait permettre de combler un besoin existant dans la communauté de développement C++ en bio-informatique. La version publique du code sera bientôt disponible sur le compte BitBucket du laboratoire (https://bitbucket.org/labjacquespe).

CHAPITRE 4

DEVELOPPEMENT D'UN OUTIL DE CORRELATION GENOMIQUE

Présentation

Ce chapitre décrit un ensemble d'outils nommé *Genomic Efficient Correlator* (GeEC) dont le développement a débuté en 2013, ayant comme principal objectif de rendre accessible la création et l'analyse de matrices de corrélation massives. L'une des premières applications de l'outil est de fournir des matrices de corrélations à partir des milliers d'ensembles de données disponibles par le portail officiel des données de l'*International Human Epigenome Consortium* (IHEC) (http://epigenomesportal.ca/ihec/grid.html).

Les outils évaluant la corrélation ou la ressemblance de résultats d'expériences génomiques ne sont pas rares dans le domaine. Cependant, la majorité de ces outils ne sont appropriés que pour un faible nombre de fichiers. Nous avons créé un ensemble d'outils capable de générer une matrice à partir de plusieurs milliers de fichiers et ce de manière distribuée. De plus, l'outil passe par l'utilisation du format HDF5, rarement utilisé en génomique, et la routine de conversion est à notre connaissance unique dans le domaine.

Dans le cadre de ce chapitre, nous allons décrire l'outil ainsi que démontrer sa validité en l'appliquant sur deux ensembles tests, respectivement d'un peu moins de deux cents et d'environ quatre mille fichiers génomiques. L'analyse de ces résultats démontre l'utilité potentielle de l'outil GeEC.

Contribution

J'ai personnellement contribué à environ 95% de la structure et des éléments associés. Plus précisément, j'ai écrit l'entièreté du module de conversion, le module de corrélation et le module d'analyse. La ré-écriture de certains de ces modules en C++ est à la charge de Jonathan Laperle, présentement étudiant à la maitrise dans le laboratoire. J'ai rédigé l'entièreté de l'article en collaboration avec mon directeur. Les analyses ont été faites conjointement avec mon directeur.

THE GENOMIC EFFICIENT CORRELATOR (GEEC) TOOLSET

Alexei Nordell-Markovits[1], Jonathan Laperle[2], Marc-Antoine Robert[1] and Pierre-Étienne Jacques[1,2]

[1] Département de Biologie, Université de Sherbrooke, Quebec J1K 2R1, Canada
[2] Département d'Informatique, Université de Sherbrooke, Quebec J1K 2R1, Canada.

Abstract

Modern sequencing techniques have produced an exponential increase in the amount of available datasets and sequencing experiments. While previous genomic bioinformatics tools were aimed at analysing at most a few dozen of datasets, developers must now think in term of scaling to thousands when developing their analysis software. The Genomic Efficient Correlator (GeEC) toolset is a similarity tool that can generate large-scale similarity matrices from thousands of experiments and assist the user in fundamental analysis tasks.

Keywords

NGS, bioinformatics, analysis, correlation

1. Introduction

High-throughput sequencing technologies have undergone a revolution in the past decade. The number of reads generated per dollar has increased at a rate superior to Moore laws (Shendure *et al.*, 2004), enabling new and novel experimental techniques to be developed using sequencing technologies(Godwin et al, *2016*). This has led the technology to be adapted to generate a wide range of biological studies such as the analysis of protein-DNA interactions, epigenetic marks and transcriptomics. As a result we observe the accumulation of a critical mass of sequencing experiments that can be publicly downloaded and analyzed using modern data analysis techniques.

However, while the bioinformatics community has produced many analysis tools that are accessible to life scientists, these tools have mostly focused on the analysis of a small number of datasets and they tend to run on local machines. To properly exploit the thousands of datasets now available requires developers to build tools that are adapted to processing large amount of datasets employing modern high performance computing (HPC) methods.

In light of this situation we have created the Genomic Efficient Correlator (GeEC), an ensemble of programs aimed at producing and analyzing large-scale similarity matrices. The first module we implemented aims at transforming the genomic data from different formats into the HDF5 (Hierarchical Data Format V5) ("The HDF Group. Hierarchical Data Format, Version 5,") format which is a highly efficient scientific vector format, allowing the possibility of substantially improving the speed of subsequent analyses. We then implemented a similarity module that is engineered to run on large compute clusters, allowing this type of analysis to be run on massive amount of datasets. This enables the creation of large similarity matrices from genomic datasets in a reasonable time frame and to efficiently scale up, particularly when compared to currently existing correlation tools. Similarity matrices of these sizes allow researchers to employ novel analysis techniques to find and validate relationships and GeEC offers certain tools to assist scientists in implementing these techniques.

2. Methods

The Similarity components of GeEC is implemented in Python and uses a number of well-known scientific libraries for most data manipulation and analysis tasks. It is divided in three modules: the data conversion is handled by the GeEC-Prep module, while data analyses are handled by GeEC-Sim and GeEC-Analysis (Figure 4.1). The tools are designed to run in parallel both on local computers and with the aid of job dispatching tools on distributed compute nodes. Before describing each module, let us begin by explaining the choices we made regarding the internal representation of the data as well as the similarity measures we tested.

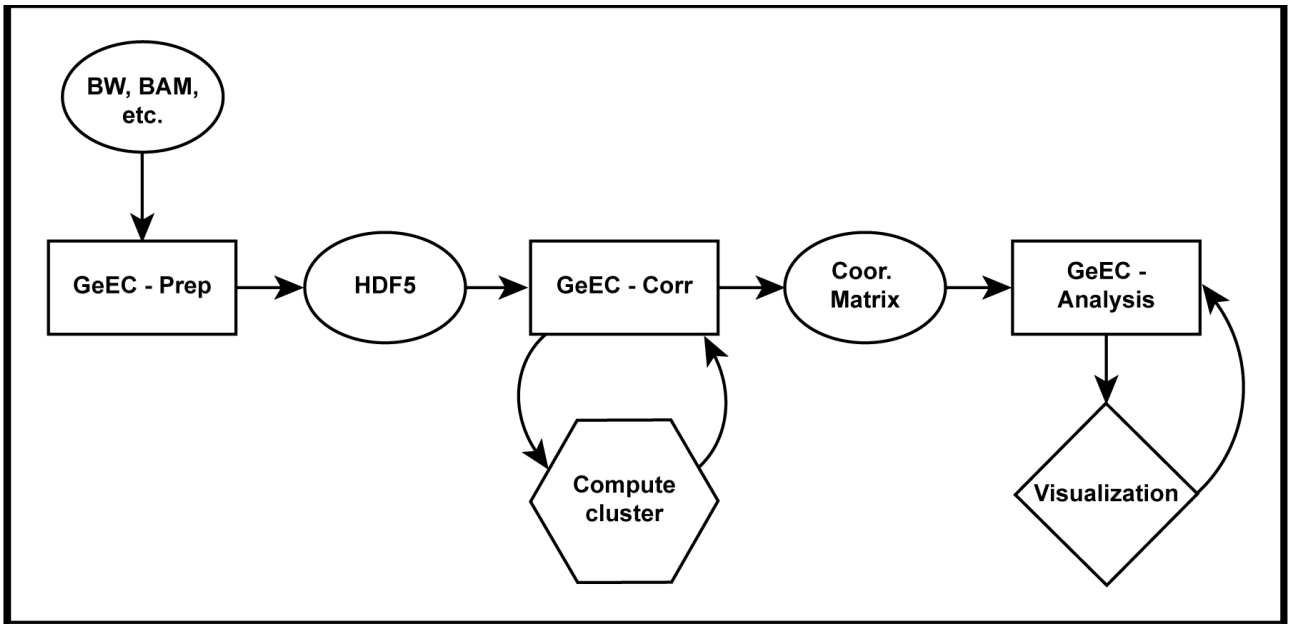

Figure 4.1 General workflow of the similarity module of GeEC.

2.1 Choice of internal data representation

Initial attempts to generate our large scale similarity matrices were hampered by the multiplicity of data formats that can be used to store aligned read data and by the uncertain validation of these formats. When downloading data from multiple sources, it was frequently found that several of these datasets needed further processing to be uniform in their representation. Additionally, with the exception of the BigWig (Kent et al. 2010) and BAM (Li et al. 2009) formats, most genomic formats are optimized for readability or disk space rather than rapid access. To alleviate both the data coherence and speed issues, we decided to convert all input files into HDF5.

The HDF5 format is designed to store and organize large amounts of numerical data. Supported by the non-profit HDF group it is a high-performance format aimed at efficient parallel I/O for high volume data. While most genomic formats are optimized for either sequential read (e.g. BedGraph) or "chunk" access (e.g. BigWig), storing genomic density in HDF5 format allows extremely fast access to the entire dataset. The HDF5 format implicitly supports attaching metadata to the stored datasets, allowing to both conserve and extend information in a way that is difficult with most rigid genomic formats. Additionally, we note that the HDF5 format has been used previously by other groups for storing genomic data or even implementing genomic specified formats with HDF5 as their foundations (https://github.com/glennhickey/hal). While these specific formats do offer extra target functionalities, we found the flexibility of interfacing directly to HDF5 files was more than sufficient for our needs.

While parsing and loading genomic data for a given organism can take anywhere between a few seconds to tens of minutes depending on the size of the file, loading the equivalent data from HDF takes between milliseconds and seconds. This is due to the internal representation of HDF5 data being a highly optimized contiguous vector. This implies that when storing data from a genomic experiment, we store information for the entire genome, including missing

data that are stored with a value of 0. While this does increase the space used, it guarantees that the size and corresponding loading time of the HDF5 format is fixed for any given organism and resolution. Conversely genomic formats tend to scale to the density and distribution of reads, making it harder to estimate the running time and space required of a given analysis. For all of the above reasons, we decided to use HDF5 as the unifying internal data representation format for our tool, and implemented the necessary modules to perform this task.

2.2 Choice of similarity measures

To produce the type of analysis required by the similarity module GeEC requires valid measures. A similarity measure is roughly defined as a function that returns a large value for two similar objects and a low or negative value for extremely dissimilar object.

While there exists a large volume of literature on available similarity measures both inside and outside of the bioinformatics community (Clark, 2013), it is fair to say that comparing two genomic experiments is often done using either Pearson or Spearman Correlation (Kong *et al.*, 2002). In an effort to offer novel metrics, we decided to also test the Maximal Information Coefficient (MIC) (Reshef *et al.*, 2011), a much more recent measure that has begun to be used in the fields of genomics. Finally, a simple "Top" metric (defined lower) was also implemented for users seeking an intuitive evaluation of the similarity between two genomic datasets. We consequently decided to use the term "similarity measure" rather than "correlation measure" when discussing these comparison functions.

Pearson Correlation is a measure that evaluates the linear relationship between two sets of values and returns the product-moment correlation coefficient (***r***) ranging between -1 and +1.

A Pearson ***r*** close to 1 means that the values vary in a similar fashion, while a ***r*** close to 0 means no obvious linear relationship between the two sets of values. Spearman correlation coefficient is defined as exactly the Pearson correlation, except that the input values are transformed into their ranked equivalent before applying the function. The main difference lies in the Pearson correlation requiring the values to be related by a linear relationship, while Spearman correlation will return a high value if any monotonic relationship exists between the two sets of values.

The MIC measure evaluates the strength of the relationship between two given variables. Unlike the previous measures, MIC evaluates both the linear and non-linear relationship of the data and claims to perform well when multiple types of relationships exist between two variables. While a complete technical description is beyond the scope of this paper, the measure is summarized in Figure 4.2, adapted from the original paper. This method has been known for some time, but only recently has the computational power necessary to calculate it been commonly available (Speed, 2011). Unfortunately, the recursive nature of its execution makes it difficult to execute on data as large the entire human genome, even with modern computing power.

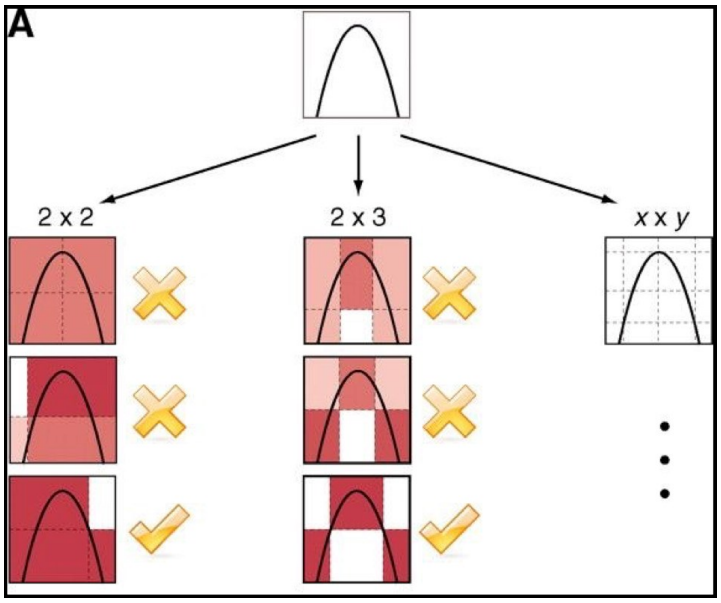

Figure 4.2 Visual representation of the MIC procedure. The algorithm recursively divides the function into squares and looks for the division that offers the strongest (deepest red) relationship.

The "Top-X" metric is defined as follows. For a given value of x [1-100], the X percent bins with the highest values in the two compared genomic datasets are identified and overlapped, and the percentage of overlapping bins is returned. As such, a score of 1 represents two experiments that have exactly the same X% most enriched regions, while a score of 0 would represent two experiments for which the X% most enriched regions are completely different. Thought this is not a formal similarity measure, we believe this type of measure gives a quick and easily understandable score of what regions are enriched in two given experiments. Moreover, it is simple to implement and fast to run. However, the value of X is suggested to vary depending of the type of experiment compared (e.g. for broad histone marks X could be between 10 and 20%, while for experiments targeting transcription factors one may want X ~1-2%).

2.3 The GeEC-Prep module: HDF5 conversion

Converting genomic format to HDF5 can be split into three parts as exemplified in Figure 4.3. Data parsing is handled by interfacing with the NGS++ Python wrapper, a library that allows reading most genomic formats. If the input data is in read format (SAM, BAM, BED), the genomic density signal is calculated by summing the reads at a base pair resolution for any given position (more precisely from the start position of the read to the start + read length position), while input data already in density signal format (WIG, BigWig, BedGraph) is read directly. The signal is next binned throughout the genome using a user-defined resolution by averaging the values in a given bin. Writing the signal to an HDF5 file is done with the `h5py` module (Collette et al., 2015) with data stored and identified at the scaffold (e.g. chromosome) level. For instance, a file containing data from the canonical human chromosomes will have 23 (or 24 depending whether chrY is present) entries (scaffolds), and the size of each vector will be constant across datasets, no matter the amount of data present in each file.

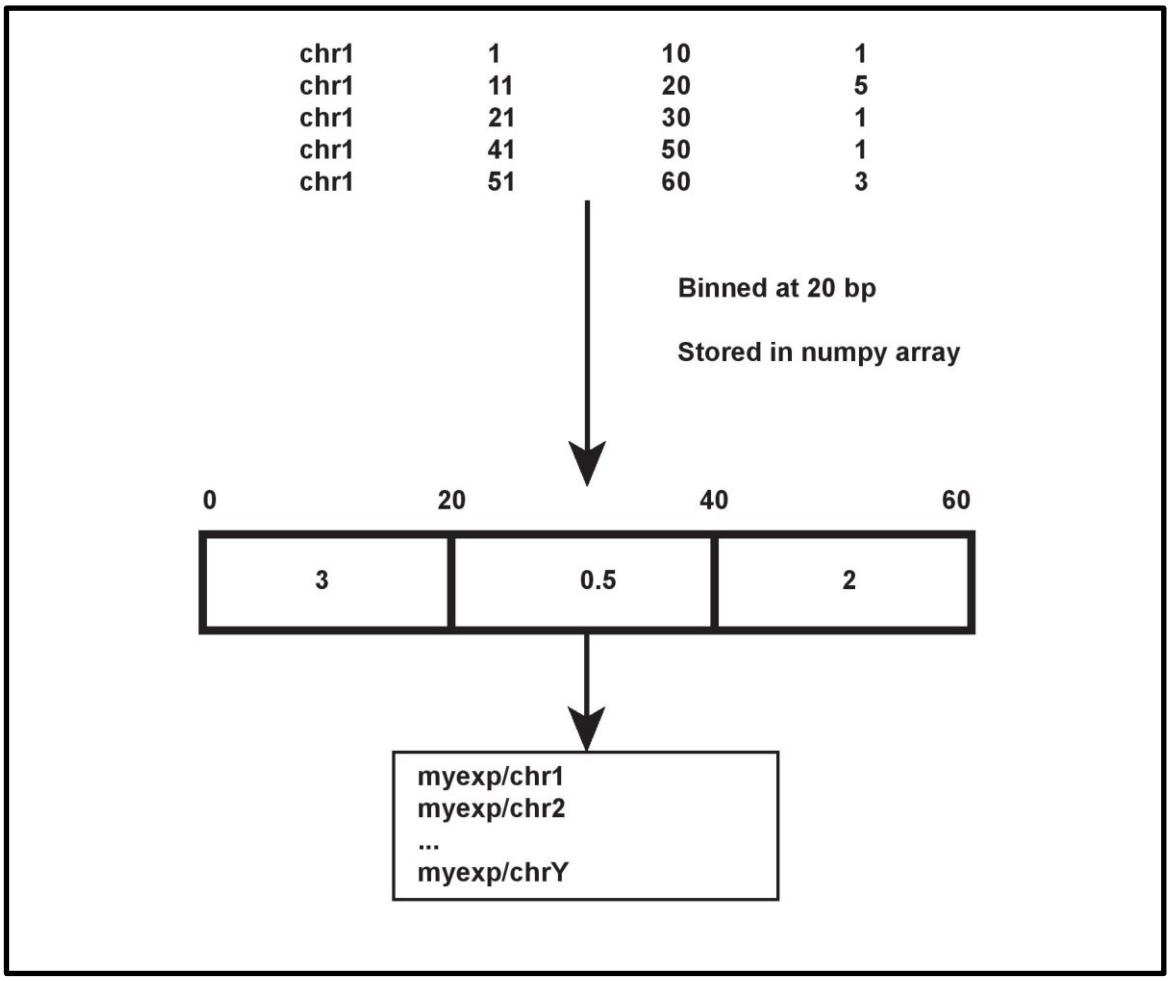

Figure 4.3 Summary of the three steps for creating an HDF5 file from genomic data. The finale step represents writing to the HDF5 file. In this case, data is store per chromosome.

To handle conversion of large amounts of datasets, the GeEC-Prep module is MPI compatible, allowing it to run seamlessly on distributed computing nodes or by running multiple threads on a local computer. It should be noted that loading human genomic data for conversion imposes an important memory burden of up to 1 GB per dataset per thread. This limit should be taken into account when running large batches on memory-limited hardware.

2.4 The GeEC-Corr module: generating the similarity matrix

Producing a similarity (e.g. correlation) matrix is a $O(n^2)$ task that would be trivial for our data sizes considering that it is rare to work with more than a few thousands elements when working with human NGS datasets. However, the size of individual genomic dataset, which can be labelled as "fat data", complicates the manipulations in memory. At a resolution of 1 bp, a single dataset takes over 2 GB of RAM. This precludes working with more than a handful of datasets. The I/O necessary to compare thousands of datasets when we can only load a few at a time per computer is therefore prohibitive. However, working with binned data not only reduces the data size to a manageable level, but as demonstrated in the result sections, may be a superior representation for certain type of analysis.

The GeEC-Corr module was designed to work with job dispatching tools found on most high performance computing clusters. As such, a single execution of the module can run anywhere from a single pairwise comparison to an entire matrix of millions of comparisons, allowing great flexibility when dispatching the tasks to a distributed computing cluster. Once generated, the results from several batches can be directly concatenated with no intermediate steps required.

Additionally, it is possible to select or mask specific regions of the datasets. This can be used to both ignore unwanted regions (e.g. blacklisted regions) and to select a subset of regions of interest (e.g. only genes). This is done via command line options pointing to user-provided BED files containing the corresponding regions.

In the context of the GeEC-Corr module, a single pairwise comparison is done by the following steps on a HDF5 file containing all the datasets:

- Read the two dataset names to compare
- Read the files containing the regions to ignore or select (optional)

- Open the HDF5 file
- For each scaffold of each dataset
    - Load the scaffold data from the HDF5
    - Remove or mask appropriate regions (optional)
    - Measure the similarity using the appropriate function
    - Store the result
- Close the HDF5 file

Most similarity functions used come from either the NumPy (Hugunin, 1995) or Scikit-learn (Jones et al., 2001) library, but the module is written to also allow user-defined functions. The results are preserved at a scaffold level to allow more precise data exploration should the user desire it.

Once the data has been generated for all pairwise comparisons, a single script can be run to transform the scaffold results into a tab-delimited result matrix, for now using a weighted average of the similarity values based on each scaffold proportion in the total genome size.

2.5 The GeEC-Analysis module: analyzing the results

The GeEC-Analysis module is in fact a number of small scripts designed to facilitate the analysis of the previously generated matrix. Written in Python 2.7 and using NumPy, SciPy (Pedregosa et al. 2011), and Scikit-learn it provides a number of easy to use functionalities:

- Hierarchical clustering
- Heatmap and dendogram visualization
- Validation by Adjusted Rand index (ARI)
- Sub-matrix extraction

2.5.1 ARI metric

The ARI validation, extensively used in the results section, is a well-known external cluster validation metric (Handl, Knowles, and Kell, 2005). It is based on the Rand index (RI).

The RI is a measure of similarity between two clusters. In this context, a true positive (TP) is when two similar items are assigned to the same cluster. A true negative (TN) is when two dissimilar items are assigned to different objects. A false positive (FP) is when two dissimilar items are assigned to the same cluster while a false negative (FN) is when two similar objects are assigned to different clusters. Given the above, the Rand index is defined as such.

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

The value given by the Rand index are between 0 and 1, with 1 the clusters generated match perfectly with the ground truth (e.g.: gold standard, validation data, labelled data), while a score of 0 represents a situation where every association done by the algorithm is different to that of the ground truth. There are several known problems with the RI metric including the fact that index varies substantially with the number of clusters (Morey and Agrest, 1984).

The Adjusted Rand index (Hubert and Arabie, 1985) is a version of the RI that seeks to correct for chance. Specifically it corrects for the fact that a random cluster assignation has a greater chance of being correct with a small number of clusters then a larger (e.g. : If there are 10 000 clusters, it is unlikely you will randomly select the correct one). The additional calculations needed are well explained in the supplementary parts of (Yeung et Ruso, 2001). An important distinction is that unlike the RI, the ARI can potentially have a negative value [-1,1] if the

assignations are substantially worse then expected by chance. However in most cases, the metric is positive.

As such, the ARI is a powerful metric for validating that the information encoded in a representation (correlation in this case) can be used by a clustering algorithm to correctly group the data. Offering this functionality to users allows both validation and exploration possibilities.

2.5.2 Hiearchical clustering

The clustering algorithm used by GeEC is a standard single-linkage hierarchical algorithm as implement in the Scikit-learn package. For the purpose of obtaining a defined number of clusters, the merge tree was cut at the specific high needed to obtain the required number of clusters. This cut is independent of any other factors (clustering density, merge proximity, etc.).

3. Results

In this section we discuss two sets of results that were used to validate and explore the functionalities of the GeEC toolset. While we offer comparisons to similar software towards the end of this section, we note that to the best of our knowledge, there is no other toolset aimed specifically at generating large-scale similarity matrix from NGS datasets. As such we have concentrated on demonstrating the validity and usefulness of the toolset.

The use of the ARI is an important element of our validation method. In the two tests, the datasets had a set of metadata containing both "positive" labels expected to participate to the clustering (e.g. chromatin mark) and "negative" labels (e.g. sequencing center). This allowed us to evaluate if the generated similarity matrix obtained after varying some parameters contained relevant relationships information extracted after the clustering method.

As discussed previously, the GeEC toolset also supports the exclusion of regions from the analysis process. In the case of the two tests, we chose to exclude regions that tend to show artificially high signal and blacklisted by Anshul Kundaje as part of the ENCODE project (https://sites.google.com/site/anshulkundaje/projects/blacklists). These regions cover features such as, centromeres, telomeres and satellite repeats. They are known to influence the results of Pearson correlations and contribute little relevant information.

3.1 Test #1

Our first experiment contained 167 datasets generated through the ENCODE project (http://genome.ucsc.edu/ENCODE/terms.html) and mapped onto the hg19 reference genome and stored in various genomic formats by the ENCODE Data Coordination Center (http://genome.ucsc.edu/ENCODE/downloads.html). Labels extracted from the metadata of these datasets are described in Table 4.1.

Table 4.1 Description of the labels

Label category	Description
bioRep	Biological replicates (ex: rep1, rep2)
chromMark	Chromatin mark (ex: H3K4me3, H3K36me3)
cellType	Cell types (ex : MCF7, GM12828)
tissue	Tissue type (ex: Kidney, Lung)
consortium	Project that generated the data (ex: ENCODE, Roadmap)
seqCenter	Sequencing center (ex: Broad, UCSD)
dataFormat	Data format used to store the data (ex: WIG, BAM)

This experiment was designed to characterize the impact of many parameters and identify a default parameter set to use for the second experiment. More specifically the parameters explored were:

- Similarity Measure: Pearson / Spearman / MIC
- Binning size: 100 bp / 1 Kb / 10 Kb / 50 Kb / 100 Kb
- Regions: Whole Genome / Genes / TSS / Regulatory regions / Blacklisted regions

Before describing these explorations, we will first present the ARI results obtained using one combination of the parameters with all the datasets to establish a benchmark, then the results obtained with subsets of the datasets. Note that the complete matrices of ARI results are presented in Annex B.

3.1.1 Clustering all datasets with Pearson on bins of 1 Kb from the whole genome

As shown in the column "All" of Table 4.2, the first striking observation based on the ARI values is that both the *bioRep* and the *chromMarks* labels were clustered fairly well using 1 Kb bins and Pearson correlation on the whole genome (after removing the ENCODE blacklisted regions). We were initially expecting the labels of *bioRep* to give a superior ARI value than the *chromMark* labels; on reflection however, the opposite is perfectly understandable for two reasons. First, the clustering requested for biological replicates is incredibly precise since in this experiment there is on average 2 biological replicates. While this grouping by biological replicate may seem natural, it happens often for instance that the replicates are not sequenced together but rather with some other samples from very similar cell type, then some batch artefacts may artificially push two datasets from the same batch to be slightly more similar overall than with their real biological replicate. Second and mostly, it is well know that high-level associations are often easier to identify then low-level fine ones. This is in part due to the respective size of the clusters with larger higher-level clusters (e.g. the chromatin marks category containing only five different labels with up to 50 elements) being more forgiving to minute miss-classifications. The well-defined clusters of the *chromMarks* can be easily identified in Figure 4.4.

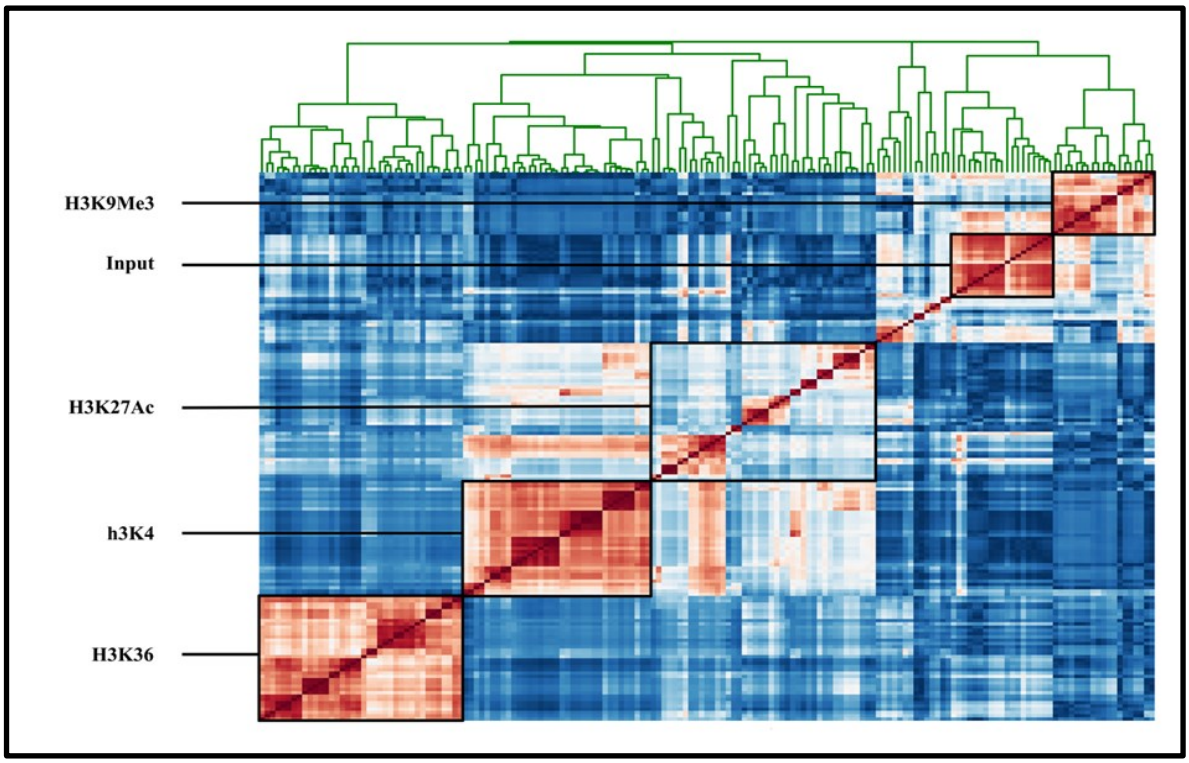

Figure 4.4 Heatmap of clustered data. Most squares represent clustered chromatin marks. Input are experiments used as control data during the sequencing process.

Using all the datasets, the ARI values for the five other label categories (from *cellType* to *dataFormat*) are around 0, clearly showing that these labels do not represent strong clustering relationships.

Table 4.2 ARI results for all the subgroups of datasets obtained using Pearson on bins of 1 Kb with datasets covering the whole genome without the blacklisted regions, or selecting only the blacklisted regions

Group Label Category	All	H3K4me3	H3K36me3	H3K27ac	H3K9me3	Input	Blacklisted
bioRep	0.51 (60)	0.65 (15)	0.67 (14)	0.84 (13)	0.61 (9)	0.23 (9)	0.32 (60)
chromMark	0.64 (5)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	0.02 (5)
cellType	0.04 (11)	0.47 (8)	0.62 (8)	0.83 (10)	0.37(7)	0.21(7)	0.28 (11)
tissue	-0.05 (7)	0.08 (4)	0.37 (4)	0.34 (7)	0.06 (4)	0.19 (4)	0.11 (7)
consortium	0.07 (4)	0.05 (4)	0.41 (4)	0.22 (4)	0.85 (4)	0.49 (4)	0.58 (4)
seqCenter	0.01 (7)	0.15 (7)	0.39 (6)	0.16 (4)	0.71 (4)	0.49 (5)	0.76 (7)
dataFormat	0.05 (4)	0.09 (4)	0.61 (4)	0.08 (4)	0.77 (4)	0.75 (3)	0.14 (4)
Count	**175**	**43**	**40**	**37**	**29**	**28**	**175**

The numbers in parentheses correspond to the number of different labels per category and per dataset subgroup and the last line contains the number of datasets per group.

3.1.2 Further cluster analysis

As expected and confirmed by the ARI score and the clusters obtained, the strongest relationship in our data is the one between chromatin marks. However, we wished to further explore the data and see if it could detect weaker relationships. This was achieved by extracting subgroups based on the *chromMark* label and reclustering those elements. For example, in the third row of Table 4.2, all 43 elements whose *chromMark* label were H3K4me3 were clustered and their ARI scores calculated based on the other label categories. Naturally, this means that the ARI for the *chromMark* for these sub-groups is equal to 1.

After *chromMark* the expected strongest relationship is the *cellType* label and this was confirmed by the results obtained on the subgroups. The ARI scores for *cellType* jumped from 0.04 when all the datasets are used, to values between 0.21 and 0.83 for the subgroups, and the ARI values for *bioRep* were also substantially improved.

The subgroups clustering scores allow other interesting insights. When observing the input sub-group, the ARIs result for the information rich categories (*chromMar*k, *bioRep, cellType*) are poor. This is expected and follows the logic of using input data as control elements. However, the high quality clustering along the *seqCenter* could be somewhat worrying at first sight. While it is well-known that specific protocols and experiments may generate identifiable bias, it was unexpected that this bias would be so well-defined as to be useable as a classifying value by a simple hierarchical clustering.

We also noted that the ARI results vary fairly significantly according to the chromatin mark used to generate the subclusters. Interestingly, the results may somewhat correlate with the proportion of the genome covered by the mark. Point source signals, such as H3K27ac (mainly enriched at enhancers) and H3K36me3 (on gene bodies) offer superior ARI results on categories such as *celltype*, while a larger signal such as H3K9me3 (repressive over large regions) offers results similar to the input category. Further study is required to ascertain why

this is happening, but it may be that the overall signal/noise ratio of marks covering a higher proportion of the genome are more susceptible to the experimental biases mentioned earlier.

The last column of the table represent a supplementary experiment run in part out of curiosity of the ARI results on the *input* column. We clustered the data using all the elements, but rather than removing the ENCODE blacklisted regions introduced earlier; we only selected those regions of the genome. The astoundingly high ARI score of 0.76 for the *seqCenter* demonstrates that the bias, at least for these datasets, can easily be identified and associated with specific consortium and even better with specific sequencing centers. This discovery helps demonstrate the use of our approach both as a validation and as an exploration method.

Finally we note that the range of improvements obtained on *the cellType* label varied significantly when using the *chromMark* subgroups. While we have no hypothesis to explain this large distribution, it is too early to suppose that certain chromatin marks offer a systematically stronger *cellType* relationship then others. We believe important insight could be obtained from repeating this procedure with larger and more diverse datasets and these insights could be used to confirm or infirm this possibility.

3.1.3 The impact of using different metrics

The previous results were obtained using the Pearson similarity measure. However, other metrics are available and to evaluate the impact of these similarity measures, we first used bins of 1 Kb covering the whole regions. Unfortunately, we found that the MIC metric was prohibitive to run on such large data. We then decided to only select the 2 Kb regions around the TSS as these regions only cover ~0.5% of the genome.

To our surprise and as shown in Table 4.3, the ARI values obtained with Spearman rank correlation were substantially worse than the values obtained with Pearson and MIC. A rapid analysis of the results suggests that it could be caused by the transformations on the data such that the distribution of the data covers a huge range of values; transforming these values into rank scores may artificially inflate the difference between two consecutive values in the same experiment. The ARI values obtained with MIC on the TSS are of decent utility, but took approximately 50 hours to run the analysis, which is about ten times more than the other metrics. It appears that operating MIC on the entire genome is of such computational complexity that it was impractical. Considering these results, we set aside Spearman and while MIC used in a genomic context warrants further study from a research standpoint, its excessive running time makes it a poor fit for large-scale usage. It may however be of use when evaluating specific regions such as a subset of TSS or specific gene groups.

Table 4.3 ARI results on the chromMark category using various similarity measures on 1 Kb bins covering the whole genome or only TSS

1 Kb	Pearson	Spearman	MIC	Top (1%)
Whole genome	0.64	0.16	NA	0.68
TSS	0.65	0.29	0.44	0.56

Finally the ARI values obtained with the Top 1% metric were similar to Pearson correlation ones on these datasets. Considering the general simplicity of this metric, this was unexpected, but does confirm our working hypothesis that the most enriched regions for a given experiment may contain the most valuable relationships. That said, using the Top metric does require the setting of an extra parameter that is the proportion of the genome to keep for comparison. As show in Table 4.4 and as expected, selecting a too large fraction of the

genome with bins of 1 or 10 Kb significantly affect the results. Note that the running time of this metric is comparable to Pearson one. While the overall performance of this metric makes it both a useful addition to the GeEC toolset and is worthy of further study, it should still be used for now with caution and cross-validation. As such, the Pearson correlation score is the current best default choice when generating a correlation matrix.

Table 4.4 Progression of ARI results on the chromMark category using the Top metric at various levels of genome selection on 1 Kb and 10 Kb bins

Top Bin	1%	5%	10%	25%	50%	75%
1 Kb	0.68	0.67	0.57	0.52	0.22	0.07
10 Kb	0.54	0.66	0.65	0.43	0.24	0.15

3.1.4 The impact of modifying the bin size

We next evaluated the impact of a wide range of bin size using the Pearson similarity measure on the entire genome. In other words, we wanted to evaluate how the ARI value of 0.64 obtained for the chromMark category with bins of 1 Kb will be affected by changing the bin size. As shown in Table 4.5, the highest ARI value was obtained using bins of 10 Kb, closely followed by bins of 1 Kb and 5 Kb. Based on these ARI values, it seems that compactness in the 1 to 10 Kb range best represents the size of features relevant for proper clustering, which also corresponds to the range the gene length in human. Binning beyond this size then aggregate features or perhaps more importantly, aggregates potentially large regions with no/low signal between features with the actual features, reducing their relevant information.

Reducing the bin size to 100 bp, which is approximately the resolution of the ChIP-seq assay, does not seem to improve the results. This is likely due to information being easier to extract in higher-level features.

Table 4.5 ARI results for the chromMark subgroup using Pearson on varying bin sizes covering the entire genome

100 bp	1 Kb	5 Kb	10 Kb	50 Kb	100 KB
0.52	0.64	0.63	0.70	0.46	0.48

While the ARI values tend to be within a similar range to each other in the 1 to 10 Kb level of binning, there is a practical advantage to work with larger bins as there is a direct inverse relationship between the size of the bins and the number required to cover the genome. Indeed, both the speed of generating a correlation matrix and the amount of memory it requires are directly related to the size of its representation.

We also wanted to evaluate the impact of calculating the correlations on different biologically important regions of the genomes, using the Pearson metric on bins of 1 and 10 Kb. Table 4.6 compares the ARI values obtained when running an analysis using the same metric and binning but with different subsets of the genome. While it was expected that using only regulatory regions would offer a worse score, the lower score obtained by using only gene regions came as a surprised, particularly when compared with the results obtained using the TSS.

At a 10 Kb resolution, it is natural that “bleeding” occurs, that is to say that short regions such as TSS (defined here as 2 Kb regions) will incorporate information from a region much larger

then there actual size. This can help explain why the TSS offer high quality results as the bins used will incorporate both upstream and downstream elements of the actual TSS, resulting in a rich mix of information. However, this does not explain why using the gene region offers such poor results. Even further, results generated using 1 Kb bins offer a similar, if less severe, pattern despite the fact that at this resolution the "bleeding" effect would be minimal.

Table 4.6 ARI results for different portions of the genomes at 1 and 10 Kb

Regions / Bin size	All	TSS (4%)	Genes (7%)	Regulatory regions (17%)
1Kb	0.64	0.65	0.56	0.56
10Kb	0.70	0.65	0.49	0.58

3.2 Test #2

Our second experiment aimed a substantially larger number of datasets. We selected 4749 datasets from the available data of the August 2014 data freeze from the International Human Epigenome Consortium (IHEC). Together, these experiments cover multiple cell-types and chromatin marks and incorporated data from multiple type of experiments such as 381 WGB-Seq datasets, 832 RNA-Seq datasets and 1951 Chip-Seq experiments targeting multiple histone marks. Together these datasets take slightly over 5 TB of disk space. The purpose of this analysis was to demonstrate the scalability of our tool. Additionally, as discussed later on, this served as a preliminary test for future services to be offered by GeEC.

The correlation matrix was generated according to the previously determined parameters, i.e. using the Pearson similarity measure on bins of 10 Kb covering the whole genome, except for the ENCODE blacklisted regions that were removed. The analyses were conducted on the Mammouth Parallel 2 (MP2) computing cluster at Université de Sherbrooke. The conversion task was split into sub-tasks of roughly 200 files each and then combined into a single HDF5 file. This step took roughly a wall-time of 4 hours on 700 CPUs (2800 CPU hours). Generating the correlation map was done using 96 CPUs and took roughly a wall-time of 2 hours (192 CPU hours). Some minor format conversion problems were encountered, causing us to drop 639 datasets at that time. However, we successfully identified the problems later and the necessary bugfixes were integrated into our toolset. The complete results are available in Annex C.

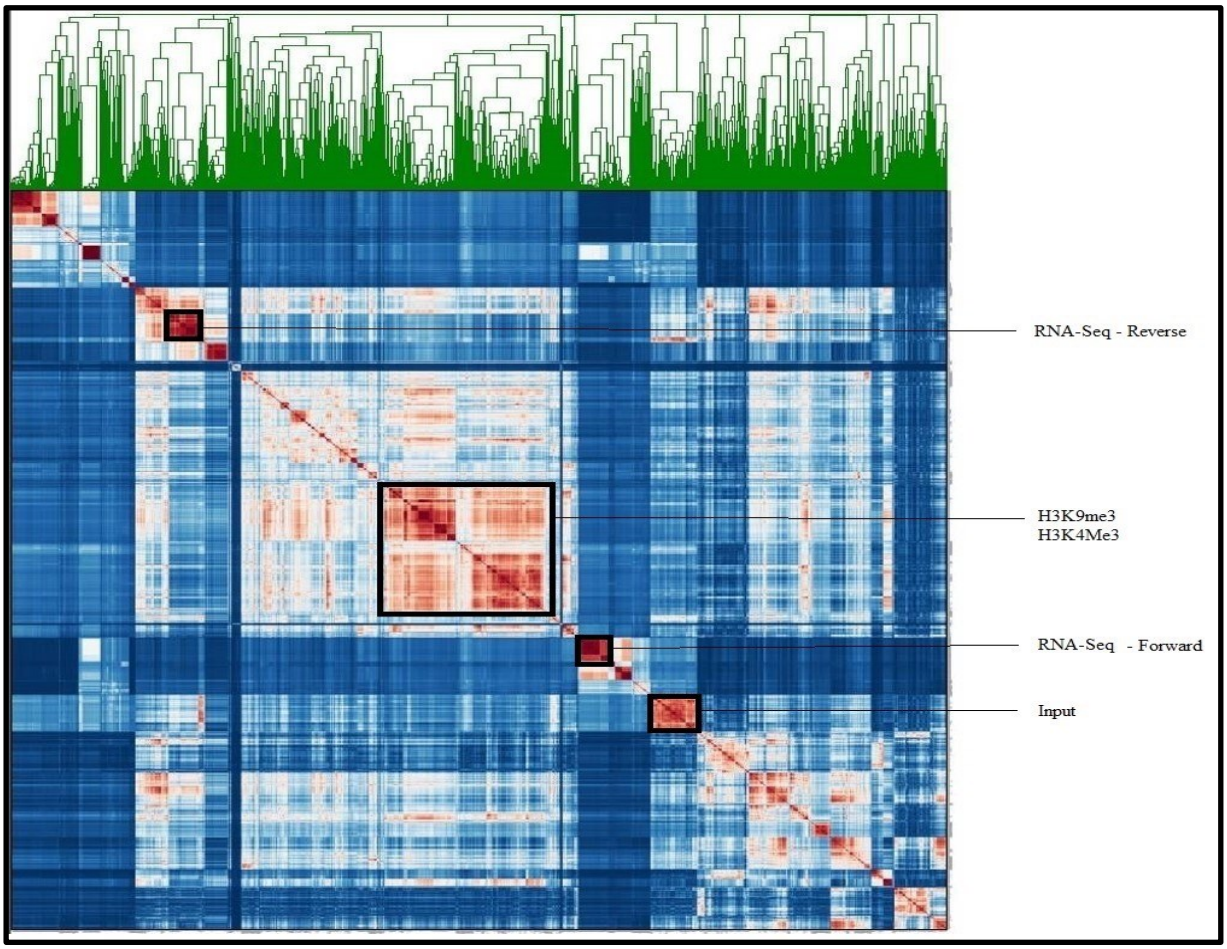

Figure 4.5 Heatmap of the clustered data. We have labeled certain groups to demonstrate the appropriate clusters. The large quantity of clustered experiments is responsible for the lack of definition of some clusters.

To evaluate the quality of the clustering of the correlation matrix illustrated in Figure 4.4, we used the same approach as in the first test, but using the labels directly extracted from the metadata provided by IHEC. As the number of experiments used is roughly 20 times higher than in test #1, we did not expected ARI values as high as previously. Indeed, as shown in Table 4.7 the values were overall lower in absolute, but the relatively much higher values obtained for the *chromMark*, *bioRep* and *cellType* compared to consortium negative control

category clearly indicate that our method is working. Further exploration revealed ARI values revealed two possible causes. First, it appears that the labelling of the datasets provided by the multiple groups involved in the consortium was not yet standardized. For instance, some datasets generated by McGill contains the *tissue* label "Kidney", while equivalent datasets generated by the Epigenome Roadmap were rather labeled as "Kidney_left" or "Kidney_right". In this example three different labels where used for data that that we would expect to cluster together. (e.g. all H3K4me3 datasets generated from the kidney should be clustered together and not with those from the heart, but not necessarily grouped by the left or right kidney). Secondly, manual validations show certain data to have been normalized using a routine that transforms data to a floating point value while other data contains raw discrete tag count.

Table 4.7 ARI results obtained by the Pearson correlation on bins of 10 Kb covering the whole genome (except blacklisted regions)

	All	H3K27ac	openChrom	RNAseq	input
repBiol	0.39 (1747)	0.289 (110)	0.26 (218)	0.51 (96)	0.09 (124)
Assay / Chrom mark	0.35 (160)	1 (1)	1 (4)	0.06 (2)	1.00 (1)
Assay category	0.14 (6)	1 (1)	0.12 (2)	1.00 (1)	1.00 (1)
cellType	0.18 (330)	0.289 (110)	0.32 (182)	0.46 (95)	0.09 (124)
consortium	0.04 (3)	0.07 (3)	0.12 (2)	0.05 (3)	0.22 (2)
count	4110	203	522	1077	311

While ARI scores obtained are lower than we would have desired, it also demonstrates that a correlation matrix can still be used to extract relationships, even when the data used is diverse and somewhat noisy. This validates our method as a good choice for the upcoming web interface that we will discuss further on.

3.2.1 Comparison of the GeEC toolset with other tools

Others have also published tools to generate correlations between genomic datasets on a pairwise or memory limited basis, including the tools ACT (Jee et al., 2011), bigWigCorrelate, DeepTools (Ramírez et al., 2014) and java-genomics-toolkit (Palpant, n.d.). In comparison, GeEC is designed to work in a distributed computing environment to generate very large similarity matrices. The other tools are designed to support only few file formats such as BigWig, BAM and WIG, while GeEC currently supports eight genomic file formats, insuring easy compatibility with available data. However, and as mentioned previously, GeEC requires a pre-processing phase to transform the data into the HDF5 format.

We decided to compare GeEC with three recent tools offering correlation options, all using the Pearson metric. DeepTools is a set of tools that is easy to integrate in the galaxy software. Java-toolkit is an example of a tool designed to assist in genomic analysis, while BigWigCorrelate is a script built on the backbone of the Kent BigWig library. However, repeated problems getting Java-toolkit to run on the Mammouth computing cluster lead us to set aside this tool. First, we validated that all tools gave similar correlation coefficients by calculating a correlation matrix from the same 10 datasets for each tool, then we calculated a Pearson coefficient correlation between these matrices (Table 4.8). While it may seem surprising that the correlations result where not identical, all three tools use different binning techniques and potential pre-process their data differently, explaining the differences. We next compared running times, with and without pre-processing in the case of GeEC, on BigWig

datasets of various sizes coming from IHEC. As expected, much of the running time of GeEC is front-loaded to the preparation phase. Consequently, once the data has been transformed, it is very fast to generate the correlation matrix. We also note that while DeepTools offers a good set of options, is integrated in the Galaxy platform and performs well on BAM format data (data not shown), its BigWig reader seems to suffer from performance problems. This leads us to conclude that bigWigCorrelate is best for point correlation for users comfortable with command line tools, DeepTools is well suited for point correlations with via the Galaxy platform, while our tool is best suited for large number of datasets. This can be clearly observed in Figure 4.7. GeEC and bigWigCorrelate running time intersect when generating the matrix with 30 elements and GeEC outperforms the other method when working on more than 30 elements (Figure 4.5). While bigWigCorrelate is faster when working with a handful of elements, as soon as the desired dataset exceed a few dozen, GeEC is increasingly faster. In our test #2 where several thousand elements where used, a tool such as bigWigCorrelate would have required a prohibitive running speed.

Table 4.8 Pearson r of the correlations obtained on a 10 by 10 matrix

	GeEC	**bigWigCorrelate**	**DeepTools**
GeEC	1	0.84	0.81
bigWigCorrelate	0.84	1	0.87
DeepTools	0.81	0.87	1

The fundamental difference between our toolset and similar ones remains the scale of the operations. Tools such as bigWigCorrelate and DeepTools are designed to locally generate small amount of correlation easily. Our toolset requires a larger set-up effort but is designed to generate large similarity matrices using a distributed computing model. As such, while both

types of tools may have some similarity, their scales are orthogonal, with both answering separate if not quite distinct needs.

Table 4.9 Execution time for three different correlation tools on different group of datasets

Group / Tool	10 small	10 medium	10 big	30 files	60 files	90 files
GeEC	P: 2 379s R: 350s T: 2 750s	P: 6 067s R: 300s T: 6 300s	P: 12 342s R: 340s T: 14 682s	P: 22 251s R: 7 058s T: 29 309s	P: 37 743 R: 26 661s T: 64 404.s	P: 61 254s R: 52 952s T: 115 206
bigWigCorrelate	1 918s	2 771s	3 476s	30 009s	116 059s	259 153s
DeepTools	119 342s	116 235s	132 540s	N/A	N/A	N/A

P: Preparation time. R: Correlation running time. T: Total

Finally we note that the computed times were measured using a preliminary version of our toolset. Current improvements in implementing C++ versions of GeEC-Prep and GeEC-Corr modules has substantially reduced the time needed to transform BigWig files into HDF5 and calculate the correlations, making GeEC much more competitive even when working with smaller sets of data.

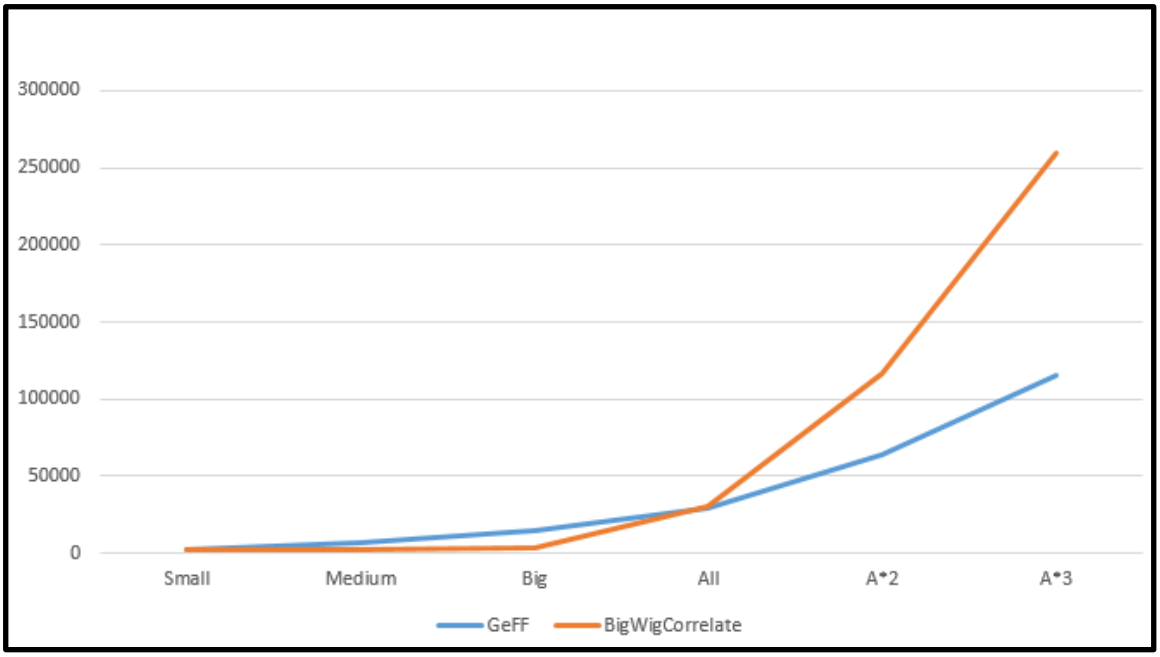

Figure 4.7 Comparison of running time for different group of files. While Kent is fairly invariant to the size of the files, GeFF increase approximately linearly. However as the number of files increases, the gain from the HDF5 conversion becomes apparent and GeFF rapidly becomes the more efficient choice.

3.2.2 Web integration and future developments

The GeEC toolset is currently being used to generate large-scale similarity matrices from more than 6000 datasets for the IHEC Data Portal (http://epigenomesportal.ca/ihec/). There are multiple services planned or in development that use such matrices; we present a few of the one's that we expect to be available in the near future.

3.2.2.1 User data upload

Above all, users will be able to upload their own datasets and use all discussed functionalities to compare their data with the publicly available human data from the IHEC data portal, as well as eventually with public datasets from model organisms. An interface that will be incorporated inside the Galaxy framework provided by the GenAP project (genap.ca) and running on some of the Compute Canada nodes, will allow users to upload their datasets in their private session and generate a correlation matrix including their data. Users will therefore be able to gradually upload more and more datasets and compare them with the ever-increasing number of the hosted public data.

3.2.2.2 Heatmap visualization

Users will be able to select all or a subset of the matrix they wish to view and have a publication quality heatmap generated. Optionally, this visualization can include a dendogram of the associated hierarchical clustering.

4. Conclusion and availability

In this paper we have presented GeEC, an ensemble of tools that can be used to generated large similarity matrices from thousands of epigenomic datasets and assist the user in certain post-processing analysis tasks. We have discussed how transforming the data into the HDF5 format can benefit other analysis tasks and we have validated that the generated similarity

matrix contains relevant relationships by clustering and evaluating the clusters using the ARI score. Additionally, we have demonstrated the utility of our toolset by discussing its current and planned integration in the IHEC data portal. The GeEC will be continually updated to reflect the need of the user base and will soon be publicly released through the lab's Bitbucket account (https://bitbucket.org/labjacquespe).

Conflict of Interest: None declared

Acknowledgements

We would like to thank the Encode project for providing the datasets used to validate the GeEC tool and the staff of the IHEC for valuable feedback.

References

Andrew Collette et al. 2015. "HDF5 for Python." http://www.h5py.org/

Clark, M. 2013. *A Comparison Of Correlation Measures*. *Nd.edu*. http://www3.nd.edu/~mclark19/learn/CorrelationComparison.pdf.

Handl, Julia, Joshua Knowles, and Douglas B Kell. 2005. "Computational Cluster Validation in Post-Genomic Data Analysis." *Bioinformatics* 21 (15): 3201–12. doi:10.1093/bioinformatics/bti517.

Jee, Justin, Joel Rozowsky, Kevin Y Yip, Lucas Lochovsky, Robert Bjornson, Guoneng Zhong, Zhengdong Zhang, et al. 2011. "ACT: Aggregation and Correlation Toolbox for Analyses of Genome Tracks." *Bioinformatics* 27 (8): 1152–54. doi:10.1093/bioinformatics/btr092.

Jim Hugunin. 1995. “The Python Matrix Object: Extending Python for Numerical Computation.” In *Proceedings of the Third Python Workshop*. URL:http://www.python.org/workshops/1995-12/papers/hugunin.html.

Jones, Eric, Travis Oliphant, Pearu Peterson, and others. 2001. “SciPy: Open Source Scientific Tools for Python.” http://www.scipy.org/.

Kent, W. J., a. S. Zweig, G. Barber, a. S. Hinrichs, and D. Karolchik. 2010. “BigWig and BigBed: Enabling Browsing of Large Distributed Datasets.” *Bioinformatics* 26 (17): 2204–7. doi:10.1093/bioinformatics/btq351.

Kong, Augustine, Daniel F Gudbjartsson, Jesus Sainz, Gudrun M Jonsdottir, Sigurjon a Gudjonsson, Bjorgvin Richardsson, Sigrun Sigurdardottir, et al. 2002. “A High-Resolution Recombination Map of the Human Genome.” *Nature Genetics* 31 (3): 241–47. doi:10.1038/ng917.

Li, Heng, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. 2009. “The Sequence Alignment/Map Format and SAMtools.” *Bioinformatics* 25 (16): 2078–79. doi:10.1093/bioinformatics/btp352.

Palpant, Tim. “Java Genomic Toolkit.” http://palpant.us/java-genomics-toolkit/.

Pedregosa, F, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, et al. 2011. “Scikit-Learn: Machine Learning in {P}ython.” *Journal of Machine Learning Research* 12: 2825–30.

Ramírez, Fidel, Friederike Dündar, Sarah Diehl, Björn A Grüning, and Thomas Manke. 2014. “deepTools: A Flexible Platform for Exploring Deep-Sequencing Data.” *Nucleic Acids Research* 42 (Web Server issue): W187–91. doi:10.1093/nar/gku365.

Reshef, David N, Yakir A Reshef, Hilary K Finucane, Sharon R Grossman, Gilean McVean, Peter J Turnbaugh, Eric S Lander, Michael Mitzenmacher, and Pardis C Sabeti. 2011. “Detecting Novel Associations in Large Data Sets.” *Science* 334 (6062): 1518–24. doi:10.1126/science.1205438.

Shendure, Jay, Robi D Mitra, Chris Varma, and George M Church. 2004. “Advanced Sequencing Technologies: Methods and Goals.” *Nature Reviews. Genetics* 5 (5): 335–44. doi:10.1038/nrg1325.

Speed, Terry. 2011. “Mathematics. A Correlation for the 21st Century.” *Science* 334 (6062): 1502–3. doi:10.1126/science.1215894.

“The HDF Group. Hierarchical Data Format, Version 5.” http://www.hdfgroup.org/HDF5/.

ANNEX C

Table 4.10 Complete results of Test #2 using 10 KB bins and Pearson

	repBiol	assay	assayCat	cellType	tissue	project	nbDatasets
all	0.39	0.35	0.14	0.18	0.04	0.04	4110
RNAseq	0.51	0.06	1.00	0.46	0.27	0.05	1077
openChrom	0.26	0.30	0.13	0.32	0.26	0.13	522
chromAcc	0.32	1.00	1.00	0.32	0.33	1.00	354
h3k27me3	0.32	1.00	1.00	0.32	0.21	0.03	275
h3k27ac	0.29	1.00	1.00	0.29	0.29	0.08	203
h3k4me1	0.24	1.00	1.00	0.24	0.23	0.01	268
h3k36me3	0.22	1.00	1.00	0.22	0.15	-0.11	284
h3k9me3	0.19	1.00	1.00	0.19	0.17	-0.04	253
h3k4me3	0.16	1.00	1.00	0.16	0.06	0.02	311
wgb	0.14	1.00	1.00	0.14	0.11	0.15	382
input	0.09	1.00	1.00	0.09	0.04	0.22	311

CHAPITRE 5

DISCUSSION ET CONCLUSION

5.1 Introduction

Je précise d'abord que ce chapitre a pour objectif d'offrir une vision personnelle sur les travaux que j'ai accomplis et de me permettre de détailler certains points qui n'avaient pas leur place dans les chapitres précédents. Dans ce sens, la discussion est écrite uniquement en mon nom et n'implique aucunement les autres auteurs ou collaborateurs avec qui j'ai réalisé ces travaux. De plus, je me permets de discuter de certains travaux qui n'ont pas donné de résultats directs mais qui ont influencé d'autres travaux qui sont aussi présentés dans la thèse.

Le chapitre se divise ainsi : j'offre d'abord une revue rapide de l'atteinte ou non des objectifs définis dans mon introduction, je décris ensuite les difficultés principales rencontrées durant mes travaux et j'explicite les solutions employées pour les surmonter, je me permets par la suite de critiquer mes travaux, particulièrement certaines étapes qui, en rétrospective, auraient été appropriées pour éviter ces problèmes, puis je termine avec une conclusion rapide.

Il est important de noter que j'occulte de cette discussion les travaux entourant le code de lecture du format BigWig puisqu'il a été suffisamment couvert dans le chapitre 3.

5.2 Résumé et évaluation des objectifs de mes travaux

Pour poursuivre la discussion amorcée dans l'introduction, dans laquelle je couvre les objectifs des travaux ayant menés à cette thèse, je vais d'abord discuter de l'objectif #1 de recherche dont les efforts n'ont pas porté fruits. Cet objectif a directement influencé des aspects des objectifs #2 et #3 et a occupé un temps significatif de mes premières années de doctorat. Je souhaite donc décrire plus spécifiquement l'influence qu'il a eu.

5.2.1 Objectif #1 : Écriture d'un algorithme de positionnement nucléosomal et écriture d'un algorithme d'apprentissage machine pour l'identification de régions enrichies

Au début de mes travaux de recherches, un de mes objectifs était d'utiliser mes connaissances en forage de données pour créer un nouvel algorithme afin d'assister les chercheurs à cibler les zones d'intérêts dans certains types de données de séquençage. Mes premiers efforts se sont donc orientés vers le développement d'un algorithme permettant de réaliser la déconvolution des données nucléosomales et ainsi d'offrir un aperçu visuel clair du positionnement potentiel des nucléosomes. Les efforts subséquents, influencés par la popularité de l'apprentissage machine, cherchaient à utiliser des réseaux de neurones pour permettre l'apprentissage et l'identification de régions génomiques à partir de données étiquetées.

L'algorithme de déconvolution permettait de prendre une expérience de séquençage nucléosomal, et d'identifier le positionnement probable des nucléosomes. L'approche récursive permettait une analyse implicite de la mobilité des nucléosomes dans une région donnée. De plus, le rendu pouvait être offert sous forme visuelle pour des régions ciblées, une fonctionnalité appréciée par les chercheurs qui ont testé les premières versions de l'algorithme.

Le programme d'apprentissage machine tentait de retrouver des régions génomiques similaires à celles déjà étiquetées par un chercheur dans d'autres expériences. Bien que moins avancé dans son développement que l'algorithme de déconvolution, les résultats étaient prometteurs.

Dans les deux cas cependant, les algorithmes se sont frappés à des problèmes importants de validation. C'est suite à l'incapacité de démontrer la fiabilité des analyses et la lenteur handicapante de la boucle de rétroaction que les algorithmes ont été écartés en faveur du développement d'outils plus dirigés. Cependant, ces travaux ont influencé les autres objectifs de manière importante.

La nécessité d'utiliser un ensemble d'utilitaires différents pour permettre la lecture des données de sources multiples à directement contribuer à établir l'importance du module de lecture de format NGS++. Pour les premiers mois de développement des algorithmes, j'étais constamment ralenti par des problèmes mineurs mais constants de lecture, de validation et de formatage; erreurs qui n'étaient aucunement liées à des travaux de recherche. L'établissement d'un outil qui permettrait la lecture uniforme des données génomiques m'est donc apparu comme un besoin important du domaine.

Les problèmes de validation rencontrés m'ont amené à modifier mon approche et mes attentes envers l'utilisation de données étiquetées. Dans l'analyse de données classique, dans laquelle j'ai été formé, il était habituel d'avoir accès à des données possédant un étiquetage bien établi qui permettait de tester et valider un algorithme de segmentation ou d'analyse. Suite à ces travaux et à la constatation de la difficulté d'obtenir des données similaires en bio-informatique génomique, j'ai assouplis ce que je considérais acceptable comme étiquetage de validation. Cela m'a permis d'utiliser la méthode présentée dans la validation de l'outil GeEC.

Malgré l'échec de ces efforts, je considère quand même que le temps investi a été une étape d'apprentissage importante ayant menée à la réalisation de mes autres travaux.

5.2.2 Objectif #2 : Faciliter le développement d'outils et la lecture de données génomiques en C++ (NGS++)

Mon objectif était de rendre accessible une librairie de code spécialisé offrant des fonctions permettant la manipulation de formats et de données de séquençage haut débit. Plus spécifiquement, je voulais permettre à un développeur de ne pas avoir à connaître les détails des formats de données génomiques avant d'écrire un outil de manipulation et d'analyse de ces données

D'un point de vue technique, cet objectif a été atteint. La publication de la librairie NGS++ a rendu disponible un ensemble d'outils puissants et, au moment de la publication, uniques qui permet à un développeur bio-informatique d'intégrer rapidement des données de séquençage dans un nouveau programme.

Cependant, la contribution d'une librairie ne peut dépasser son adoption par la communauté de développement. Deux ans après sa publication, il est honnête de dire que la librairie n'est plus activement utilisée. Le nombre de téléchargements a diminué au point d'être presque nul et la librairie n'a pas été citée dans des publications récentes d'outils. L'adoption de la librairie par la communauté est donc l'échec principal de l'objectif et je l'aborderai plus en détails dans la section 5.4.1.

5.2.3 Objectif #3 : Rendre accessible la génération de matrices de corrélation génomique (GeEC)

Mon objectif était de démontrer que la génération de matrices de corrélation de grandes tailles était accessible dans un contexte de génomique haut débit. Spécifiquement, je voulais

permettre de générer des matrices d'une taille supérieure à ce qui était précédemment possible et de démontrer qu'une matrice de cette taille serait un atout pour des travaux de recherche.

L'outil GeEC offert rencontre les critères principaux désirés. Non seulement, il est possible de l'utiliser pour générer des matrices faites à partir de milliers d'expériences de séquençage mais j'ai démontré l'utilité de ces matrices en analysant les résultats d'une matrice de plusieurs milliers d'éléments. De plus, les choix d'architecture informatique de l'outil vont permettre d'étendre les fonctionnalités pour viser des comparaisons ciblées ou pour introduire des mesures d'informations supplémentaires.

Cependant, GeEC, dans sa forme actuelle, souffre d'une faible accessibilité. Pour atteindre la pleine performance de l'outil, l'étape de pré-traitement des données bénéficie fortement d'une architecture de stockage qui doit être gérée de manière distincte de l'outil. De plus, l'étape de corrélation doit être exécutée sur une architecture de calcul distribuée si l'usager souhaite obtenir le plein potentiel de l'outil. Ces deux aspects ne sont pas directement encadrés par l'outil et doivent être pris en charge par un technicien qualifié, compétence qui n'est pas disponible à toutes les équipes de recherches.

5.3 Défis rencontrés et solutions

Je discute ici de certains des défis rencontrés durant la réalisation de mes objectifs. J'expose les solutions qui ont été retenues et dans certains cas, celles qui ont été essayées et ensuite délaissées.

5.3.1 NGS++

Les travaux nécessaires pour la production de NGS++ retiennent plus du développement logiciel que de la recherche traditionnelle. Dans ce sens, les défis rencontrés concernent majoritairement des choix technologiques et des décisions d'architecture interne.

5.3.1.1 Mauvaise définition de certains formats

La multiplicité des formats utilisés en bio-informatique est un problème que j'ai explicité dans l'introduction, Cependant, pendant l'écriture du module de lecture de données, nous avons constaté que certains formats avaient plusieurs définitions. Spécifiquement, la définition du format GFF précisée dans les sites du UCSC, de Ensembl et de Gmod, offrait de légères différences. Même chose pour celle du format GTF, qui différait également dans plusieurs sites. L'introduction d'une meilleure uniformisation des formats ainsi que la définition du format GFF 2.2 a éliminé ces erreurs depuis. Il est à noter cependant, qu'à la période de l'écriture de NGS++ (2012-2013), elles étaient toujours présentes.

Une correspondance privée avec les sites concernés n'a pas permis de résoudre le problème et l'absence de « propriétaire » du format a empêché de consulter une autorité responsable. J'ai donc décidé de me fier à un vote majoritaire des données. Les différences étant minimes, j'ai codé un lecteur de format pour chaque variation et j'ai exécuté les lecteurs sur des blocs de données téléchargées de la base de données BioMart. Le lecteur qui a réussi à lire le plus de données en lançant un minimum d'erreurs se retrouve aujourd'hui dans la librairie.

5.3.1.2 Accessibilité via d'autres langages

L'évaluation initiale de l'article de NGS++ par Oxford Bioinformatics était majoritairement favorable, mais une des revues avait la requête spécifique de fournir une version de la librairie dans un langage de scriptage, soit R, Perl ou Python. Au moment de la l'évaluation par le journal, il n'y avait aucune expérience ou expertise dans l'équipe de développement sur comment accomplir ce type de conversion. J'avais à peine touché à des langages non-compilés et la demande me semblait excessive considérant le délai de trois mois offert pour accomplir les changements nécessaires.

Suite à l'échange de plusieurs courriels avec l'éditeur en charge, nous avons établi un compromis. Plutôt que de réécrire une version de la librairie, nous avons accepté d'écrire un « wrapper » autour de la librairie qui permettrait à des développeurs Python d'y accéder.

Plusieurs méthodes existent pour offrir une interface en Python à une librairie en C++, mais les options les plus populaires à ce moment était ctypes (https://docs.python.org/2/library/ctypes.html), Swig (http://www.swig.org/) et Cython (http://cython.org/). Après avoir consulter les membres d'un laboratoire voisin ayant une expertise en Python, nous avons décidé d'essayer ctypes et Swig. Quelques semaines d'essais nous ont amenés à la conclusion que l'écriture de l'interface en Swig serait un choix supérieur à long terme, mais la complexité relative de l'outil comparé à ctypes et les restrictions de temps imposés par le processus de revue d'articles nous ont poussés à écrire l'interface en ctypes.

Le résultat est une interface incomplète mais utilisable qui donne un accès intégral aux modules de lectures de données et un accès partiel aux fonctions d'exécutions de la librairie. La rapidité relative d'écriture de cette interface m'amène à conclure qu'il aurait été possible d'offrir une interface complète à la librairie, ce qui aurait grandement aidé la diffusion de la librairie, un point abordé plus tard dans ce chapitre.

5.3.1.3 Validation de la librairie

Tel qu'exposé dans le chapitre 2, la complexité de la librairie est un élément clé des possibilités offertes. Par contre, cette même complexité, en partie due à l'utilisation de métaprogrammation, augmente exponentiellement les chances d'erreurs. Sommairement, le code utilisant la métaprogrammation n'est pas validé au moment de la compilation de la librairie, mais plutôt au moment de la compilation du programme qui utilise ce code. Il est possible de développer de manière extensive une librairie utilisant la métaprogrammation sans jamais provoquer d'erreur de compilation si les fonctions de cette librairie ne sont pas utilisées et testées durant le développement.

La première version de la librairie, écrite en 2010-2011 pour une utilisation interne du laboratoire, s'est retrouvée victime de cette problématique. Plusieurs dizaines de fonctions et de classes centrales à la librairie n'étaient testées que pour les usages très spécifiques du laboratoire et aucun autre test n'était rajouté. Quand est venu le temps d'en élargir l'utilisation, les programmes généraient une telle quantité d'erreurs qu'il est devenu évident que la librairie était inutilisable dans son état présent.

J'ai donc commencé à la réécrire en intégrant des concepts de développement par tests (TDD). Utilisant l'outil Gtest (https://github.com/google/googletest) de Google pour assister le processus, j'ai écrit un ensemble exhaustif de tests qui permettait de valider l'usage de métaprogrammation à mesure que celle-ci était intégrée à la librairie. Malgré l'écriture de plus de 700 tests disponibles à la compilation, un des arbitres au moment de la soumission de l'article a trouvé de multiples erreurs. Le résultat final du processus de développement a mené à l'écriture de plus de 1000 tests distincts qui ont garanti la robustesse du code publié.

5.3.2 GeEC

L'écriture et la validation du module de corrélation se rapprochent de la recherche dans les domaines de l'analyse de données. Contrairement à NGS++, il fallait non seulement relever les défis de développement logiciel, mais également démontrer que l'application de l'outil pouvait servir à informer des travaux de recherche classiques.

5.3.2.1 Gestion et formatage des données

Les premiers prototypes de GeEC utilisaient l'interface Python de NGS++ pour lire les données. Il était donc facile d'utiliser des données provenant de multiples formats génomiques pour effectuer les tests. Cependant, il était flagrant que, à l'exception possible du format BigWig, la lecture des données serait beaucoup trop lente. De plus, une méthode de stockage des métadonnées devenait une option souhaitable considérant la diversité des données traitées.

Le choix de convertir les données en HDF5 s'est imposé en grande partie grâce à la disponibilité de l'expertise nécessaire. Bien qu'aucun membre de l'équipe ne possédait les connaissances appropriées pour la gestion d'immenses données, les techniciens du superordinateur « Mammouth » avaient souvent travaillés avec ce format.

L'utilisation du format HDF5 n'a tout de même pas été sans difficulté. Si l'interface d'accès a pris au plus quelques jours à rendre fonctionnel, un temps considérable a été perdu sur des erreurs mineures. L'implémentation parallèle de HDF5 n'a jamais offert un niveau de stabilité suffisant pour que je le considère hors de la phase de prototypage.

Malgré qu'il soit, en théorie, possible de stocker des données HDF5 de tailles indéterminées, j'ai rencontré deux limites. Quand une expérience était gardée à une résolution trop précise (en

pratique 10 pb et moins), son temps d'accès augmentait de manière exponentielle. Par ailleurs, quand le nombre d'expériences stockées dans un fichier HDF5 dépassait une certaine taille (estimée à 10+GB), des erreurs d'écriture et de lectures fréquentes survenaient. Des discussions sur les forums appropriés et avec l'équipe de techniciens de l'Université de Sherbrooke n'ont pas permis de trouver de solution à ces problèmes.

L'erreur reliée à la taille du fichier HDF5 a été la cause de l'échec de plusieurs séquences d'expériences. L'erreur n'apparaissant que pour un certain volume de données, il m'était très peu clair pourquoi ces expériences avaient échouées tandis que d'autres, très similaires mais plus petites, n'offraient aucune erreur.

L'erreur reliée à la résolution d'un seul fichier est responsable d'une utilisation massive et inutile de ressources durant la période de validation. Quand j'ai exécuté les tests à une résolution de 10pb, j'avais précédemment réalisé des tests à une résolution de 100pb et plus. Mes estimés étaient qu'un test à 10 pb prendrait environ 2 fois plus de temps que celui à 100pb. En pratique, le test a pris plus de 15 fois le temps estimé et a vidé les banques d'heures de travail qui nous étaient allouées sur le superordinateur utilisé. Sans être une difficulté critique, ceci a retardé la réalisation des tests des mois subséquents.

Pour éviter cette situation, il a fallu limiter la résolution. Heureusement, les validations subséquentes ont démontré que générer la corrélation à une résolution en dessous de 1000pb offre des résultats de moindre qualité.

5.3.2.2 Distribution des calculs

Après que les tests initiaux aient confirmé que générer de grandes matrices prendrait un temps prohibitif, deux options technologiques pour rendre le calcul distribué ont été envisagées, soit Hadoop MapReduce ou MPI en combinaison avec un outil de planification de tâches :

Hadoop MapReduce est une implémentation très populaire du paradigme MapReduce utilisée pour distribuer des tâches de calculs sur des architectures distribuées. L'implémentation gère la distribution des calculs ainsi que la combinaison des résultats. Sa popularité est telle que l'implémentation est disponible sur la majorité des grappes de calculs académiques ainsi que sur les services d'infrastructure de l'industrie (AWS/Azure/etc.). Suite à une décennie de développement, la technologie est bien rodée et possède de nombreuses librairies (MrJob (https://pythonhosted.org/mrjob/)).

MPI est un protocole de communication qui permet d'assurer la communication entre différents nœuds/CPUs exécutant un travail qui nécessite un partage de l'information. Les outils de planification de tâches sont des programmes spécialisés retrouvés dans le cadre d'une grappe de calculs haute performance. Ils assistent les usagers de ces machines à distribuer leurs tâches de calculs sur les multiples nœuds/CPUs.

Lors de l'évaluation de la possibilité d'utiliser Hadoop MapReduce, le protocole semblait parfaitement adapté. Cependant, je n'avais aucune expérience pratique avec le paradigme MapReduce et l'implémentation n'était pas disponible par défaut sur les grappes de calculs de l'Université de Sherbrooke.

En contrepartie, le protocole MPI était régulièrement utilisé dans des travaux de recherche à l'Université de Sherbrooke et les outils de planification étaient activement déployés et supportés sur les grappes de calculs de l'Université. Des tests ont démontré qu'il serait facile d'exécuter les corrélations de manière distribuée et le protocole MPI offrait les options

nécessaires pour contourner les problèmes liés aux formats HDF5 mentionnés précédemment. Il a donc été décidé d'adopter ces technologies plutôt que Hadoop MapReduce.

Dans ce sens, la décision finale d'opérer la corrélation en utilisant les outils de planification et le protocole MPI a été prise en grande partie en ligne avec les pratiques courantes de la grappe de calculs locale. Les technologies offraient les options nécessaires mais ont fortement liés le code développé à un certain type d'infrastructure de calculs.

J'ai depuis développé ma connaissance de Hadoop MapReduce et il est probable que si je devais refaire le choix précédent, la robustesse ainsi que la grande disponibilité des outils entourant cette technologie auraient orienté mon choix dans cette direction. Il est probable qu'une telle implémentation serait plus portable.

5.4 Critique des travaux

Dans cette section je couvre certaines critiques personnelles concernant mes travaux. Cette section n'est aucunement exhaustive, mais présente les améliorations que je considère potentiellement les plus importantes. Dans certain cas, des travaux futurs pourraient palier à certains de ces problèmes.

5.4.1 NGS++

Bien que la publication initiale de la librairie NGS++ a suscité un certain intérêt parmi la communauté des développeurs en bio-informatique, je constate que quelques années après sa publication, la librairie n'a pas réussi à maintenir une présence active. Cette section est donc plus une critique rétrospective qui, je l'espère, pourrait être utile à un lecteur souhaitant publier une librairie dans un contexte similaire.

5.4.1.1 Choix du langage et accessibilité de la librairie

Le choix initial d'implémenter NGS++ en C++ est une décision qui je considère a nuit à la librairie et n'a pas considéré quelques aspects importants. Très sommairement, deux questions devraient être posées lors de la sélection d'un langage pour l'écriture d'une librairie de codes :

1) Est-ce que le langage est adapté aux besoins qui cherchent à être répondus?
2) Est-ce que le langage est utilisé par la communauté visée, c'est-à-dire la communauté de bio-informatique génomique ?

La nécessité de la première question paraît évidente. Dans le cas de nos besoins, je considère que le C++ est un langage approprié. Sans être spécifiquement adapté aux tâches de prototypage et de manipulation de données, le langage est un couteau suisse, suffisamment puissant et rapide pour répondre aux besoins ciblés dans l'objectif.

La deuxième question est plus subtile. Le potentiel d'une librairie est égal aux nombres de développeurs qui l'utilisent pour accélérer le développement de nouveaux outils. La librairie peut offrir une interface puissante et conviviale, mais cette interface n'est qu'un accélérateur de

développement qui, en soit, ne contribue pas à faciliter la recherche. Il en découle qu'une librairie se doit d'être accessible aux développeurs de la communauté visée.

Je n'ai considéré cette question que vers la fin développement de NGS++ et un survol informel de la communauté de développeurs m'a fait réaliser que les langages communs étaient ceux associés à l'analyse de données (Perl, R, Python, Matlab) avec un faible taux d'adoption du C/C++. Il était d'emblée probable qu'une librairie en C++ aurait des difficultés à percer dans la communauté, puisque ce langage n'est que très peu utilisé par cette dernière.

Malgré cela, nous avons décidé de distribuer la librairie telle quelle avec l'ajout d'une interface partielle en Python suite à des commentaires de revues. Des développeurs qui ont essayé la librairie, plusieurs ont spécifiquement fait la demande d'une amélioration de l'interface Python, ce qui renforce la pertinence de cette critique. Les raisons de l'absence de ces améliorations après la publication sont abordées dans la section 5.4.1.2.

Avec quelques années de recul, la solution est simple bien que longue en temps de développement. Une des forces du C++ est l'existence d'un nombre d'outils permettant à une librairie d'un langage d'être accessible par des programmes en d'autres langages. Si à la publication de la librairie, des interfaces robustes en Python et R avaient été disponibles et si la présence de ces interfaces avait été mise de l'avant dans les exemples d'utilisation, il est probable que la librairie aurait connu un plus grand succès.

5.4.1.2 Support après la publication

Le temps alloué pour offrir un support aux utilisateurs potentiels de NGS++ suite à sa publication n'avait pas été considéré au départ. J'ai approché la publication de la même manière qu'une publication classique, c'est-à-dire que mes attentes étaient que suite à

l'acceptation du papier, je commencerais à travailler sur le prochain projet, possiblement construit sur les fondations de NGS++. La documentation exhaustive ainsi que les quelques exemples fournis sur le site web devaient être suffisants pour les développeurs.

Je n'avais donc pas prévu investir d'efforts importants dans la librairie directement après la publication et d'autres projets sont venus absorber la majorité de mes efforts. Pourtant les quelques mois suivant la publication initiale de la librairie ont vu le plus de téléchargements et de courriels contenant des questions et parfois des requêtes d'amélioration.

À cette étape, il aurait été idéal de répondre aux requêtes des usagers et d'implémenter, rapidement, les modifications nécessaires pour garder l'intérêt démontré. De plus, cette période aurait été le moment idéal pour communiquer directement avec les développeurs et leurs offrir une assistance directe pour que NGS++ soit utile à leurs efforts de développement. Si, suite à ces efforts, la librairie avait joué un rôle dans la publication d'un outil important, la visibilité de la librairie aurait été grandement améliorée.

Ce ne fut pas le cas. Les quelques utilisateurs qui ont fait des requêtes d'options supplémentaires ont rapidement compris que le développement futur de la librairie se ferait très lentement. De plus, je n'ai jamais offert mon aide aux développements d'outils qui auraient potentiellement utilisés NGS++. Environ six mois après la publication, l'intérêt était rendu minime et je suis convaincu qu'une meilleure gestion des efforts après la publication aurait potentiellement amené un résultat différent.

Ma conclusion est donc simple. Suite à la publication, il est important d'avoir un plan pour répondre rapidement aux demandes des utilisateurs et de démontrer que la librairie reçoit, malgré sa publication, des améliorations rapides et constantes. Cet effort continu aide à instaurer un climat de confiance chez les utilisateurs et permet potentiellement d'assurer la pérennité de la librairie publiée. La décision de continuer à investir ou non une part importante de temps dans le projet devrait donc être prise quelques mois après la publication quand l'utilisation active de la librairie pourra être mieux mesurée.

5.4.2 GeEC

Étant en développement actif, je vais éviter les critiques techniques de GeEC, celles-ci étant possiblement caduques au moment de la lecture. Je parle uniquement d'une facette du papier présenté au chapitre 4 que je considère comme un manque important.

5.4.2.1 Absence d'exemple de découverte avec l'outil

La validation des résultats donnés se situe presque exclusivement au niveau de la reproduction. En vérifiant qu'un partitionnement des matrices de corrélation permet de retrouver des relations connues, nous avons démontré que GeEC est capable d'accomplir un travail de vérification de résultats.

Cependant, aucune démonstration fournie n'a montré que les matrices pouvaient servir à identifier de nouvelles relations. Certains des résultats présentaient indirectement ce potentiel (les hauts scores associés à des données issues des mêmes sites de séquençage), mais la question de la découverte n'a pas été approfondie. Il n'existe donc aucune formule qu'un usager intéressé par cette possibilité puisse réutiliser sur ses propres données, ce qui relègue pour le moment l'option de la découverte à des laboratoires qui ont accès à des chercheurs expérimentés en analyses numériques.

Il aurait été intéressant de faire cette démonstration, mais la décision a été prise de mettre le focus sur l'intégration de GeEC dans le portail du IHEC. Ceci dit, cette même intégration pourra être une aide précieuse quand viendra le moment d'utiliser GeEC dans un cadre de découvertes. L'étiquetage des données du portail IHEC, combiné avec les corrélations du GeEC devraient permettre d'identifier des régions d'intérêts dans de nouvelles données.

Voici un exemple hypothétique : une expérience utilisant des cellules cancéreuses du foie cible une certaine marque d'histone (ex : H3K4Me3). GeEC génère la corrélation entre cette expérience et toutes les autres données de H3K4me3 du portail. Cette matrice permet de constater que le score de corrélation de l'expérience et des autres données du foie est plus bas que la norme. Une analyse permet de constater que l'expérience corrèle mieux avec les données du poumon que du foie, permettant d'émettre l'hypothèse que l'expérience vient de cellule du poumon (ex : potentiel de métastase).

Je suis donc confiant que GeEC et les matrices générées pourront être utilisés comme aide directe à la recherche.

5.5 Direction de la bio-informatique et impact de mes travaux

Je suis de l'avis que la bio-informatique génomique se dirige dans une direction où il sera fréquent que l'analyse de données se fasse sans production de données à l'interne du laboratoire. La diffusion et l'accessibilité des expériences de séquençage vont permettre à des laboratoires d'exécuter des analyses précises sans nécessairement devoir générer de nouvelles données. Dans cette même ligne d'idée, les méta-analyses regroupant un grand nombre d'expériences de sources multiples vont se multiplier.

Ces méta-analyses vont nécessiter que les groupes de recherches aient accès à des ressources informatiques de plus en plus puissantes. Dans l'optique que les analyses bio-informatiques sont des tâches qui nécessitent beaucoup de ressources pour la phase de génération des résultats (GeEC peut prendre des centaines de CPU mais pour 2-3 jours) mais peu de ressources pour la longue phase d'analyse, il est probable que ces ressources seront partagées et gérées de manière distribuée entre les groupes de recherches, similairement au *cloud computing*.

Je considère que GeEC est parfaitement conçu pour s'intégrer dans ce nouveau paradigme. L'outil gère des calculs distribués et il a déjà démontré qu'il pouvait être utilisé sur de grandes quantités de données. De plus, il serait facile de l'intégrer avec plusieurs bases de données génomiques pour permettre, à un usager, d'analyser des données de plusieurs sources utilisant un nombre de métriques et de paramètres différents. Ces calculs pourraient être exécutés sur des ressources distribuées et ainsi minimiser la nécessité d'un chercheur d'avoir à gèrer ses propres ressources informatiques.

Par ailleurs, je suis d'avis que sa flexibilité lui permettra également d'intégrer de nouvelles métriques et d'être intégré dans de nouveaux environnements d'analyses distribuées. Je considère donc que GeEC est bien positionné pour pouvoir s'adapter aux besoins d'analyses futures du domaine.

5.6 Résumé des travaux

Cette section présente un bref résumé des travaux et des contributions de l'ensemble de la thèse. Elle est offerte pour le lecteur qui voudrait avoir un aperçu rapide de sa valeur scientifique.

Les contributions informatiques et techniques de la thèse se situent principalement au niveau du code et des outils rendus disponibles par le cadre de ces recherches. L’article présenté au chapitre 2 décrit une librairie qui permet à l’usager, d’une part de manipuler de manière accessible et transparente une multitude de formats génomiques, et, d’autre part, d’implanter rapidement des algorithmes utilisant ces données. Ce chapitre est accompagné d’une documentation en ligne complète et d’exemples. La librairie a, entre autres, été utilisée pour construire l’outil du chapitre 4.

Le chapitre 3 présente une classe C++ conçue afin de rendre disponible un outil de lecture de fichiers BigWig. Cette classe offre une interface simple d'utilisation basée sur le modèle des itérateurs Java et C++. Elle permet à un usager d'intégrer facilement la lecture de code BigWig dans leurs utilitaires. Sa structure moderne permet de facilement le modifier et l'améliorer.

Finalement, le chapitre 4 présente un ensemble d'outils nommé GeEC qui permet de générer et d'analyser des matrices de corrélation massives faites à partir de données d'expériences de séquençage haut débit. Cet outil est le premier du genre à permettre la génération de matrices avec plusieurs milliers d'expériences génomiques en employant le calcul distribué et la disponibilité de superordinateurs. Dans ce chapitre, nous démontrons qu'il est possible de générer des matrices à partir de milliers d'expériences de séquençage et que les outils d'analyse fournis avec GeEC facilitent l'extraction d'information de cette matrice de corrélation. GeEC est déjà utilisé dans une collaboration avec le portail de IHEC et devrait bientôt être disponible au grand public.

5.7 Conclusion

La possibilité de contribuer à la recherche bio-informatique liée à la génomique haut débit a été pour moi une expérience multidisciplinaire à la fois formatrice, épanouissante et occasionnellement frustrante. Étant issu du domaine de l'analyse de données et du développement plus classiques, la transition vers des travaux directement appliqués à la biologie a été un choc culturel important qui a nécessité l'apprentissage d'un nouveau « langage de recherche ». Au début de mes travaux, nous étions peu à franchir le mur entre les départements de biologie et d'informatique et, plusieurs fois, j'ai eu la chance de constater que les attentes et, parfois même, les termes utilisés en recherche varient fortement entre les domaines.

En contrepartie, cette même séparation m'a permis de développer une vision élargie et d'adopter des aspects des deux domaines. Apprendre à apprécier la rigueur et la minutie des travaux de recherche en biologie tout en gardant l'attitude d'abstraction et de réutilisation favorisées en informatique m'a permis d'améliorer mes talents de chercheur.

La recherche en bio-informatique est autant, sinon plus, active qu'au début de mes travaux. La disponibilité toujours croissante des données de séquençage ainsi que la nécessité de continuer à développer des outils adaptés offrent des opportunités de recherche uniques aux analystes et développeurs qui osent le saut de la multidisciplinarité. Je suis convaincu que l'apport d'idées de pointe du domaine de l'informatique ne fera qu'enrichir les découvertes en biologie.

Du coté de mes travaux, l'utilisation de GeEC dans le portail du IHEC démontre l'utilité de l'outil et la possibilité de collaborer avec des organismes de grande importance. C'est une des raisons pour lesquelles le laboratoire du Pr. Jacques a décidé de continuer activement le développement de l'outil. Spécifiquement, entre le moment où les tests ont été réalisés et la présente écriture, les modules GeEC-Prep et GeEC-Corr ont été réécrits en C++ et s'exécutent chacun près de 20 fois plus rapidement. Suite à cette optimisation, je suis convaincu que GeEC aura sa place dans ce portail de données publiques. C'est avec plaisir que j'observe la suite de certains de mes travaux et j'espère que cette base pourra servir à rendre accessible des types d'analyses qui étaient précédemment inatteignables pour les chercheurs.

BIBLIOGRAPHIE

Andrew, S. (2010). FastQC: a quality control tool for high throughput sequence data. http://www.bioinformatics.babraham.ac.uk/projects/fastqc.

Annunziato, A. T. (2008). DNA Packaging: Nucleosomes and Chromatin. Nature Education *1*, 26.

Barski, A., Cuddapah, S., Cui, K., Roh, T.-Y., Schones, D. E., Wang, Schones, D.E., Wang, Z., Wei, G., Chepelev, I., Zhao, K. (2007). High-resolution profiling of histone methylations in the human genome. Cell *129*(4), 823–37.

Belton, J.-M., McCord, R. P., Gibcus, J. H., Naumova, N., Zhan, Y., & Dekker, J. (2012). Hi-C: a comprehensive technique to capture the conformation of genomes. Methods (San Diego, Calif.) *58*(3), 268–76.

Benjamini, Y., & Speed, T. P. (2012). Summarizing and correcting the GC content bias in high-throughput sequencing. Nucleic Acids Research *40*(10), 1-14.

Brunelle, M., Nordell Markovits, A., Rodrigue, S., Lupien, M., Jacques, P.-É., & Gévry, N. (2015). The histone variant H2A.Z is an important regulator of enhancer activity. Nucleic Acids Research *43*(20), 9742–56.

Cairns, J., Spyrou, C., Stark, R., Smith, M. L., Lynch, A. G., & Tavaré, S. (2011). BayesPeak - An R package for analysing ChIP-seq data. Bioinformatics *27*(5), 713–714.

Chen, K., Xi, Y., Pan, X., Li, Z., Kaestner, K., Tyler, J., Dent, S., He, X., Li, W. (2013). DANPOS: dynamic analysis of nucleosome position and occupancy by sequencing. Genome Research *23*(2), 341–51.

Chilana, P. K., Palmer, C. L., & Ko, A. J. (2009). Comparing bioinformatics software development by computer scientists and biologists: An exploratory study. ICSE Workshop on Software Engineering for Computational Science and Engineering, 72–79.

Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L., & Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Research *38*(6), 1767–71.

Coulombe, C., Poitras, C., Nordell-Markovits, A., Brunelle, M., Lavoie, M.-A., Robert, F., & Jacques, P.-É. (2014). VAP: a versatile aggregate profiler for efficient genome-wide data representation and discovery. Nucleic Acids Research *42*, W485-93.

Cuddapah, S., Jothi, R., Schones, D. E., Roh, T.-Y., Cui, K., & Zhao, K. (2009). Global analysis of the insulator binding protein CTCF in chromatin barrier regions reveals demarcation of active and repressive domains. Genome Research *19*(1), 24–32.

Dabney, J., & Meyer, M. (2012). Length and GC-biases during sequencing library amplification: a comparison of various polymerase-buffer systems with ancient and modern DNA sequencing libraries. BioTechniques *52*(2), 87–94.

Diaz, A., Park, K., Lim, D. A., & Song, J. S. (2012). Normalization, bias correction, and peak calling for ChIP-seq. Statistical Applications in Genetics and Molecular Biology *11*(3), Article 9.

Dvir, A., Conaway, J. W., & Conaway, R. C. (2001). Mechanism of transcription initiation and promoter escape by RNA polymerase II. Current Opinion in Genetics & Development *11*(2), 209–14.

Eberwine, J., Sul, J.-Y., Bartfai, T., & Kim, J. (2014). The promise of single-cell sequencing. Nature Methods *11*(1), 25–7.

ENCODE Project Consortium. (2012). An integrated encyclopedia of DNA elements in the human genome. Nature *489*(7414), 57–74.

Ernst, J., & Kellis, M. (2012). ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods*, *9*(3), 215–6.

Farnham, P. J. (2009). Insights from genomic profiling of transcription factors. Nature Reviews Genetics *10*(9), 605–16.

Fejes, A. P., Robertson, G., Bilenky, M., Varhol, R., Bainbridge, M., & Jones, S. J. M. (2008). FindPeaks 3.1: a tool for identifying areas of enrichment from massively parallel short-read sequencing technology. Bioinformatics *24*(15), 1729–30.

Ferraiuolo, M. A., Sanyal, A., Naumova, N., Dekker, J., & Dostie, J. (2012). From cells to chromatin: capturing snapshots of genome organization with 5C technology. Methods *58*(3), 255–67.

Flores, O., & Orozco, M. (2011). nucleR: a package for non-parametric nucleosome positioning. Bioinformatics *15*(27), 2149-2150.

Fullwood, M. J., Liu, M. H., Pan, Y. F., Liu, J., Xu, Bin Mohamed, Y., Orlov, Y. L., Velkov, S., Ho, A., Mei, P. H., *et al*. (2009). An oestrogen-receptor-alpha-bound human chromatin interactome. Nature *462*(7269), 58–64.

Gaffney, D. J., McVicker, G., Pai, A. A., Fondufe-Mittendorf, Y. N., Lewellen, N., Michelini, K., Widom, J., Gilad, Y., Pritchard, J. K. (2012). Controls of nucleosome positioning in the human genome. PLoS Genetics *8*(11), e1003036.

Gévry, N., Hardy, S., Jacques, P.-É., Laflamme, L., Svotelis, A., Robert, F., & Gaudreau, L. (2009). Histone H2A.Z is essential for estrogen receptor signaling. Genes & Development *23*(13), 1522–1533.

Gilmour, D. S., & Lis, J. T. (1985). In vivo interactions of RNA polymerase II with genes of Drosophila melanogaster. Molecular and Cellular Biology *5*(8), 2009–18.

Goecks, J., Nekrutenko, A., Taylor, J., & Team, T. G. (2016). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biology *11*(8), R86.

Goodwin, S., McPherson, J. D., & McCombie, W. R. (2016). Coming of age: ten years of next-generation sequencing technologies. Nature Reviews Genetics *17*(6), 333–51.

Greenfield, P., Duesing, K., Papanicolaou, A., & Bauer, D. C. (2014). Blue: correcting sequencing errors using consensus and context. Bioinformatics *30*(19), 2723–32.

Guttman, A. (1984). R-trees. ACM SIGMOD Record *14*(2), 47.

Hagen, J. B. (2000). The origins of bioinformatics. Nature Reviews Genetics *1*(3), 231–6.

Hake, S. B., & Allis, C. D. (2006). Histone H3 variants and their potential role in indexing mammalian genomes: the "H3 barcode hypothesis". Proceedings of the National Academy of Sciences of the United States of America *103*(17), 6428–35.

Hallam, S. (2013). *A Data-Driven History of Bioinformatics*. University of Chicago Press.

Head, S. R., Komori, H. K., LaMere, S. A., Whisenant, T., Van Nieuwerburgh, F., Salomon, D. R., & Ordoukhanian, P. (2014). Library construction for next-generation sequencing: overviews and challenges. BioTechniques *56*(2), 61–4.

Hsi-Yang Fritz, M., Leinonen, R., Cochrane, G., & Birney, E. (2011). Efficient storage of high throughput DNA sequencing data using reference-based compression. Genome Research *21*(5), 734–40.

Iyer, V. R. (2012). Nucleosome positioning: bringing order to the eukaryotic genome. Trends in Cell Biology *22*(5), 250–6.

Jiang, C., & Pugh, B. F. (2009). Nucleosome positioning and gene regulation: advances through genomics. Nature Reviews Genetics *10*(3), 161–72.

Kelley, D. R., Schatz, M. C., & Salzberg, S. L. (2010). Quake: quality-aware detection and correction of sequencing errors. Genome Biology *11*(11), R116.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., Pringle, T. H., Zahler, A. M., & Haussler, D. (2002). The human genome browser at UCSC. Genome Research *12*(6), 996–1006.

Kent, W. J., Zweig, a. S., Barber, G., Hinrichs, a. S., & Karolchik, D. (2010). BigWig and BigBed: Enabling browsing of large distributed datasets. Bioinformatics *26*(17), 2204–2207.

Kharchenko, P. V, Tolstorukov, M. Y., & Park, P. J. (2008). Design and analysis of ChIP-seq experiments for DNA-binding proteins. Nature Biotechnology *26*, 1351–1359.

Knierim, E., Lucke, B., Schwarz, J. M., Schuelke, M., & Seelow, D. (2011). Systematic comparison of three methods for fragmentation of long-range PCR products for next generation sequencing. PloS One *6*(11), e28240.

Kornberg, R. D., & Thomas, J. O. (1974). Chromatin structure; oligomers of the histones. Science *184*(4139), 865–8.

Landt, S. G., Marinov, G. K., Kundaje, A., Kheradpour, P., Pauli, F., Batzoglou, S., Bernstein, B. E., Bickel, P., Brown, J. B., Cayting, P., *et al*. (2012). ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. Genome Research *22*(9), 1813–31.

Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology *10*(3), R25.

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics *25*(14), 1754–60.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R. ; 1000 Genome Project Data Processing Subgroup. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics *25*(16), 2078–9.

Li, H., & Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. Briefings in Bioinformatics *11*(5), 473–83.

Li, H., Ruan, J., & Durbin, R. (2008). Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Research *18*(11), 1851–58.

Li, R., Yu, C., Li, Y., Lam, T.-W., Yiu, S.-M., Kristiansen, K., & Wang, J. (2009). SOAP2: an improved ultrafast tool for short read alignment. Bioinformatics *25*(15), 1966–7.

Mardis, E. R. (2010). The $1,000 genome, the $100,000 analysis? Genome Medicine *2*(11), 84.

Meneghini, M. D., Wu, M., & Madhani, H. D. (2003). Conserved histone variant H2A.Z protects euchromatin from the ectopic spread of silent heterochromatin. Cell *112*(5), 725–36.

Meyer, C. A., & Liu, X. S. (2014). Identifying and mitigating bias in next-generation sequencing methods for chromatin biology. Nature Reviews Genetics *15*(11), 709–21.

Mueller-Planitz, F., Klinker, H., & Becker, P. B. (2013). Nucleosome sliding mechanisms: new twists in a looped history. Nature Structural & Molecular Biology *20*(9), 1026–32.

Nelson, M. (1995). *C++ Programmers Guide to Standard Template Library*. John Wiley & Sons.

Nordell Markovits, A., Joly Beauparlant, C., Toupin, D., Wang, S., Droit, A., & Gevry, N. (2013). NGS++: a library for rapid prototyping of epigenomics software tools. Bioinformatics *29*(15), 1893–4.

Ozsolak, F., & Milos, P. M. (2011). RNA sequencing: advances, challenges and opportunities. Nature Reviews Genetics *12*(2), 87–98.

Papadias, D., & Theodoridis, Y. (1997). Spatial relations, minimum bounding rectangles, and spatial data structures. International Journal of Geographical Information Science *11*(2), 111–38.

Park, P. J. (2009). ChIP-seq: advantages and challenges of a maturing technology. Nature Reviews Genetics *10*(10), 669–80.

Pepke, S., Wold, B., & Mortazavi, A. (2009). Computation for ChIP-seq and RNA-seq studies. Nature Methods *6*(11), S22–S32.

Peterson, W. W. (1957). Addressing for random-access storage. IBM Journal of Research and Development *1*, 130–146.

Quinlan, A. R., & Hall, I. M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics *26*(6), 841–2.

Rhee, H. S., & Pugh, B. F. (2011). Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. Cell *147*(6), 1408–19.

Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., & Mesirov, J. P. (2011). Integrative genomics viewer. Nature Biotechnology *29*(1), 24–6.

Ross, M. G., Russ, C., Costello, M., Hollinger, A., Lennon, N. J., Hegarty, R., Nusbaum, C., & Jaffe, D. B. (2013). Characterizing and measuring bias in sequence data. Genome Biology *14*(5), R51.

Sanger, F., Air, G. M., Barrell, B. G., Brown, N. L., Coulson, A. R., Fiddes, C. A., Hutchison III, Slocombe, P. M., & Smith, M. (1977). Nucleotide sequence of bacteriophage phi X174 DNA. Nature *265*(5596), 687–95.

Schirmer, M., Ijaz, U. Z., D'Amore, R., Hall, N., Sloan, W. T., & Quince, C. (2015). Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. Nucleic Acids Research *43*(6), e37–e37.

Schling, B. (2011). *The Boost C++ libraries*. XML press.

Schmidt, D., Wilson, M. D., Spyrou, C., Brown, G. D., Hadfield, J., & Odom, D. T. (2009). ChIP-seq: using high-throughput sequencing to discover protein-DNA interactions. Methods *48*(3), 240–8.

Shendure, J., & Ji, H. (2008). Next-generation DNA sequencing. Nature Biotechnology *26*(10), 1135–45.

Shendure, J., & Lieberman Aiden, E. (2012). The expanding scope of DNA sequencing. Nature Biotechnology *30*(11), 1084–94.

Sims, D., Sudbery, I., Ilott, N. E., Heger, A., & Ponting, C. P. (2014). Sequencing depth and coverage: key considerations in genomic analyses. Nature Reviews Genetics, *15*(2), 121–32.

Struhl, K., & Segal, E. (2013). Determinants of nucleosome positioning. Nature Structural & Molecular Biology *20*(3), 267–273.

Talbert, P. B., & Henikoff, S. (2006). Spreading of silent chromatin: inaction at a distance. Nature Reviews Genetics *7*(10), 793–803.

Taub, M. A., Corrada Bravo, H., & Irizarry, R. A. (2010). Overcoming bias and systematic errors in next generation sequencing data. Genome Medicine *2*(12), 87.

Taylor, J. (n.d.). Tools for manipulating biological data, particularly multiple sequence alignments. https://bitbucket.org/james_taylor/bx-python/overview

Tyler, A. D., Christianson, S., Knox, N. C., Mabon, P., Wolfe, J., Van Domselaar, G., Graham, M. R., Sharma, M. K. (2016). Comparison of Sample Preparation Methods Used for the Next-Generation Sequencing of Mycobacterium tuberculosis. PloS One *11*(2), e0148676.

Vakoc, C. R., Mandat, S. A., Olenchock, B. A., & Blobel, G. A. (2005). Histone H3 lysine 9 methylation and HP1gamma are associated with transcription elongation through mammalian chromatin. Molecular Cell *19*(3), 381–91.

Vignali, M., Hassan, A. H., Neely, K. E., & Workman, J. L. (2000). ATP-dependent chromatin-remodeling complexes. Molecular and Cellular Biology *20*(6), 1899–910.

Watson, J. (1990). The human genome project: past, present, and future. Science *248*(4951), 44–49.

Ye, T., Krebs, A. R., Choukrallah, M.-A., Keime, C., Plewniak, F., Davidson, I., & Tora, L. (2011). seqMINER: an integrated ChIP-seq data interpretation platform. Nucleic Acids Research *39*(6), e35.

Yeung, K. Y., & Ruzzo, W. L. (2001). Principal component analysis for clustering gene expression data. Bioinformatics *17*(9), 763–74.

Zhang, Y., Liu, T., Meyer, C. a, Eeckhoute, J., Johnson, D. S., Bernstein, B. E., Nusbaum, C., Myers, R. M., Brown, M., Li, W., & Liu, S. (2008). Model-based analysis of ChIP-Seq (MACS). Genome Biology *9*(9), R137.

Zheng, W., Chung, L. M., & Zhao, H. (2011). Bias detection and correction in RNA-Sequencing data. BMC Bioinformatics *12*, 290.

Zlatanova, J., & Thakar, A. (2008). H2A.Z: view from the top. Structure *16*(2), 166–79.