Développement de méthodes et d'algorithmes pour la caractérisation et l'annotation des transcriptomes avec les séquenceurs haut débit

Nicolas Philippe

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ **M**ONTPELLIER **II**
Sciences et **T**echniques du **L**anguedoc

THÈSE

présentée au Laboratoire d'Informatique de Robotique
et de Microélectronique de Montpellier pour
obtenir le diplôme de doctorat

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Information, Structures, Systèmes**

Développement de méthodes et d'algorithmes pour la caractérisation et l'annotation des transcriptomes avec les séquenceurs haut débit

par

Nicolas PHILIPPE

Soutenue le 29 septembre 2011, devant le jury composé de :

Directeur de thèse
M. Éric RIVALS, directeur de recherche ..... LIRMM, Université Montpellier II, France

Codirectrice de thèse
Mme. Thérèse COMMES, professeur ..... CRBM, Université de Montpellier II, France

Rapporteurs
M. Roderic GUIGO, directeur de Recherche ..... CRG, Barcelone, Espagne
M. Gregory KUCHEROV, directeur de Recherche ..... LIGM, Paris, France

Président du jury
M. Jamal TAZI, professeur ..... IGMM, Montpellier, France

Examinateurs
M. Didier AUBOEUF, chargé de Recherche ..... INSERM, Lyon, France
M. Dominique LAVENIER, directeur de Recherche ..... IRISA, Rennes, France

Remerciements

J'exprime tout d'abord mes profonds remerciements à mon directeur et ma directrice de thèse, Eric Rivals et Thérèse Commes. D'une part, Eric m'a apporté la rigueur, la justesse et les démarches méthodiques pour concevoir et développer le mieux possible une idée, il m'a appris à ressortir les éléments fondamentaux d'un travail lors de l'écriture d'une publication afin de transmettre un message scientifique clair et précis. D'autre part, Thérèse m'a apporté tous les fondamentaux de la génomique ce qui m'a permis de mieux comprendre les problématiques biologiques, elle a toujours été disponible pour moi que ce soit le soir, le week-end et même pendant ses vacances, mais surtout, elle m'a souvent éclairé sur les problèmes biologiques reliant des besoins en informatique et en mathématiques, autour de réflexion et de discussion, soulevant des points cruciaux lors de mes recherches. De façon générale, tous deux m'ont fait confiance. Ils m'ont laissé une grande liberté lors de mes créations et ils m'ont encadré d'une façon exemplaire.

Il est naturel que je remercie toute mon équipe de travail avec notamment Anthony Boureux, Elias Bousamra, Mireille Galloni et Florence Ruffle pour les conseils et les travaux que nous avons pu partager pendant mes années de thèses.

Je remercie très cordialement mon jury de thèse. D'une part mes deux rapporteurs Roderic Guigo et Gregory Kucherov qui sont deux scientifiques de renommée internationale. Ils ont pris de leur temps précieux pour lire méticuleusement mon manuscrit et juger de manière constructive mon travail. D'autre part, mes trois examinateurs Didier Auboeuf, Dominique Lavenier et Jamal Tazi qui ont accompagné mes rapporteurs lors de ma soutenance. Grâce à leur expérience et leurs connaissances pointues du domaine de la bioinformatique, ils ont tous les cinq fait vivre un débat productif avec des questions pertinentes qui m'ont fait réfléchir et ont souvent menées à des perspectives intéressantes.

Je vais maintenant remercier différentes personnes qui, chacune à leur façon, ont pu un moment donné m'apporter de l'aide tout au long de mon doctorat.

En premier lieu ma famille, tout particulièrement Bruno et Marie-José Philippe, mes parents, qui m'ont permis d'entreprendre des études supérieures dans les meilleures conditions. Je leur en suis reconnaissant pour tous les sacrifices qu'ils ont dû accomplir afin de subvenir à mes besoins. Mais en plus de cela, ils ont avec Thomas Philippe, mon frère, Camille Godron, ma conjointe, Bernard et Marie-Christiane Godron, mes beaux-parents, Romain Godron, mon beau-frère, ainsi que autres grands-mères, grands-pères, oncles, tantes, cousins et cousines, sans oublier Mauka et Zéphyr mes deux petits chats, su m'apporter de l'enthousiasme, de la confiance, de l'amour et de l'attention avec une générosité sans compter. Je suis fier et chanceux d'avoir une si belle famille. D'ailleurs, je suis heureux de pouvoir remercier Camille, Bernard et Marie-Christiane Godron pour tout le temps qu'ils ont pu passer à relire ma thèse afin d'améliorer certaines tournures de phrases, mais aussi pour chasser le maximum de fautes d'orthographe qui ont pu s'échapper de « mon clavier ». Je rajoute évidemment un bonus supplémentaire à Camille qui a su m'épauler tout au long de mon, ou plutôt notre, dur labeur, notamment en me conseillant et en me rassurant lors des moments les plus délicats et en me pardonnant de ses moments de solitude lors des mes soirées tardives de travail.
Puis, je remercie mes parents, Camille et mes beaux-parents pour s'être occupés avec le plus grand soin des préparatifs de ma soutenance : à la clé du bon champagne et des petites mignardises confectionnées par amour.

Quant à Mikaël Salson, ami de longue date et précieux collaborateur, je ne peux qu'être enthousiaste à l'idée de travailler avec lui. Nous avons déjà accompli plusieurs projets ensemble depuis nos années universitaires. J'ai eu la chance de pouvoir collaborer avec cet homme admirable durant ma thèse, notamment sur le développement des logiciels *Gk arrays* et CRAC. Je le remercie aussi pour les aides supplémentaires qu'il m'a apportées en tikZ. D'ailleurs, j'avoue que c'est entièrement sa faute si j'ai passé du temps à me perfectionner lors de la conception de mes présentations orales (pourtant je n'aurais pas pensé qu'il était aussi fin graphiste, il y a de ça quelques années...).

Dans cette catégorie des personnes soucieuses de bien faire les choses, je remercie Alban Mancheron et Floréal Morandat pour leur performance en informatique. Leur côté « geek » m'a souvent rendu service dans diverses situations, entre autres lors du packaging de mes logiciels (Alban) ou encore la conception d'un style de thèse en LaTeX ultra fonctionnel (Floréal). Enfin, je remercie Thierry Lecroq et Martine Léonard qui font partie de

mes professeurs préférés de l'université de Rouen et avec qui j'ai pu collaborer durant ma thèse.

D'un point de vue personnel, je souhaite remercier tous mes amis qui ont suivi mon cheminement, tout en ayant partagé avec moi des moments de détente et de bonheur. Pour n'en citer que quelques-uns avec qui j'ai passé mes dernières vacances (par ordre alphabétique des noms de famille) : Emmanuel Bénard, Guillaume Buwalda, Elsa Dollé, Camille Godron, Jean-Daniel Lomenede, Laëtitia Plisson, Camille Serrecourt, Élodie Serurier-Duceau, Mikaël Salson, Nicolas Toublanc et Stéphanie Verbreugh.

Je tiens aussi à remercier mes colocataires Thomas Bailly, Raphaël Chartier et Laurent Vertu pour leurs différentes contributions lors de la phase finale de mon doctorat, comme m'avoir écouté lors des sessions blanches de ma soutenance. J'insiste tout particulièrement sur Thomas qui, de son plein gré, a su se montrer serviable en s'occupant des différents invités présents à mon pot de thèse, en compagnie de ma famille.

Pour finir, je me permets de faire un petit clin d'œil à Roger Federer et au FC-Barcelone de Lionel Messi qui m'ont permis de me ressourcer lors de mes journées et soirées intensives de réflexion et d'écriture.

Pour toutes les personnes qui m'auraient apporté de l'aide et qui sont déçues de ne pas être remerciées, j'en suis par avance sincèrement désolé. Cependant, sachez par avance qu'il n'y a personne que je ne souhaite pas remercier.

Table des matières

Introduction

La recherche dans le domaine des sciences de la vie a pâti durant quelques années d'une stagnation des techniques de séquençage. On assiste depuis peu à l'émergence de nouvelles technologies pour analyser les séquences d'ADN : les séquenceurs haut débit (SHD). Par rapport aux méthodes classiques de séquençage, il s'agit d'une augmentation d'au moins un facteur 1 000 en capacité qui s'accompagne d'une importante simplification des méthodes [von Bubnoff, 2008]. Les SHD offrent la possibilité de générer des millions, voire des milliards de séquences, appelées *reads*, avec une profondeur sans précédent. Depuis l'arrivée de ces nouveaux systèmes (Roche 454®, Illumina® et SOLiD) [Shendure et Ji, 2008], le séquençage des génomes a fait un bond en avant [Via *et al.*, 2010]. Ils ont notamment révolutionné l'étude des transcriptomes à l'échelle du génome.

Des techniques transcriptomiques, telles que la *Digital Gene Expression* (DGE) [Morrissy *et al.*, 2009] et le *RNA-Sequencing* (RNA-Seq) [Sultan *et al.*, 2008; Wang *et al.*, 2009], permettent aujourd'hui d'explorer la globalité des transcriptomes. Ces techniques ouvertes de la transcription ont, sous l'impulsion du projet ENCODE, conduit à un changement de paradigme dans les stratégies d'étude et d'annotation des transcriptomes et des génomes [The ENCODE Project Consortium, 2007; Rozowsky *et al.*, 2007].

Une analyse transcriptomique, en combinant RNA-Seq et DGE, permet de répertorier, de quantifier, voire reconstruire tous les transcrits d'une cellule, même les plus rares. Parmi ce type de transcrits se trouvent des ARN non-codants régulateurs [Mercer *et al.*, 2009; Derrien et Guigó, 2011] ; des variants d'épissage créateurs de protéines [Dutertre *et al.*, 2010; Sammeth *et al.*, 2008] ; et aussi des recombinaisons de gènes par translocation chromosomique ou trans-épissage, appelées chimères [Li *et al.*, 2008d; Maher *et al.*, 2009c;

Edgren *et al.*, 2011; Stephens *et al.*, 2011; Nacu *et al.*, 2011]. La détection de ces événements biologiques représente un enjeu majeur de la biologie actuelle.

Mais les générations de SHD s'enchaînent, produisant des séquences de plus en plus longues et volumineuses [Pareek *et al.*, 2011], permettant de répondre à encore plus de questions biologiques. Au-delà de l'enthousiasme soulevé par ces réalisations technologiques impressionnantes, cette production toujours plus importante de données à analyser et à interpréter nécessite un accroissement des besoins algorithmiques et de stockage, et constitue de nouveaux défis pour les bioinformaticiens [Trapnell et Salzberg, 2009]. En effet, l'analyse de ces milliards de *reads* nécessite des besoins méthodologiques que seule l'informatique est en mesure de satisfaire de façon précise et rapide. Il faut développer des méthodes relevant de la statistique et de l'algorithmique, élaborer des nouvelles et originales stratégies d'analyse, ainsi que mettre à disposition des moyens informatiques (stockage et calcul) très importants.

En bioinformatique, l'ADN est perçu comme un texte où l'alphabet est composé de quatre lettres : A (pour adénine), C (pour cytosine), G (pour guanine) et T (pour thymine). L'algorithmique du texte est une branche de l'informatique qui permet, notamment, d'aligner ces mots d'ADN (ou séquences) entre eux ou sur une séquence plus longue, mais les algorithmes classiques ne permettent pas de manipuler des milliards de séquences. Pourtant, avec des *reads* issus de SHD et en présence d'un génome de référence, aligner *reads* et génome constitue une stratégie possible et intéressante. Nous voulons localiser et analyser ces *reads* afin d'identifier et de caractériser l'ensemble des transcrits présents dans un tissu cellulaire.

L'enjeu algorithmique est de taille car de nombreux points sont à prendre en considération : Comment élaborer des structures informatiques permettant le passage à l'échelle de cette masse de *reads* ? Comment distinguer les erreurs des événements biologiques ? Comment détecter les phénomènes biologiques qui sont faiblement exprimés dans un transcriptome ? De plus, rappelons que les *reads* sont assimilés à de courtes séquences par rapport à un génome ; alors comment les positionner sur ce génome avec le moins d'ambiguïté possible ? Tant de questions complexes à analyser en même temps !

L'objectif de ma thèse est de proposer des algorithmes et des méthodes permettant d'analyser l'ensemble des informations dans le cadre de la transcriptomique.

Première partie

Cette partie constitue l'état de l'art de ma thèse. J'y introduis les définitions, les notations et les concepts nécessaires à la compréhension de la suite du travail.

Chapitre 1. Il s'agit dans ce chapitre d'établir le contexte scientifique de mes recherches. La génomique étant un domaine ample, je définis les problèmes qui nous intéressent. Je me concentre notamment sur la complexité des analyses transcriptomiques et leur évolution. La naissance des premières analyses ouvertes du transcriptome basées sur le séquençage a engendré des besoins en informatique dans le traitement des séquences d'ADN. Le vocabulaire informatique relatif à l'étude des séquences génomiques est aussi défini dans ce chapitre.

Chapitre 2. Tout ce chapitre est consacré aux séquenceurs haut débit (SHD). Je commence par retracer leur parcours et leur impact sur l'annotation des transcriptomes. Avec une profondeur de séquençage sans précédent, ils ouvrent la voie à l'identification de nouveaux mécanismes biologiques , à un séquençage systématique des génomes à prix réduit et à l'identification des mutations entre espèces ou à plusieurs stades d'un développement cellulaire. Néanmoins, ils produisent une quantité astronomique de séquences (ou *reads*) et des erreurs. Une étude biologique sur ces *reads* passe par leur traitement avec des méthodes et des algorithmes informatiques. Je détaille, pour chaque question biologique, les algorithmes, les points forts et les faiblesses des outils d'analyse actuels pour le traitement des *reads*. J'insiste particulièrement sur la difficulté de concevoir des algorithmes capables de détecter des événements complexes tout en étant capable de gérer des milliards de *reads*.

Seconde partie

Cette deuxième partie concerne mes travaux de recherches. J'ai répondu avec l'aide de mes directeurs de thèse et d'autres collaborateurs à plusieurs questions posées lors du traitement des transcriptomes par SHD, autour de trois chapitres.

Chapitre 3. Dans ce chapitre, une étude bioinformatique et statistique sur la DGE est proposée afin d'évaluer l'impact combiné de la longueur des *reads*, de l'attente aléatoire du *mapping* et des erreurs de séquences lors de l'analyse des *reads*. Nous montrons dans cette analyse que le *mapping* exact suffit pour localiser des séquences sur un génome tel que l'humain ou la souris. À partir, de cette analyse j'ai développé un pipeline d'annota-

tion pour la DGE. Ce pipeline permet non seulement de faire des études globales sur la répartition des transcrits mais également facilite les études biologiques orientées vers la recherche de nouveaux transcrits spécifiques de tissus.

Le travail sur ce pipeline est en cours de rédaction pour une publication dans une revue internationale.

Il comprend une stratégie méthodologique permettant de diminuer au maximum l'influence des erreurs de séquences sur l'annotation, tout en conservant le maximum de transcrits (même les plus rares). Ce travail a fait l'objet d'une publication dans Nucleic Acids Research en 2009 [Philippe *et al.*, 2009].

Chapitre 4. À partir de ce premier travail, j'ai cherché à aller plus loin dans l'amélioration des procédures d'analyse des données de transcriptomes. Un des aspects que nous avons cherché à exploiter est le fait qu' au delà d'une simple information de *mapping*, un read issu d'une expérience de RNA-Seq intègre d'autres informations partagées entre les *reads*. J'ai ainsi pu montrer le besoin d'effectuer des requêtes rapides sur les *reads*. Cela nous a amené à concevoir, moi et quelques collaborateurs, la structure d'indexation baptisée *Gk arrays* qui permet d'organiser ces millions de *reads*. La difficulté réside ici dans le volume gigantesque de données à traiter. En effet, d'un côté des structures d'indexation classiques telles que les arbres ou les tables de suffixes sont trop gourmandes en mémoire pour ce passage à l'échelle. D'un autre côté, les structures d'indexation compressées sont trop lentes à construire et à interroger. Les *Gk arrays* offrent un bon compromis entre mémoire et temps de construction. Ils peuvent être interrogés très rapidement à l'aide de requêtes, par exemple pour récupérer des informations cruciales partagées entre les *reads*. Ce travail a fait l'objet d'une publication dans BMC Bioinformatics [Philippe *et al.*, 2011].

Chapitre 5. Enfin, en s'appuyant sur les *Gk arrays*, j'ai développé, avec l'aide de collaborateurs, le logiciel CRAC spécialisé dans le traitement du RNA-Seq. C'est un programme d'analyse de *reads* qui intègre sa propre phase de *mapping* à la détection de causes biologiques telles que les SNV, les indels ou les jonctions d'épissage. CRAC se base sur une double indexation i/ des *reads* à l'aide des *Gk arrays* et ii/ du génome à l'aide d'un index compressé de type FM-index. Nous avons aussi développé un pipeline de simulation, en y intégrant notamment *FluxSimulator* (Sammeth, unpublished software) pour simuler du mieux possible une expérience de RNA-Seq intégrant des mutations telles que les SNV, indels, et chimères. À l'aide de cette simulation, nous comparons CRAC sur la qualité et la correction du *mapping* avec les outils (Bowtie [Langmead *et al.*, 2009], BWA [Li et Durbin, 2009], SOAP2 [Li *et al.*, 2009b], GSNAP [Wu et Nacu, 2010], GASSST [Rizk et Lavenier, 2010],

BWA-SW [Li et Durbin, 2010]). Nous le comparons également sur les outils d'analyse capable de détecter les jonctions d'épissages (TopHat [Trapnell *et al.*, 2009], MapSplice [Wang *et al.*, 2010], GSNAP [Wu et Nacu, 2010]).

Ainsi, en alliant le *mapping* aux informations contenues dans les *reads*, nous offrons un algorithme beaucoup plus précis, voire plus sensible, que les outils d'analyse actuels. CRAC permet notamment de distinguer les erreurs de séquences des causes biologiques, mais aussi de détecter précisément les jonctions d'épissage et le point de cassure des chimères.

Rappelons que la détection de chimères sur une expérience de RNA-Seq suscite l'élaboration d'outils précis car elles sont souvent très faiblement exprimées dans un transcriptome et sont par nature complexes à détecter avec des parties localisées à différents endroits sur le génome. Ces nouveaux transcrits suscitent aussi un réel défi biologique car ils peuvent être impliqués dans de nombreux processus cellulaires physiologiques et pathologiques et sont fréquemment décrits dans les cancers. Nous examinons les prédictions de chimères de CRAC pour une banque de cellules cancereuses de K562 (www.sanger.ac.uk/PostGenomics/encode/RGASP.html) et une banque de cellules normales ERR030856 (http://trace.ddbj.nig.ac.jp/DRASearch/experiment?acc=ERX011226). Ce travail est en cours de rédaction pour une publication dans une revue internationale.

Conclusion et discussions. Je finis par une brève conclusion résumant mes principales contributions et publications. J'ouvrirai sur des perspectives et mes projets à court terme.

Première partie

État de l’art

CHAPITRE

1 Contexte scientifique

Il s'agit dans ce premier chapitre de situer le contexte scientifique de ce manuscrit de thèse. Nous y introduisons les notions clés essentielles à la compréhension des différents thèmes abordés dans la suite du travail. Dans un premier temps, nous effectuons un résumé sommaire de la génomique chez les eucaryotes. Dans un second temps, nous détaillons davantage certains aspects de la transcriptomique. Puis dans une dernière partie, nous définissons le vocabulaire de la bioinformatique directement lié au traitement des séquences.

Sommaire

1.1 Introduction à la génomique

La *génomique* est l'étude des *génomes*. Chez les eucaryotes, le *génome* est l'ensemble du matériel génétique d'un individu ou d'une espèce codé dans son *ADN*. Tout ce matériel génétique permet, entre autres, la synthèse des *protéines* ou la régulation des *gènes* via tous les *ARN*.

Tous ces événements ont lieu au sein de chaque cellule, lui permettant ainsi de vivre et de se renouveler. Mais comment ADN et ARN sont-ils reliés ? Comment fonctionne la synthèse des protéines ? Qu'est-ce qu'un gène ?

Nous nous sommes tous posés ce type de questions lors de nos premiers pas en génomique, alors il semble important de fixer ces quelques axiomes avant de rentrer davantage dans le cœur du problème.

1.1.1 L'ADN

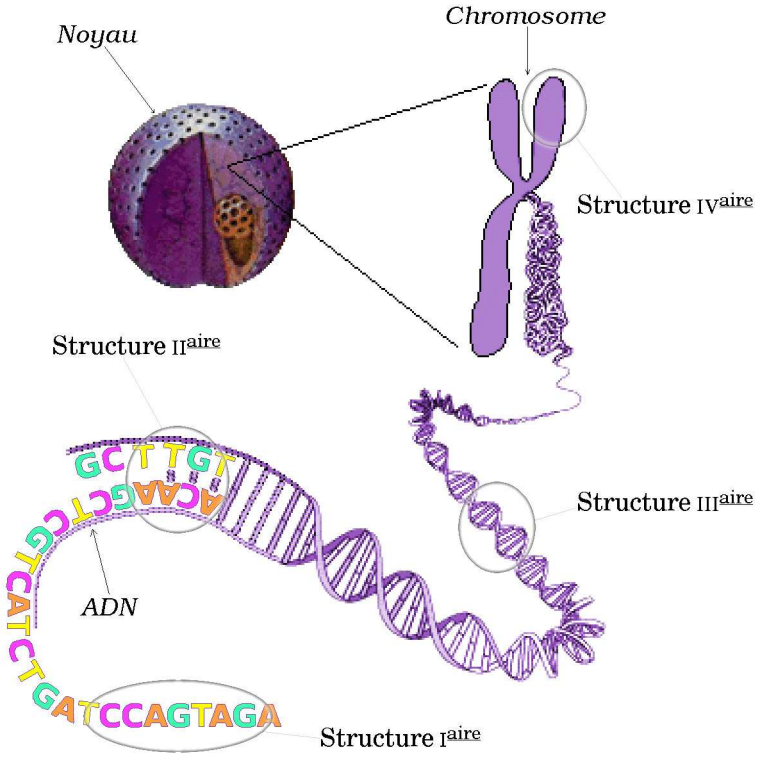
Depuis le milieu du XX^{e} siècle, nous savons que le vecteur de l'information génétique est l'acide désoxyribonucléique, plus connu sous son acronyme : ADN. Ce dernier est une *macromolécule*, c'est-à-dire une molécule composée de plusieurs molécules. Les constituants de base de l'ADN sont les *nucléotides*. Ils sont au nombre de quatre et se distinguent en deux catégories selon la *base azotée* (ou *base*) qui les compose : d'un côté les purines avec l'*Adénine* et la *guanine*, de l'autre les pyrimidines avec la *cytosine* et la *Thymine*. Les noyaux purines et pyrimidines sont ainsi complémentaires grâce à leurs propriétés physico-chimiques respectives permettant leur appariement par liaisons hydrogène, c'est-à-dire que la thymine est associée à l'adénine et la cytosine à la guanine. On parle de *liaisons faibles* permettant la construction de la structure tertiaire, c'est-à-dire la double hélice de l'ADN. Cette dernière est constituée de deux *brins* d'ADN complémentaires enlacés en spirale.

Si les *liaisons faibles* sont responsables de l'appariement des bases, les *liaisons fortes* sont responsables de l'agencement des bases sous forme de squelette. Celui-ci est semblable à un fil, il n'est pas ramifié, et permet donc de définir les structures primaire et secondaire de l'ADN. Ces liaisons confèrent aux brins d'ADN une orientation, dénotée $5'$–$3'$ (en raison de la position des atomes de carbone portant les liaisons fortes). Les deux brins complémentaires ont de ce fait une orientation opposée. Les différentes structures sont illustrées par la figure 1.1.

1.1.2 L'ARN

Les molécules d'ARN diffèrent des molécules d'ADN en trois points essentiels : i/ elles sont constituées d'une seule chaîne de nucléotides, plutôt que d'une chaîne double ; ii/ la thymine est remplacée par l'*uracile*, qui est également une pyrimidine. En outre, la structure moléculaire de l'*uracile* est très proche de celle de la thymine, ce qui lui permet aussi

FIGURE 1.1 : Les différentes structures de l'ADN.



Représentation des structures I^{aire}, II^{aire}, III^{aire} et IV^{aire} de l'ADN : du noyau chromosomique jusqu'à la séquence nucléotidique.

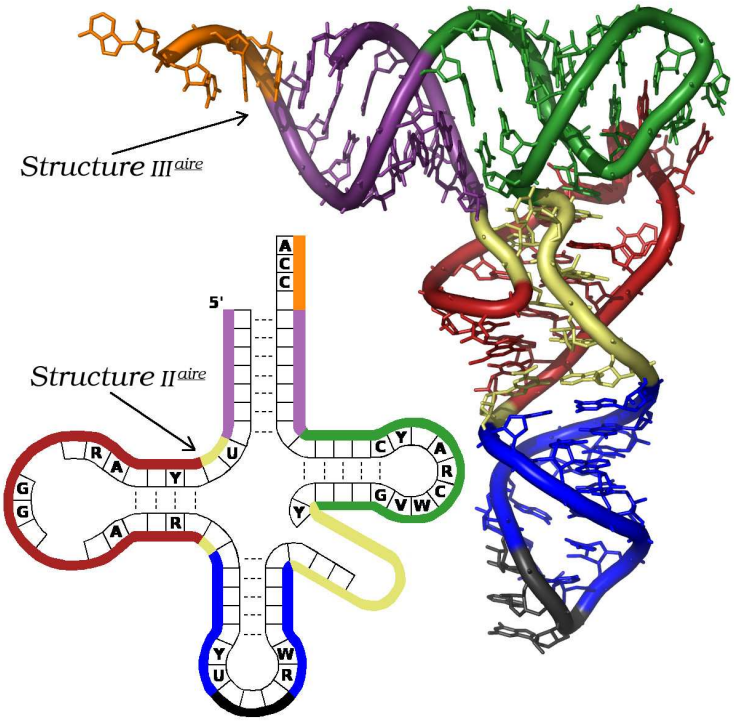
de s'apparier à l'adénine ; iii/ la composition nucléotidique diffère avec un désoxyribose pour l'ADN contre un ribose pour l'ARN.

De même que l'ADN correspond au support du code génétique, l'ARN peut être décrit comme un support génétique intermédiaire de nos gènes pour fabriquer les protéines dont elles ont besoin. En fait, dans la cellule, l'ARN est une copie locale d'une région de l'un des brins de l'ADN. En effet, il est synthétisé à partir d'un brin complémentaire de l'ADN, en substituant juste l'uracile à la thymine. Ensuite, chez les eucaryotes la molécule d'ARN synthétisée reste dans le noyau et est traitée par un complexe enzymatique. Ce mécanisme s'appelle *l'épissage* : certaines séquences appelées introns sont excisées, les exons restant se relient ensuite entre eux. Il peut y avoir un mécanisme d'épissage alternatif, augmentant ainsi le nombre de possibilités d'ARN messager mature (voir section 2.6 du chapitre 2). Dans tous les cas, l'ARN produit est plus court, passe dans le cytoplasme et devient un *ARN mature*. Tout ce mécanisme est appelé la *transcription* (voir section 1.2.1).

Les ARN ainsi produits peuvent avoir trois grands types de fonctions, ils peuvent être support de l'information génétique d'un ou plusieurs gènes codant pour des protéines (on parle alors d'ARN messagers), ils peuvent adopter une structure secondaire et tertiaire stable et accomplir des fonctions catalytiques (par exemple l'ARN ribosomique), ils peuvent enfin servir de guide ou de matrice pour des fonctions catalytiques accomplies par des facteurs protéiques comme c'est le cas par exemple des microARN.

Comme pour l'ADN, plusieurs niveaux de représentations de l'ARN sont possibles (FIGURE 1.2).

FIGURE 1.2 : Les différentes structures de l'ARN.



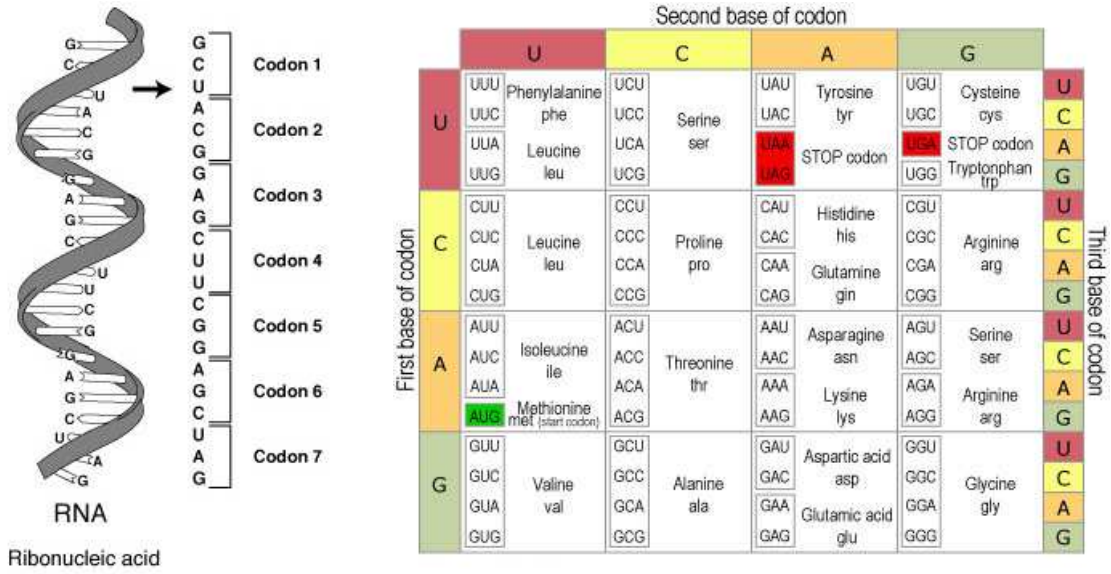
Représentation des structures II^{aire} et III^{aire} de l'ARN.

1.1.3 La protéine

La *protéine* est aussi une macromolécule. Elle est synthétisée par les cellules à partir de l'information contenue dans le génome et véhiculée par l'ARN. Par exemple, lors du processus de *transcription*, si le transcrit porte sur la synthèse d'une protéine (un ARNm), alors il est *traduit* sous forme de protéine. La *traduction* est une étape qui consiste à lire les nucléotides de l'ARNm par bloc de 3, appelé *codon* (FIGURE 1.3(a)). La lecture s'exécute dans le sens $5' - -3'$ de l'ARNm. Ensuite, chaque *codon* lu doit correspondre à un *acide aminé* parmi les 20 possibles, pour que la *traduction* puisse continuer (FIGURE 1.3(b)). Enfin, la lecture s'arrête lorsqu'un *codon STOP* est lu. La protéine est ainsi construite par l'accumulation des acides aminés [Crick *et al.*, 1961].

Les protéines jouent un rôle essentiel dans la fonction d'un organisme. Elles ont une activité permanente au sein d'une très grande majorité des processus cellulaires [Harvey Lodish, 2000] : les réactions de biocatalyse (enzymes), le transport et le stockage des nutri-

FIGURE 1.3 : La traduction d'un ARNm.



(a) (b)

Dans la figure 1.3(a), un ARNm est traduit. La *traduction* s'exécute par la lecture des *codons* de sa partie 5′ vers sa partie 3′ (bloc de 3 nucléotides). La figure 1.3(b) représente les acides aminés possibles qui sont associés à des *codons*. On s'aperçoit que la traduction s'arrête après la lecture du codon 7. En effet, le *codon STOP* marque l'arrêt de la lecture.

ments, la régulation des hormones, la défense immunitaire (anticorps), et bien d'autres encore.

FIGURE 1.4 : Le principe de la synthèse protéique.

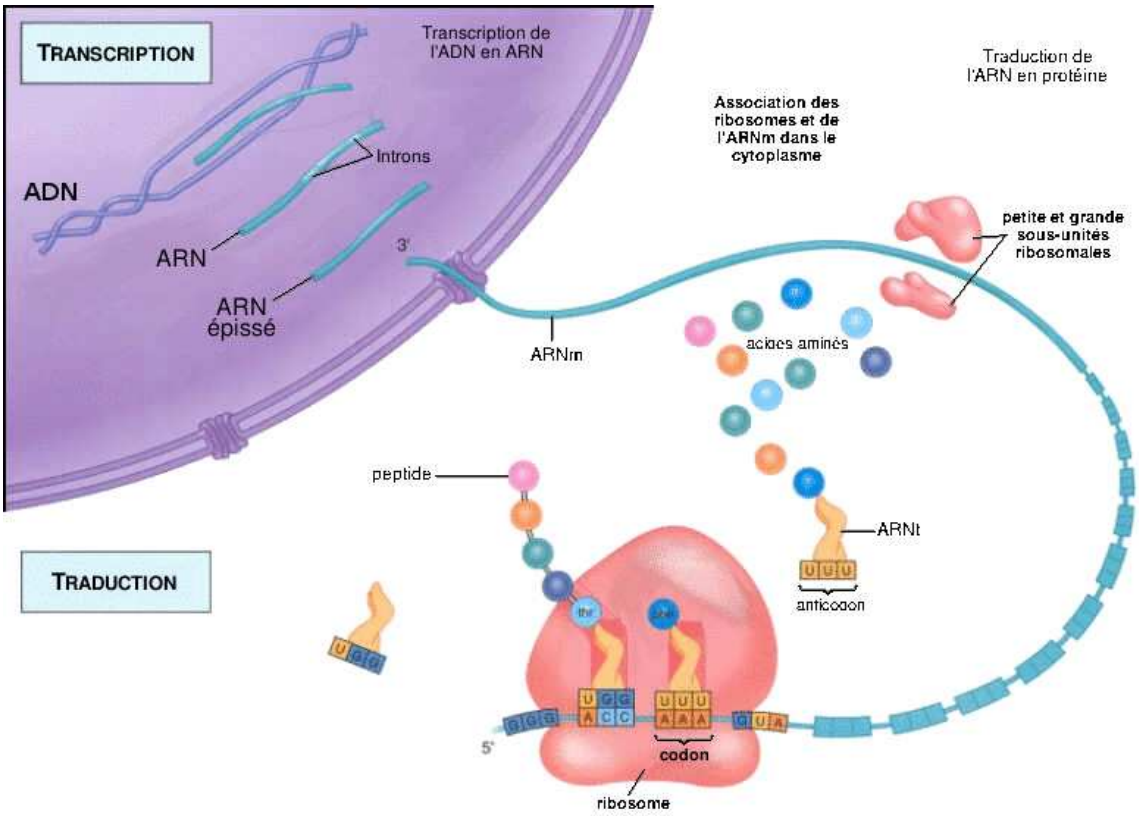


Illustration du processus de transcription d'un ARNm, puis de sa traduction en protéine.

1.1.4 Le gène

Un *gène*, dans sa définition la plus simple et concise, est une unité d'information génétique. En effet, le génome est souvent comparé à une encyclopédie dont les différents volumes seraient les chromosomes. Quant aux gènes, ils composent les phrases contenues dans ces volumes. Des phrases qui sont écrites dans un langage génétique représenté par les quatre bases de l'ADN.

Il faut cependant noter que parallèlement à cette définition simple du gène, il existe des définitions plus précises issues du progrès des connaissances. La première date de 1909 par De Vries, elle est basée sur les concepts des généticiens Morgan et Mendel. Depuis, le concept de gène n'a cessé d'évoluer. Maintenant, nous définirons un gène comme une séquence génomique représentant la zone de transcription d'ARN. Ainsi, chez les eucaryotes, les portions d'appariement entre ADN et ARN constituent les exons et les autres portions constituent les introns (FIGURE 1.5). Cependant, sa définition tend à se complexifier encore si l'on prend en compte l'ensemble des phénomènes biologiques identifiés au cours de ces dernières années [Gerstein *et al.*, 2007] (on y reviendra dans les chapitres suivants).

FIGURE 1.5 : La représentation d'un gène.

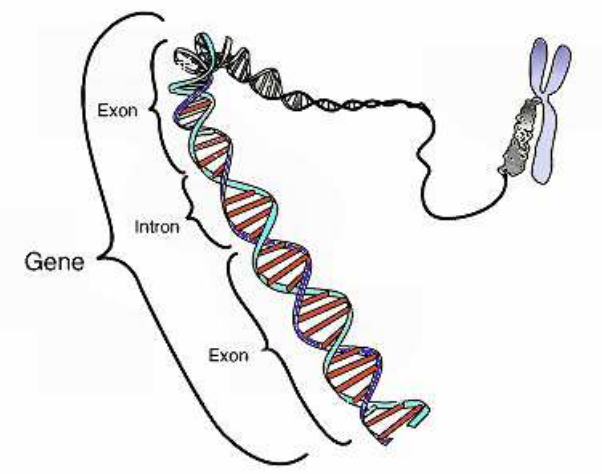


Illustration d'une portion d'un génome eucaryote qui s'apparente à un gène. Ce dernier étant composé d'exons et d'introns.

1.1.5 Le génome

Les notions précédentes constituent les éléments essentiels à la compréhension des problématiques actuelles de la biologie moléculaire chez les eucaryotes. L'étude des gènes a été longtemps limitée à celle d'un seul gène ou d'une portion d'un génome. Depuis les années 90, les études ont changé d'échelle pour passer à des analyses globales portant sur le fonctionnement d'un organisme, d'un organe, d'un cancer, etc, le tout à l'échelle du génome (et non plus à celle d'un seul gène). C'est sous le terme de génomique que se définit cette discipline de la biologie moderne. Elle est née de l'évolution concomitante des technologies et de l'informatique, elle comprend principalement 2 aspects : d'une part la génomique structurale qui va s'intéresser à la compréhension des génomes par l'étude de leur séquence, la cartographie des gènes et les comparaisons entre espèces, d'autre part la génomique fonctionnelle qui englobe l'étude des transcriptomes et des protéomes pour arriver à déterminer la fonction des gènes (voir section 1.2).

Dans le cadre des programmes de génomique structurale, la première étape est celle du séquençage. Pour donner quelques chiffres, les premiers séquençages de génomes complets sont apparus dans les années 1990 et une dizaine de génomes de référence a vu le jour dans la période de 1990 à 2000. Le site du NCBI recense actuellement plus de 180 génomes disponibles (http://www.genomenewsnetwork.org/resources/sequenced_genomes/genome_guide_p1.shtml). Les programmes d'annotation constituent le deuxième volet, ils nécessitent la mise en commun massive des connaissances et des résultats, c'est le cas du consortium *ENCODE* pour « the Encyclopedia of DNA Elements » créé pour l'annotation des génomes de l'homme et de la souris (http://genome.ucsc.edu/ENCODE/index.html), en répertoriant l'ensemble des éléments fonctionnels d'un génome incluant les protéines, les transcrits et les éléments régulateurs qui contrôlent les gènes [The ENCODE Project Consortium, 2007].

Parallèlement de nombreux projets de séquençage voient le jour, pour ne citer que les plus marquants, le programme des 1000 génomes humains portés par le *1000 Genomes Consortium* et le projet *Cancer* initié par *ENCODE* dont la perspective est de coordonner des études à grande échelle sur plus de 50 types de cancers différents et correspondant à plus de 25 000 analyses [Via *et al.*, 2010; International Cancer Genome Consortium *et al.*, 2010]. L'accroissement de ces informations va de pair avec l'augmentation continue du nombre de base de données qui recensent ces informations [Galperin, 2007].

1.2 Introduction à la transcriptomique

La *transcriptomique* est l'étude des *transcriptomes*. Le *transcriptome* est l'ensemble complet de tous les *transcrits* d'une cellule, ainsi que leur quantité, à un instant donné dans le développement cellulaire ou pour une condition physiologique spécifique. Un *transcrit* est la dénomination de la séquence nucléotidique correspondant à un ARN lors de la *transcription*.

1.2.1 La transcription

La *transcription* fait partie, avec la *réplication* et la *traduction*, des axiomes de la théorie fondamentale de la biologie moléculaire [Crick, 1970]. Cette théorie explique le transfert des informations entre les biopolymères (les nucléotides et les protéines) dans les organismes vivants. L'ADN, l'ARN et les protéines sont les principaux protagonistes au cœur de cette théorie (voir la figure 1.4).

Les gènes, quant à eux, constituent les régions de l'ADN qui contiennent les informations nécessaires pour constituer la structure primaire des *ARN messager* (ARNm). Ils illustrent les mécanismes de transcription, d'épissage des ARNm, et de leur traduction en protéine. La transcription nécessite l'intervention d'un ARN polymérase (ARN responsable de la synthèse de l'ARN). La transcription est initiée dans la région promoteur du gène. Pendant la transcription, la *chromatine* s'ouvre et les deux brins de l'ADN sont temporairement séparés. La polymérase se lie sur l'un des deux brins (brin matrice), puis commence la catalyse de la partie $3'$ vers la partie $5'$ de ce brin matrice, c'est-à-dire la construction d'une chaîne ribonucléique qui est complémentaire au fragment transcrit. La chaîne ainsi produite est équivalente au fragment sur l'autre brin (le brin codant) de l'ADN (voir figure 1.6) [Bruce Alberts, 2002].

Cependant, de nombreux ARN sont transcrits en dehors des gènes et par conséquent ne sont pas traduits en protéines. La *transcription* est parfois indépendante des gènes. Elle conduit à la synthèse d'un ARN à partir d'une région de l'ADN mais pas forcément d'une région génique. Cette synthèse est produite à l'aide d'une enzyme appelée *ARN polymérase*.

FIGURE 1.6 : Le processus de transcription.

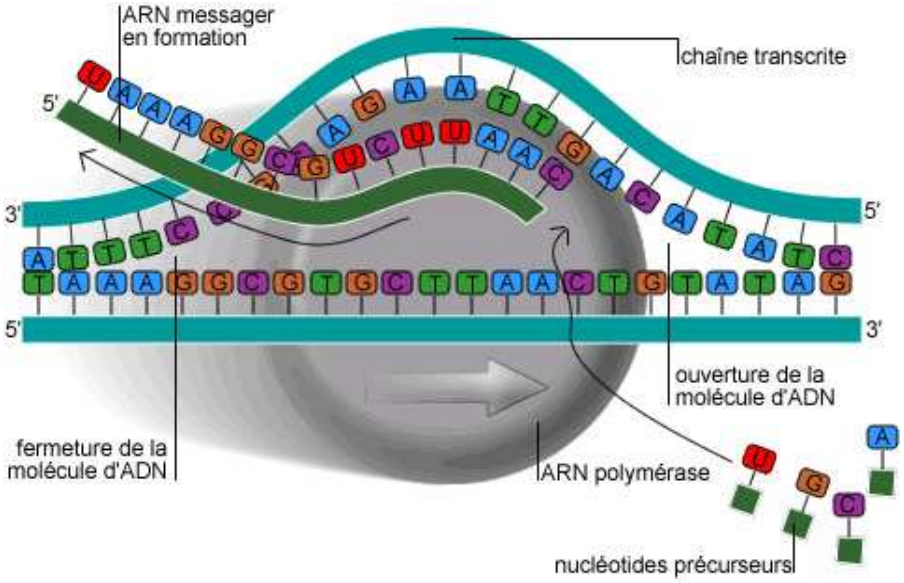


Illustration du processus de transcription. Nous pouvons observer l'ARN polymérase se lier sur le brin matrice lors de l'ouverture de la chromatine. Un ARNm se construit par catalyse du brin matrice. La chaîne ainsi produite est complémentaire au fragment transcrit du brin matrice mais équivalente au fragment sur le brin codant.

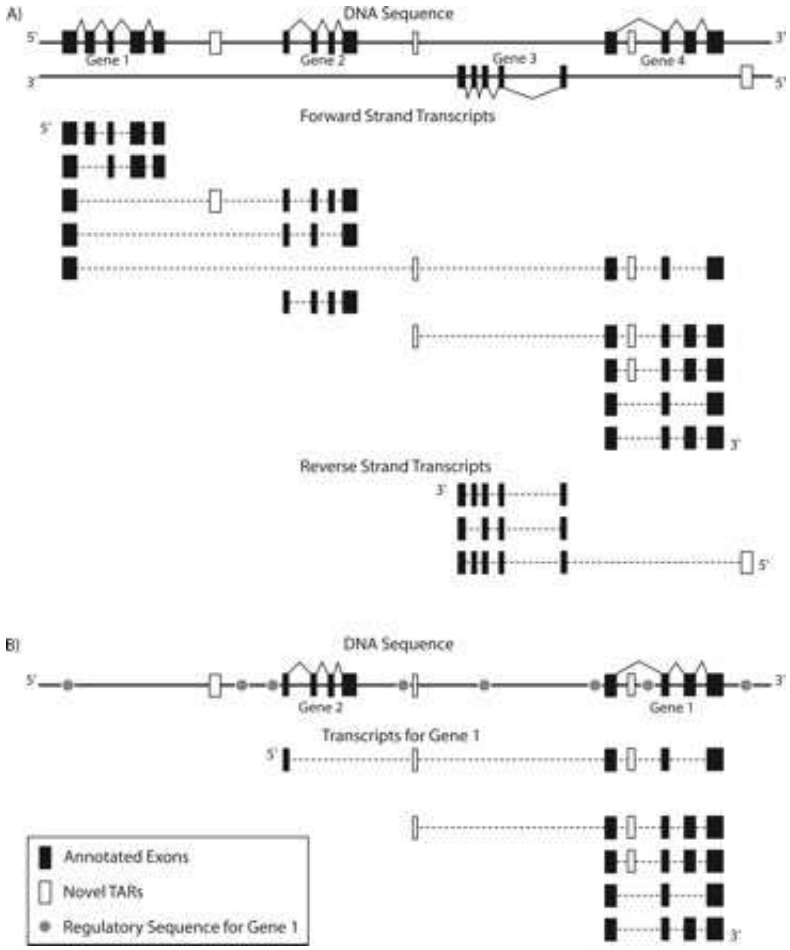
1.2.2 L'essor de la transcriptomique

Au début du siècle, le séquençage des génomes apparaissait comme la clé de tous les mystères. Pourtant, le décryptage du génome humain [Consortium, 2001] a montré que les informations contenues dans les séquences d'ADN ne permettaient pas de comprendre le mode d'action d'un organisme vivant et les différences entre les espèces. Pour cause, très peu de gènes sont identifiés et l'écart génétique qui sépare les différents mammifères est faible : les gènes de la souris sont à 90% similaires à ceux de l'homme [Waterston *et al.*, 2002]. Si peu de gènes, et si semblables d'une espèce à l'autre ! Comment croire qu'ils contiennent à eux seuls les clés du vivant ?

Depuis quelques années, les séquences complètes des premiers génomes sont disponibles et accessibles sur la toile, ce qui accélère la progression de nombreux projets en biologie moléculaire, réduisant des années de travail à des mois, des mois à des jours. En fait, toutes ces informations disponibles sur les génomes ont permis d'aborder la biologie de manière globale et de développer de nouvelles méthodes d'études des transcriptomes.

Des approches qui ont permis : la détection de nouveaux gènes dans les organismes ; la mise en évidence de structures complexes de gènes avec une multiplicité d'exons souvent très petits, mais aussi l'existence de zones transcrites non-codantes en dehors des gènes [Johnson *et al.*, 2005; Gerstein *et al.*, 2007] (FIGURE 1.7). Des nouvelles pistes ont ainsi été mises en exergues et ont permis d'identifier de nouveaux mécanismes de régulation cellulaires. Par exemple, des études ont montré qu'une majeure partie du génome, chez l'homme, est transcrite [Bertone *et al.*, 2004; The ENCODE Project Consortium, 2007; Rozowsky *et al.*, 2007] et que la proportion et le rôle des ARN ne codant pas pour des protéines (ARNnc) est largement sous estimée [Amaral *et al.*, 2008]. Ce type d'études, qui démontre la complexité de la transcriptomique, permettra sans doute d'aider à mieux comprendre l'expression et la régulation des gènes dans les cellules. À peine a-t-on édité quelques séquences complètes des génomes, que se présente un nouveau défi, encore plus grand, les interpréter.

FIGURE 1.7 : La structure complexe des gènes.



Cette figure illustre la complexité de l'annotation du génome humain. Avec les nouvelles données en matière de transcription, nous observons l'existence de plusieurs transcrits qui s'apparentent à la combinaison de plusieurs gènes ce qui montre leur complexité.

1.2.3 Les premières techniques

À ce jour, l'étude des transcriptomes par séquençage haut débit est devenue incontournable. C'est la raison pour laquelle un chapitre entier lui est dédié (chapitre 2). Nous allons par conséquent nous concentrer, ici, uniquement sur les deux approches transcriptomiques qui ont été les premières utilisées. Une première méthode fondée sur l'hybridation sur puce des ARN, une autre basée sur les premières techniques de séquençage.

La première méthode, plus connue sous les termes *macroarrays*, *microarrays* ou encore *oligochips* (selon l'étude), est une méthode ciblée [Marshall, 2004]. Elle repose sur le principe suivant : des fragments d'ADN (sondes) sont immobilisés de façon ordonnée sur un support solide, puis un ensemble d'ADNc marqué (identifiant les ARN cibles) est mis en présence du support. Enfin, grâce aux propriétés d'appariement de l'ADN, la double hélice de l'ADN est reconstruite, ce qui conduit à la formation d'hybrides moléculaires. Le signal mesuré traduit l'apparition de ces hybrides et l'intensité du signal leur abondance relative.

L'autre approche est une méthode numérique par séquençage Sanger. Dans le jargon biologique, on parlera de méthode SAGE (Serial Analysis of Gene Expression) [Velculescu *et al.*, 1995] ou longSAGE (séquences SAGE plus longues) [Saha *et al.*, 2002]. Elle repose sur une technique précise de séquençage d'un ensemble ADNc (identifiant les ARN cibles). La region séquençée correspond à une région 3′ des ADNc définie par un site de restriction (voir section 2.2.1 du chapitre 2), plus ou moins courte suivant qu'il s'agisse de SAGE (14bp) ou de LongSAGE (21pb). Au final, chaque transcrit est identifié par cette courte séquence appelée *tag*. En outre, grâce à cet ancrage, plus un transcrit est présent dans l'échantillon, plus le *tag* correspondant est fréquent. La méthode numérique est donc quantitative. Plusieurs variantes de cette technique ont été utilisées permettant des étiquetages en 3′ ou 5′ des transcrits. Parmi elles nous trouvons la méthode CAGE (Cap Analysis of Gene Expression) principalement utilisée pour étudier l'architecture des régions promotrices [Harbers et Carninci, 2005].

Toutefois, rares sont les laboratoires qui n'ont pas recours aux *microarrays* à un stade ou un autre de leurs travaux, en raison des améliorations croissantes apportées par une production industrielle. Ces améliorations ont permis une qualité de standardisation rendant possible des comparatifs d'une plate-forme à une autre [Shi *et al.*, 2006]. En revanche, contrairement aux méthodes numériques, les *microarrays* ne permettent pas de rechercher des transcrits encore inconnus, on parle alors de méthode fermée. Les méthodes SAGE et longSAGE, quant à elles, sont ouvertes car il n'y a pas d'*a priori* sur ce qui est exprimé dans une cellule. Ces méthodes permettent d'analyser les résultats avec une grande

facilité, même provenant de laboratoires différents puisque les données reposent sur un simple séquençage de *tags* et leur dénombrement [Khattra *et al.*, 2007]. Ainsi, l'analyse comparative des données se fait d'autant plus facilement [Nielsen *et al.*, 2006]. Néanmoins, le protocole contient une phase de clonage biologique qui constitue la principale difficulté de la technique. En effet, ce protocole a besoin de précision biologique et reste coûteux en temps. Ces derniers sont des critères fondamentaux dans de nombreux domaines, notamment dans le diagnostic médical. Même si ce n'est pas l'objet de ce chapitre, notons que depuis peu, les technologies à haut débit ont largement contribué à une simplification des méthodes SAGE et longSAGE et permis une profondeur d'analyse des transcrits sans précédent [Blow, 2009].

1.2.4 Les génomes et les transcriptomes sont indissociables

Cependant quelle que soit la méthode utilisée, l'exploitation des données de transcriptomes ne peut s'appuyer que sur une solide connaissance des génomes (séquençage, annotation, caractérisation des sites de régulation, conservation, etc) pour permettre une interprétation globale de l'information biologique.

Les scientifiques parlent souvent de génome comme d'une unité simple, mais il s'agit en fait d'un concept largement idéalisé. En pratique, il s'agit d'une entité très variable, d'une espèce à une autre, aussi bien en terme de nombre de bases nucléotidiques, de chromosomes que de gènes. Lorsqu'on veut initier une analyse de séquences, il est nécessaire, dès la première étape, comme dans un texte, de déterminer l'ordre des pages, et les marques de ponctuation.

Les *facteurs de transcription* illustrent bien cette nécessité d'avoir des informations solides à la fois sur le génome et le transcriptome. En effet, toutes les cellules d'un organisme possèdent le même génome, c'est-à-dire la même information génétique contenue dans des gènes avec une régulation sélective (activation des gènes caractéristiques d'un type cellulaire et répression des gènes inadéquats) qui confère aux cellules leur identité [Fisher, 2002].

Cette régulation transcriptionnelle implique d'étroites coopérations entre différents mécanismes impliquant les facteurs de transcription qui se lient sur des sites précis situés sur le génome, et des mécanismes épigénétiques qui modifient physiquement la chromatine. La plupart des mécanismes épigénétiques sont ubiquitaires, ils vont permettre la condensation ou le relâchement de la chromatine, ce dernier phénomène favorisant la transcription. La présence de facteurs de transcription spécifiques va conditionner la transcription et ainsi la spécificité cellulaire. Chaque type cellulaire possède son propre

réseau de facteurs de transcription qui sont nécessaires à l'établissement et au maintien de son identité [Sieweke et Graf, 1998].

Dans l'idéal, pour comprendre parfaitement le comportement d'un facteur de transcription donné, les informations doivent être analysées dans un contexte cellulaire en considérant une foule de paramètres incluant l'architecture nucléaire, les domaines de la chromatine, les territoires chromosomiques. Une dérégulation de la transcription dans la cellule peut perturber l'identité cellulaire et ainsi favoriser le développement de tumeurs cancéreuses [Thorne *et al.*, 2009].

La combinaison d'un ensemble d'informations, recueillies dans un système cellulaire donné, aide à l'annotation et la caractérisation des régions non-codantes des génomes, ainsi qu'à la compréhension de leur rôle biologique [Alexander *et al.*, 2010] : la caractérisation d'une nouvelle famille de longs ARN non-codants (ARNlnc) a permis la découverte de nouveaux mécanismes de régulation [Rinn *et al.*, 2007; Derrien et Guigó, 2011].

1.2.5 Les ARN non-codants

En 2004, une nouvelle technologie dénommée *Tiling Arrays* fait son apparition. Elle est fondée sur le même principe que celui des microarrays à une exception : le génome tout entier, et non plus une toute petite partie, est concernée. Cette particularité rend cette méthode ouverte avec un balayage massif du génome ce qui assure une couverture de la quasi totalité des transcrits exprimés [Bertone *et al.*, 2004].

C'est à partir de cette technologie que des travaux issus de l'université de Yale [Bertone *et al.*, 2004; Johnson *et al.*, 2005] et la société Affymetrix [Cheng *et al.*, 2005; Rozowsky *et al.*, 2007] ont mis en évidence une transcription largement étendue dans le génome humain. De nombreuses régions transcrites ont ainsi été mises en évidence en dehors des gènes connus et des régions identifiées comme codantes car elles ont échappé à tout programme de prédiction : cette nouvelle transcription correspond essentiellement à des ARN non-codants (ARNnc) [Kapranov *et al.*, 2007]. Ces derniers sont assimilés à de la « matière noire », allusion à un concept de l'astrophysique : « l'univers contiendrait beaucoup plus de matière que ce que l'on observe, du fait de l'existence d'une matière noire invisible autour des galaxies ». En effet, les transcrits représentent plus de la moitié de l'ADN d'un génome, alors que les gènes qui codent des protéines ne correspondent, eux, qu'à 1% de cet ADN ! En somme, la moitié de l'ADN se transcrit en ARN qui ne donne pas de protéine. A quoi servent ces transcrits ?

Ce nouveau phénomène de transcription massive constitue l'un des points centraux du

projet ENCODE, qui comprend l'analyse d'environ 1% du génome humain avec une précision inégalée [The ENCODE Project Consortium, 2007; Rozowsky *et al.*, 2007]. Par ailleurs, l'abondance croissante des données transcriptomiques contrastant avec le décompte révisé à la baisse des gènes de protéines laisse supposer que le transcriptome a un rôle essentiel dans la régulation des gènes. Un autre élément indirect allant dans ce sens est l'abondance des facteurs de transcription (voir section 1.2), au nombre de plusieurs centaines de milliers, en particulier dans les régions introniques et intergéniques du génome [Mardis, 2007].

En revanche, un argument contraire parle de surestimation de la transcription. Il semblerait en effet qu'une partie de cette matière noire ne soit que le simple reflet d'un bruit de fond transcriptionnel introduit par différents biais protocolaires [van Bakel *et al.*, 2010]. Ce point fait toujours l'objet de débats et l'utilisation de méthodologies complémentaires pourrait apporter des précisions sur la caractérisation de ce répertoire transcriptionnel.

1.3 Introduction à la bioinformatique

La bioinformatique joue un rôle déterminant avec la conception d'outils de stockage et de calcul. Toutes ces méthodes qui vont de l'analyse du génome à la modélisation de l'évolution d'une population dans un environnement donné, en passant par la modélisation moléculaire, l'analyse d'image, le stockage des séquences génomiques et la reconstruction d'arbres phylogénétiques ont et vont encore apporter des avancées considérables à la connaissance des espèces.

1.3.1 La bioinformatique et les séquences biologiques

L'analyse de séquences biologiques est souvent réalisée *in vitro*, par les chercheurs en biologie moléculaire, en physiologie ou encore en biochimie. Cependant, depuis l'invention du séquençage de l'ADN et surtout depuis la naissance des séquenceurs haut débit, une analyse *in vitro* est impossible. En effet, un nombre exponentiellement de séquences de génomes est généré, dont l'annotation reste à effectuer. C'est à ce niveau que la bioinformatique tient son intérêt en permettant une analyse *in silico* des données à traiter. La bioinformatique aborde les études sous un aspect plus formel où certaines théories développées dans le cadre de l'algorithmique du texte trouvent une application dans le traitement de ces séquences. En outre, il ne s'agit pas simplement d'appliquer des théories aux problèmes biologiques, mais de les adapter, d'en développer de nouvelles plus spéci-

fiques.

Dans le cadre du séquençage à haut débit, la première difficulté consiste à organiser une énorme masse d'information et de la rendre facilement accessible à l'ensemble de la communauté des chercheurs (voir section 2.3 du chapitre 2). La deuxième difficulté consiste à concevoir et développer des outils d'analyse capables de gérer cette même masse de données (voir section 2.4 du chapitre 2).

1.3.2 Les notations et définitions relatives aux séquences

Dans les chapitres 3, 4 et 5, des méthodes bioinformatiques sur l'algorithmique du texte ont été développées pour répondre à des problèmes biologiques sur le traitement des séquences. Ces méthodes étant formelles, nous avons besoin de définir des notions élémentaires sur l'algorithmique du texte : elles seront utilisées tout au long de ce manuscrit. En effet, les séquences biologiques sont des portions d'ADN qui sont considérées comme du texte.

Définition 1.1. *Un alphabet est un ensemble fini de symboles définit par* Σ. *Le cardinal* σ *de cet ensemble est le nombre de symboles qui le compose. Dans le cas de l'ADN, on obtient* $\Sigma = \{A,C,G,T\}$ *et* $\sigma = 4$.

Étant donné un alphabet Σ, une suite finie et ordonnée de symboles de Σ est appelée *chaîne*. Ainsi une chaîne qui est composée de m symboles appartient à Σ^m. Lorsque la chaîne représente une séquence biologique, la dénomination *séquence* est utilisée. Le terme *mot* est employé pour désigner une chaîne dans un cadre abstrait. Par opposition, le terme *motif* est utilisé pour exprimer une connotation biologique.

Notons que pour les séquenceurs haut débit (SHD), quelques termes spécifiques sont couramment employés. Les *reads* sont associés aux courtes séquences générées par les SHD. Le *mapping* est l'action qui consiste à positionner des *reads* sur une séquence génomique ou transcriptomique. Le *splicing* est l'action qui consiste à détecter les jonctions d'épissage dans les *reads* (voir section 2.6.1 du chapitre 2).

Définition 1.2. *Une séquence ou un* read w *de taille* m *est un mot composé d'une suite finie de lettres de* Σ *tel que,* $w = w[0]w[1]\dots w[m-1]$ *où* $w[0]$ *est la première lettre et* $w[m-1]$ *la dernière lettre de* w.

Définition 1.3. *Un mot* p *est un préfixe d'un mot* w *s'il existe un mot* v *tel que* $w = pv$. *L'ensemble des préfixes de* w *est noté* $Pref(w)$.

Définition 1.4. *Un mot s est un suffixe d'un mot w s'il existe un mot u tel que $w = us$.*
L'ensemble des suffixes de w est noté $Suff(w)$.

Définition 1.5. *Un mot f est un facteur d'un mot w s'il existe deux mots u et v tels que $w = ufv$. Notons que tout préfixe d'un suffixe est par conséquent un facteur.*
L'ensemble des facteurs de w est noté $Fact(w)$.

Remarque 1. *Le terme k-mer est employé pour désigner un facteur de longueur k.*

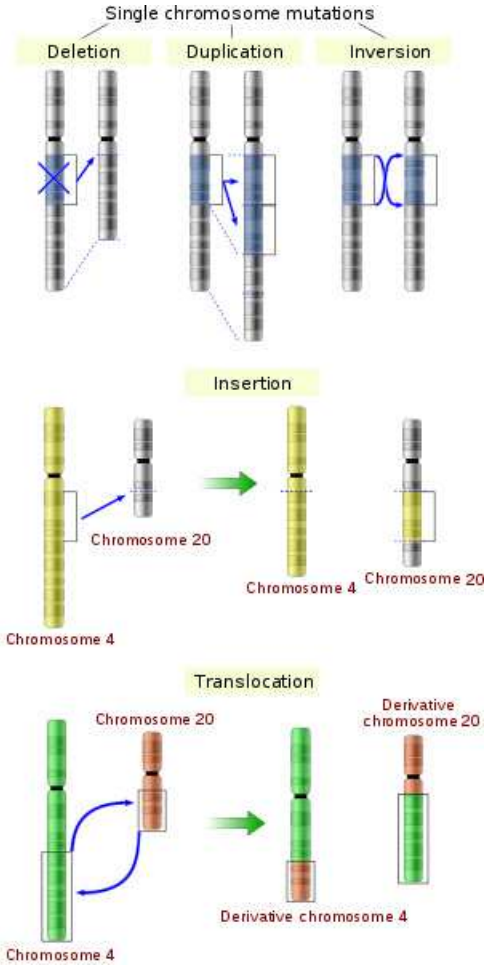
Exemple 1.1. Le préfixe, le suffixe et le facteur.
Soit w = ACAGT une séquence d'ADN, alors :
$Pref(w)$ = {A, AC, ACA, ACAG, ACAGT},
$Suff(w)$ = {T, GT, AGT, CAGT, ACAGT},
$Fact(w)$ = {A, C, G, T, AC, CA, AG, GT, ACA, CAG, AGT, ACAG, CAGT, ACAGT}.

1.4 Mutations biologiques

Dans nos principaux résultats (chapitre 3 et chapitre 5) nous identifions des mutations biologiques, c'est-à-dire des différences biologiques entre deux séquences d'ADN. Ces mutations sont des changements physico-chimiques de l'ADN dans les cellules. Elles sont souvent causées par des *facteurs extérieurs* tels que les radiations, les virus, les produits chimiques mutagènes, etc [Kryston *et al.*, 2011], par des *transposons* (des morceaux d'ADN hébergés qui peuvent se déplacer dans le génome) [Goryshin *et al.*, 2000], ou encore par des problèmes survenant lors du processus de réplication de l'ADN. Des *mutations somatiques* qui se développent au sein des cellules et peuvent être à l'origine de l'apparition de tumeurs [Kan *et al.*, 2010]. Certaines mutations vont rester sans effet, d'autres vont conduire au fil des générations à une évolution des fonctions afin de répondre à de nouveaux besoins [Meyer, 2008], et d'autres vont engendrer des défauts et mener à toutes sortes de maladies génétiques, neurologiques, etc [Kan *et al.*, 2010; Kryston *et al.*, 2011; Lee *et al.*, 2010; Stephens *et al.*, 2009].

Il y a plusieurs types de mutations, elles peuvent n'affecter qu'un seul ou un petit nombre de nucléotides, comme elles peuvent provoquer des grands changements au niveau des gènes et des chromosomes : duplication, inversion, suppression, déplacement ou réarrangement d'un gène ou d'une portion de chromosome (FIGURE 1.8).

FIGURE 1.8 : Les différents type de mutations biologiques.



1.4.0.1 Les petites mutations

Cette catégorie rassemble les mutations qui affectent de un à quelques nucléotides, qui peuvent être des substitutions, délétions ou insertions.

Les mutations ponctuelles, *SNV* (Single Nucleotide Variant) . Elles sont souvent causées par des facteurs extérieurs chimiques ou l'exposition aux radiations par des problèmes lors de la réplication, la recombinaison ou la réparation de l'ADN. Leur caractéristique est tout simplement la modification d'un seul nucléotide dans la séquence [Freese, 1959a]. La substitution peut être de deux natures différentes : une transition ou une transversion [Freese, 1959b]. La plus commune est la transition qui échange soit, deux purines

entre elles (A ↔ G), soit deux pyrimidines entre elles (C ↔ T). Les transversions sont plus rares et se manifestent par un échange entre une purine et une pyrimidine (C/T ↔ A/G). Notons que deux mutations successives d'un nucléotide peut faire revenir la séquence à l'état original.

Les SNV qui sont présents dans la partie codante des gènes peuvent avoir plusieurs effets : abolir la traduction, induire la formation d'une protéine avec une fonction atténuée, ou une protéine avec une fonction différente. Tout dépend du codon produit par le SNV. Ces derniers seront qualifiés de : i/ mutation silencieuse si le codon obtenu code pour le même acide aminé que l'ancien ; ii/ mutation faux-sens si le codon produit correspond à un autre acide aminé ; iii/ mutation non-sens si le codon produit est un codon stop. De la même manière l'expression des gènes peut être affectée par des mutations dans leurs régions régulatrices.

Remarque 2. *En bioinformatique, les termes SNP (Single Nucleotid Polymorphism) et SNV (Single Nucleotid Variant) sont souvent confondus à tort. Bien qu'ils désignent tous deux la variation d'un seul nucléotide, on parle de SNP quand la mutation est conservée au sein d'une espèce dans une fraction significative de la population (polymorphisme génétique) alors que le SNV est une mutation intrinsèque à un individu, une mutation qui peut se manifester à n'importe quel moment de la vie (cellulaire).*

Les insertions. Elles peuvent être causées par des éléments génétiques mobiles (transposons, rétrovirus, etc) ou par des défauts lors de la réplication, la recombinaison ou la réparation de l'ADN. Leur caractéristique est l'insertion d'une séquence de nucléotides dans la séquence d'origine. Les insertions introduites dans des régions codantes d'un gène peuvent avoir toutes sortes de conséquences, par exemple abolir la fonction, altérer les jonctions d'épissage de l'ARNm (les sites d'épissage étant déplacés), ou encore décaler le cadre de lecture des codons. Les insertions peuvent dans certains cas être neutralisées par l'excision de l'élément transposable.

Les délétions. Leur caractéristique est la suppression d'une séquence de nucléotides dans la séquence d'origine. Comme les insertions, elles peuvent provoquer des altérations profondes de la fonction des gènes. En revanche, elles sont généralement irréversibles.

Remarque 3. *En bioinformatique, les insertions et les délétions sont souvent classées dans une même catégorie : les* indels *(de l'anglais* insertions-deletions*). En effet, lors d'un alignement de deux séquences, une insertion dans une séquence peut aussi être vue comme une*

délétion dans l'autre séquence , d'où le terme indels pour factoriser insertions-deletions. *Dans certains outils de* mapping*, on emploiera aussi le terme* gap *pour désigner un* indel.

1.4.0.2 Les grandes mutations

Cette catégorie rassemble les grandes mutations qui affectent de larges portions d'ADN et la structure chromosomique. Dans leurs versions extrêmes, des chrosomosomes entiers peuvent être gagnés ou perdus.

Les amplifications ou duplication de gènes. Elles mènent à l'augmentation du nombre de copies des gènes affectés.

Les grandes délétions. Leur caractéristique est une suppression d'une grande partie d'une séquence chromosomique, avec comme conséquence la perte possible d'un grand nombre de gènes.

Les réarrangements. Ce sont des mutations dont l'effet est de modifier l'ordre des séquences chromosomiques. Elles comprennent : i/ les translocations chromosomiques (voir section 2.6.2 du chapitre 2) ; ii/ les inversions chromosomiques, c'est-à-dire l'inversion de l'orientation d'un segment chromosomique (FIGURE 1.8).

En conclusion, les mutations présentes dans les séquences génomiques affectent les organismes vivants à divers degrés selon la nature et le type de mutation que ce soit dans l'altération, la perte ou l'évolution d'une fonction, avec des conséquences dans le développement des maladies, mais aussi dans la variabilité génétique entre les individus. Ces mutations sont des changements dans la séquence d'ADN de l'organisme. Tous ces changements peuvent être interprétés par des méthodes bioinformatiques. C'est l'un des objectifs principaux du chapitre 5.

CHAPITRE

2 Émergence du haut débit dans l'étude des transcriptomes

Dans ce chapitre nous commençons par retracer le parcours des séquenceurs haut débit (SHD) et leur impact sur l'annotation des transcriptomes. Nous exposons comment leur production massive de séquences (les reads*) ainsi que les erreurs qu'ils génèrent ont engendré des besoins en bioinformatique. Nous détaillons les techniques biologiques et les différents algorithmes informatiques existants pour le traitement des* reads *obtenus par SHD. Dans un deuxième temps, nous nous focalisons essentiellement sur le RNA-Seq en abordant tous les problèmes d'épissage et leurs rôles dans les organismes, avec notamment les transcrits chimères et leur influence dans les cancers.*

Sommaire

La connaissance du transcriptome est essentielle pour déterminer l'ensemble des transcrits exprimés dans les cellules et les tissus, mais aussi pour comprendre le développement des cellules et des maladies. Les principales caractéristiques de la transcriptomique sont : i/ cataloguer tous les transcrits de toutes les espèces tels que les ARNm, les ARNnc ou encore d'autres petits ARN ; ii/ déterminer la structure transcriptionnelle des gènes, c'est-à-dire les sites de départ et d'arrivée (5' et 3'), les jonctions d'épissage et autres

modifications post-transcriptionnelles ; iii/ mesurer le niveau d'expression des transcrits, à différents stades ou conditions cellulaires.

Contrairement aux microarrays, les méthodes de séquençage déterminent directement la séquence ADNc de l'ARN. Les premiers séquenceurs *Sanger* utilisaient des banques d'EST (Expressed Sequence Tags) [Boguski *et al.*, 1994] mais cette méthode n'était généralement pas quantitative. Ensuite, des techniques plus élaborées comme les méthodes SAGE et LongSAGE ont été développées mais elles sont très laborieuses et plutôt qualifiées de bas débit (très coûteuses en temps et ne permettent pas de séquencer les transcrits rares d'une cellule) [Gerhard *et al.*, 2004]. Pour pallier ce problème, des méthodes plus quantitatives ont été développées, comme la DGE (Digital Gene Expression) et le RNA-Seq (RNA Sequencing), sur une nouvelle génération de séquenceurs. La première ne séquence qu'une partie bien précise du transcrit : le *tag* ou la signature. La deuxième méthode permet, en théorie, de déterminer la séquence complète des transcrits (et non plus une seule partie) tout en restant quantitative.

Récemment, une nouvelle vague de séquenceurs haut débit (voire très haut débit) vient de faire son apparition. Ils produisent une collection encore plus volumineuse de *reads* d'une longueur variant de 50 *pb* à 2 *kpb*.

2.1 Séquenceurs haut débit

Avant le passage aux séquenceurs à haut débit (SHD), les séquenceurs *Sanger* ont dominé l'industrie des génomes sur presque deux décennies et ont permis de nombreuses et considérables réalisations telles que le séquençage et l'assemblage complet du premier génome humain [ConsortiumInternational, 2004]. Cependant, malgré ces avancées sans précédent, le besoin de nouvelles technologies encore plus performantes s'est fait sentir, afin de pouvoir notamment cerner les transcriptomes [Metzker, 2010].

L'arrivée des SHD sur le marché a modifié les approches scientifiques et les applications biologiques. L'atout majeur offert par les SHD est le volume gigantesque de *reads* pour un coût abordable et une rapidité exceptionnelle. Entre les technologies Sanger et les SHD, on passe de plusieurs années de séquençage à seulement quelques heures, pour produire 100 fois plus de *reads*, avec un coût 1000 fois plus réduit [von Bubnoff, 2008]. Cette puissance de séquençage repousse les limites de la biologie actuelle. Pour l'étude de l'expression des gènes,par exemple, les microarrays sont remplacés par les SHD qui sont capables d'identifier et de quantifier des transcrits rares encore inconnus. Ils peuvent donner des informations sur des épissages alternatifs ou des variants de gènes[Wold et Myers, 2007; Wang *et al.*, 2009]. De plus, la faculté de séquencer le génome complet de beaucoup

d'organismes ouvre la possibilité de faire des études comparatives et évolutives à grande échelle, ce qui était inimaginable il y a encore quelques années. Une importante application des SHD pourrait être le séquençage d' une multitude de génomes humains afin de renforcer nos connaissances et compréhensions sur la façon dont les différences génétiques affectent la santé ou les maladies et la réponse aux traitements [Schuster *et al.*, 2010].

2.1.1 Les différents types de séquenceurs

Concentrons nous essentiellement sur les trois SHD les plus utilisés dans la communauté scientifique : le séquenceur FLX de Roche 454®, les séquenceurs GA et HiSeq 2000 de Illumina® et le séquenceur SOLiD de Applied Biosystems®. Depuis leur apparition, ces nouvelles technologies ont déjà apporté des avancées importantes dans beaucoup de domaines, comme en génomique avec le projet de séquençage 1 000 *génomes* [Via *et al.*, 2010], en transcriptomique avec la détection de gènes de fusion dans les cancers [Maher *et al.*, 2009b] et dans bien d'autres applications. Les séquenceurs produisent des millions (voire milliards) de *reads* pour aborder de telles analyses. La quantité de données générées pour une expérience est de l'ordre de 450 M*pb* pour FLX, 18 – 35 G*pb* pour Illumina® et 30 – 50 G*pb* pour SOLiD, et la longueur respective des *reads* est en moyenne de 330 *pb* pour le premier, 75 – 100 *pb* pour le second et 50 *pb* pour le dernier [Metzker, 2010]. Ainsi, selon l'application sous-jacente, nous avons la possibilité de choisir entre des séquences courtes ou longues et des séquences peu ou fortement couvertes.

2.1.2 Les séquenceurs et les erreurs générées

Néanmoins, et ce, quel que soit le séquenceur, l'analyse des données SHD nécessite souvent d'organiser judicieusement les *reads* afin de détecter précisément des informations telles que des SNP (single nucleotide polymorphism) [Li, 2011] ou des variants d'épissage [Trapnell *et al.*, 2010]. Dans de tels cas, ce n'est pas seulement la quantité totale de données qui est nécessaire, mais aussi la qualité des *reads* car les taux d'erreurs et les biais systématiques ont une influence sur leur traitement [Shendure et Ji, 2008]. En d'autres termes, pour évaluer la performance des SHD, une étude sur la qualité des séquences doit être approfondie. Récemment, les différents SHD ont été comparés sur leurs erreurs respectives [Suzuki *et al.*, 2011]. Cette analyse montre que les types d'erreurs sont différents selon les SHD et qu'il existe des biais qui leur sont intrinsèques, par exemple Illumina® produit un taux d'erreurs qui augmente sur la longueur du *read*, et il en produit davantage sur les nucléotides G et C.

2.1.3 Les séquenceurs et les applications

2.1.3.1 Le bilan

Finalement, le choix du séquenceur est souvent dépendant du domaine biologique. Dans l'étude du transcriptome, Illumina® monopolise les applications car il propose un bon compromis entre la quantité et la longueur des *reads* permettant ainsi de cerner précisément les jonctions des exons et l'expression des transcrits. En revanche pour les applications d'assemblage, Roche 454® semble toujours avoir une longueur d'avance grâce à sa production de séquences longues. Quant à SOLiD, il est apprécié dans les applications génomiques du type ChIP-Seq ou encore séquençage des exomes où l'utilisation de très courtes séquences n'est pas un problème (pas d'épissage) alors qu'une bonne couverture (nombre de *reads* qui couvrent la zone) est primordiale. Notre domaine étant le transcriptome, nous ne parlerons par la suite que du séquenceur Illumina® qui présente un bon compromis entre nombre de *reads* et longueur des *reads* comparé à SOLiD qui produit des *reads* trop courts et Roche 454® qui ne produit pas assez de *reads*.

2.1.3.2 Le futur

Actuellement, les analyses du transcriptome par les SHD sont réalisées par séquençage des ADNc correspondant aux ARN. Ces approches présentent plusieurs inconvénients technologiques qui limitent l'interprétation des transcriptomes et leur compréhension biologique. En fait, des ADNc erronés peuvent être créés lors de la transcription inverse liés principalement : i/ à la tendance de la polymérase à générer de nombreuses erreurs à cause d'un rendement variable de la rétro-transcription (RT) selon les enzymes utilisées et les conditions expérimentales [Houseley et Tollervey, 2010] ; ii/ à la possibilité d'un changement de matrice (*template switching*) lors de la synthèse qui conduit à des séquences avec inversion d'orientation ; iii/ à des synthèses indépendantes de l'amorce, à partir d'un amorçage interne provenant de structures secondaires présentes sur l'ARN [Ozsolak *et al.*, 2009]. Par ailleurs, la plupart des techniques transcriptomiques à haut débit séquencent un ADNc double brin, ce qui pose des problèmes pour déterminer le brin sur lequel est exprimé l'ARN.

Parce que presque toutes les technologies d'analyse du transcriptome souffrent aujourd'hui des limitations brièvement résumées ci-dessus, des séquenceurs du futur sont en cours d'expérimentation. Ils proposent de séquencer directement les molécules d'ARN en limitant les étapes de préparation qui engendrent des difficultés liées à la RT, l'amplification, et la fragmentation des ADNc [Ozsolak *et al.*, 2009].

2.2 Différentes techniques du transcriptome par séquençage

La profondeur de séquençage est l'un des principaux avantages d'une étude transcriptomique par SHD par rapport au microarrays (voire tiling arrays) : il n'y a pas de limite critique pour quantifier un transcrit alors que pour les microarrays il faut que le signal soit au dessus du bruit de fond pour considérer une quelconque transcription. Dans une seule expérience de SHD, nous observons des *reads* représentant des transcrits très faiblement exprimés et d'autres très fortement exprimés [Trapnell et Salzberg, 2009].

2.2.1 La DGE

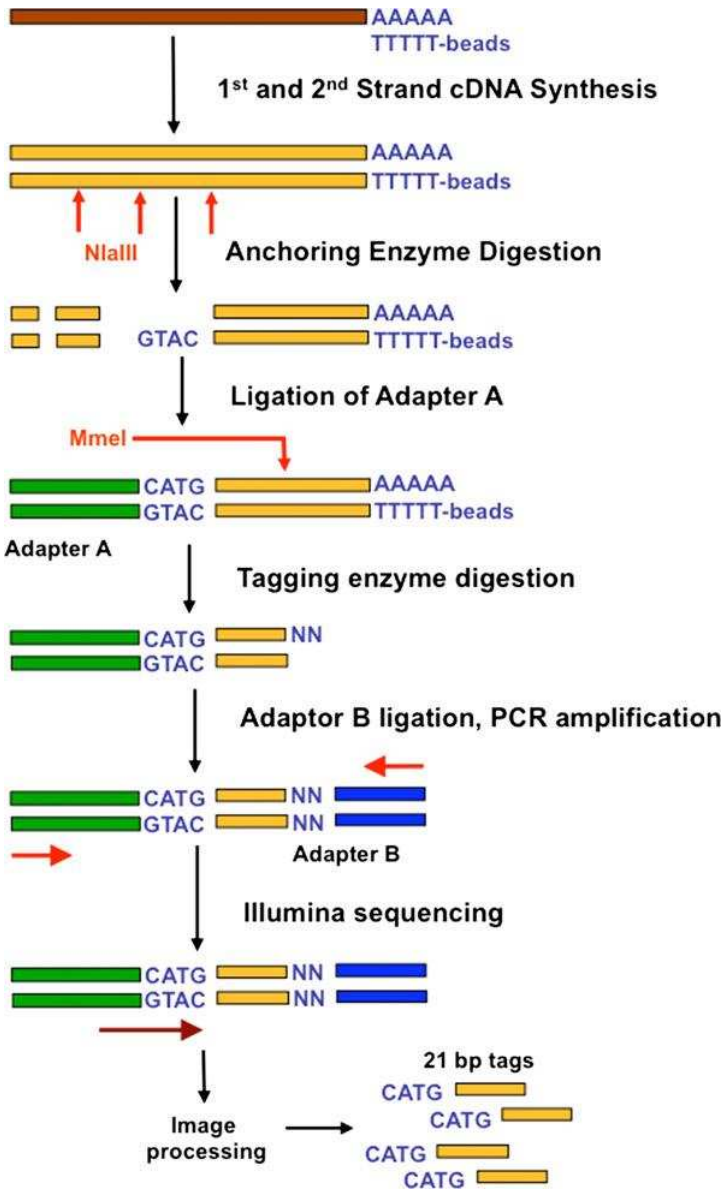
2.2.1.1 Le protocole

Les méthodes numériques pour mesurer le niveau d'expression des gènes, plus connues sous le nom de DGE (Digital Gene Expression), se décomposent en deux catégories : l'analyse en série de l'expression des gènes (SAGE) avec un ancrage de *tags* en fin de transcrit (région 3′) puis l'analyse de la coiffe de l'expression des gènes (CAGE) avec cette fois un ancrage en début de transcrit (région 5′) [Velculescu et Kinzier, 2007; Kawaji *et al.*, 2006]. Le protocole DGE peut se résumer en quatre étapes : i/ les ARNm sont rétrotranscrits en ADNc ; ii/ les ADNc double brins sont synthétisés et subissent une hydrolyse par une enzyme de restriction comme Nla3 ou Sau3a (enzyme d'ancrage) ; iii/ après fixation d'un premier adaptateur, l'hydrolyse par une 2^{e} enzyme, en général Mmel (enzyme de marquage), génère un *tag* situé en 3′ de l'adaptateur ; iv/ un deuxième adaptateur est fixé à l'extrémité 3′ du *tag* et les ADNc ainsi obtenus sont séquencés par SHD après un dernier enrichissement par une réaction en chaîne de la polymérase de son terme anglais PCR (Polymerase Chain Reaction). Le protocole est illustré dans la figure 2.1.

2.2.1.2 Les avantages

En conséquence, les transcrits identiques sont signés par un même *tag* ce qui rend ces méthodes numériques quantitatives et reproductibles : principaux points forts de la DGE [Wu *et al.*, 2010]. De plus, les *tags* peuvent être organisés dans un catalogue, ces données sont disponibles sur internet pour plusieurs espèces (http://cgap.nci.nih.gov/SAGE). Étant donnée la puissance des séquenceurs nouvelle génération, de nouvelles classes de *tags* sont générées incluant des facteurs de transcriptions généralement peu abondants, des *tags* antisens et des séquences introniques qui viennent compléter le catalogue DGE déjà existant [Morrissy *et al.*, 2009]. La précision de la méthode LongSAGE est conservée ce qui permet de quantifier rigoureusement des transcrits rares représentant de nouveaux

FIGURE 2.1 : Une expérience de DGE classique.



Cette figure illustre les différentes étapes pour générer une banque de *tags* DGE. Pour commencer, chaque ARNm (marron) a subi une synthèse en ADNc double brins. Les ADNc (or) sont ensuite digérés par l'enzyme Nla3 (flèches rouges verticales). Seuls les fragments attachés sur la bille à oligo(dT) sont conservés. Puis, l'adaptateur A (vert) est accroché sur le site de restriction pour que l'enzyme Mmel puisse faire son travail de digestion. Dans le sens de la digestion (flèche rouge verticale), un deuxième adaptateur B est ligaturé (bleu) pour marquer l'arrêt de cette digestion 2 *pb* avant la fin de la digestion. Notons que des *primers* de PCR (flèches rouges horizontales) sont introduits sur les adaptateurs A et B pour enrichir les *tags*. Enfin, le séquençage (flèche marron horizontale) est réalisé avec la technologie Illumina® pour obtenir des *tags* DGE de 21 *pb* (longueur maximale pour la technique DGE).

exons ou gènes [Kapranov *et al.*, 2007]. Finalement, le couplage des méthodes numériques avec des SHD se présente comme une solution précise et quantitative pour l'étude des transcriptomes à grande échelle.

2.2.1.3 Les limites

En revanche, la DGE atteint ses limites en terme de longueur de séquences puisque les enzymes de marquage permettent de ne générer que des *tags* de 21 nucléotides. La DGE présente plusieurs limites : tout d'abord, elle ne permet pas de caractériser les transcrits qui ne possèdent pas de sites d'ancrage, même s'ils sont rares. Les transcrits alternatifs ou variants d'épissage qui possèdent le même 3′ UTR ne peuvent être différenciés car ils vont être identifiés par un même *tag* ce qui va impliquer une perte d'information et une surreprésentation du *tag* (le taux d'expression de chaque transcrit est cumulé dans un seul *tag*). Une des caractéristiques de la DGE est la position du *tag* sur le transcrit, il correspond au dernier site en 3′ de la séquence. Cependant, étant donnée la profondeur de la technique, des faux positifs vont être séquencés dans le cas d'une digestion partielle des sites de restriction [Nicolae et Măndoiu, 2011].

Malgré tout, la DGE reste une technique importante (de la transcriptomique) car elle mesure précisément le taux d'expression des transcrits et présente une grande sensibilité comparée à d'autres techniques. Nous nous appuierons sur la robustesse de cet aspect quantitatif de la DGE pour élaborer des méthodes statistiques (voir chapitre 3). Des améliorations ont aussi été proposées, en utilisant par exemple des *tags* plus longs (super SAGE, 26 *pb* proposé dans la technologie SOLID) ou encore en couplant des *tags* produits par deux enzymes différentes (TDGS) [Saha *et al.*, 2002; Rivals *et al.*, 2007].

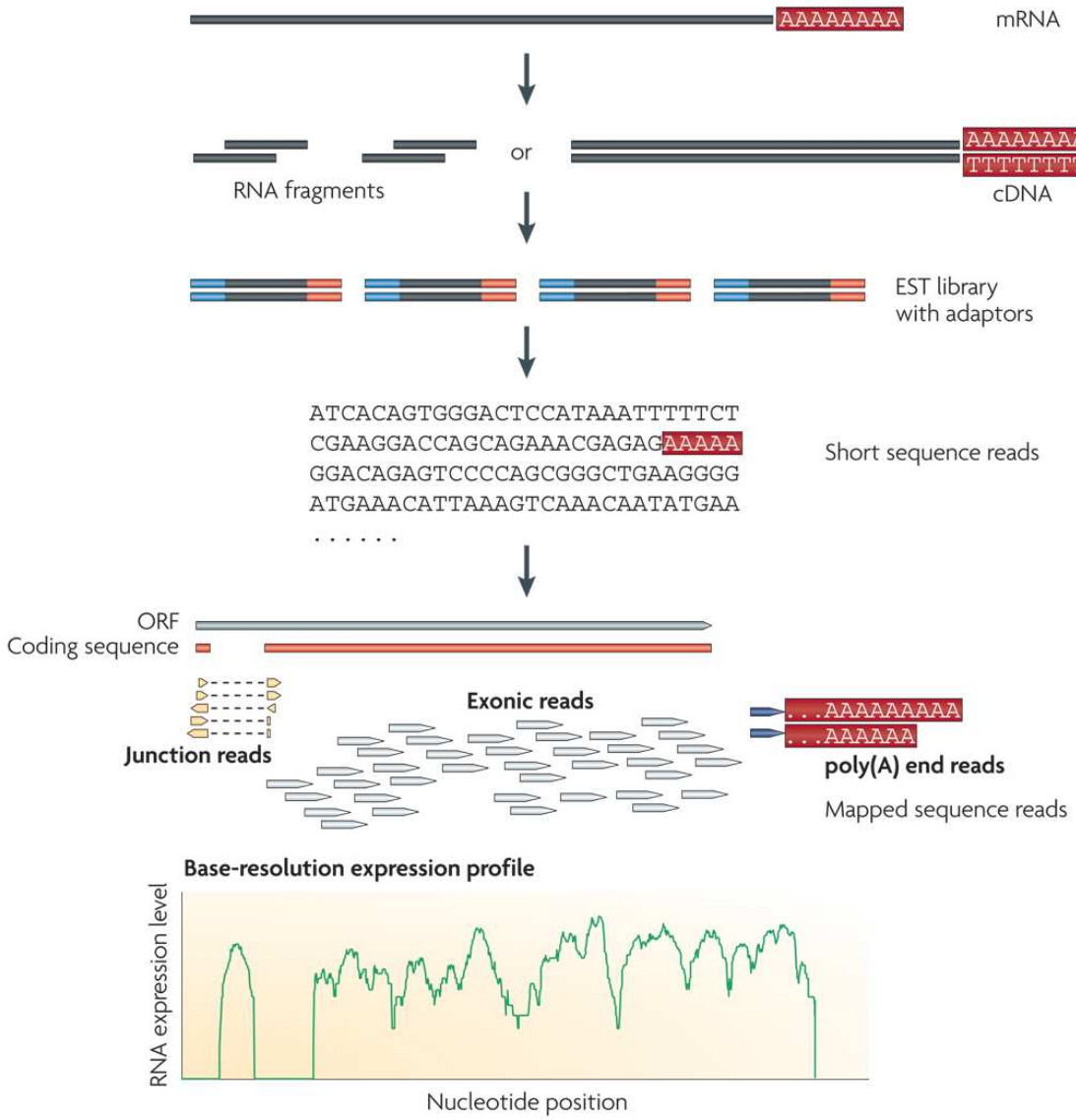
2.2.2 Le RNA-Seq

2.2.2.1 Le protocole

Le RNA-Seq (RNA-Sequencing) a été développé avec les SHD. À partir d'une population d'ARN transformée en ADNc double brins et une étape de fragmentation, un adaptateur est ajouté à chacune des extrémités (FIGURE 2.2). Chaque molécule résultante est, avec ou sans amplification, séquencée par haut débit de façon à obtenir des *reads*. Ces derniers sont généralement longs de 50 *pb* à 2 *kpb*, tout dépend de la technologie de séquençage utilisée. En principe, n'importe quelle technologie SHD peut être adaptée à du RNA-Seq [Holt et Jones, 2008], que ce soit le séquenceur Illumina® [Sultan *et al.*, 2008], le séquenceur SOLiD [Cloonan *et al.*, 2008] ou le séquenceur Roche 454® [Emrich *et al.*, 2007]. Notons que

le séquenceur Illumina® permet de générer un ou deux *reads* à partir de chaque transcrit : on parlera de *single reads* ou *paired reads*.

FIGURE 2.2 : Une expérience de RNA-Seq classique.



Contrairement aux autres, les longs ARN sont tout d'abord fragmentés. Ensuite ils sont tous (les longs aussi bien que les petits) transformés en une collection d'ADNc double brins par un protocole de transcription inversée. Puis, des adaptateurs de séquençage sont ajoutés aux extrémités de chaque fragment, et des *reads* sont ainsi obtenus. Ces derniers proviennent de trois endroits des transcrits : dans les exons, sur les jonctions (ou splicing) et dans les queues poly(A) (ou poly(A) end-*reads*). Les *reads* peuvent ensuite être positionnés sur une séquence consensus (génome ou transcriptome) afin de reconstruire la structure et l'expression des gènes comme illustré en bas de l'image ; un ORF de levure avec un intron est ainsi visible.

2.2.2.2 Les avantages

Le RNA-Seq est la seule technique permettant de couvrir l'intégralité des transcrits, que ce soit en 5′ UTR ou en 3′ UTR mais aussi au niveau des jonctions des différents exons qui les composent. Avec cette possibilité de séquencer l'intégralité des transcrits, le RNA-Seq possède un net avantage sur ses prédécesseurs. Il a d'ailleurs été très vite utilisé chez la souris et l'humain [Nagalakshmi *et al.*, 2008].

À partir d'un échantillon cellulaire, il est possible de séquencer, puis d'aligner des *reads* sur un génome, un transcriptome de référence ou de réaliser un assemblage *de novo*, tout en envisageant de reconstruire la structure complète de tous les transcrits. Le RNA-Seq permet aussi une mesure assez précise du taux d'expression de ces transcrits. D'autre part, la profondeur de séquençage est telle que nous avons la possibilité de retrouver la structure de transcrits rares, ou faiblement exprimés, qui pourraient être impliqués dans certaines pathologies.

2.2.2.3 Les limites

Cependant, même si cette nouvelle technique semble sans limites pour annoter des transcriptomes, elle présente quelques faiblesses. Pour commencer, la fragmentation des ARN peut compliquer la quantification et la précision. En effet alors que les petits ARN tels que *microRNAs (miRNAs), Piwi-interaction RNAs (piRNAs), short interfering RNAs (siRNAs)* sont directement séquencés, les plus longs doivent être fragmentés en petits morceaux de $200-500$ *pb* pour être compatible avec les SHD.

Le protocole de fragmentation peut se faire à deux niveaux : la fragmentation des ARN (hydrolyse et nébulisation) ou la fragmentation des ADNc (traitement par DNase I et ionisation). Chacune des deux méthodes introduit des biais à sa façon. La fragmentation des ARN permettrait de corriger les biais en 3′ des transcrits obtenus à partir de fragments de l'ADNc [Mortazavi *et al.*, 2008]. En effet, la transcription inverse favorise la transcription des régions 3′ au détriment des extrémités 5′ des ARN. Cependant l'uniformité dans la couverture des transcrits quelle que soit la méthode est difficile à obtenir ce qui affaiblit l'aspect quantitatif du RNA-Seq [Wang *et al.*, 2009]. Néanmoins, des améliorations sont actuellement proposées afin de corriger cette hétérogénéité d'expression le long des transcrits [Wilhelm *et al.*, 2010; Roberts *et al.*, 2011]. Une autre considération clé concerne la fabrication de banques de données RNA-Seq orientées. Rappelons que dans le protocole standard, des ADNc double brins sont séquencés. Par conséquent pour un transcrit qui à la base est orienté, nous produisons des *reads* sur les deux brins de son ADNc. Construire des banques orientées aurait l'avantage de conserver l'information sur l'orientation des trans-

crits [Parkhomchuk *et al.*, 2009; Cloonan *et al.*, 2008]. Cet argument est important pour annoter des gènes, spécialement lorsqu'on recherche des transcrits qui sont antisens aux gènes (souvent non-codants) [Quéré *et al.*, 2004; Morris, 2009].

En revanche, il est particulièrement laborieux de construire ce type de banque à cause du nombre d'étapes supplémentaires lors de la préparation, étapes qui peuvent engendrer de nouveaux biais [Wang *et al.*, 2009]. À ce jour, la plupart des expériences ont été conçues à partir des procédures de RNA-Seq non-orientés.
Le RNA-Seq semble être une solution d'avenir pour l'annotation des transcriptomes parce qu'il est capable de reconstruire un transcrit dans son intégralité, tout en restant quantitatif. Cependant cette étape de reconstruction rend la méthode moins sensible que la DGE pour la quantification des taux d'expression. Pour cette dernière le transcrit est représenté par un *tag* alors que pour le RNA-Seq plusieurs *reads* couvrent un même transcrit.

2.2.2.4 Croiser la DGE et le RNA-Seq

En conclusion, les techniques DGE et RNA-Seq semblent être complémentaires pour l'annotation des transcriptomes car les faiblesses de l'une semblent combler celles de l'autre :

1. Le protocole DGE conserve toujours l'orientation des transcrits, par conséquent la construction de banques non-orientées avec le RNA-Seq (beaucoup moins contraignantes) ne devient plus un problème puisque l'orientation est donnée par le *tag* DGE.
2. Le problème de coupure des régions 3' du transcrits par RNA-Seq est corrigé en présence du *tag* DGE car il est positionné sur la dernière occurrence du site de restriction en 3' du transcrit.
3. L'information limitée que la DGE apporte sur la structure des transcrits est directement comblée par l'information exhaustive apportée par le RNA-Seq.
4. Chacun des transcrits pourrait être organisé dans une base de données avec comme identifiant le *tag* DGE et sa faculté d'ancrer un transcrit toujours au même endroit.
5. La surreprésentation des *tags* DGE induite par le regroupement de transcrits alternatifs peut être théoriquement corrigée grâce aux *reads* RNA-Seq puisqu'ils identifient tous les variants d'épissage.
6. Enfin, les éventuelles imprécisions de quantification du RNA-Seq sont directement amorties par le taux d'expression du *tag* qui est représentatif de celui du transcrit (à condition de l'avoir bien identifié à l'aide du point n°5) ce qui facilite les comparaisons entre expérimentations.

2.2.3 Le ChIP-Seq

L'immunoprécipitation de la chromatine (ChIP) [Collas et Dahl, 2008] est une méthode qui permet de déterminer les sites de liaison à l'ADN de protéines particulières. Elle donne accès à une représentation des interactions protéine–ADN qui ont lieu dans le noyau de la cellule vivante ou dans les tissus. Cette technique est rapidement devenue une méthode de choix pour l'identification à grande échelle des interactions facteurs de transcription-ADN, ou aussi pour la caractérisation des différents états de la chromatine en suivant les modifications des histones. Le ChIP-Seq (ChIP-Sequencing) est une technique permettant d'identifier ces sites de liaison de l'ADN [Mardis, 2007; Boyle *et al.*, 2008; Kharchenko *et al.*, 2008]. Les fragments du génome enrichis au cours de la procédure de ChIP sont soumis aux SHD. Ces derniers génèrent des *reads* qui peuvent être localisés sur le génome de référence afin d'identifier les zones enrichies, c'est-à-dire les régions qui contiennent les sites de liaisons. Cependant, les SHD produisent des erreurs de séquences ce qui induit des fausses localisations [Dohm *et al.*, 2008a].

Par conséquent,au niveau des régions concernées pour valider des liaisons, il faut une couverture minimale obtenue avec une profondeur de séquençage adaptée au génome étudié. Une possibilité pour mesurer ce minimum requis en terme de couverture est d'intégrer un contrôle négatif à l'expérience, puis de comparer sa couverture aux autres données séquencées (en utilisant par exemple un anticorps qui ne reconnaît aucun facteur de transcription, ou encore un type cellulaire qui n'exprime pas le facteur de transcription).

2.2.4 Le WGSS

Parallèlement aux principales techniques d'études des transcriptomes et de la régulation de la transcription que nous venons de décrire, de nombreuses méthodes d'analyses des génomes ont vu le jour avec l'apparition des SHD. Parmi celles-ci, le WGSS (Whole Genome Shotgun Sequencing) qui consiste à séquencer massivement les fragments de l'ADN d'un génome.

La technique est beaucoup plus simple à mettre en œuvre que les deux précédentes puisqu'il n'y a pas d'étape de rétro-transcription des ARN en ADNc double brins. En fait, en ayant directement de l'ADN double brins au départ, il suffit de reprendre les étapes du protocole du RNA-Seq à partir de l'étape de fragmentation. Cette technique est surtout utilisée pour séquencer des génomes. D'ailleurs, l'idée de séquencer directement l'ADN n'est pas récente puisqu'elle a déjà été utilisée avec la technologie Sanger pour séquencer et assembler complètement la première séquence du génome humain [ConsortiumInternational, 2004]. Néanmoins depuis le haut débit, cette technique est plus largement utilisée avec

notamment le projet de séquençage 1 000 *génomes* [Via *et al.*, 2010]. L'intérêt de séquencer le plus de génomes possibles (sains ou malades) rend l'annotation des transcriptomes plus simple et plus complète. En effet, rappelons que l'annotation des transcriptomes dépend beaucoup de la qualité de la séquence du génome d'origine.

2.2.5 La nouvelle dimension de la transcriptomique

Comparées aux méthodes basées sur les puces à ADN (cf section 1.2 du chapitre 1), des techniques comme la DGE, le RNA-Seq, le WGSS ou encore le ChIP-Seq, offrent la possibilité de découvrir de nouveaux transcrits car se sont des méthodes ouvertes. Comparées aux tiling arrays, elles permettent une meilleure garantie de détecter ces nouveaux transcrits car ils sont très souvent peu exprimés. L'intervalle de détection (*dynamic range*) de ces techniques est largement plus étendu que celles des méthodologies basées sur l'hybridation. Des études récentes avec l'appui des méthodes de haut débit ont permis de découvrir un grand nombre de nouvelles régions transcrites chez la souris [Mortazavi *et al.*, 2008], l'humain [Velculescu et Kinzier, 2007] et d'autres génomes.

D'un point de vue biologique, la découverte de ces nouveaux transcrits est d'un intérêt primordial. Pour la plupart, ils ne semblent pas coder pour des protéines et leur fonction reste dans l'ensemble difficile à déterminer [Rozowsky *et al.*, 2007; Mercer *et al.*, 2009]. D'un côté, s'il est actuellement admis que de multiples classes d'ARNnc participent aux mécanismes essentiels de la machinerie cellulaire comme l'inactivation spécifique de l'expression de certains gènes par les microARN [Amrouche *et al.*, 2011], les connaissances sur leur nombre, leur localisation et leur rôle restent partielles. C'est le cas notamment des longs ARN non codants (ARNlnc) même si des études très récentes ont démontré des fonctions clés dans la régulation de l'expression des gènes [Rinn *et al.*, 2007; Gupta *et al.*, 2010; Derrien et Guigó, 2011; Morris, 2009].

En fait, les transcriptomes semblent beaucoup plus complexes que nous l'imaginions. Ajoutons à cela les nombreux variants d'épissage dont l'implication dans des maladies telles que les cancers a déjà été mise en évidence [Sammeth *et al.*, 2008; Dutertre *et al.*, 2010].
Le haut débit va offrir de nouvelles directions pour mieux comprendre les transcriptomes et développer les applications biologiques.

2.3 Structures d'indexation et haut débit

L'émergence des SHD transforme progressivement la science des génomes. Le séquençage massif est maintenant employé pour mesurer divers processus biologiques à l'échelle du génome complet. Les avantages technologiques et le faible coût du séquençage mettent ces approches expérimentales à la portée de nombreux projets de recherches. Cependant, ces technologies ne cessent d'évoluer avec des productions de données qui sont de plus en plus longues et en plus grande quantité générant ainsi des problèmes liés à leur traitement. De ce fait, pour mieux exploiter ces données, il faut mieux les organiser afin d'accéder le plus rapidement possible aux informations qui nous intéressent. D'ailleurs, les structures de données informatiques, permettant de stocker et d'organiser les données, sont déjà utilisées dans quasiment tous les outils, que ce soit pour le *mapping* (section 2.4), pour la détection de jonctions (section 2.6) ou encore pour la détection des erreurs et du polymorphisme génétique (section 2.5). Dans cette section nous détaillons les principales structures qui indexent de l'ADN : les structures d'indexation classiques et compressées.

2.3.1 Les structures d'indexation classiques

Généralement, les structures d'indexation quelles qu'elles soient, cherchent à indexer les facteurs d'un texte. Dans un premier temps, le choix des suffixes semblait judicieux. En effet, un texte possède un nombre de facteurs de l'ordre de n^2 alors qu'il ne contient qu'un nombre linéaire de suffixes. Or, dans des structures comme les arbres ou les tables des suffixes, il est possible d'accéder simplement aux préfixes de chacun des suffixes, c'est-à-dire aux facteurs du texte.

Néanmoins, dans une application de SHD, on ne va pas s'intéresser à tous les facteurs du texte mais seulement à ses k-mers, c'est-à-dire tous ses facteurs de longueur k. D'ailleurs dans cette optique, une table de hachage permet aussi de ranger efficacement les facteurs d'une taille fixe d'un texte (voir chapitre 4).

2.3.1.1 Les arbres des suffixes

L'arbre des suffixes est une structure d'indexation encore très utilisée, notamment pour la recherche de répétitions dans une séquence génomique mais aussi pour de nombreuses autres applications en algorithmique du texte [Gusfield, 1997]. En effet, il permet, de rechercher n'importe quel motif, de longueur m, dans un texte de longueur n en temps $O(m)$. Ainsi, la recherche prend un temps qui dépend uniquement de la longueur des motifs.

Définition 2.1. *Soit T le texte à indexer, alors l'arbre des suffixes de T doit remplir les deux conditions suivantes :*

i/ Les feuilles de l'arbre contiennent un numéro qui correspond à la position de début du suffixe dans T.

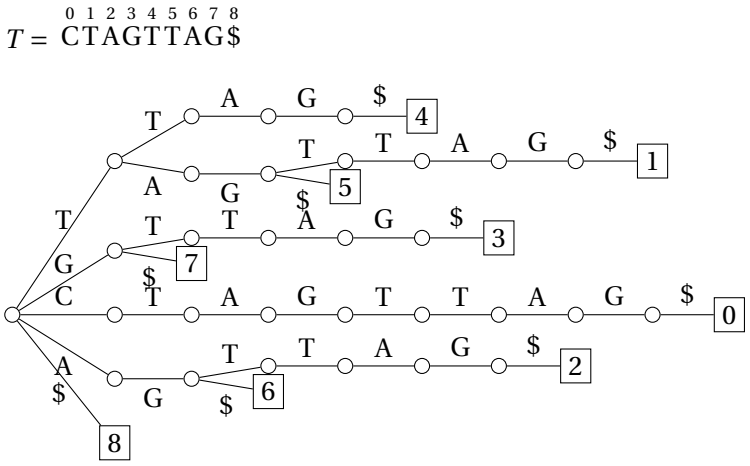
ii/ Les branches peuvent être étiquetées de différentes manières :

- *par une chaîne de caractères de longueur supérieure ou égale à 1,*
- *par un couple (p, l) qui correspond à la sous-chaîne commençant à la position p, de longueur l, dans T.*

iii/ La première lettre des étiquettes de chaque branche est différente en sortie d'un nœud de l'arbre.

En général, on ajoute un terminateur à la fin du texte pour éviter que certains suffixes ne se terminent sur des nœuds de l'arbre, c'est-à-dire un caractère spécial non présent dans le reste du texte. Dans notre exemple \$ est notre terminateur.

Exemple 2.1. L'arbre des suffixes.



Les arbres compacts des suffixes ont été introduits pour réduire une partie de l'espace mémoire utilisée par les arbres des suffixes[McCreight, 1976]. Le principe général est de factoriser les nœuds des arbres des suffixes qui ne contiennent qu'un seul fils.

Définition 2.2. *Un arbre compact des suffixes doit respecter les deux propriétés suivantes :*

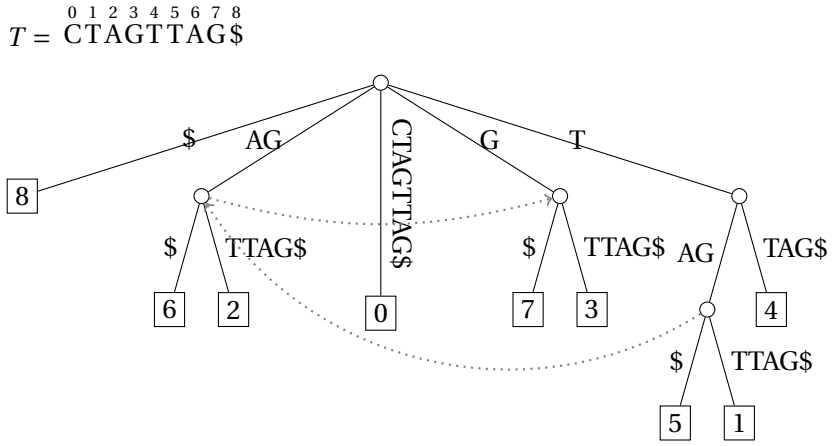
i/ Chaque nœud interne possède au moins deux fils.

ii/ Toutes les étiquettes des branches sortant d'un même nœud commencent par une lettre différente.

Par la suite, l'algorithme de construction des arbres compacts des suffixes a été amélioré avec l'utilisation de liens suffixes [McCreight, 1976; Ukkonen, 1995]. Ce sont des liens

qui vont du nœud représentant le facteur a·u, avec $a \in \Sigma$ et $u \in \Sigma^*$, au nœud représentant le facteur u, s'il existe. Dans l'exemple 2.2, les liens suffixes sont représentés par des flèches en pointillés (les flèches dont le nœud destination est la racine ne sont par convention pas représentées). Nous pouvons observer dans la figure de l'exemple 2.2 que les liens suffixes permettent de cerner les parties communes de T (trois sous-arbres possédant des branches avec les mêmes étiquettes). En conséquence, les liens suffixes permettent à la fois de faciliter la construction de l'arbre compact des suffixes et de réduire l'espace mémoire utilisé par la structure [Ukkonen, 1995].

Exemple 2.2. L'arbre compact des suffixes.



Rechercher un motif M dans T. La recherche de M est relativement simple. En partant de la racine de T, il faut suivre la branche dont l'étiquette commence par la première lettre de M. En suivant cette branche, on arrive alors à un nouveau nœud. Puis on recommence le processus en continuant de lire le motif. Ainsi, le processus s'arrête dans deux conditions. La première lorsqu'on ne peut plus descendre dans l'arbre par la lettre lue dans M. Dans ce cas, il n'y a pas d'occurrence du motif dans le texte. La deuxième lorsqu'on a pu lire l'intégralité de M. Dans ce cas, si on s'arrête sur un nœud de l'arbre (ou au milieu d'une branche pour l'arbre compact), alors le nombre de feuilles présentes dans le sous-arbre correspond au nombre d'occurrences de M dans T (chaque feuille contient la position du début de l'occurrence dans T) ; sinon on s'arrête forcément sur une feuille et il n'y a qu'une seule occurrence de M dans T (qui se trouve à la position indiquée par le numéro de la feuille).

Par exemple, si l'on recherche le motif AG dans l'arbre T de l'exemple 2.2, on s'arrête sur

un nœud qui contient deux fils qui ont pour valeurs 2 et 6. Ainsi, il y a deux occurrences de AG dans T aux positions 2 et 6.

Rechercher des répétitions de M dans T. Une propriété de ce type d'arbre est la possibilité de détecter les facteurs qui apparaissent plusieurs fois très rapidement. En effet, si la lecture de M se termine sur un nœud, alors le nombre de fils sortant du nœud indique le nombre d'occurrences de M. Ainsi nous sommes capables de faire cette requête en temps linéaire de la taille de M. Ensuite, pour récupérer les positions des répétitions dans T, il faut récupérer la valeur de chacune des feuilles des fils concernés. Ainsi, on peut voir dans l'exemple 2.2 que le facteur AG est répété et apparaît deux fois dans T alors que le facteur TTA n'apparaît qu'une seule fois dans T.

Finalement, l'arbre des suffixes s'avère être une solution efficace pour la recherche de motifs dans un texte. Cependant, cette solution est très coûteuse avec une consommation mémoire de $10n$ octets en moyenne et jusqu'à $20n$ dans le pire des cas (n étant la taille de T) [Kurtz, 1999]. Cette contrainte empêche de gérer des grands ensembles de données tels que des *reads* de SHD ou encore des génomes de mammifères. Par exemple, le génome humain (hg19) qui contient environ 3 milliards de nucléotides devrait occuper 45 GO en mémoire [Kurtz, 1999].

2.3.1.2 Les tables des suffixes

En informatique, le stockage des nœuds des arbres consomme beaucoup de mémoire, notamment à cause de la sauvegarde des liens entre les pères et les fils. Une structure plus légère (sans ces liens) mais qui reste basée sur la même idée : la table des suffixes [Manber et Myers, 1990].

Définition 2.3. *Soit T un texte à indexer, alors une table des suffixes s'obtient en deux étapes :*
i/ L'ensemble des suffixes de T sont triés par ordre lexicographique.
ii/ Les positions des suffixes triés sont enregistrées dans une table.

Exemple 2.3. La table des suffixes.

$$T = \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8}{\text{CTAGTTAG\$}}$$

8	6	2	0	7	3	5	1	4
\$	A	A	C	G	G	T	T	T
	G	G	T	\$	T	A	A	T
	\$	T	A		T	G	G	A
		T	G		A	\$	T	G
		A	T		G		T	\$
		G	T		\$		A	
		\$	A				G	
			G				\$	
			\$					

Rechercher un motif M dans un arbre T. Du fait que tous les suffixes sont triés, une recherche par dichotomie d'un motif M dans la table est réalisable. Cependant, une telle recherche prend plus de temps qu'un simple parcours d'un arbre. La procédure de recherche n'est donc pas optimale comparée à celle sur les arbres des suffixes. Pour remédier à ce problème, il est possible d'utiliser une table additionnelle : la table LCP (Longest Common Prefixes). Cette table stocke la longueur du plus long préfixe commun entre deux suffixes consécutifs de la table des suffixes. Avec ces informations supplémentaires, les comparaisons redondantes sont évitées ce qui permet d'exécuter une recherche plus rapide mais toujours moins qu'avec les arbres des suffixes ($O(logn + m)$ contre $O(m)$ pour les arbres).

Finalement, les tables des suffixes sont une alternative aux arbres des suffixes. Elles sont moins gourmandes en mémoire puisque ne sont stockées que les positions des suffixes. En contre partie, le temps de construction est plus long du fait que les suffixes sont triés au préalable [Manber et Myers, 1990] pour l'algorithme original ou (voir https://code.google.com/p/libdivsufsort/wiki/SACA_Benchmarks) pour des algorithmes plus récents. Cependant, il faut quand même compter au minimum un entier pour enregistrer chaque suffixe dans la table. Par conséquent, l'indexation du génome humain dans une table des suffixes n'est pas raisonnable.

2.3.2 Les structures compressées

Il semblerait que les structures construites sur le principe d'enregistrer l'ensemble des suffixes d'un texte ne soit pas la solution la plus appropriée pour enregistrer une grande quantité de données. Récemment des structures d'indexation compressées ont été

élaborées afin de pouvoir organiser massivement des informations. Ce type de structure est essentiellement utilisé dans le haut débit pour l'indexation complète des génomes (voir section 2.4.1.2). Les structures d'indexation compressées qui existent à ce jour sont divisées en deux familles : les index utilisant la compression LZ-78 et les index utilisant un échantillonnage de la table des suffixes [Salson, 2009]. Dans tous les cas, cela permet de diminuer considérablement l'espace mémoire requis pour indexer un texte, tout en conservant la possibilité de lire son contenu sans le décompresser. Par exemple, une méthode d'échantillonnage telle que le *FM-Index* rend possible le chargement et la lecture du génome humain (hg19) en mémoire avec moins de 2 GO. Cependant, ce gain en mémoire se perd essentiellement en temps avec des requêtes qui peuvent prendre entre 100 et 1 000 fois plus de temps qu'avec les structures classiques[Ferragina *et al.*, 2009].

En conclusion, toutes les structures peuvent défendre leur intérêt avec d'un côté, une construction et un accès rapide aux données et de l'autre, un gain considérable en espace mémoire. D'ailleurs, nous allons voir par la suite que toutes vont être utilisées à des fins différentes.

2.4 Outils de *mapping*

En plus du problème d'indexation, il faut en parallèle extraire l'information des données SHD. Or, les techniques de séquençage évoluent et permettent la découverte d'unités transcrites de plus en plus complexes à caractériser. Le *mapping* est généralement la première étape du processus d'analyse. Cette étape consiste à localiser des *reads* sur un génome de référence, c'est-à-dire trouver la position d'origine du *read*. À cause de la quantité de données générées, il est nécessaire de développer des outils bioinformatiques adéquats qui sont capables de faire ce positionnement rapidement et sans ambiguïté. Notons aussi, que les SHD produisent un nombre d'erreurs à traiter qui est proportionnel à la quantité de données séquencées. Nous sommes donc confrontés à la fois à des problèmes méthodologiques et algorithmiques.

Les SHD évoluent continuellement avec des *reads* qui peuvent être plutôt courts avec une longueur de 50 – 100 *pb* et d'autres beaucoup plus longs d'une longueur de 150 *pb* à 2 *kpb*, puis une quantité de données qui augmente considérablement, jusqu'à atteindre plusieurs milliards de *reads* [Metzker, 2010]. Nous avons décidé d'organiser les outils de *mapping* en suivant les stratégies d'indexation (FIGURE 2.3). La liste des outils n'est pas exhaustive et des informations supplémentaires sont disponibles sur *SEQanswers* (http:

//seqanswers.com/). Notons que certains de ces outils de *mapping* ont été supplantés par des index compressés : SOAPv1 [Li *et al.*, 2008c] remplacé par SOAPv2 [Li *et al.*, 2009b] ou MAQ [Li *et al.*, 2008b] par BWA [Li et Durbin, 2009].

2.4.1 Les outils pour les séquences courtes

2.4.1.1 Indexation des *reads*

Tout d'abord, il y a les outils qui sont fournis directement avec les SHD comme le logiciel ELAND (Cox, unpublished software) de la technologie Illumina® ou SeqMap [Jiang et Wong, 2008]. Ces derniers associent les *reads* à des graines espacées en utilisant la méthode du « pigeonhole »[1], ensuite ces graines sont indexées dans une table de hachage. Rappelons qu'une graine de taille k représente une portion consécutive du *read* de taille k qui se localise sur le génome. Une graine espacée de taille k ne représente pas forcément une portion consécutive (il peut y avoir des différences entre le *read* et le génome). Nous reviendrons sur le principe de graine et filtration en section 2.4.2.

Exemple 2.4. Le principe du « pigeonhole ».
Pour bien comprendre le principe, prenons deux séquences de longueur 10 avec une graine de longueur 8 :

	0	1	2	3	4	5	6	7	8	9
r_1	a	a	c	a	g	c	t	t	g	t
r_2	a	a	c	t	g	c	t	t	c	t
s	#	#	#	–	#	#	#	#		

Bien que les nucléotides en position 3 et 8 soient différents, les deux séquences sont indexées par la même graine ### – #### dans la table de hachage.

Notons que les graines de ELAND sont construites en fonction des erreurs de séquences (la moyenne des qualités par position). Une graine peut être pondérée par le nombre de # qui la constitue. La graine de l'exemple 2.4 a un poids de 7.

Ensuite lors de l'alignement, le génome est haché à la volée ce qui rend l'algorithme beaucoup plus rapide et moins consommateur que les alignements classiques comme BLAT et BLAST. Dans ces derniers, des longues séquences génomiques sont indexées à l'aide de graines simples, en utilisant le principe du *seed and extend* (cf. sous-section 2.4.2.2), et des alignements dynamiques du type Smith-Waterman sont utilisés

1. On a plus de *reads* que de graines alors que chaque *read* doit être associé au moins à une graine.

[Kent, 2002; Altschul *et al.*, 1990]. Néanmoins, les séquences d'une longueur supérieure à 32 *pb* ne sont pas supportées et au maximum deux substitutions sont tolérées. De plus, la recherche approchée implique une analyse statistique assez complexe afin d'estimer la probabilité qu'une séquence approximative identifie la vraie localisation du *read* sur le génome (dans le chapitre 3 une étude similaire est réalisée sur de la recherche exacte de courtes séquences). Le logiciel libre MAQ est fondé sur le même principe avec quelques améliorations [Li *et al.*, 2008b]. En effet, celui-ci gère des *reads* plus longs (jusqu'à 63 *pb*) et présente une phase de vérification supplémentaire pour limiter les artefacts dus aux erreurs de séquences. Par défaut, la méthode garantit un positionnement des 28 premiers nucléotides des *reads* avec au maximum deux substitutions, le reste de la séquence étant extrapolé. La phase de vérification est introduite après le *mapping*, elle consiste à filtrer les séquences potentielles par un score en faisant la moyenne de la qualité des substitutions dans la partie extrapolée. Ainsi, MAQ présente une analyse plus spécifique et plus accessible que le logiciel privé ELAND.

2.4.1.2 Indexation du génome

Les solutions précédentes traitent les *reads* au préalable avant de les localiser sur le génome, la quantité de mémoire utilisée va directement dépendre de la quantité de *reads* à indexer. Alors que les SHD sont réputés pour produire des quantités gigantesques de *reads*, les tables de hachage le sont plutôt pour être gourmandes en mémoire. Ce double effet impacte les solutions qui vont finir par atteindre une limite.

Maintenant, supposons le cas inverse : rechercher des *reads* sur un génome indexé. En effet, le génome contient une quantité impressionnante de données auxquelles nous voulons accéder le plus rapidement possible. Imaginons le génome comme un annuaire avec un accès direct (un index) sur les séquences recherchées. Il y a dix ans, une équipe avait eu l'idée originale d'adapter un index pour une longue séquence d'ADN en la découpant en petites portions continues de taille k (suffisamment grande) pour créer le logiciel SSAHA [Ning *et al.*, 2001].

Après l'apparition des SHD, d'autres se sont inspirés de cette idée dans SOAPv1 [Li *et al.*, 2008c], ils indexent le génome (plutôt que les *reads*) à l'aide de tables de hachage et consultent les séquences à la volée. Cette consultation inversée permet de localiser les séquences cinq fois plus rapidement que MAQ et offre les mêmes résultats en terme de prédiction pour des séquences ≤ 60 *pb*. Malheureusement cet avantage a un coût, celui de l'espace mémoire nécessaire pour loger la table de hachage du génome. Celle-ci peut

atteindre jusqu'à 20 fois l'espace utilisé par le texte d'origine. Ainsi, pour que cette idée soit utilisable en pratique, il faut indexer chacun des chromosomes indépendamment.

L'utilisation d'arbres (ou de tables) des suffixes permet d'identifier précisément les facteurs communs entre *reads* et génome. Ces structures ont l'avantage de regrouper les facteurs communs d'une même séquence dans un même endroit ce qui permet d'économiser du temps de calcul sur d'autres structures comme les tables de hachage où chaque copie de la séquence demande un traitement différent. Mais ces structures sont très coûteuses en mémoire : pour les arbres, en plus des suffixes, il faut aussi enregistrer les informations relatives à chaque nœud de l'arbre telles que les adresses du père et des fils.

Une méthode basée sur les arbres des suffixes étendus a été conçue pour faire de la recherche approchée (substitutions et indels) de *reads* sur un génome [Hoffmann *et al.*, 2009]. Dans ce travail, le génome est indexé sans être compressé dans un arbre pour regrouper les portions qui sont communes. Sur un génome tel que l'humain, il faut traiter les chromosomes indépendamment pour une consommation mémoire raisonnable (voir section 2.3.1), ce qui reste problématique pour certaines applications telles que la recherche de gène de fusion.

Afin de pouvoir interroger le génome en un seul bloc, la solution a été d'opter pour une structure d'indexation compressée [Salson, 2009] qui a l'avantage de pouvoir indexer tous les chromosomes à la fois tout en étant beaucoup moins coûteuse en mémoire [Langmead *et al.*, 2009; Li et Durbin, 2009]. En conséquence, il est possible d'indexer un génome compressé en mémoire tout en conservant une lecture sans décompression. Par contre, Bowtie n'est pas du tout adapté à des *reads* longs : il ne permet de localiser que des séquences avec trois substitutions au maximum sur le génome à l'aide d'un algorithme de *backtracking*. Cette recherche approchée se fait en temps exponentiel de la taille de chaque *read*.

Néanmoins, pour une recherche de $8.84M$ de données de 35 *pb* sur le génome humain, Bowtie procure un gain de 100X en temps par rapport à SOAPv1 pour un index qui utilise seulement $2GO$ en mémoire, en plus la qualité des localisations est similaire. D'ailleurs, une version SOAPv2 a été développée en adaptant ce type d'index compressé [Li *et al.*, 2009b; Li et Durbin, 2009] et l'équipe de MAQ a aussi adopté ce type d'index avec deux algorithmes : i/ un premier, BWA [Li et Durbin, 2009], pour des séquences courtes (jusqu'à $\simeq 100$ *pb*) qui autorise quelques substitutions et un gap ; ii/ un second algorithme, BWA-SW [Li et Durbin, 2010], conçu pour des séquences plus longues, qui effectue un alignement basée sur des heuristiques de *Smith-Waterman-like*. BWA-SW est plus lent et moins précis que BWA, mais sur des séquences plus longues (> 150 *pb*), il est meilleur.

2.4.2 Les approches par filtration et les graines

Avec l'élongation des *reads*, augmente aussi la probabilité qu'ils contiennent une différence, et principalement une erreur, par rapport à la séquence du génome de référence. Il faut donc dépasser les limites des outils qui n'autorisent qu'un petit nombre et types de différences (par ex. une ou deux substitutions pour Bowtie), tout en passant à l'échelle des millions de *reads*.

Le positionnnement de *reads* sur un génome peut être vu comme une recherche de motif approchée dans un texte, avec le *read* pour motif et le génome pour texte. Ainsi, on peut concevoir des stratégies de positionnement efficaces bâties sur des principes de filtrations rapides déjà utilisées pour la recherche de motif. Toutes les méthodes de recherche de motifs approchée rapides adoptent ce principe. La recherche se décompose en deux étapes : i/ la filtration, qui en évaluant un critère simple élimine des positions du texte ne pouvant pas être similaires au motif, ii/ la vérification, qui examine parmi les positions sélectionnées celles où débute un occurrence approchée du motif. Le critère de filtration peut être *avec perte* (lossy) ou *sans perte* (lossless) selon qu'il écarte ou non des vraies occurrences (qu'il aurait dû conserver). En général, le critère de filtration est de complexité linéaire, voire sous-linéaire à calculer, tandis que la vérification peut être quadratique. L'efficacité tient dans la capacité du filtration de la première étape.

Les méthodes de filtration ont aussi été utilisées pour accélérer la recherche exacte de motifs. Pour le *mapping* dans un génome, MPSCAN développe un algorithme de *set pattern matching* basé sur un filtration par la composition en q-grams. La littérature de « stringology » comporte de multiple méthodes de filtration [Pevzner et Waterman, 1995] bien connues en bioinformatique puisqu'elles sont utilisées dans BLAST, FASTA, BLAT, etc [Kent, 2002]. Deux parmi les plus connus et efficaces ont été adaptés au *mapping* de *reads* : d'une part le lemme des q-grams [Owolabi et McGregor, 1988; Jokinen et Ukkonen, 1991] qui fût exploité avec succès pour la recherche multiples de similarités dans QUASAR ou SWIFT [Burkhardt *et al.*, 1999; Rasmussen *et al.*, 2006], et les graines espacées d'autre part [Ma *et al.*, 2002; Li *et al.*, 2003; Burkhardt et Kärkkäinen, 2003]. Ici, nous donnons une courte description des fondements de ces méthodes.

2.4.2.1 Mapping exact : l'algorithme de MPSCAN

MPSCAN est un outil dédié au *mapping* exact d'un ensemble de *reads* sur un génome ou un transcriptome de référence. Il recherche les positions du texte où chaque *read* a une occurrence exacte. Ce type de recherches s'applique lorsque les *reads* sont très courts et donc ne contiennent que peu d'erreurs. MPSCAN se distingue des méthodes pionnières de

mapping car : il recherche tous les reads en une seule passe sur le génome sans indexer ce dernier, et garantit une réponse exacte (toutes les occurrences et rien que des occurrences).

MPSCAN implémente un algorithme original pour rechercher des reads de longueur m. Au lieu de considérer la comparaison par paires de symboles, il teste des égalités de k-mers. L'ensemble des motifs est résumé dans une expression régulière simple, où chaque position contient l'ensemble des k-mers présents dans tous les *reads* à cette position. Pour qu'à un indice du texte commence une occurrence d'un *read*, il faut *qu'à chaque position* de la fenêtre de longueur m, le k-mer appartienne à l'ensemble de k-mers répertoriés dans l'expression régulière. Ainsi, en choisissant bien la longueur k, la probabilité que le filtre conserve un indice est faible. En pratique, les k-mers sont stockés dans des entiers et la recherche de l'expression régulière simple se fait grâce à l'algorithme dit « Backward Non-Deterministic DAWG Matching » [Navarro et Raffinot, 2002], qui exploite le parallélisme d'instruction sur des vecteurs de bit. MPSCAN peut ainsi rechercher des millions de *reads* sur le plus long chromosome humain en quelques minutes et se révéler aussi rapide que Bowtie [Rivals *et al.*, 2009]. L'analyse de la complexité montre que la filtration permet un temps de recherche moyen sous linéaire en regard de la longueur du texte et que cette complexité est optimale [Navarro et Fredriksson, 2004; Rivals *et al.*, 2009].

Concernant l'incidence du choix de la recherche exacte plutôt qu'approchée, notre travail développé au chapitre 3, montre que pour des reads < 30 *pb*, il est plus opportun de rechercher exactement les préfixes de longueur 20 des reads, plutôt que les *reads* complets de manière approchée, du fait de l'augmentation de la probabilité d'erreur avec la longueur des *reads*. En effet, le nombre total de localisations uniques sur le génomes est plus élevé dans le premier cas que dans le second.

2.4.2.2 *Mapping* approché pour les *reads* plus longs

Pour toutes les approches précédentes, un petit nombre de substitutions est accepté, avec quelquefois des indels très courts pour quelques exceptions. Pourtant, la longueur des données générées par les SHD ne cesse d'augmenter, il est donc nécessaire de développer des méthodes performantes capables de détecter des indels plus longs.

Le filtre des q-grams permet la recherche à p erreurs près d'un motif. Dans le pire des cas les erreurs sont distribuées régulièrement dans une occurrence approchée du motif. Ainsi, le sous-mot compris entre deux erreurs est identique entre le motif et le texte. Le lemme du q-gram indique que dans une occurrence approchée ayant $\leq p$ erreurs, il existe au moins un sous-mot exact de longueur $\lfloor m - p + 1/q + 1 \rfloor$. Ce critère de filtration, appelé

seed and extend, est repris notamment dans GASSST et GSNAP [Rizk et Lavenier, 2010; Wu et Nacu, 2010], qui le combine avec un index de tous les q-grams ou k-mers du génome. GASSST est adapté au traitement de *reads* génomiques du type WGSS alors que GSNAP est plutôt adapté à des *reads* transcriptomiques du type RNA-Seq.

On voit que dans ce filtre les q-grams sont les témoins d'une occurrence approchée potentielle. Ils imposent la contrainte que toutes les identités sont adjacentes dans cette fenêtre de longueur q. Par exemple, pour un *read* de 50 *pb* qui contient 3 erreurs de séquence, il n'y a pas de portions plus petites que 12 *pb* qui ne sont pas localisées, par conséquent les graines ont une taille de 12 au maximum [Owolabi et McGregor, 1988; Jokinen et Ukkonen, 1991].

Outre les méthodes précédentes sur les graines qui considèrent le génome, une autre approche qui utilise ces notions de distance mais cette fois-ci en appliquant un traitement sur les *reads* est le comptage par q-gram. SHRiMP et RazerS utilisent ce principe [Rumble *et al.*, 2009; Weese *et al.*, 2009], le premier étant moins précis sur l'alignement, intéressons nous à l'algorithmique du second. Les q-gram sont calculés à partir d'un *code* : la transformation qui permet d'aligner chaque séquence vers un fragment du génome.

Exemple 2.5. Le principe des q-grams.
Pour bien comprendre le principe, examinons le *code* t sur une séquence de longueur 8 avec un fragment du génome de longueur 11 :

	0	1	2	3	4	5	6	7	8	9	10	11
r	a	c	a	$-$	$-$	g	t	c	g	t		
g	a	c	a	t	g	g	$-$	c	g	t	a	t
t	M	M	M	I	I	M	D	M	M	M		

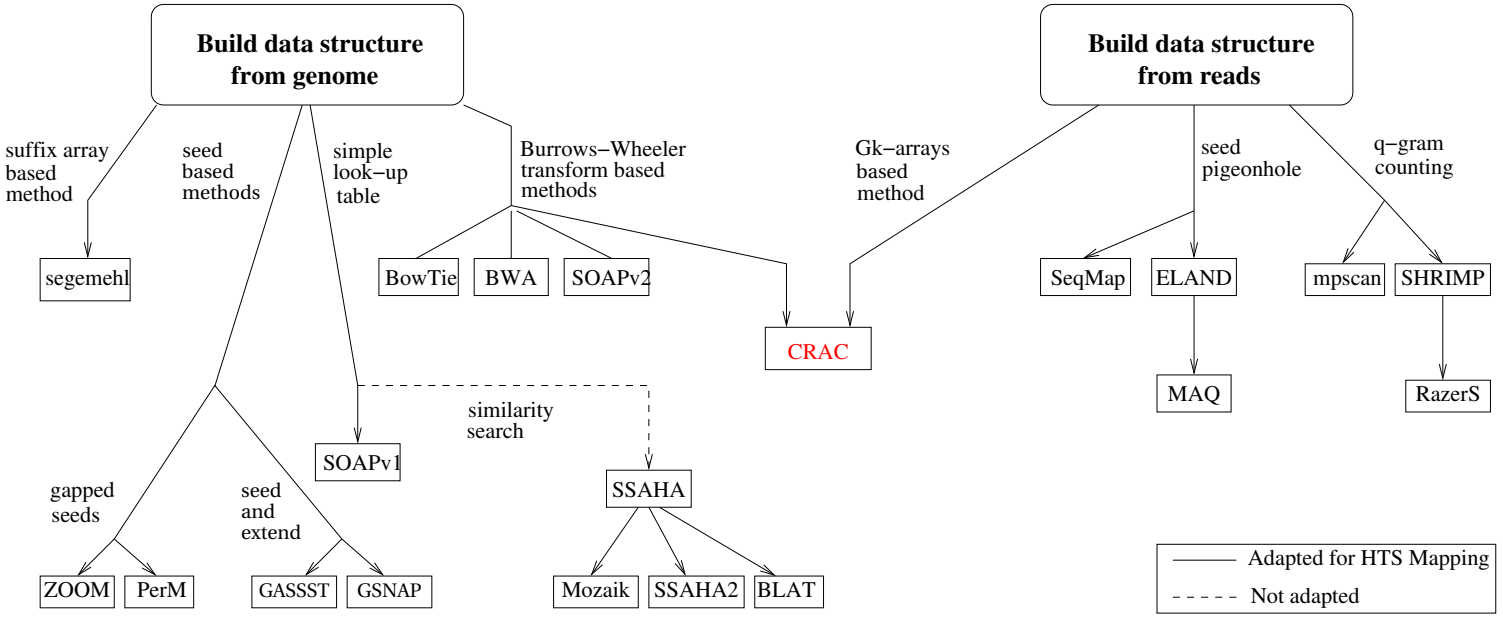
Les q-match sont les portions de longueurs q représentant une séquence identique entre le génome et la séquence. Dans l'exemple ci-dessus, en partant de l'hypothèse que $q = 3$, nous obtenons deux occurrences du 3-gram MMM aux positions 0 et 7 dans le *code*. En pratique, ce qui nous intéresse c'est le meilleur alignement entre une séquence et le génome. RazerS va chercher à maximiser ce critère en utilisant la méthode SWIFT sur chaque *code* [Rasmussen *et al.*, 2006] : il va prédire des alignements potentiels en s'intéressant au comptage des q-matchs du *code*, en d'autres mots il va conserver les alignements qui possèdent un nombre suffisant de q-matchs (en fonction d'un seuil spécifié en paramètre du programme). Ensuite, les alignements restants sont analysés à l'aide de vecteurs de bits pour ne conserver que les meilleurs [Myers, 1999].

Les graines espacées sont apparues pour améliorer encore les outils de recherche de similarité de séquences tels que BLAST, qui utilisaient déjà cette idée d'une graine contiguë témoin d'un match potentiel. Pour cette question, les dissimilarités entre séquences peuvent être telles qu'ils existent des alignements où aucune graine contiguë suffisamment longue ne matche. L'idée à l'origine des graines espacées est de relâcher la contrainte de contiguïté tout en requérant un même nombre d'identité [Burkhardt et Kärkkäinen, 2003]. Une graine espacée est une fenêtre de longueur $w > q$ dans laquelle on obstrue quelques (*i.e.*, $w - q$) positions. Cette dispersion des identités sur une plus grande largeur de fenêtre a permis d'augmenter la sensibilité de la filtration (avec pertes). Cependant, ce gain est balancé par le fait que la conception des graines (qui choisit leur longueur et les positions a obstruer) exige un preprocessing dont la complexité est exponentielle (NP-difficile), quelle que soit la formulation du problème de conception [Kucherov *et al.*, 2006; Ma et Li, 2007; Nicolas et Rivals, 2008].

Les graines espacées spécifiques pour la question du *mapping* ont été conçues et utilisées avec efficacité dans ZOOM et PerM [Lin *et al.*, 2008; Chen *et al.*, 2009]. Cependant, l'indexation des graines espacées est plus complexe et gourmande que celle de simples k-mers. En outre, dans le cas de ZOOM la conception des graines dépend très spécifiquement des paramètres du *mapping*, obligeant l'utilisateur à changer de graines pour des paramètres différents et rendant l'utilisation peu souple [Rivals *et al.*, 2009].

Dans le contexte du *mapping*, les graines espacées ont permis de surcroît de modéliser la dépendance entre positions erronées dans les *reads* et les dépendances entre positions dans le codage couleur des *reads* SOLiD. Cette idée est implémentée dans le logiciel storm [Noé *et al.*, 2010]. Dans cet outil, des graines espacées sont conçues à l'aide de *Iedera* [Kucherov *et al.*, 2006] tenant compte à la fois la variabilité de l'ADN et des erreurs de séquences.

FIGURE 2.3 : Organisation des différents logiciels de *mapping*.



Cet organigramme range les différents outils de *mapping* selon leur type d'indexation. Certains outils organisent le génome, d'autres les *reads*, alors que certains s'occupent des deux. De façon générale, la structure choisie influence les performances : des méthodes sont plus rapides mais d'autres seront moins gourmandes. En outre, la façon d'organiser les informations a une influence sur la sensibilité et la précision des résultats (conférer le chapitre 5).

2.4.3 Les limites du *mapping*

Cependant le *mapping* seul est encore loin de répondre aux attentes des biologistes [Trapnell et Salzberg, 2009]. En effet, certains outils deviennent limités à cause de l'évolution rapide des données SHD. Pour des techniques transcriptomiques telles que le RNA-Seq, positionner des séquences sur un génome ne suffit plus. De savoir qu'il y a $10M$ de *reads* ayant une localisation unique sur un génome est une première information mais on ne peut les traiter un par un : il faut les ranger de façon plus sélective en identifiant précisément ceux qui sont affectés par une cause (erreurs de séquences, SNV, indels, jonctions d'épissage, etc). Ce besoin est devenu incontestable depuis l'apparition du RNA-Seq car les portions séquencées des ARN transcrits permettent non seulement de savoir quels gènes sont activés dans une cellule, mais aussi d'inférer précisément les limites des exons des régions transcrites sans aucune connaissance *a priori*. Ainsi, en exploitant une telle expérience, nous pouvons en théorie découvrir la structure de gènes classiques (exon-intron), mais aussi des ARN non-codants, voire des ARN chimères, c'est-à-dire les transcrits de fusion et les transcrits de trans-épissage intra-chromosomique (chapitre 5).

2.5 Erreurs de séquences et polymorphisme

Les SHD produisent un faible taux d'erreurs pendant le séquençage, par exemple Illumina® peut générer des millions de *reads* avec un ratio moyen d'erreurs inférieur à 1% [Dohm *et al.*, 2008a]. Cependant sur la quantité des *reads* qui sont générés, ces erreurs peuvent complexifier les analyses bioinformatiques comme l'assemblage de séquences, le re-séquençage des génomes et la localisation de séquences sur un génome ou un transcriptome [Schröder *et al.*, 2009].

Lors du positionnement de *reads* sur un génome, si ces derniers sont affectés par des erreurs, alors l'analyse du polymorphisme est plus difficile [Keime *et al.*, 2007]. En effet, nous avons d'un côté des SNV (Single Nucleotide Variant) entre deux individus où tous les *reads* qui les contiennent ne se localisent pas parfaitement sur le génome ; puis de l'autre côté nous avons des erreurs de séquences où les *reads* erronés ne se localisent pas parfaitement sur le génome non plus. Maintenant, imaginons que la partie concernée du génome n'est recouverte que par 2 *reads*. Est-ce un SNV très peu exprimé ou une erreur qui s'est produite sur 2 *reads* sur la même région du génome ?
Deux possibilités sont envisageables :

– Une même erreur a peu de chance de se reproduire sur deux *reads* identifiés au même

endroit sur le génome, sauf pour un transcrit fortement représenté par une quantité exorbitante de *reads*.
- Si deux *reads* se localisent de manière approchée sur le génome, alors qu'au même endroit d'autres *reads* sont parfaitement localisés. Nous pourrions affirmer que les deux premiers *reads* sont erronés et partagent la même erreur. Mais là encore, il est possible d'avoir un allèle bien exprimé qui se localise parfaitement sur le génome et un autre (plus rare) qui contient un SNV.

Finalement, il est souvent compliqué de distinguer une erreur de séquence d'un polymorphisme.

Des équipes se sont attachées à résoudre ce problème selon deux directions de recherches : i/ soit, corriger les erreurs de séquences [Salmela et Schröder, 2011; Schröder *et al.*, 2009] ; ii/ soit, détecter directement les mutations biologiques ponctuelles (SNV, indels courts) [Li, 2011]. Notons que les deux stratégies sont fortement liées puisque si nous arrivons à corriger les erreurs, nous savons écarter les causes biologiques aussi. Cependant, dans la deuxième stratégie il n'y a pas de phase de correction, ce qui rend la procédure bioinformatique moins coûteuse en ressources. En contrepartie la première stratégie est plus précise. En outre, les deux stratégies peuvent être cumulées.

2.5.1 La correction des erreurs

La démarche est la même quel que soit l'outil utilisé : si r est un *read* qui ne contient pas d'erreur, alors il faut chercher tous les *reads* erronés r' qui dérivent de r.
Dans SHREC [Schröder *et al.*, 2009], un *trie* (un arbre ordonné sur l'ordre lexicographique) est utilisé pour identifier r (un *candidat*) et les r' (ses *voisins*). Le principe est le suivant :

1. Un *trie* est construit à partir d'une collection de *reads*. Sur les branches de l'arbre chaque nucléotide est séparé par un nœud et la feuille contient le nombre de fois que le *read* est présent dans la collection. En d'autres termes, la valeur de chaque feuille constitue le poids de la branche qui lui est associée.
2. Les fourches, c'est-à-dire les nœuds qui contiennent plusieurs branches, sont identifiées. Pour chacune de ces fourches, la branche qui a le plus gros poids est considérée comme un *candidat* (un *read* sans erreur) et les autres branches sont, soit ses *voisins* (*reads* erronés), soit d'autres *candidats* (tout dépend du poids de ces branches par rapport à celui du père).

Dans Coral [Salmela et Schröder, 2011], une table de hachage est utilisée. Voici le principe ;

1. Les *reads* sont segmentés en k-mers. Ces k-mers servent de clés dans une table de hachage. Chacune de ces clés a pour valeur un poids : le nombre d'occurrences du k-mer dans la collection de *reads*.
2. Un ensemble de séquences consensus est construit par assemblage des *reads* à l'aide de la table de hachage. Coral part du principe suivant : si les *reads* partagent un même k-mer alors ils ont tous la même position d'origine sur le génome ou transcriptome de référence.
3. Un alignement multiple avec l'algorithme de Needleman–Wunsch [Needleman et Wunsch, 1970] est effectué afin d'aligner les *reads* sur l'ensemble des séquences consensus. Ainsi, lorsque les *reads* chevauchent une substitution ou un indel, en fonction du poids des k-mers, il est possible de distinguer les erreurs des SNV.

2.5.2 La détection des SNV/SNP

Cette idée de détecter les SNV sans corriger les erreurs est adoptée dans le pipeline samtools [Li *et al.*, 2009a]. L'idée générale est d'utiliser les chaînes CIGAR[2] générées par des outils de *mapping* dans un fichier de résultats qui est au format SAM [Li *et al.*, 2009a]. À partir de cette chaîne, les alignements sont reconstruits pour tous les *reads* sur le génome. Cette démarche permet de détecter un ensemble de *reads* qui couvrent une même région du génome atteinte par une substitution ou un indel court. Puis, en fonction de la qualité des séquences (disponible en format fastq pour chaque position des *reads*), il est possible de discerner les erreurs de séquences des causes biologiques.

Cet algorithme *SNP calling* a été quelque peu modifié très récemment, à cause d'une grande quantité de faux positifs générée. La principale source du problème est la présence des indels : les technologies produisent des *reads* plus longs et en plus grande quantité ce qui augmente la probabilité d'en identifier. Ainsi, le *SNP calling* produit fréquemment des erreurs en la présence d'un indel autour d'un potentiel SNV entre génome et *reads* [Li et Homer, 2010]. En effet, les indels sont plus complexes à détecter pour les outils de *mapping* (voir section 2.4) et souvent une insertion ou une suppression courte est confondue avec une substitution. Une solution possible serait de faire un réalignement local des *reads* qui contiennent des indels mais ce processus serait contre productif de la méthode qui se veut économe en ressources consommées. Donc à la place, un profil HMM (Hidden Markov Models) est construit pour évaluer la probabilité de l'alignement et ainsi limiter le nombre de faux positifs [Li, 2011].

2. La chaîne CIGAR d'un *read* est un code représentant son alignement sur le génome.

2.6 Jonctions d'épissage

Dans cette section, nous nous situons essentiellement dans un contexte de transcriptomique. Comme toutes les techniques de haut débit, le RNA-Seq évolue avec les SHD. Le plus gros travail à accomplir dans l'analyse de ce type de données est l'alignement des *reads* sur un génome ou un transcriptome de référence. En effet, le problème de *mapping* de *reads* RNA-Seq fait partie des grands défis de la bioinformatique car contrairement au *mapping* classique, déjà complexe, beaucoup de *reads* sont affectés par les jonctions exon/exon.

Par convention, on distinguera trois types de jonctions : celles constituées de deux exons ordonnés dans un même gène (les épissages classiques), celles qui apparaissent dans des régions non-annotées ou non-connues (les épissages non-codants) et les autres (les épissages chimériques). En pratique, s'il existe quelques outils bioinformatiques pour détecter les épissages classiques (TopHat, MapSplice, QPALMA, ou encore GSNAP [Trapnell *et al.*, 2009; Wang *et al.*, 2010; De Bona *et al.*, 2008; Wu et Nacu, 2010]), le cas des ARN non-codants, rares ou chimériques restent ouverts [Maher *et al.*, 2009b]. Nous exposons ici les différents épissages classiques et chimériques.

2.6.1 Les épissages classiques

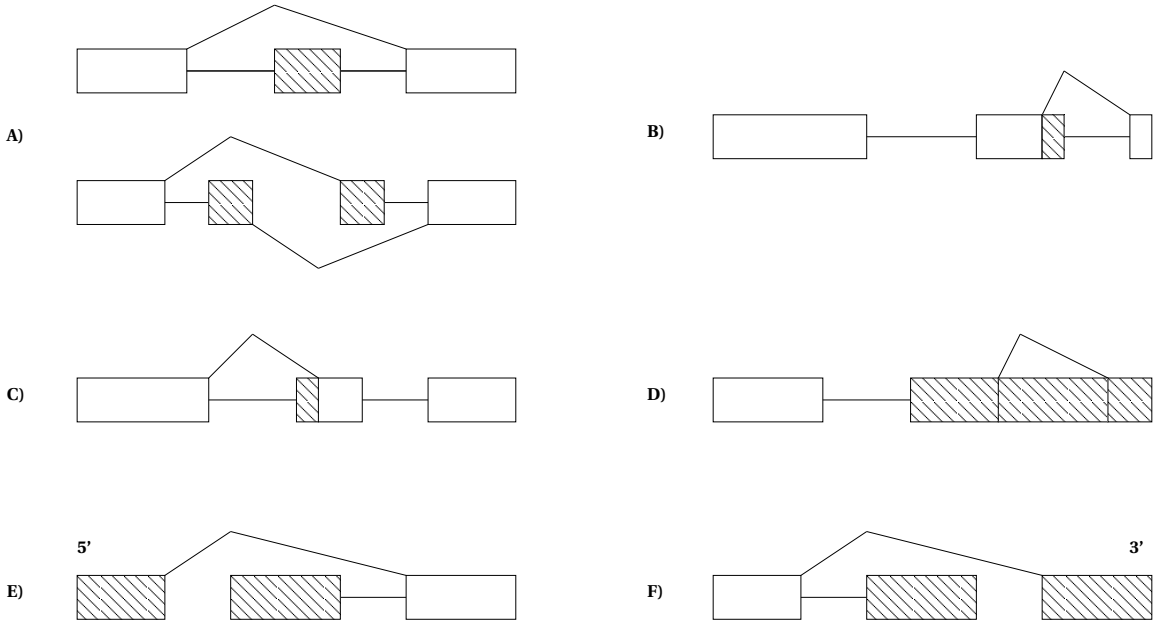
Généralement, un épissage est le processus qui conduit le passage du pré-messager au messager chez les eucaryotes (élimination des séquences introniques). Le transcrit épissé est ainsi représenté par des exons et leurs jonctions. Le transcrit canonique est constitué de tous les exons du gène, les jonctions ainsi générées sont par conséquent canoniques. Cependant, dans certains cas il peut y avoir des transcrits alternatifs du même gène ; le gène produit plusieurs transcrits isoformes, par combinaison des exons inclus dans l'ARNm (des transcrits qui n'ont pas toutes leurs jonctions canoniques). Des estimations montrent qu'entre 60% et 75% des gènes humains sont affectés par au moins un transcrit alternatif [Kim *et al.*, 2004].

Cette découverte a changé, de façon irréversible, les perspectives sur les problèmes d'annotation des gènes [Black, 2003]. Pour cause, les épissages alternatifs ont un rôle fondamental dans la régulation de certains gènes chez les eucaryotes, ils sont d'ailleurs impliqués dans de nombreuses maladies chez l'homme, notamment dans des cancers [Morrissy *et al.*, 2009; Nagalakshmi *et al.*, 2008]. En effet, la protéine résultante d'un transcrit alternatif peut être différente avec parfois même des fonctions antagonistes [Florea, 2006]. Du coup, cette protéine peut cohabiter avec la protéine canonique sur la même cellule et le

phénotype dépendra donc du taux d'expression de chacune des deux protéines [Florea, 2006].

À partir d'un gène, nous pouvons classer les transcrits alternatifs en quatre catégories. Celles-ci sont détaillées dans la figure 2.4 avec de (A) à (D) : exclusion/inclusion d'exon, exon alternatif en 3', exon alternatif en 5' et rétention d'intron. Ensuite dans les régions UTR (5' et 3'), des variants d'épissage ((D) et (E) dans la figure 2.4) sont souvent rencontrés mais ils sont complexes à identifier et à caractériser.

FIGURE 2.4 : Les différents types d'épissage alternatif.



(A) Exclusion/inclusion d'exon ; (B) Exon alternatif en 3' ; (C) Exon alternatif en 5' ; (D) Rétention d'intron ; (E) et (F) 5' et 3' variant d'UTR (Untranslated Regions). Cette illustration représente les exons par des rectangles et les introns par des traits. Les événements d'épissage alternatif (exons ou portions d'exons) sont hachurés et les jonctions simples sont par conséquent non hachurées.

Une étude récente constate que les transcrits alternatifs augmentent la diversité des gènes chez un individu ce qui explique un paradoxe sur le fait que les organismes (simples ou complexes) se ressemblent dans leur nombre de gènes [Sammeth *et al.*, 2008]. Ils proposent une caractérisation automatique des différents variants d'épissage dans les tissus dans le but de comparer les différents transcrits d'un même gène. Ce travail pourrait révéler des molécules qui seraient responsables d'une différence de phénotype. Ainsi, en comparant des cellules saines et cancéreuses d'un même tissu, on pourrait découvrir des variants qui sont directement impliqués [Dutertre *et al.*, 2010].

Détections des jonctions. Nous pouvons observer dans la section 2.4.2.2 qu'il existe deux grandes approches différentes pour localiser des *reads* sur un transcriptome : i/ les approches bwt (Burrows-Wheeler transform) qui autorisent au mieux un gap court, ii/ les approches à base de graines qui autorisent des gaps plus longs mais nécessitent plus de ressources de calculs (par exemple, GSNAP prend $\simeq$ 8X de temps que TopHat pour environ $\simeq 1,5$X plus de *reads* détectés dans des jonctions d'épissage [Garber *et al.*, 2011]). L'approche i/ s'avère efficace lorsque le génome de référence est disponible alors que l'approche ii/ est plus sensible si le transcriptome de référence est disponible [Garber *et al.*, 2011].

Quoi qu'il en soit, les deux approches se retrouvent limitées pour détecter les jonctions d'épissage où la plupart des gaps sont beaucoup trop grands pour être localisés. Il existe cependant quelques outils de détection de *splicing*. Ils se divisent en deux catégories :

Les outils qui procèdent en 2 étapes comme TopHat [Trapnell *et al.*, 2009] et Mapsplice [Wang *et al.*, 2010]. Dans une première étape, les *reads* sont localisés en utilisant des outils de la catégorie i/ (décrite au début du paragraphe) tels que Bowtie[Langmead *et al.*, 2009] ou BWA[Li et Durbin, 2009]. Dans une seconde étape, les *reads* qui ne sont pas localisés, au cours de la première étape, sont découpés en plusieurs segments. Ces segments sont eux mêmes localisés indépendamment sur le même génome. Ensuite, la région génomique qui les entoure est analysée pour les relier entre eux et ainsi créer des jonctions. Cette seconde étape est souvent très coûteuse en temps car il faut tester toutes les combinaisons possibles entre les segments.

La deuxième catégorie d'outils de *splicing* concerne ceux qui utilisent des graines multiples comme QPALMA et GSNAP [De Bona *et al.*, 2008; Wu et Nacu, 2010]. Le principe est le suivant : tout d'abord le génome est organisé dans une table de hachage avec comme identifiants les différents k-mers du génome ; ensuite, des graines de longueur k sont construites à partir des *reads* et du génome haché, puis elles sont étendues au maximum pour détecter précisément les parties non localisées des *reads* : c'est le principe du *seed and extend* (cf. section 2.4.2.2). Les candidats potentiels sont ensuite examinés avec une méthode plus sensible, telle que l'algorithme de Smith-Waterman pour distinguer précisément la jonction lors de l'alignement des *reads* sur le génome.
La première catégorie d'outils de *splicing* est globalement moins coûteuse en ressources (mémoire et temps). En contre partie, elle est parfois moins sensible car des épissages peuvent être localisés malencontreusement lors de la première étape du processus [Garber *et al.*, 2011].

Reconstruction des transcrits. Définir un catalogue précis et détaillé de tous les transcrits et de leurs isoformes est l'un des défis majeurs dans l'étude des transcriptomes. Le RNA-Seq est la première technique à grande échelle qui couvre quantitativement tous les transcrits, même les plus rares. Pour parvenir à leur reconstruction, il faut néanmoins regrouper les différentes informations contenues dans les *reads*. Cette étape est un exercice difficile pour trois raisons majeures :

- les gènes possèdent en moyenne plusieurs transcrits isoformes, quel isoforme produit quels *reads* ?
- les *reads* sont courts et certains gènes ne sont pas couverts sur toute leur longueur, comment restructurer le gène précisément ?
- les *reads* proviennent aussi bien de l'ARNm mature (avec seulement les exons) que des ARN précurseurs qui ne sont épissés que partiellement (avec des séquences introniques lors du phénomène de rétention d'introns [Sammeth *et al.*, 2008]), comment reconstruire le transcrit mature ?

Plusieurs méthodes existent pour reconstruire les transcrits qui sont plus ou moins dépendantes du génome de référence. En d'autres termes, certaines méthodes comme GMorSE [Denoeud *et al.*, 2008] ou Cufflinks [Trapnell *et al.*, 2010] commencent par localiser les *reads* sur le génome et assemblent les *reads* qui se recouvrent ensuite. En contraste, des méthodes telles que Trans-ABySS [Robertson *et al.*, 2010] assemblent directement les *reads* sans génome de référence. Quoi qu'il en soit, tous les algorithmes ont déjà fait leurs preuves en construisant des milliers de transcrits et transcrits alternatifs. La question est plutôt de savoir quel est celui qui répond le mieux à ce problème biologique. Si le génome de référence n'existe pas, alors la question ne se pose pas. Dans le cas inverse, les méthodes qui en dépendent (GMorSE, Cufflinks) semblent gagner en sensibilité car elles peuvent s'appuyer sur les nombreuses annotations qui sont disponibles sur les génomes. En outre, dans le cas où le génome ou transcriptome de référence est affecté par des réarrangements majeurs, comme dans les cancers [Stephens *et al.*, 2011, 2009], le choix est plus délicat et dépend surtout du but de l'analyse. D'ailleurs, dans ces cas complexes, une approche hybride combinant les deux types d'algorithmes est peut-être une bonne solution, que ce soit pour capturer les informations connues ou les nouvelles. Notons tout de même qu'en pratique les algorithmes indépendants des génomes ont besoins de ressources considérables [Garber *et al.*, 2011].

Estimation du niveau d'expression. La quantification et la normalisation des taux d'expression des transcrits sont des éléments importants pour comprendre comment les transcrits sont exprimés et évoluent sous différentes conditions cellulaires ou stades de dévelop-

pement. L'émergence des *microarrays* a été accompagnée par le développement de méthodes statistiques [Grant *et al.*, 2005].

Bien qu'en théorie ces méthodes soient aussi applicables sur du RNA-Seq, l'utilisation de la couverture des *reads* pour mesurer la quantification permet de bénéficier d'informations supplémentaires, par exemple la distribution de l'expression de chacun des isoformes sur le gène [Trapnell *et al.*, 2010].
Pour mesurer la distribution des *reads* sur un gène, des lois statistiques telles qu'une distribution de Poisson sont utilisées [Jiang et Wong, 2009]. Cependant, les distributions des *reads* sont variables entre différents réplicas d'un échantillon, ainsi une distribution statistique qui suit une loi normale ne prend pas en compte cette variabilité [Langmead *et al.*, 2010]. L'idéal serait donc d'estimer cette variabilité de façon empirique entre les réplicas, à condition d'en avoir suffisamment, ce qui est loin d'être envisageable dans la réalité. Pour pallier le problème, des approches paramétriques telles que EdgeR modélisent cette variabilité biologique [Robinson *et al.*, 2010]. Une autre approche est la technique du RPKM (Reads Per KiloBase Per Million Mapped) qui consiste à dénombrer le nombre de *reads* qui sont sur un millier d'exons parmi un million de *reads* [Mortazavi *et al.*, 2008].

Cependant, même si ces approches sont significatives pour des expressions différentielles de gènes, les conclusions biologiques sont à prendre avec précaution, d'autant plus que nous avons vu dans la sous-section 2.2.2 qu'il existait des biais protocolaires lors de la préparation du RNA-Seq.

2.6.2 Les épissages chimériques

Rappelons que dans les cellules eucaryotes, le patrimoine génétique constituant le génome est partitionné dans le noyau sous forme d'ADN associé à des protéines pour former la chromatine. Quant aux ARN messagers, ils servent principalement d'intermédiaires pour faire circuler précisément les informations contenues dans le génome. Formées lors de l'épissage, les informations sont ensuite conservées depuis le génome selon une organisation colinéaire (sans changer l'ordre d'apparition des nucléotides).

Une représentation de ces deux différents alignements colinéaires est donnée dans la figure 2.5 (a). Cependant, au cours de ces dernières années une autre classe d'ARN a été découverte remettant en cause l'hypothèse d'une stricte conservation linéaire de l'information entre l'ADN et les ARN. En effet, il existe des transcrits qui sont composés de deux parties du génome complètement distinctes (chromosomes, brins, gènes, etc). Ils sont qualifiés de chimères ou transcrits chimériques, à cause de leur structure non conformiste.

Ces chimères ont plusieurs origines. Les premières décrites sont les transcrits de fu-

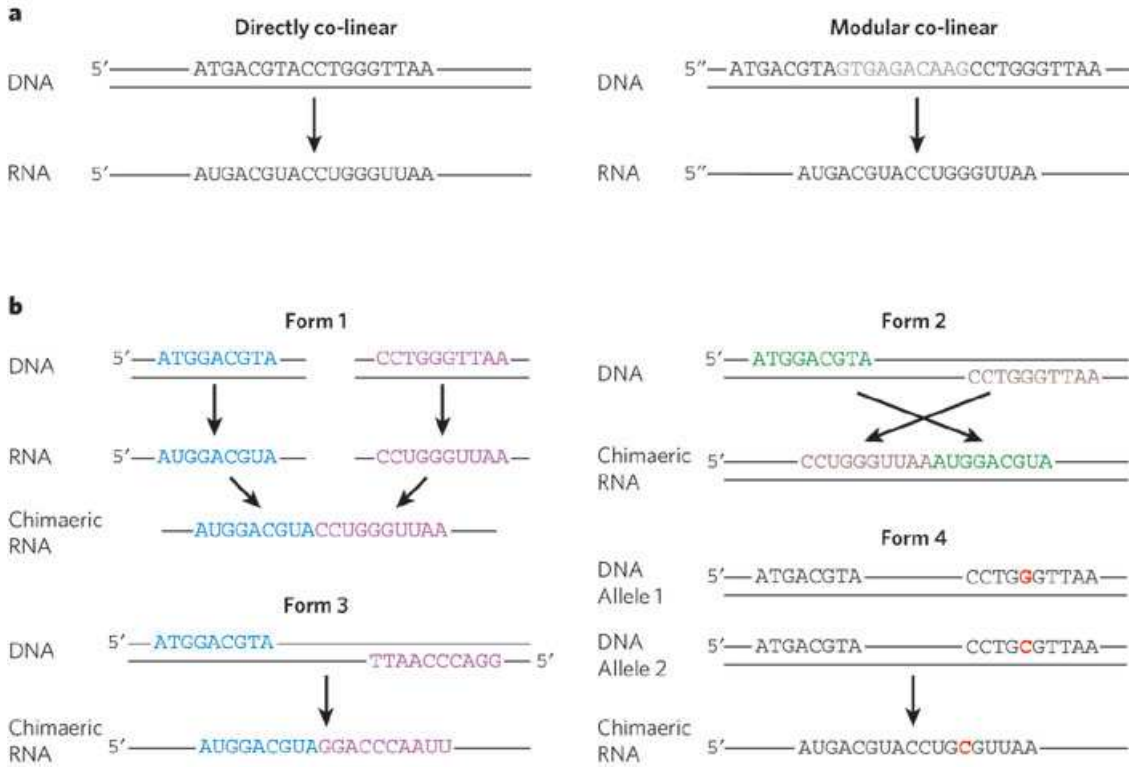
sion : transcrits chimériques exprimés dans les cellules cancéreuses à partir d'un gène de fusion issu d'une translocation chromosomique. Depuis leur mise en évidence il y a plusieurs années, leur implication dans les mécanismes tumoraux est largement reconnue [Stephens *et al.*, 2011]. Ces transcrits constituent des marqueurs moléculaires primordiaux pour le diagnostic et le suivi des maladies. L'intérêt des SHD, tout particulièrement avec le RNA-Seq, est rapidement apparu, avec notamment la détection de nouveaux transcrits de fusion issus de translocations [Gingeras, 2009]. Récemment, ces mêmes données ont permis la caractérisation d'autres catégories de chimères qui sont induites au cours du processus d'épissage (sans translocation chromosomique) par des mécanismes encore inexpliqués : i/ transcription de deux gènes consécutifs en même temps, on parle alors de chimères par trans-épissage [Parra *et al.*, 2006; Maher *et al.*, 2009b; Nacu *et al.*, 2011] ; ii/ transcription de deux gènes à cause d'un glissement de la polymérase sur une région homologue, on parle alors de chimères par glissement [Li *et al.*, 2008d]. L'identification et la caractérisation de ces ARN chimères constituent un enjeu primordial pour la compréhension des mécanismes tumoraux.

Identifier les jonctions chimériques est donc un problème complexe, tant les formes chimériques sont diverses et les connaissances sur le sujet sont faibles. En conséquence, malgré quelques approches sur des données de RNA-Seq *paired* [Li *et al.*, 2011; Sboner *et al.*, 2010] ou d'autres en croisant des données génomiques et transcriptomiques de la même lignée cellulaire [McPherson *et al.*, 2011], très peu d'outils sont capables d'identifier précisément la globalité des chimères d'une expérience. De plus, le manque de connaissances biologiques sur le sujet fait que très peu de chimères sont caractérisées à ce jour.

2.6.2.1 Les différentes formes de chimères

Des études sur la structure des ARN de différentes espèces ont révélé que des séquences provenant de régions différentes du génome sont en définitives reliées entre elles dans un même transcrit mature (FIGURE 2.5 (b)) : i/ avec des ARN qui peuvent être transcrits sur des gènes différents : transcrits de fusion lors d'un réarrangement du génome ou d'autres chimères formées lors du processus d'épissage par glissement ou trans-épissage (Form1), ii/ sur le même chromosome mais avec un réarrangement des séquences exoniques sur le transcrit mature (Form2), iii/ sur le même chromosome mais sur les deux brins de l'ADN (Form3), ou iv/ sur le même chromosome mais sur deux allèles différents d'un même gène (Form4).

FIGURE 2.5 : Les différentes formes de chimères.



(**a**) Les alignements colinéaires sont caractérisés par deux formes : soit la séquence est directement transférée (à gauche), soit la séquence est épissée puis transférée (à droite). (**b**) Les alignements non-colinéaires ou chimères sont répartis en quatre catégories (la production d'un ARN précurseur n'est représentée que dans la première catégorie (**Form 1**) mais elle apparaît dans toutes). **Form1** représente la formation d'un transcrit chimérique à partir de locus de différents gènes sur le génome (à cause d'un réarrangement ou au cours du processus d'épissage). **Form2** illustre la formation d'une chimère provenant d'un seul gène mais avec un réarrangement des exons. **Form3** illustre la formation d'une chimère provenant de deux ARN précurseurs qui sont transcrits sur des brins opposés. Et enfin **Form4** décrit la formation d'une chimère à partir de transcrits recombinés de deux allèles du même gène, un contenant le SNP (G → C), l'autre non.

Bien que la plupart des chimères qui sont étudiées s'accompagne de translocations chromosomiques (transcrits de fusion) [Stephens *et al.*, 2011], des études ont montré la présence de chimères chez des patients cancéreux en l'absence de translocations [Maher *et al.*, 2009b].

En effet, un transcrit chimérique peut être le produit de l'épissage de deux gènes consécutifs, on peut parler de trans-épissage [Parra *et al.*, 2006]. De tels transcrits peuvent être non seulement détectés dans des cellules cancéreuses, mais ils peuvent être aussi présents dans des tissus sains [Li *et al.*, 2008a].

Un type de chimère par réarrangement d'exons (Form2 de la FIGURE 2.5) impliquant l'isoforme SEC14L2 a été décrit par [Kapranov *et al.*, 2007]. Fonctionnellement, cette chimère permet d'ouvrir une plus grande zone de lecture pendant la mise en place des acides aminés 56-93 de SEC14L2 après l'acide aminé 218 de la protéine. Une autre étude montre qu'entre 4 et 6% des gènes (au moins) seraient impliqués dans la formation de chimères lors de l'épissage bien qu'elles soient très peu exprimées [Parra *et al.*, 2006].

2.6.2.2 Les chimères et les cancers

Les transcrits de fusion forment l'une des plus grandes classes des altérations génétiques, celle résultant des réarrangements chromosomiques [Maher *et al.*, 2009c]. Les transcrits de fusion sont connus pour leur caractère oncogène (catégorie de transcrits dont l'expression favorise la survenue de cancers) dans les leucémies, les lymphomes et les sarcomes, avec par exemple le transcrit de fusion BCR-ABL présent dans les leucémies myéloïdes qui sert de prototype (FIGURE 2.6) [Klein *et al.*, 1982].

Toutefois, il s'avère que plus de 80% des transcrits de fusions connus sont attribués aux leucémies, aux lymphomes, et aux sarcomes bénins qui représentent 10% des cancers. En revanche, pour l'ensemble des cancers épithéliaux qui représentent à eux seuls 80% des cancers mortels, seulement 10% des transcrits de fusion ont été répertoriés [Maher *et al.*, 2009c]. À partir de données SHD, de nouveaux cas ont été récemment identifiés dans les cancers de la prostate comme TMPRSS2-ERG et dans les cancers du poumon avec EML4-ALK. Dans le passé, il était difficile de les identifier tellement leur niveau d'expression est faible dans les tissus en question. Leur rôle oncogène a été vérifié ce qui en fait des cibles thérapeutiques potentielles.

Finalement, les transcrits de fusions jouent aussi un rôle dans la formation de cancers dans les cellules épithéliales même si leur présence est rare dans les cellules [Tomlins *et al.*, 2005; Soda *et al.*, 2007].

Cette découverte n'aurait pas été possible sans les SHD, car, depuis le haut débit, de

FIGURE 2.6 : Le transcrit de fusion BCR-ABL.

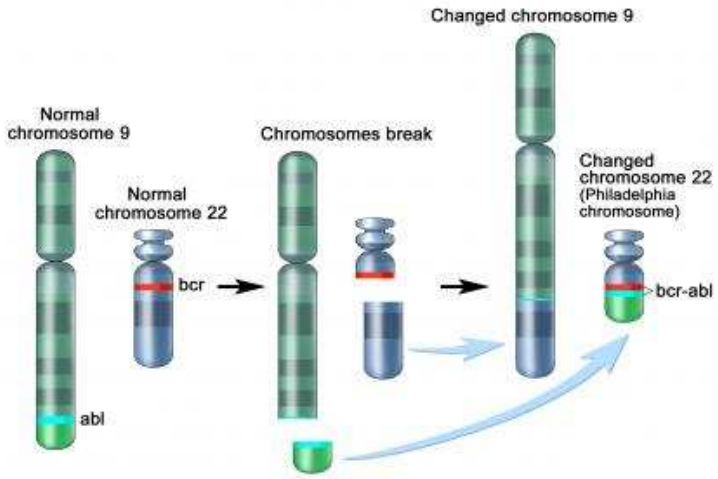


Illustration de la translocation génomique entre les gènes BCR (chr 22) et ABL (chr 9). Un ARN fonctionnel est transcrit au niveau de la translocation sur le chromosome 22 pour ainsi générer le transcrit de fusion BCR-ABL. Cette chimère est fortement impliquée dans les leucémies myéloïdes. Notons que la translocation réciproque n'est pas exprimée donc n'induit pas de chimère.

nouvelles translocations sont identifiées et le séquençage massif de données transcriptomiques a permis de caractériser certains événements récurrents. Chez les patients atteints du cancer du sein, des chimères sont présentes presque systématiquement tout en étant très faiblement exprimées. C'est le cas de ETV6-NTRK3 ou MYB-NFIB impliqués dans la formation de carcinomes adénoïdes kystiques dans le sein [Edgren *et al.*, 2011].

Les transcrits de fusion, c'est-à-dire en présence d'une translocation chromosomique, ont un rôle prépondérant dans la formation des cancers, cependant de nombreux cancers sont décelés en l'absence d'une quelconque translocation. Le RNA-Seq permet de détecter les chimères générées au cours du trans-épissage (FIGURE 2.7 (**B**)). L'une des premières de ce genre est JAZF1-JJAZ1 détectée dans des cellules stromales de l'endomètre [Li *et al.*, 2008a]. Cette chimère est identique à celle produit à partir d'un gène de fusion détecté dans des cas de sarcome du même tissu. Ces observations suggèrent qu'un phénomène de trans-épissage pourrait précéder l'apparition d'une translocation chromoqomique.

La chimère HHLA1-OC90 est surexprimée dans les cellules du tératocarcinome (tumeur de l'ovaire ou du testicule) pendant que CD205-DCL1 est surexprimée dans les cellules du lymphome de Hodgkin (LH) [Nacu *et al.*, 2011]. Citons aussi la chimère pro-

duite entre les oncogènes RBM14 et RBM4 qui génère une protéine de fusion de la famille des coactivateurs transcriptionnels CoAZ [Brooks *et al.*, 2009] ou encore celle formée à partir de RBM6 et RBM5 qui est présente dans de nombreux tissus cancéreux et associé à de nombreuses tumeurs du sein [Nacu *et al.*, 2011].

A ma connaissance, aucune chimère par glissement (FIGURE 2.7 (**B**)) n'a encore été caractérisée ni chez l'humain, ni même chez d'autres vertébrés, et non plus dans les cancers. Les premières identifications ont été faites chez la levure avec les gènes SPT7 et LYS12 et le rôle fonctionnel de ce type de chimère reste inexpliqué. En tout cas, les protéines de chacun des gènes concernés ne sont pas affectées. En outre, pour que ce type de chimère puisse se former, il faut qu'il y ait une petite région d'homologie entre les deux gènes cibles [Li *et al.*, 2008d].

FIGURE 2.7 : Les chimères formées au cours de l'épissage.

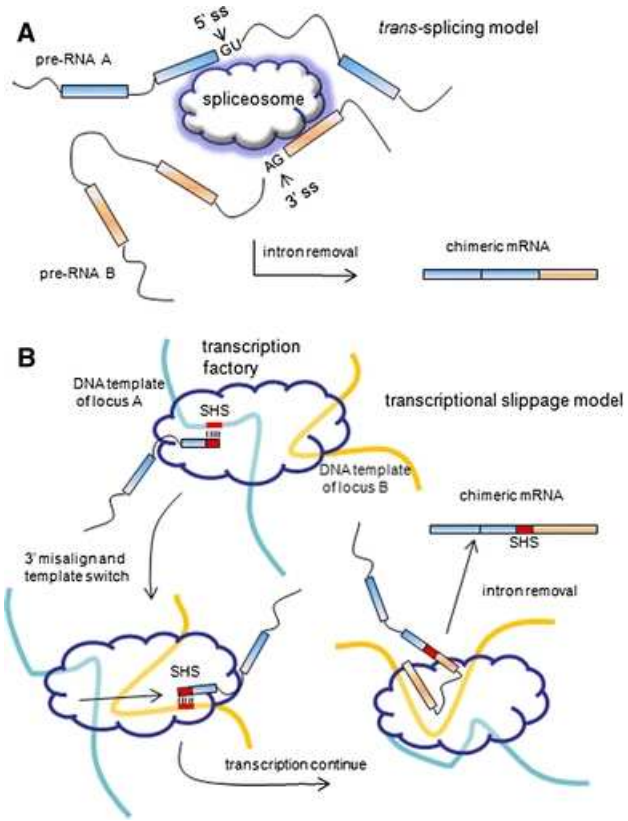


Illustration des deux types de chimères qui sont générées au cours du processus d'épissage : chimère par trans-épissage (**A**) et chimère par glissement (**B**). (**A**) Les ARN précurseurs A et B proviennent respectivement des régions 5' et 3' des sites d'épissage. Les deux précurseurs génèrent ensuite une chimère lors de la transcription. (**B**) Les petits rectangles rouges symbolisent des courtes séquences homologues entre deux gènes distincts (SHS). Les locus A et B sont actifs et partagent un facteur de transcription. La transcription commence au niveau du locus A, puis la polymérase glisse au niveau de la SHS, pour enfin finir sa transcription au niveau du locus B.

2.6.2.3 L'identification des chimères par SHD

Nous savons que les techniques de SHD permettent d'identifier des transcrits de fusion dans les cellules cancéreuses, et ce même s'ils sont peu exprimés [Nacu *et al.*, 2011; Gingeras, 2009]. Encore mieux, des nouvelles formes de chimères sont maintenant identifiées chez des patients sains ou malades.

Toutefois, le monde des chimères semble vaste et complexe et à ce jour, nous ne sommes pas en mesure de caractériser précisément toutes les chimères d'un tissu cellulaire. En raison de la profondeur de séquençage, de la masse de données générée, des erreurs de séquences, des biais biologiques lors de la mise au point des techniques, et d'autres facteurs peut-être encore inconnus, la détection bioinformatique des chimères est difficile. Tous ces problèmes peuvent amener à des erreurs algorithmiques et malencontreusement à la détection de fausses chimères. C'est pour cela que les outils bioinformatiques capables de détecter les chimères ne sont pas faciles à concevoir. Il en existe quelques-uns, que nous pouvons classer en trois catégories.

RNA-Seq et *paired*-reads . Une alternative à la méthode du RNA-Seq avec la technologie Illumina® est la possibilité de séquencer les extrémités de chaque brin, on parle alors de séquences binômes ou *paired*-reads . Ainsi, si l'ARN d'origine mesure 500 *pb*, nous nous retrouvons avec des paires de *reads* qui, une fois repositionnées sur le génome, devraient se retrouver espacées de $\simeq$ 500 *pb* si le brin provient d'un même exon. Cette technique est utilisée pour favoriser la détection des jonctions d'épissage et la reconstruction des transcrits [Trapnell *et al.*, 2010].

À l'aide d'outils de *mapping* robustes (voir section 2.4), nous pouvons prédire un réarrangement chromosomique dans le cas où les deux *reads* ne vérifient pas ce critère (localisation sur des chromosomes ou des gènes différents). Cette solution a été adoptée par la majeure partie des logiciels comme FusionSeq et FusionHunter pour citer les plus récents [Sboner *et al.*, 2010; Li *et al.*, 2011].
La méthode est la suivante : i/ localisation des binômes sur le génome pour vérifier la cohérence des *reads* (procédure décrite juste avant), ii/ les candidats restants sont filtrés selon la qualité des séquences et l'annotation des gènes, iii/ les paires de gènes homologues sont écartées. En effet, si les *reads* sont localisés dans deux gènes différents mais très homologues, il est possible que cette potentielle chimère soit tout simplement un artefact de la procédure de *mapping* en i/. L'utilisation des *paired*-reads a ses avantages et ses inconvénients.

D'un côté : i/ il est possible de lever plus facilement quelques ambiguïtés, en utilisant le lien entre les deux *reads*. Par exemple, si l'un des deux *reads* est dans une région répétée et l'autre non, la paire de *reads* peut quand même être traitée normalement alors que cela resterait ambigu sans le deuxième *read* ; ii/ l'utilisation de deux *reads* courts (2x50) plutôt qu'un grand (1x100) fournit une meilleure couverture du transcrit sur toute sa largeur [Maher *et al.*, 2009c] ; iii/ certains points de cassures ne sont pas recouverts et de ce fait indétectable ; iv/ la détection de transcrits de fusion est plus facile, il suffit de localiser indépendamment les deux *reads* avec un logiciel efficace, puis de relier leur localisation ensuite [Sboner *et al.*, 2010].
D'un autre côté : a/ l'étape de séquençage *paired*-reads peut introduire des biais technologiques [McManus *et al.*, 2010] ; b/ l'utilisation de *reads* plus longs augmente les possibilités de chevaucher le point de cassure des transcrits de fusion ; c/ les SHD génèrent de plus en plus de données donc le problème de couverture mentionné en ii/ et par la même occasion impliquant le iii/ n'en sera plus un.

Finalement avec l'évolution des SHD, il existe deux arguments majeurs en faveur de l'utilisation du *paired*-reads pour identifier les chimères : lever les ambiguïtés du *mapping* et utiliser une procédure simple pour détecter avec assurance les chimères. Cependant, le point de cassure est très important pour identifier l'annotation et la fonction de celles-ci [McPherson *et al.*, 2011] et cet aspect reste le gros point négatif de ce type de stratégie.

RNA-Seq et WGSS. Comparé aux microarrays, le RNA-Seq peut être utilisé pour une caractérisation *de novo* du transcriptome. Cependant, la transcription inverse (RT) utilisée dans le protocole RNA-Seq pour la préparation de l'ADNc semble produire des fausses chimères du même type que *Form3* dans la figure 2.5 [Houseley et Tollervey, 2010]. Cet effet a deux conséquences : la première est l'ajout d'une difficulté algorithmique au développement de méthodes pour identifier les chimères (sachant que la conception est complexe à la base) ; la deuxième est l'impossibilité de distinguer les fausses des vraies chimères de Form3.

Pour résoudre ce problème, une idée possible est de coupler pour un même échantillon du RNA-Seq et du WGSS. En effet, la deuxième technique n'est pas affectée par ce biais, elle permet donc de mettre en exergue les vrais transcrits de fusion. Avec un tel procédé, nous pouvons nous permettre d'utiliser du RNA-Seq (*single*-read) en évitant tous les inconvénients cités dans le précédent paragraphe. En effet, les méthodes bioinformatiques sont plus simples pour analyser les données génomiques que transcriptomiques (pas de jonction d'épissage, moins d'erreurs dues à la préparation, etc). Tout en filtrant les faux positifs algorithmiques grâce au WGSS, nous pouvons détecter précisément les points de cassure

et garder une bonne couverture des gènes avec le RNA-Seq. Cette procédure est utilisée dans le logiciel Comrad [McPherson *et al.*, 2011].

Cependant cette approche a deux inconvénients. Tout d'abord, elle a un coût : tous les laboratoires n'ont pas les moyens de produire des expériences RNA-Seq et WGSS en parallèle sur un même échantillon cellulaire. Ensuite, cette approche permet de détecter précisément les transcrits de fusion. Cependant cette approche n'enlève pas le besoin d'outils performants pour la détection de tout type de chimères.

RNA-Seq et chimères par trans-épissage. Ici la philosophie est différente. L'idée est de créer un programme capable de détecter précisément une seule catégorie : les chimères par trans-épissage. En effet, la problématique pour la détection de ce type de chimères ne varie pas beaucoup de celle des jonctions classiques d'épissage, d'un point de vue bioinformatique (mêmes chromosomes, mêmes brins). Seuls quelques critères s'ajoutent : i/ les deux exons doivent appartenir à deux gènes distincts ; ii/ les deux gènes doivent être proches l'un de l'autre afin de pouvoir posséder les mêmes facteurs de transcription. Il est envisageable de vérifier ce dernier point à l'aide d'une base de données contenant toutes les refseq. Cette procédure a déjà permis de découvrir et de valider des nouvelles chimères par trans-épissage [Nacu *et al.*, 2011].

Toutefois, aussi précise soit elle, cette approche est spécifique à un sous ensemble de chimères et de ce fait ne permet pas non plus de détecter l'ensemble complet des chimères d'une expérience.

Deuxième partie

Résultats

CHAPITRE

3 Méthodes pour annoter des reads sur un génome

Ce chapitre expose un panel de méthodes pour exploiter au mieux des données du transcriptome issues des SHD. À l'aide de ces méthodes, nous proposons une stratégie pour annoter des nouveaux transcrits sur un génome à l'aide de la DGE et du RNA-Seq, tout en limitant les problèmes causés par des artefacts technologiques et méthodiques.

Sommaire

Les SHD produisent une quantité massive de *reads*. Une stratégie possible est de positionner ces *reads* sur un génome afin d'identifier des aspects biologiques, comme l'étude de la fonction des gènes ou encore le rôle de la transcription dans les régions non-codantes [Bertone *et al.*, 2004]. Intéressons nous à la technique DGE qui identifie chaque séquence par un *tag* correspondant à la signature d'un transcrit (voir section 2.2.1 du chapitre 2). Dès lors, il est possible de cataloguer chacun des transcrits et de mesurer leur activité dans une cellule [Kim *et al.*, 2007]. Si, en plus, le *tag* ne possède qu'une et une seule localisation sur le génome, nous déterminons la position génomique d'origine du transcrit, ce qui nous offre la possibilité d'obtenir une annotation fonctionnelle complète du transcrit [Rivals *et al.*, 2007; Barski *et al.*, 2007; Boyle *et al.*, 2008]. En fait, les localisations multiples sont

écartées simplement parce qu'elles engendrent une ambiguïté sur la position d'origine du transcrit, ce qui rend l'annotation plus complexe.

D'année en année, les SHD produisent de plus en plus de séquences. Celles-ci peuvent être aussi de plus en plus longues ce qui semble un avantage pour les localiser sans ambiguïté sur un génome [Saha *et al.*, 2002]. Néanmoins, des investigations récentes permettent de constater qu'une large proportion des séquences ne peut être localisée [Keime *et al.*, 2007; Rivals *et al.*, 2007]. Enfin, les techniques de séquençages adaptées au transcriptome telles que la DGE et le RNA-Seq nous permettent actuellement de découvrir une quantité importante de nouvelles régions transcrites, appelée « RNA dark matter » ou matière noire, ce qui ouvre une nouvelle voie d'exploration [Kim *et al.*, 2007; Kapranov *et al.*, 2007; van Bakel *et al.*, 2010]. L'analyse d'une stratégie de positionnement représente donc un enjeu majeur dans la génomique afin d'augmenter la capacité à prédire un maximum de positions génomiques pour les transcrits.

Les *reads* sont assimilés à des courtes séquences présentes en nombre remarquable, ou dont la répartition est remarquable sur le génome. La première question qui nous vient à l'esprit est : comment positionner des *reads* sur un génome avec le moins d'ambiguïté possible ?
Une façon de procéder serait de positionner chaque *read* (extrait d'un vrai transcrit) à un seul et bon endroit sur le génome. Mais avec quels critères peut-on réaliser cette procédure ? En effet, dans une expérience SHD, nous avons toujours un pourcentage de *reads* qui se localisent mais pas forcément au bon endroit sur le génome.

Supposons que pour être dans les conditions idéales, il faut un minimum de *bruit de fond* et un maximum de *prédictions*, tels que :

- le *bruit de fond* est l'attendu moyen de positionnement aléatoire, c'est-à-dire la proportion de séquences qui sont localisées sur un génome aléatoire (de la même taille que le génome de référence),
- les *prédictions* sont les séquences qui ne contiennent pas d'erreur(s) et qui sont positionnées à un endroit unique sur le génome de référence.

Remarque 4. *Le* bruit de fond *mesure une proportion de séquences qui ne sont pas localisées à l'endroit où elles devraient l'être. En d'autres termes, c'est une estimation du nombre de* reads *qui ne sont pas localisés au bon endroit sur le génome. Par conséquent, il faut minimiser ce* bruit de fond.

Nous nous intéressons à ce problème en démontrant qu'il existe une longueur mini-

male pour réduire efficacement le *bruit de fond*. Ensuite, nous proposons une étude statistique pour mesurer l'impact des erreurs de séquences sur les *prédictions*. Afin d'avoir un bon compromis entre *bruit de fond* et *prédictions*, nous proposons une stratégie de positionnement d'une collection de séquences d'ARN (ou plutôt ADNc) générée par des SHD sur un génome.

Ensuite, nous vous proposons un pipeline, qui à partir d'une collection de *reads*, applique notre stratégie de positionnement afin d'extraire des unités transcrites : les *prédictions*. Une fois cette action réalisée, il faut les cataloguer par annotation, chacune étant associée à un transcrit codant ou non-codant (plus précisément la partie UTR ou CDS d'un exon, un intron, etc...). De plus, plusieurs sources d'informations (tiling arrays, Sage Genie, RNA-Seq) sont intégrées à notre pipeline afin de renforcer l'annotation de nouvelles zones transcrites (celles qui sont annotées dans les régions introniques et intergéniques).

Enfin, nous avons évalué cette stratégie avec notre propre collection de *reads* générée avec la technique DGE. Des candidats potentiels ont été validés biologiquement par Q-PCR.

3.1 Modélisation du *bruit de fond*

De part leur courte taille, par rapport à une séquence de référence, les *reads* sont similaires à des *mots rares* ou *exceptionnels*. C'est pourquoi nous nous sommes inspirés de l'ouvrage de Stéphane Robin, François Rodolphe et Sophie Schbath, qui traite, entre autres, de la recherche de ces mots exceptionnels dans les séquences d'ADN par des méthodes heuristiques [Robin *et al.*, 2007]. En effet, la question du *bruit de fond*, consistant à savoir si un *read* est positionné par hasard ou non à un endroit sur son génome de référence, est importante. Une approximation par une loi de Poisson composée semble être appropriée pour apporter une solution rapide et efficace à ce problème. Elle permet de simuler le comportement du positionnement d'un ensemble de *mots rares* sur une séquence aléatoire, suffisamment longue. Grâce à cette méthode, nous pouvons varier la longueur des *reads* et observer le comportement aléatoire que cela engendre lors de leur positionnement sur une séquence génomique.

Une partie des définitions de la sous-section 3.1.1 sont directement tirées de cet ouvrage et elles sont nécessaires pour la compréhension de notre méthode en sous-section 3.1.2.

3.1.1 Définitions et notations des outils statistiques

3.1.1.1 Modèle de Bernoulli

Définition 3.1. *Un modèle de Bernoulli consiste à considérer une séquence X de longueur n comme une concaténation de lettres $X[0]X[1]\dots X[n-1]$ toutes indépendantes les unes des autres et prenant leur valeur dans l'alphabet Σ avec les probabilités respectives $\mu(a)$, $\mu(c)$, $\mu(g)$ et $\mu(t)$.*

On dit que le modèle est de type M0 lorsque la probabilité d'apparition est égale pour chacun des nucléotides, tel que $\mu(a) = \mu(c) = \mu(g) = \mu(t) = \frac{1}{4}$.

3.1.1.2 Loi de poisson

Définition 3.2. *Une loi de Poisson est une loi de probabilité qui décrit le comportement d'un ensemble fini d'événements rares se produisant sur une séquence, si ces évènements se produisent avec une fréquence moyenne connue.*

3.1.1.3 Occurrence d'un mot sur une séquence de Bernoulli

Soit X une séquence de Bernoulli générée par M0 de longueur n. Si l'on s'intéresse aux occurrences d'un mot w de longueur m, $w = w[0]w[1]\dots w[m-1]$, une première information importante est la probabilité que ce mot apparaisse en une position quelconque de la séquence X. Cette probabilité sera appelée *probabilité du mot* et sera notée $\mu(w)$.

$$\mu(w) = \mu(w[0]) \times \mu(w[1]) \times \cdots \times \mu(w[m-1]) = \prod_{j=0}^{m-1} \mu(w[j]). \tag{3.1}$$

L'équation (3.1) donne la probabilité que le mot w commence en une position quelconque de la séquence. La position d'une occurrence de w dans la séquence est donnée par la position de sa première lettre. On définit l'indicatrice $Y_i(w)$ d'occurrence du mot w en position i par :

$$Y_i(w) = \begin{cases} 1 & \text{si } X[i]X[i+1]\dots X[i+m-1] = w[0]w[1]\dots w[m-1], \\ 0 & \text{sinon.} \end{cases} \tag{3.2}$$

Pour toute position i comprise entre 0 et $n-m$, on a :

$$\mathbb{P}\{Y_i(w) = 1\} = \mu(w)$$

Les variables $Y_i(w)$ sont donc des variables de Bernoulli de paramètre $\mu(w)$ et leur espérance vaut :

$$\mathbb{E}[Y_i(w)] = \mu(w). \tag{3.3}$$

Pour les positions i situées au-delà de $n-m$ dans la séquence X, il n'y a plus assez de place pour que le mot se forme et donc $\mathbb{P}\{Y_i(w)=1\}=0$.

3.1.1.4 Comptage des occurrences

Le nombre d'occurrences (ou *comptage*) $N(w)$ du mot w dans la séquence X peut se définir à partir des indicatrices d'occurrences $Y_i(w)$:

$$N(w)=\sum_{i=0}^{n-m} Y_i(w). \tag{3.4}$$

On peut ainsi calculer l'espérance de $N(w)$ ou *comptage attendu* d'après (3.3) :

$$\mathbb{E}[N(w)]=\sum_{i=0}^{n-m}\mathbb{E}[Y_i(w)]=(n-m+1)\mu(w). \tag{3.5}$$

Pour avoir plus de détails sur le *comptage attendu*, vous pouvez consulter [Robin *et al.*, 2005].

3.1.1.5 Recouvrement de deux occurrences

Les occurrences d'un même mot peuvent se chevaucher dans une séquence. Ainsi la séquence

$$X=\overset{0}{\mathrm{G}}\overset{1}{\mathrm{A}}\overset{2}{\mathrm{C}}\overset{3}{\mathrm{A}}\overset{4}{\mathrm{C}}\overset{5}{\mathrm{A}}\overset{6}{\mathrm{C}}\overset{7}{\mathrm{C}}\overset{8}{\mathrm{T}}\overset{9}{\mathrm{T}}\overset{10}{\mathrm{A}}\overset{11}{\mathrm{C}}\overset{12}{\mathrm{A}}\overset{13}{\mathrm{C}}$$

contient trois occurrences du mot w =ACAC en position $i=1,3$ et 10 dans X. Or les deux dernières lettres de l'occurrence en position 1 constituent les deux premières lettres de l'occurrence en position 3. Ce phénomène est défini comme *le recouvrement* de deux occurrences, on parle aussi *d'occurrences chevauchantes*. On définit une *indicatrice de recouvrement* $\varepsilon_u(w)$ pour u compris entre 1 et m. Cette indicatrice vaut 1 si le mot w peut se recouvrir sur u lettres, c'est-à-dire si ces u dernières lettres sont égales à ses u premières lettres :

$$\varepsilon_u(w)=\begin{cases}1 & \text{si } (w[m-u]\,w[m-u+1]\dots w[m-1])=(w[0]\,w[1]\dots w[u-1]),\\ 0 & \text{sinon.}\end{cases}$$

Par construction, on a toujours $\varepsilon_m(w)=1$ puisque tout mot peut se recouvrir entièrement. On appellera *mot non recouvrant* un mot w pour lequel toutes les indicatrices $\varepsilon_u(w)$, sauf $\varepsilon_m(w)$, sont nulles.

3.1.1.6 Approximation du comptage par une loi de Poisson composée

Les *occurrences chevauchantes* perturbent le *comptage attendu* des occurrences par une loi poissonnienne pour un mot recouvrant. Ces mots ont en effet tendance à apparaître par paquets - ou *trains* - d'*occurrences chevauchantes* (les *wagons* sont les occurrences successives dans le *train*). L'indépendance des occurrences du modèle de Poisson n'est donc plus satisfaite, même asymptotiquement.

En étudiant d'abord les occurrences de *trains* puis en s'intéressant au nombre de *wagons* par *train*, on peut appliquer une loi poissonnienne sur les occurrences des *trains* qui par définition ne se chevauchent pas dans la séquence (cf. [Robin *et al.*, 2005] pour avoir plus de détails sur l'approximation).

Définition 3.3. *Une approximation du comptage des occurrences par une loi de Poisson composée est une approximation poissonnienne adaptée pour les mots recouvrants.*

3.1.1.7 Comptage ou espérance du *nombre de trains*

Lorsqu'une séquence X possède plusieurs *occurrences chevauchantes* successives, on parle de *train*. Chaque occurrence w du train est appelé un *wagon*.

Définition 3.4. *Le* nombre de trains $T(w)$ *de w est le nombre d'occurrences de w qui commencent un train. Une occurrence de w qui commence un train se distingue des autres par le fait qu'elle n'est pas recouverte à gauche par une occurrence précédente de w. Si on note $a(w)$ la probabilité qu'une occurrence donnée de w soit recouverte par la gauche, alors l'espérance du nombre de trains est égale à la proportion $[1-a(w)]$ de l'espérance du nombre d'occurrences :*

$$\mathbb{E}[T(w)] = [1-a(w)] \times \mathbb{E}[N(w)] \tag{3.6}$$

Notons que pour un mot *non recouvrant*, $a(w) = 0$ et le *nombre de trains* est identique au nombre d'occurrences ; on retrouve alors l'approximation du comptage $\mathbb{E}[N(w)]$ par une variable aléatoire de Poisson.

3.1.1.8 *Période* d'un mot

Définition 3.5. *Un entier $p \in \{0,\dots,m-1\}$ est une* période *du mot w si et seulement si deux occurrences de w peuvent se produire avec un décalage de p positions ($\varepsilon_{m-p}(w) = 1$) ce qui implique la périodicité suivante : $w[j] = w[j+1]$ pour tout $j \in \{0,\dots,k-p\}$. En d'autres termes, p est une période de w si le suffixe de longueur $m-p$ de w est égal au préfixe de longueur $m-p$ de w. Notons que 0 est toujours une période et que $m-1$ est la* période maximale. *L'ensemble des* périodes *d'un mot w est dénoté par $P(w)$*

Définition 3.6. *La* période primaire $\Pi(w)$ *du mot* w *est la plus petite de ses* périodes $p \in \{0,\dots,m-1\}$ *sous la condition* $p > 0$.

Remarque 5. *Un mot* w *peut toutefois avoir plusieurs* périodes *mais il n'a au plus qu'une seule* période primaire.

Exemple 3.1. La période et la période primaire.

$$w = \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7}{\mathrm{AATAATAA}}$$

w possède l'ensemble des *périodes* $P(w) = \{0,3,6,7\}$ et a pour *période primaire* $\Pi(w) = \{3\}$.

Dans le cas d'un recouvrement, on sait que w possède une *période primaire*, cela signifie que seul le préfixe $w[0]w[1]\dots w[p-1]$ est recouvert par un autre mot. De ce fait, la composante $a(w)$ ne va donc dépendre que des probabilités associées à ce préfixe.
Soit $\pi(w[j], w[j+1])$ la probabilité d'avoir un enchaînement des nucléotides $w[j]$ et $w[j+1]$ sur la séquence X, alors :

$$a(w) = \prod_{j=0}^{\Pi(w)-1} \pi(w[j], w[j+1]). \tag{3.7}$$

3.1.1.9 *Autocorrélation* d'un mot

Définition 3.7. *Soit* w *un mot de taille* m, i.e. $w \in \Sigma^m$, *et soit* $P(w)$ *l'ensemble des périodes de* w. *L'autocorrélation* c *de* w *est une représentation de* $P(w)$. *C'est un vecteur binaire de longueur* m *tel que :*

$$\begin{cases} c[i] = 1 \iff i \in P(w) & \forall 0 \le i < m, \\ c[i] = 0 & \textit{sinon.} \end{cases} \tag{3.8}$$

Exemple 3.2. L'autocorrélation.
Prenons trois mots w_1, w_2 et w_3 de longueur $m = 6$:

$$\begin{aligned} w_1 &= \overset{0\ 1\ 2\ 3\ 4\ 5}{\mathrm{ATATAT}} & \Rightarrow c_1 &= 101010 \\ w_2 &= \mathrm{AGGAGG} & \Rightarrow c_2 &= 100100 \\ w_3 &= \mathrm{ACAGAT} & \Rightarrow c_3 &= 100000 \end{aligned}$$

Définition 3.8. *Soit* Γ_m *l'ensemble des autocorrélations de tous les mots* w *d'une certaine longueur* m *:*

$$\Gamma_m := \{c \in \{0,1\} \cap \{m\} \mid \exists w \in \Sigma^m : c = P(w)\} \tag{3.9}$$

Les autocorrélations dans Γ_m peuvent être partitionnées selon leur *période primaire*, les classes ainsi générées pour tout $0 \leq p < m$ sont dénotées par $\Gamma_{m,p}$ où p est ici un entier $\in [0, m-1]$.

Définition 3.9. *Soit* $\Psi(c,\sigma)$ *la* population *d'une autocorrélation c sur l'alphabet de* σ *lettres. La* population *d'une autocorrélation donnée correspond au nombre total de séquences qu'on peut construire avec un alphabet* Σ *qui ont cette même autocorrélation.*

Exemple 3.3. La population.
Pour un alphabet Σ = {A,C,G,T} de cardinal $\sigma = 4$ et une autocorrélation $c = 101$, on peut construire l'ensemble des mots {ACA,AGA,ATA,CAC,CGC,CTC,GAG,GCG,GTG,TAT,TCT,TGT} et de ce fait, $\Psi(c,\sigma) = 12$.

3.1.2 Définitions des modèles

Considérons d'une part un génome cible X de taille n, l'alphabet $\Sigma = \{A,C,G,T\}$ et σ le cardinal de cet alphabet. D'autre part, considérons un ensemble de q séquences $\{w_1,\ldots,w_q\}$ de longueur m.
Nous voulons déterminer les probabilités suivantes :

i/ $A(m,n)$: la probabilité qu'une séquence aléatoire w de longueur m ne soit pas localisée sur le génome aléatoire X de taille n ;

ii/ $B(m,n)$: la probabilité qu'une séquence aléatoire w de longueur m ne soit localisée qu'une seule fois sur le génome aléatoire X de taille n.

Définissons formellement ces probabilités en utilisant le comptage $N(w)$ correspondant au nombre de localisations de w sur X :

$$A(m,n) = \mathbb{P}\{N(w) = 0\} \quad \text{et} \quad B(m,n) = \mathbb{P}\{N(w) = 1\} \tag{3.10}$$

Du fait que la fréquence d'apparition des *reads* sur un génome est censé être petite, nous pouvons assimiler le comptage $N(w)$ à une somme de variables aléatoires de Bernoulli $Y_i(w)$ suivant une loi de Poisson (définition 3.2) pour peu que : l'espérance du comptage soit petite, la séquence soit suffisamment grande et les indicatrices $Y_i(w)$ soient indépendantes. Cependant, en considérant d'emblée une séquence génomique X comme un modèle de Bernoulli (définition 3.1), nous ne respectons pas la dernière condition. En effet, les indicatrices $Y_i(w)$ ne sont pas indépendantes, premièrement parce que les lettres $X[i]$ composant la séquence X ne sont pas indépendantes mais surtout parce que $Y_i(w)$ et $Y_{i+d}(w)$ font intervenir des lettres communes de X avec une distance $d < m$.

Cependant, la séquence génomique X doit être suffisamment grande pour que les mots w soient *rares*. En d'autres termes, la longueur m des *reads* doit respecter la condition $m = O(log(n))$;

Finalement, une approximation par une loi de Poisson est envisageable pour répondre au problème. Néanmoins, lors du positionnement de *reads* sur un génome, il est évident que les occurrences d'un même *read* peuvent se chevaucher. Ainsi le comptage $N(w)$ ne peut être approché par une loi de Poisson « simple » mais par une loi de Poisson pour les mots à *occurrences chevauchantes*, dite loi de Poisson composée, sur la séquence X (définition 3.3).

Comme nous avons pu le voir dans le chapitre 2, il existe plusieurs approches, en transcriptomique, pour séquencer des *reads*. Une première approche est le RNA-Seq pour laquelle il n'y a aucune restriction particulière dans la composition des *reads*. Une deuxième approche possible est la DGE où, au contraire, il y a une restriction. Du fait que les *reads* sont ancrés, leurs quatre premiers nucléotides sont connus et fixés. Nous proposons donc deux modèles :

i/ un premier modèle standard pour les *reads* qui n'ont pas de propriétés particulières sur leur composition,

ii/ un deuxième modèle adapté pour les *reads* possédant un site de restriction préfixe connu.

3.1.2.1 Modèle standard

Nous étudions d'abord les occurrences de *trains*, puis les nombres de *wagons* par *train* (définition 3.4). En effet, par construction, les différents *trains* ne vont pas se chevaucher dans la séquence X car cela voudrait dire que chacun de ces *trains* constituent des *wagons* d'un *train* encore plus grand. La loi de comptage du *nombre de trains* est ainsi approchée par une loi de Poisson puisque les *trains* sont indépendants (définition 3.2).

Finalement, nous pouvons modéliser nos probabilités $A(m,n)$ et $B(m,n)$ avec une approximation de notre comptage $N(w)$ en suivant une loi de Poisson composée $\mathscr{L}_{pc}(\mathbb{E}[T], a)$, où $a(w)$ est la somme des probabilités des portions de w qui se chevauchent sur la séquence X et $\mathbb{E}[T(w)]$ l'espérance du *nombre de trains* pour w (équation 3.6) :

$$A(m,n) = e^{-\mathbb{E}[T(w)]} \quad \text{et} \quad B(m,n) = (1-a(w)) \times \mathbb{E}[T(w)] \times e^{-\mathbb{E}[T(w)]} \tag{3.11}$$

La difficulté se situe sur l'approximation du comptage $\mathbb{E}[T(w)]$ et plus particulièrement sur le calcul de la composante $a(w)$ (équation 3.7), car pour une collection de q séquences $(w_1, \ldots, w_q)$ à positionner sur un génome X, l'expression des recouvrements $a(w_i), 1 \leq i \leq$

q nécessite la connaissance de l'ensemble des *périodes primaires* de $\{w_1,\ldots,w_q\}$ (définition 3.5).
Pour surmonter cette difficulté, nous nous intéressons aux *autocorrélations* associées aux séquences (définition 3.8). Rappelons que l' *autocorrélation* d'une séquence w est en fait, ni plus ni moins que la représentation binaire de l'ensemble de ses *périodes* (définition 3.7).

D'après notre hypothèse de départ, le génome X est une séquence de Bernoulli de type M0. Par conséquent, quel que soit le nucléotide à une position j quelconque sur X, chaque nucléotide de Σ a la même probabilité d'apparaître à la position suivante $j+1$. Cela nous permet de lever un premier niveau de difficulté en simplifiant la composante $a(w)$:

$$a(w) = \sum_{p\in\Pi(w)} \sigma^{-p}$$

Par conséquent w ne dépend que de $\Pi(w)$. Nous pouvons donc utiliser les autocorrélations associées aux séquences pour calculer cette composante.

En utilisant un algorithme qui, à partir d'une longueur m, calcule l'ensemble des autocorrélations Γ_m de longueur m, nous récupérons les *périodes primaires* et la *population* associées à chaque autocorrélation possible pour des mots de longueur m (définition 3.7) [Rivals et Rahmann, 2003]. Toutes ces informations nous permettent d'approcher $a(c)$, en faisant une moyenne : somme des probabilités des recouvrements des *périodes primaires* sur la somme des *populations*, pour chaque autocorrélation de Γ_m. Cette procédure nous donne donc une probabilité moyenne sur l'ensemble des mots qui se chevauchent sur la séquence X :

$$a(c) = \frac{\sum_{c\in\Gamma_m} \sum_{p\in\Pi(c)} \sigma^{-p}\cdot\Psi(c,\sigma)}{\sum_{c\in\Gamma_m} \Psi(c,\sigma)} \tag{3.12}$$

Sachant que notre génome X est de type M0, la probabilité d'apparition de tous les nucléotides de Σ est la même, l'équation 3.5 devient :

$$\mathbb{E}[N(w)] = \frac{n-m+1}{\sigma^m} \tag{3.13}$$

Au final, nous pouvons intégrer le comptage $\mathbb{E}[N(w)]$ (équation 3.13) et la probabilité de recouvrement $a(c)$ (équation 3.12) à notre modèle 3.11 pour ainsi calculer les probabilités $A(m,n)$ et $B(m,n)$. Ces dernières correspondent à des probabilités de localisations de séquences (de longueur m) sur un génome (de taille n), sous l'hypothèse que la distribution en nucléotides du génome soit équirépartie, ce qui est relativement vrai chez les vertébrés.

3.1.2.2 Modèle adapté

Le modèle précédent peut être adapté dans le cas des séquences qui ont un préfixe prédéfini, comme celles générées par la technique DGE, où le préfixe de longueur 4 (CATG pour l'enzyme Nla3, GATC pour Sau3a par exemple). La démarche pour ce modèle est globalement la même que pour le modèle standard excepté le comptage $\mathbb{E}[N(w)]$ (voir équation 3.13) :

1. les quatre premiers nucléotides de chacune des séquences sont connus, et
2. ce préfixe est identique dans toutes les séquences.

Par conséquent, pour une séquence w de type DGE, seuls $m-4$ nucléotides sont construits de façon aléatoire (le suffixe en position 4) car le site de restriction δ est connu et de longueur 4. De plus, les séquences ne peuvent plus se positionner à $(n-m+1)$ endroits sur le génome X mais seulement à $n_X(\delta)$ endroits, ce qui correspond au nombre de fois où le site de restriction δ est présent dans X. Nous obtenons donc :

$$\mathbb{E}[N(w)] = \frac{n'}{\sigma^{m-4}} \text{ avec } n' = n_X(\delta) \tag{3.14}$$

3.2 Estimations des erreurs de séquences

Des millions de *reads* sont générés dans une expérience haut débit. Même si par position l'erreur est faible (une erreur de $\simeq 1\%$ estimée chez Illumina® http://www.illumina.com/technology/sequencing_technology.ilmn), un grand nombre de *reads* sont affectés par des erreurs de séquences : si une erreur est présente dans un *read*, alors il est erroné. L'influence de ces erreurs est souvent problématique dans l'exploitation des résultats.

Par exemple, dans une expérience SHD, une première étape consiste à positionner les *reads* sur une séquence de référence, à l'aide d'un outil de *mapping* (voir section 2.4 du chapitre 2). Or, avec un logiciel de positionnement exact, une proportion de *reads* ne se localise pas. Dans cette proportion il y a, d'un côté ceux qui sont affectés par une erreur de séquençage, de l'autre ceux qui sont affectés par des causes biologiques du type SNP ou jonction d'épissage (voir chapitre 2). Pourtant, il est nécessaire de distinguer les erreurs des causes biologiques.

Nous suggérons une méthode qui, à partir d'une collection de *reads*, estime ceux qui sont affectés par une erreur. Pour ce faire, nous positionnons les *reads* exactement sur le génome avec le logiciel MPSCAN [Rivals *et al.*, 2009]. Le paramètre important dans cette

procédure est la longueur m des séquences. En effet, plus la séquence est longue et plus la probabilité qu'un de ses nucléotides soit erroné est grande.

3.2.1 Occurrence et tag

Pour une technique de séquençage dite *quantitative*, telle que la DGE, il est de coutume de différencier les séquences générées (les *reads*) du nombre de séquences différentes générées (les signatures). En effet, la quantification d'un *read* donné est représentée par le nombre de fois qu'il est séquencé.

Formellement, *les* reads peuvent être représentés par **une collection d'occurrences** et les *signatures* par **un ensemble de tags**. Cet ensemble regroupe toutes les *occurrences* qui sont identiques dans la collection, afin de n'identifier que les reads différents. De plus, on mémorise le **nombre d' occurrences** de chaque *tags* dans la variable *#occ.*

Exemple 3.4. L'occurrence et le *tag*.
Prenons cinq *reads* AAC, CAA, AAC, AAC et AAA de longueur $m = 3$. À partir de ces *reads*, nous pouvons représenter respectivement la collection d'*occurrences*, l'ensemble des *tags* et la pondération *#occ.* associée à chaque *tag* de cette façon :

1. (AAC,CAA,AAC,AAC,AAA),
2. {AAC,CAA,AAA},
3. #AAC=3, #CAA=1, #AAA=1.

3.2.2 Modélisation des estimateurs

Nous présentons des estimateurs de différentes probabilités sur une séquence de longueur m. La séquence peut être vue comme un *tag* où une *occurrence* (voir sous-section 3.2.1). Toutefois, si la séquence est une *occurrence*, elle est erronée si elle contient au moins une erreur. Par contre, si la séquence est un *tag*, elle n'est erronée que si toutes ses *occurrences* sont erronées (toutes les occurrences d'un *tag* sont par définition identiques). Par conséquent, pour qu'un *tag* ne soit pas erroné, il suffit qu'une seule de ses *occurrences* ne soit pas erronée.

De plus, dans cette étude, nous partons de l'hypothèse qu'une séquence n'est erronée que si son nombre d'occurrences *#occ.* est faible. Au vu du taux d'erreurs par position, il est en effet très peu probable qu'une même erreur se produise, au même endroit, sur plusieurs *reads* identiques. Il existe donc un seuil à partir duquel un *read* ne peut plus être considéré comme erroné même s'il ne se localise pas sur sa séquence de référence.

Soit Q une collection de q *reads* $(r_1, r_2, \ldots, r_q)$ de longueur m et G un génome de taille n. Nous pouvons considérer les quatre probabilités suivantes :

- $\mathscr{S}(m)$: la probabilité qu'un *read*, de longueur m, contienne au moins une erreur ;
- $\mathscr{X}(m)$: la probabilité *a priori* qu'un *read*, de longueur m, ne soit pas localisé sur G ;
- $\mathscr{R}(m)$: la probabilité qu'un *read* qui ne contient pas d'erreur, de longueur m, ne soit pas localisée sur G ;
- $\mathscr{M}(m)$: la probabilité qu'un *read* erroné, de longueur m, soit localisée sur G.

Nous voulons calculer la probabilité $\mathscr{S}(m)$ à partir des probabilités $\mathscr{X}(m)$, $\mathscr{R}(m)$ et $\mathscr{M}(m)$ car ces dernières sont estimables contrairement à la première. En effet, la composante $\mathscr{X}(m)$ peut être estimée par un pourcentage de *reads* non localisés sur G, à partir d'une expérience sur des données réelles. Pour estimer la composante $\mathscr{R}(m)$, nous avons besoin des *reads* qui ne contiennent aucune erreur. Pour ce faire, nous construisons l'ensemble des *tags* T à partir de la collection Q, puis nous utilisons les pondérations $\#occ.$ pour chaque t_i avec $1 \leq i \leq t$ défini en sous-section 3.2.1. Nous partons du principe qu'à partir d'un certain seuil les *reads* sont sans erreur ; cette supposition est biologiquement fondée. La composante $\mathscr{M}(m)$ peut aussi être estimée en partant de l'hypothèse qu'une séquence erronée ne contienne qu'une seule erreur. Là encore, cette deuxième supposition est justifiée puisque les *reads* sont relativement courts (< 21 *pb*) et l'erreur par position est théoriquement très faible ($\simeq 1\%$). Il suffit donc de générer une erreur sur chacun des *reads* qui ne contiennent pas d'erreur, c'est-à-dire ceux que nous avons choisis pour $\mathscr{R}(m)$, puis de compter le pourcentage des *reads* qui sont localisés sur G.

Pour relier toutes ces composantes les unes avec les autres, il faut se demander dans quelle mesure un *read* n'est pas localisé sur un génome ?

i/ soit, il ne contient pas d'erreur mais une mutation biologique ponctuelle (SNP, modification post-transcriptomique, indel, etc...) par rapport à G, auquel cas il n'est pas localisé. Formellement cette situation est donnée par $(1-\mathscr{S}(m))\cdot\mathscr{R}(m)$;

ii/ soit, il contient une erreur et n'est pas localisé sur G. Dans ce cas, nous obtenons $(1-\mathscr{M}(m))\cdot\mathscr{S}(m)$.

Nous pouvons noter que les cas *i/* et *ii/* sont exclusifs puisque dans la condition *i/* nous supposons que le *read* est sans erreur. En conséquence, tous les *reads* qui contiennent une erreur et une cause biologique telle qu'un SNP seront classés en *ii/*.
Finalement, comme *i/* et *ii/* sont exclusifs, nous pouvons écrire :

$$\mathscr{X}(m) = (1-\mathscr{S}(m))\cdot\mathscr{R}(m) + (1-\mathscr{M}(m))\cdot\mathscr{S}(m). \tag{3.15}$$

Comme nous savons déjà estimer $\mathscr{X}(m)$, $\mathscr{R}(m)$ et $\mathscr{M}(m)$, nous pouvons en déduire $\mathscr{S}(m)$:

$$\mathscr{S}(m) = \frac{\mathscr{X}(m)-\mathscr{R}(m)}{1-\mathscr{M}(m)-\mathscr{R}(m)}. \tag{3.16}$$

3.2.3 Calcul de l'erreur standard

Afin de donner une estimation la plus précise possible de $\mathscr{S}(m)$ nous proposons de calculer une estimation de son erreur standard $\alpha(m)$, en fonction de la longueur m [1]. L'idée est de mesurer un intervalle de confiance borné respectivement par la probabilité dans le pire comme dans le meilleur des cas. La technique expérimentale couramment utilisée pour donner cet intervalle est **le bootstrap** [Efron et Gong, 1983]. Le principe est le suivant :

1. construire un nombre z d'échantillons,
2. calculer l'écart type empirique de tous les échantillons $\mathscr{S}_i(m)$, pour tout $1 \leq i \leq z$, par rapport à un échantillon $\overline{\mathscr{S}}(m)$ moyen.

Si on formalise, nous obtenons :

$$\alpha(t) = \sqrt{\frac{\sum_{i=1}^{z} \left(\mathscr{S}_i(t) - \overline{\mathscr{S}}(t)\right)^2}{z-1}}. \tag{3.17}$$

Remarque 6. *Nous divisons ici par* $(z-1)$ *élements car la moyenne* $\overline{\mathscr{S}}(t)$ *est une estimation, c'est-à-dire que sa valeur exacte est inconnue. L'écart type est donc donné sous forme corrigée.*

Notons que plus z est grand et plus l'erreur standard est précise.

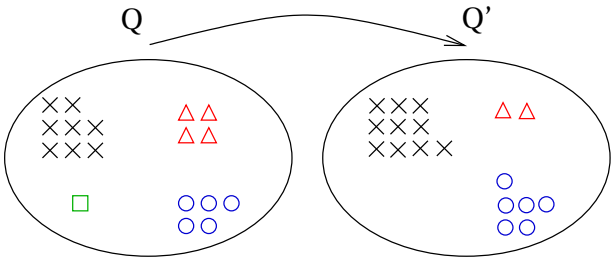
3.2.3.1 Algorithme du bootstrap sur nos estimateurs

Pour construire chaque échantillon $1 \leq i \leq z$, nous devons tout d'abord reconstruire à chaque étape i une nouvelle collection Q' à partir de la collection Q de départ, en faisant des tirages avec remises des séquences $s_j, 1 \leq j \leq q$, sachant que chaque Q' doit contenir le même nombre d'éléments que Q. Dans la FIGURE 3.1, nous avons un aperçu de ce que nous pourrions obtenir à partir d'une collection de 18 éléments.

Ensuite, une fois qu'une nouvelle collection Q' est construite, il suffit de relancer nos estimateurs dessus. D'ailleurs on peut noter que notre hypothèse sur la pondération *#occ.* reste vraie, ce qui est un critère indispensable. En effet, les tirages étant avec remises, la probabilité de piocher un élément majoritairement présent est plus grande. Par conséquent, un *tag* qui possède un nombre élevé d'occurrences dans une collection Q conservera un nombre élevé d'occurrences dans une collection Q'. Plus visuellement, en nous appuyant de la FIGURE 3.1, si on assigne des *tags* aux symboles (carré, rond, triangle, croix), nous sommes forcés de constater que nous avons plus de chance de piocher une croix qu'un carré.

1. Ne pas confondre erreur standard qui représente une marge d'erreur sur un calcul et erreur de séquence qui représente l'erreur produite lors du séquençage.

FIGURE 3.1 : Redistribution d'une collection Q par bootstrap.



3.2.3.2 Proportions de vrais/faux positifs/négatifs

À partir de nos estimateurs $\mathscr{X}(m)$, $\mathscr{R}(m)$, $\mathscr{M}(m)$ et $\mathscr{S}(m)$, nous pouvons estimer quelques proportions intéressantes. Partons du principe qu'une séquence localisée sur le génome G est un candidat **positif** et une séquence erronée est un **faux** candidat. Nous pouvons alors écrire :

1. La proportion de **faux positifs**

$$\mathscr{V}(m) = \frac{\mathscr{S}(m) \cdot \mathscr{M}(m)}{1 - \mathscr{X}(m)}. \tag{3.18}$$

2. La proportion de **vrais positifs**

$$\mathscr{W}(m) = \frac{(1 - \mathscr{S}(m)) \cdot (1 - \mathscr{R}(m))}{1 - \mathscr{X}(m)}. \tag{3.19}$$

3. La proportion de **faux négatifs**

$$\mathscr{Y}(m) = \frac{\mathscr{S}(m) \cdot (1 - \mathscr{M}(m))}{\mathscr{X}(m)}. \tag{3.20}$$

4. La proportion de **vrais négatifs**

$$\mathscr{Z}(m) = \frac{(1 - \mathscr{S}(m)) \cdot \mathscr{R}(m)}{\mathscr{X}(m)}. \tag{3.21}$$

Notons qu'il ne faut pas confondre $\mathscr{V}(m)$ et $\mathscr{M}(m)$. La proportion $\mathscr{V}(m)$ correspond au pourcentage de *reads* erronés parmi ceux qui sont localisés sur G, alors que $\mathscr{M}(m)$ correspond au pourcentage des *reads* localisés sur G parmi ceux qui sont erronés. En fait, on peut voir $\mathscr{M}(m)$ comme une surestimation expérimentale de la composante $(1 - \mathscr{A}(m))$ (en section 3.1.2) [2].

3.2.3.3 Estimation de l'erreur par position

Lorsque nous évaluons $\mathscr{S}(m)$ sur l'ensemble des *occurrences*, il est possible d'évaluer l'erreur par position (à l'échelle nucléotide). Notons p la probabilité d'avoir une telle erreur. Dans le cas où

2. une séquence erronée n'est pas tout à fait une séquence construite au hasard

une *occurrence* de longueur m ne contient pas de nucléotide erroné, sa probabilité $\mathscr{S}(m)$ est égale à $(1-p)^m$. Par déduction, une *occurrence* qui contient au moins une erreur a comme probabilité $\mathscr{S}(m) = 1-(1-p)^m$, ce qui nous donne :

$$p = 1 - \exp\left(\frac{\log(1-\mathscr{S}(m))}{m}\right). \tag{3.22}$$

Remarque 7. *L'erreur par position correspond à l'erreur estimée par les SHD (≃ 1% chez Illumina®).*

3.3 Annotation des transcrits

L'annotation des gènes et, plus généralement des transcrits, a pris une autre dimension depuis l'utilisation des SHD. Des techniques capables de mesurer l'expression des gènes, comme la DGE ou le RNA-Seq, sont devenues des choix de premier ordre. La DGE, qui combine la génération massive de transcrits avec une identification par un *tag*, suscite la naissance d'un nombre exorbitant de nouvelles unités transcrites [Robinson *et al.*, 2010; Blow, 2009]. Il semblerait néanmoins qu'une partie de cette transcription soit artefactuelle [van Bakel *et al.*, 2010], alors comment améliorer la procédure pour permettre de distinguer les unités transcriptionnelles biologiquement valides ?

Dans un premier temps, nous proposons une méthode bioinformatique permettant de sélectionner les unités transcrites, qu'on appellera des « prédictions ». Puis dans un second temps, nous détaillons la méthode biologique permettant de valider ces prédictions.

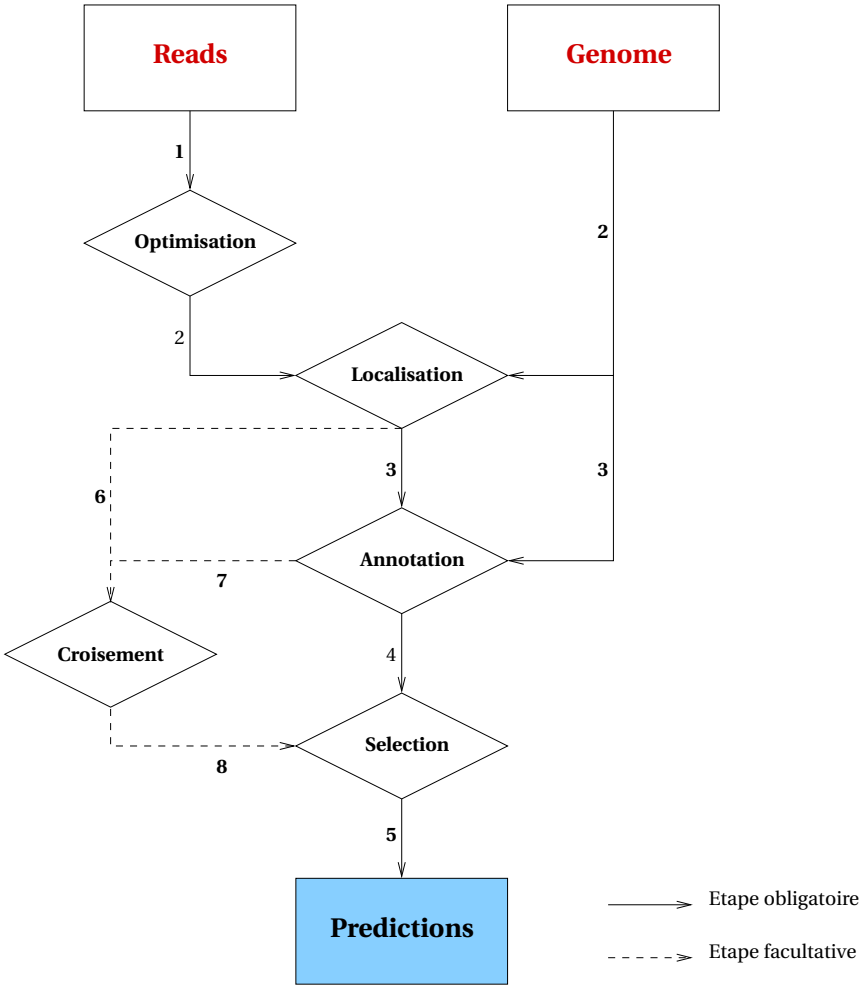
3.3.1 Pipeline transcriptomique

Dans le domaine de l'informatique, un pipeline est un assemblage de différents programmes (ou différents algorithmes). Notre pipeline transcriptomique explore, de la façon la plus exhaustive possible, les informations relatives à des *reads* transcriptomiques (DGE ou RNA-Seq). Le but est de ne sélectionner que les *tags* qui ne contiennent pas d'erreurs et qui sont associés à des régions du génome auxquelles on veut s'intéresser, comme par exemple les régions non codantes. La sélection dépend donc de certains critères qui sont mentionnés à l'exécution du pipeline par l'utilisateur. Nous allons détailler chacune des étapes du pipeline, illustrées dans la FIGURE 3.2.

3.3.1.1 L'optimisation

L'optimisation est une étape préliminaire à la localisation mais c'est une étape essentielle. Elle consiste à appliquer les méthodes des sections 3.1 et 3.2 sur la collection de *reads* afin d'en déduire la meilleure stratégie pour localiser ces *reads* sur le génome, et ainsi diminuer le *bruit de fond* et les erreurs de séquences. Dans la FIGURE 3.2, à partir de notre collection de *reads*, nous construisons notre ensemble de *tags* à partir de cette optimisation.

FIGURE 3.2 : Organigramme du pipeline transcriptomique.



3.3.1.2 La localisation

La localisation est l'étape qui consiste à positionner les *reads* sur un génome. La stratégie de *mapping* à utiliser dépend beaucoup de la technique avec laquelle les *reads* ont été séquencés. Dans le cas de la DGE couplée au génome humain, par exemple, une stratégie de « match parfait » semble être la plus appropriée car les *reads* ont une longueur de 21 nucléotides, ce qui semble être une bonne longueur pour un positionnement optimisé, d'après les résultats en section 3.4. Dans la section 2.4 du chapitre 2, un état de l'art des outils de *mapping* est proposé afin d'utiliser celui qui est adéquat à son type de données et positionner au mieux les *reads* sur un génome.

3.3.1.3 L'annotation

Pour annoter nos transcrits nous nous appuyons sur le système d'annotation automatique Ensembl (http://www.ensembl.org). Avec le UCSC (http://genome.ucsc.edu/), ils constituent les deux plus grandes sources d'informations sur l'annotation des génomes. Ensembl se présente d'abord comme un navigateur de génomes (Genome Browser) permettant d'explorer et de visualiser à différents niveaux les génomes de nombreux organismes. Mais il contient aussi une base de données ouverte dans laquelle on peut librement venir puiser, soit directement, soit à travers une interface de programmation documentée (http://www.ensembl.org/info/docs/Pdoc/ensembl/index.html). C'est à travers cette interface, que nous proposons un algorithme pour associer un transcrit à un *tag*.
Après l'étape 2 du pipeline (FIGURE 3.2) nous nous retrouvons avec trois catégories de localisations :

1. les *reads* localisés une seule fois sur le génome,
2. les *reads* localisés plusieurs fois sur le génome,
3. les *reads* non localisés sur le génome.

Seule la première catégorie nous intéresse car cela devient compliqué d'associer un transcrit à un *tag* quand ce dernier est localisé plusieurs fois sur le génome, et encore plus lorsqu'il n'est pas du tout localisé.

Nous considérons donc ce sous ensemble de *tags* et à partir de leur unique localisation génomique, nous déterminons s'ils sont associés à une région annotée par un gène, un EST ou à une région qui n'est pas annotée. Cependant, un gène peut posséder plusieurs transcrits, alors que nous voulons associer un unique transcrit pour chacun des *tags*. Sachant, que les *reads* sont issus du transcriptome, et qu'ils sont plutôt amorcés sur la partie $3'$ des ARN pour la DGE, nous pouvons établir des priorités dans le choix du transcrit : exon contre intron, UTR contre CDS.
Un dernier point à mentionner est l'existence de transcrits qui sont antisens à un gène (sur le brin opposé du gène annoté) [Quéré *et al.*, 2004; Morris, 2009]. Il est donc possible d'associer un *tag* à un transcrit qui est antisens à un gène, tout en privilégiant celui qui est sur le même brin que le gène (s'il existe).

Finalement, les priorités sont attribuées de 1 à 10 selon trois catégories. La première : le *tag* et le transcrit sont sur le même brin : la priorité est de 1 si le *tag* est dans l'UTR de l'exon, 2 s'il est dans le CDS de l'exon, 3 s'il est à cheval entre l'exon et l'intron (nous parlerons d'inxon), puis 4 s'il est dans l'intron. La deuxième : pour des brins différents, les priorités sont respectivement, $5, 6, 7$ et 8. La troisième : si le *tag* n'est pas annoté dans un gène, la priorité est de 9 s'il croise un EST et 10 si aucun des critères précédents n'est respecté. En se basant sur ces priorités, nous proposons l'algorithme 1, il calcule l'annotation d'un *tag* donné.

Algorithme 1 : Algorithme d'annotation du *tag* s sur le génome X

Data : $loc(s)$: la localisation génomique de s sur X
Result : ann(s) : l'annotation de s sur X

```
1 begin
    // p prend la plus mauvaise priorité par défaut
2   p ← 10;
    // g est un gène, t est un transcrit
3   foreach g ∈ G tq [loc_deb(g) ≤ loc(s) ≤ loc_fin(g)] do
4     forall t ∈ g do
        // Priority(t, s) est une fonction qui attribue une priorité au
           transcrit t en fonction de son annotation sur s
5       newP ← Priority(t, s);
6       if newP < p then p ← newP;
    // Annotation(p) est une fonction qui attribue l'annotation du tag s
       en fonction de sa priorité p
7   ann(s) ← Annotation(p);
8   return (ann(s));
9 end
```

3.3.1.4 Le croisement

Cette partie du pipeline a pour but de rassembler le maximum d'informations disponibles autour des *tags* qui ont les critères recherchés par l'utilisateur ; un critère possible serait de rechercher les nouvelles régions transcrites d'un génome. Dans ce cas, un *tag* qui serait exprimé et localisé dans une région du génome qui ne contient pas d'annotation constituerait une prédiction. Il est évident que, plus il y aura d'arguments dans la balance de cette prédiction, plus la probabilité qu'elle soit valide augmentera. Cette étape du pipeline est facultative, mais peut donc être fortement conseillée, selon l'attente ou les besoins de l'utilisateur.

Il y a plusieurs façons d'emmagasiner des informations complémentaires pour une prédiction, on peut cependant les répertorier en deux catégories :

1. les informations intrinsèques à la prédiction, par entrecroisement des données de transcriptome,
2. les informations d'orthologie, par entrecroisement de différents génomes.

L'entrecroisement de deux *reads*. Il se fait entre deux *reads* provenant de sources différentes. Selon la technique avec laquelle chacun a été généré, nous procédons d'une façon différente :

DGE vs DGE : La DGE est une technique quantitative qui nous permet de comparer l'expression de deux transcrits (identifiés par chaque *tag*) dans deux échantillons différents. Cette caractéristique est importante, voire indispensable dans certains cas comme l'étude de la

différence d'expression entre un tissu normal et un tissu tumoral. La comparaison de deux *tags*, issus de la DGE, nous permet aussi de diminuer les artefacts. Il est en effet moins probable qu'un *tag* soit erroné dans deux expériences différentes, plutôt qu'une (plus un *tag* est séquencé moins il a de chance d'être erroné). La banque de donnée Sage Genie (ftp://ftp1.nci.nih.gov/pub/SAGE) regroupe toutes les informations relatives à tous les *tags* (humain et souris) qui ont été exprimés au moins une fois dans une expérience.

DGE vs RNA-Seq : Le RNA-Seq est une technique qui est complémentaire à la DGE. Elle est assurément moins quantitative du fait qu'on ne séquence pas un endroit précis du transcrit mais qu'en revanche, on séquence la totalité du transcrit. Il y a donc un double intérêt de croiser les deux techniques : i/ vérifier que le *tag* DGE se situe bien en 3′ des séquences RNA-Seq ; ii/ estimer, à la fois, la structure du transcrit et son niveau d'expression dans une expérience.

DGE vs tiling arrays : Les tiling arrays nous permettent aussi d'étudier le transcriptome mais d'un œil différent car la technologie utilisée est plus proche d'une puce à ADN que d'un séquenceur. Toutefois, si des *tags* générés par DGE (ou RNA-Seq) sont déjà identifiés comme transcrits par les expressions de tiling arrays, la probabilité qu'ils soient valides augmente du fait qu'il y ait moins de chance que deux technologies différentes produisent les mêmes erreurs.

L'entrecroisement de deux génomes. Ce croisement est important, non seulement pour ajouter du poids à la validité du *tag*, mais aussi pour donner une information supplémentaire sur la zone qui est transcrite. Le croisement peut se faire à deux niveaux :

Localisation : si le *tag* est localisé sur les deux génomes, alors il existe une synténie entre les deux espèces, pour ce transcrit. Pour un transcriptome complet, on peut mesurer un certain degré de conservation entre les transcriptomes.

Annotation : cette étape est exclusivement réservée à la DGE. L'idée est d'étudier la différence d'expression d'un *tag* qui serait exprimé, à la fois, chez les deux espèces (avec la possibilité de s'intéresser à un tissu cellulaire particulier). Ce constat nous informe sur le rôle fonctionnel de certains gènes spécifiques (ou non) à une espèce.

3.3.1.5 La sélection

La sélection est le résultat du pipeline pour un échantillon de *reads* lorsque celui-ci est soumis aux différents filtres de chacune des étapes précédentes (Optimisation, Localisation, Annotation et Croisement). Ces filtres sont établis en fonction des paramètres mentionnés par l'utilisateur à l'exécution du pipeline. Imaginons, par exemple, que nous sommes en la possession d'une collection de *reads* issue du transcriptome d'une lignée de cellules souches humaines, puis que ces *reads* ont été générés avec la technique DGE. On peut régler les paramètres du pipeline pour ne conserver que les *tags* qui ne contiennent pas d'erreur de séquences, qui sont localisés dans une région intergénique chez l'humain, et qui sont aussi exprimés dans les cellules souches de la souris.

3.3.2 Méthode biologique pour valider l'expression des transcrits

Pour valider l'expression des transcrits, nous commençons par extraire des *tags* à l'aide de la phase de sélection en section 3.3.1.5. Ensuite, nous reconstruisons manuellement la zone transcrite en 5' du *tag* et si possible à l'aide de *reads* RNA-Seq ou encore des informations de tiling arrays qu'on aurait pu prendre en compte lors de la phase de croisement en section 3.3.1.4. Les candidats potentiels qui sont ainsi choisis et créés sont, à ce stade, des prédictions. Ces dernières doivent être validées biologiquement. D'abord les ADNc cibles doivent être construits, puis les prédictions sont vérifiées.

3.3.2.1 Création des ADNc cibles

Tout d'abord, il faut sélectionner des tissus dans lesquels sont exprimées les prédictions. Avec l'étape de croisement du pipeline (section 3.3.1.4), nous pouvons connaître les tissus où le *tag* est exprimé avec la banque de données *Sage Génie*. Pour notre part, nous nous intéressons aux nouveaux transcrits qui pourraient être exprimés de façon différentielle dans les cellules humaines normales ou cancéreuses. Nous avons constitué une collection d'ADNc provenant des lignées suivantes : U937 (leucémie myélomonocytaire), K562 (leucémie myéloide chronique), MDAPca1 (cancer de la prostate), SH-SY5Y (neuroblastome), MCF7 (cancer du sein), HEK (cellules embryonnaires de rein). Des lignées de cellules souches embryonnaires (ESC ; HD129) et de cellules pluripotentes induites (iPS ; M4C2) ont été généreusement fournies par l'équipe de John Devos (IRB, Montpellier).

Les ARN totaux ont ensuite été purifiés à l'aide du kit *RNeasy* de Qiagen®, puis la qualité a été analysée par la technologie *Agilent* en utilisant le *Bioanalyzer 2100* (Plateau Q-PCR, UM2). Enfin, une réaction de reverse transcription a été effectuée à partir de 1µg d'ARN totaux avec des amorces aléatoires à partir du kit *High-capacity cDNA Archive kit* de Applied Biosystems®.

3.3.2.2 Validation des prédictions

À partir des prédictions, nous construisons des amorces à l'aide du programme *Primer3Plus* (http://www.bioinformatics.nl/cgi-bin/primer3plus/primer3plus.cgi). Nous pouvons paramétrer le programme afin qu'il prenne en compte la taille des amplicons et la température d'hybridation pour la PCR. Nous avons ainsi conçu des amorces permettant une PCR avec une température d'hybridation de 60°C et un amplicon de 200 à 250 *pb*.

La réaction de PCR en temps réel est réalisée avec la technique du *SYBRgreen I* et le *light cycler 480* de Roche®. Des séries de 384 puits ont été réalisées à l'aide d'un protocole de pipettage automatique avec le robot PerkinElmer®.

3.4 Études expérimentales et résultats

Dans les résultats des sous-sections suivantes nous utilisons des données issues de séquençage. Pour la collection de *reads* que nous nommons SAGE-Sanger, nous utilisons les librairies dites de type SAGE séquencées avec la technologie Sanger. Les identifiants des librairies sont compris entre GPL1485 et GPL5624 et elles sont disponibles sur la plate-forme Gene Expression Omnibus (GEO) à l'adresse http://www.ncbi.nlm.nih.gov/geo, sachant que toutes ces librairies sont disponibles sur CAGP (Sage genie ftp://ftp1.nci.nih.gov/pub/SAGE/). Pour la collection SAGE-Illumina®, nous utilisons une librairie privée de type DGE. Elle a été séquencée avec la technologie Illumina® par l'équipe Skuld-Tech®. Pour la collection CAGE-Sanger, nous utilisons les données CAGE disponibles dans la publication de Kawaji *et al.* [Kawaji *et al.*, 2006]. Pour la collection ChIP-Seq-Illumina®, nous utilisons des *reads* génomiques (à l'inverse des trois autres qui sont des collections de *reads* transcriptomiques). Cette librairie GSM325935 est disponible sur GEO.

Enfin, en collaboration avec l'équipe de John Devos (IRB, Montpellier), des ARN de cellules souches humaines ont été séquencés avec Illumina® et la technique DGE. À l'aide de cette librairie DGE-StemCell, nous avons pu identifier et valider de nouvelles régions transcrites (voir la section 3.4.4). Pour accompagner nos *reads* DGE-StemCell, nous avons aussi utilisé une collection de 40 millions de *reads* de longueur 75 *pb* de type RNA-Seq de la technologie Illumina® provenant d'une lignée K562 du génome humain. Les données sont disponibles sur RGASP (numéro d'accession *GM12878* sur www.sanger.ac.uk/PostGenomics/encode/RGASP.html avec la permission de B. Wold).

Le génome humain sur lequel nous localisons les *reads* est la version hg18, NCBI Build 36.1 disponible sur le genome browser UCSC (http://genome.ucsc.edu).

Pour prédire la localisation génomique d'origine de nos *tags* dans chacune de nos expériences ci-dessus, nous utilisons le logiciel MPSCAN [Rivals *et al.*, 2009]. C'est un logiciel de « mapping » rapide et peu coûteux en mémoire destiné à faire de la localisation exacte de courtes séquences sur un génome de référence. Il n'est fondé sur aucune heuristique, les résultats qu'il renvoie sont donc garantis.

Dans les différents résultats où nous examinons la localisation des *reads* en fonction de leur longueur, nous recherchons leurs préfixes sur le génome. Par exemple, pour un *tag* de 21 *pb*, en fonction de la variation de sa longueur entre 14 et 21 *pb*, nous localisons ses préfixes de longueur $14, 15, \ldots$ jusqu'à 21.

3.4.1 Études statistiques sur le positionnement des *reads* sur un génome

Nous voulons déterminer comment procéder pour positionner des *reads* sur un génome, tout en minimisant les fausses localisations,c'est-à-dire minimiser le *bruit de fond* (section 3.1). En

d'autres termes, existe-t-il une longueur optimale (tout du moins minimale) à partir de laquelle le *bruit de fond* est quasiment nul ? Pour répondre à cette question, nous avons calculé les probabilités $[1 - A(m,n)]$ et $B(m,n)$ de l'équation 3.10 pour le modèle standard, en utilisant les formules de l'équation 3.11. Les composantes $[1 - A(m,n)]$ et $B(m,n)$ correspondent aux probabilités de localiser, respectivement au moins une fois et une unique fois, des *reads* aléatoires de longueur m sur une séquence de Bernoulli de la taille du génome humain ($n_{humain} = 3080436051$ nucléotides). Nous avons aussi calculé $[1 - A'(m,n)]$ et $B'(m,n)$ pour le modèle adapté, en utilisant les formules de l'équation 3.14, afin de simuler le comportement de *reads*, issus de la DGE (avec l'enzyme de restriction Nla3), sur le génome humain, c'est-à-dire avec $n_{humain}(CATG) = 27575352$ sites de restrictions.

FIGURE 3.3 : Probabilités du bruit de fond et influence de la longueur sur la capacité de prédiction de reads non ancrés.

A)

m \ $\mathbb{P}$	$1 - A(m,n)$	$B(m,n)$
	Localisé	Localisé unique
16	0.5154	0.2349
18	0.0651	0.0505
19	0.0191	0.0074
20	0.0023	0.0019
22	$1.51\ 10^{-4}$	$1.30\ 10^{-4}$
24	$9.61\ 10^{-6}$	$8.44\ 10^{-6}$
26	$6.15\ 10^{-7}$	$5.93\ 10^{-7}$
28	$3.90\ 10^{-8}$	$3.85\ 10^{-8}$
30	$9.84\ 10^{-9}$	$9.75\ 10^{-9}$

B)

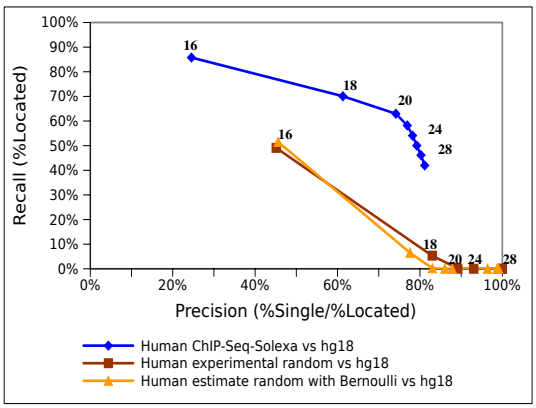


Table A : probabilités théoriques pour le modèle standard, de localiser des reads $(1 - A(m,n))$, et de les localiser de manière unique $B(m,n)$ sur une séquence de Bernoulli. La première composante varie de $\simeq 1$ pour 14 *pb* jusqu'à < 0.01 pour 20 *pb*. Alors que la deuxième trouve son maximum à 20 *pb*, avec une quasi totalité des séquences localisées de façon unique ($\frac{B(m,n)}{(1-A(m,n))}$). **Figure B :** L'influence du bruit de fond sur la capacité de prédiction est illustré avec une courbe *Precision-Recall-like*. Le *Recall* (% de séquence localisée/toutes les séquences) est tracé en fonction de la *Precision* (% de séquence localisée de façon unique/les séquences localisées) en faisant varier la longueur des séquences. La courbe bleue illustre le comportement des séquences réelles ChIP-Seq-Illumina® que l'on compare à des séquences aléatoires construites avec le modèle adapté (jaune) ou de manière empirique (marron). Les deux dernières courbes se confondent ce qui montre la validité du modèle théorique. La courbe bleue reste linéaire jusqu'à 20 et tombe un peu ensuite.

Nous pouvons visualiser le comportement de ces différentes estimations sur les tables 3.3.A et 3.4.A. Sur la table 3.4.A, la probabilité pour qu'un *tag*, issue de la DGE, soit localisé au moins une fois sur la séquence de Bernoulli (de la taille du génome humain) est de l'ordre de 46 % pour la longueur 16 (ligne où $m = 16$ dans la table 3.4.A). Rappelons que le génome humain étant assez bien réparti dans sa distribution en nucléotides, si 46% des *reads* sont attendus sur la séquence de Bernoulli, un même nombre sera alors localisé aléatoirement sur le génome humain. En d'autres termes, il est raisonnable d'assimiler cette probabilité à une P-valeur (comme dans BLAST) qui

FIGURE 3.4 : Probabilités du bruit de fond et influence de la longueur sur la capacité de prédiction de reads ancrés.

A)

m \ $\mathbb{P}$	$1-A'(m,n)$	$B'(m,n)$
	Localisé	Localisé unique
14	0.9997	0.0017
15	0.7717	0.1409
16	0.4654	0.1849
17	0.1303	0.0946
18	0.0351	0.0285
19	0.0091	0.0080
20	0.0023	0.0021
21	$5.93\ 10^{-5}$	$5.53\ 10^{-5}$
25	$2.44\ 10^{-6}$	$2.3\ 10^{-6}$

B)

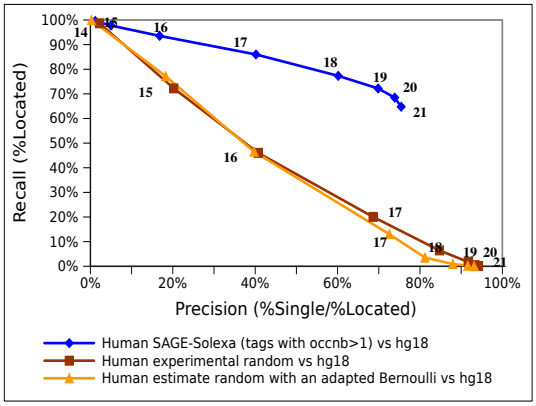


Table A : probabilités théoriques pour le modèle adapté, de localiser des reads $(1-A'(m,n))$, et de les localiser de manière unique $B'(m,n)$ sur une séquence de Bernoulli. La première composante varie de $\simeq 1$ pour 14 *pb* jusqu'à < 0.01 pour 19 *pb*. Alors que la deuxième trouve son maximum à 20 avec une quasi totalité des séquences localisées de façon unique $(\frac{B(m,n)}{(1-A(m,n))})$. **Figure B :** L'influence du bruit de fond sur la capacité de prédiction est illustré avec une courbe *Precision-Recall-like*. Le *Recall* (% de séquence localisée/toutes les séquences) est tracé en fonction de la *Precision* (% de séquence localisée de façon unique/les séquences localisées) en faisant varier la longueur des séquences. La courbe bleue illustre le comportement des séquences réelles SAGE-Illumina® que l'on compare à des séquences aléatoires construites avec le modèle adapté (jaune) ou de manière empirique (marron). Les deux dernières courbes se confondent ce qui montre la validité du modèle théorique. La courbe bleue reste à l'horizontale jusqu'à 19 – 20 et tombe un peu à 21.

mesure le *bruit de fond*, avec la règle suivante : plus la probabilité est proche de 0 et moins on a de chance de localiser une séquence au hasard. En conséquence, il est dans notre intérêt, lors de la procédure de *mapping*, de choisir des *reads* qui ont une longueur m pour avoir un taux de *bruit de fond* quasiment nul. Dans le cas du génome humain et d'après la FIGURE 3.3.A, la probabilité $[1-A(m)]$ est proche de 1 pour une séquence de longueur $m = 14$ alors qu'elle est à 10^{-7} pour une séquence de longueur 26. Cela signifie que positionner des séquences de longueur 14 sur le génome engendre un maximum de *bruit de fond*, alors qu'avec des séquences de longueur ≥ 20 le taux est $< 1\%$.

Maintenant, intéressons nous aux *reads* qui ont une localisation unique sur le génome. Sur la figure 3.4.A (entre autres), la probabilité $B'(m)$ augmente jusqu'à la longueur 16, puis diminue linéairement jusqu'à converger vers 0 pour $m = 25$.
Bien sûr, le meilleur des compromis serait d'avoir un minimum de *bruit de fond* ($[1-A'(m)] \to 0$) tout en ayant un maximum de localisations uniques ($B'(m) \to 1$). C'est pourquoi, nous avons décidé d'évaluer la meilleure longueur de m afin d'obtenir le meilleur des compromis. Pour ce faire, nous avons utilisé des courbes qui sur un axe mesurent le *Rappel* (*Recall*), c'est-à-dire le pourcen-

tage des séquences qui sont localisées parmi la collection de séquence initiale ; puis sur l'autre axe mesurent la *Précision* (*Precision*), c'est-à-dire le pourcentage de séquences qui sont localisées de manière unique parmi les séquences qui sont localisées (FIGURE 3.3.B et FIGURE 3.4.B). De plus, afin de vérifier que nos modèles théoriques sont appropriés, nous avons estimé de façon expérimentale les probabilités $A(m)$ et $B(m)$ (respectivement $A'(m)$ et $B'(m)$), en générant aléatoirement des *reads* et en mesurant le pourcentage de ceux qui sont localisés sur le génome humain. Les courbes représentant les séquences aléatoires sont comparées, avec d'un côté la courbe marron pour les séquences expérimentales, et de l'autre coté la courbe orange pour nos modèles de Bernoulli. Nous pouvons observer que les deux courbes sont fortement corrélées, que ce soit dans la figure 3.3.B ou dans la figure 3.4.B, ce qui nous assure de la pertinence de nos modèles théoriques. Pour ce qui est des données réelles, prenons par exemple la courbe bleue de la figure 3.3. Elle illustre, pour des *reads* non ancrés, le meilleur choix de la longueur. Effectivement, si le but est de maximiser, à la fois, *Rappel* et *Précision*, nous constatons que la courbe est linéaire entre 16 et $20-22$ et tombe à la verticale ensuite. Cela signifie qu'il n'y a plus vraiment d'intérêt à augmenter la longueur des séquences à partir de $m = 22$, puisqu'il y a la perte de *Rappel* qui n'est pas compensée par le gain de *Précision*.

Finalement, afin d'avoir le meilleur compromis entre le minimum de fausses localisations et le maximum de localisations uniques, il faut ajuster la longueur des séquences lors de la phase de positionnement (ou *mapping*). Ainsi, sur le génome humain, le choix de la longueur se situe aux alentours de $m = 22$ *pb* pour des *reads* non ancrés (*Recall*$\simeq 60\%$, *Precision*$\simeq 80\%$ et $1 - A(22) < 0.1\%$ dans la FIGURE 3.3) ; alors que la longueur se situe plutôt aux alentours de $m = 20$ *pb* pour des *reads* ancrés (*Recall*$\simeq 70\%$, *Precision*$\simeq 70\%$ et $1 - A'(20) < 0.3\%$ dans la FIGURE 3.4).

3.4.2 Évaluation des erreurs sur des ensembles de données réelles

Dans cette partie nous évaluons les erreurs de séquences sur trois des quatre collections de *reads* : SAGE-Illumina®, ChIP-Seq-Illumina®, SAGE-Sanger, la quatrième (CAGE-Sanger) contenant des erreurs spécifiques à la technique CAGE (analyse non montrée).

3.4.2.1 Estimation sans filtre de #*occ.*

Comme nous l'avons énoncé dans la section 3.2, l'estimation des erreurs dépend de la localisation de séquences réelles. En localisant chacune de nos trois collections nous allons déterminer leur propre composante $\mathscr{X}(m)$, pour une longueur de m que l'on fait varier. À partir de cette composante, nous voulons appliquer la formule de l'équation 3.16 afin de calculer, pour chacune des expériences SAGE-Illumina®, ChIP-Seq-Illumina®, SAGE-Sanger, la probabilité qu'une occurrence ou un *tag* contienne au moins une erreur dans sa séquence. Cependant, pour arriver à nos fins, nous appliquons la procédure décrite en sous-section 3.2.2, à commencer par la recherche du

seuil, c'est-à-dire la valeur critique sur la variable *#occ.* pour discerner les erreurs des causes biologiques. Celle-ci constituera donc notre hypothèse biologique (FIGURE 3.5). Pour trouver ce seuil, nous étudions la courbe correspondante à la proportion de *tags* qui sont localisés sur le génome en fonction de leur nombre d'occurrences *#occ.* (courbe bleue). Comme la probabilité d'avoir un *tag* erroné baisse avec *#occ.*, la courbe stagne au bout d'un certain seuil. Pour un échantillon SAGE-Illumina®, la courbe bleue nous indique que la proportion des *tags* localisés se stabilise à 80% de localisation pour un seuil de *#occ.* > 10. Notons qu'elle n'atteint pas les 100% de localisation car nous positionnons de manière exacte les *tags* sur le génome et certains contiennent des causes biologiques, comme des SNP ou des jonctions d'épissage. À partir de ce dernier, nous pouvons donc mesurer, de manière empirique $\mathscr{R}(m)$: la probabilité qu'une séquence soit à la fois biologiquement valide et localisée sur le génome. Ensuite, toujours de façon expérimentale, nous avons calculé $\mathscr{M}(m)$: la probabilité qu'un *tag* erroné soit localisé. La table 3.1 regroupe le calcul de ces différentes composantes pour nos trois échantillons. Puis, nous proposons dans la table 3.2 l'estimation de $\mathscr{S}(m)$ pour une occurrence erronée, ainsi qu'une estimation de $\mathscr{S}(m)$ dans la table 3.3 mais cette fois pour un *tag* erroné. Enfin, nous déduisons de la première table le pourcentage p correspondant à l'erreur de séquence à l'échelle nucléotidique (table 3.2).

TABLE 3.1 : Calcul des composantes pour l'estimation des erreurs de séquences.

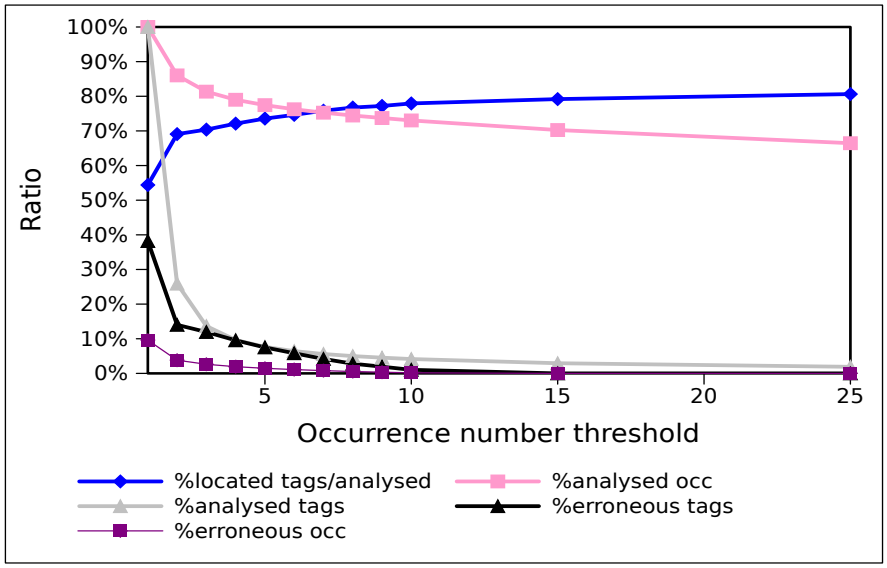
	SAGE-Sanger (1 192 500 tags)			SAGE-Illumina® (440 445 tags)			ChIP-Seq-Illumina® (929 165 tags)		
m	$\mathscr{X}(m)$	$\mathscr{R}(m)$	$\mathscr{M}(m)$	$\mathscr{X}(m)$	$\mathscr{R}(m)$	$\mathscr{M}(m)$	$\mathscr{X}(m)$	$\mathscr{R}(m)$	$\mathscr{M}(m)$
14	0.99	0.19	97.58	1.01	0.23	97.36	–	–	–
15	4.62	0.81	88.35	4.16	0.96	88.42	–	–	–
16	13.91	1.78	63.50	10.86	2.34	63.35	14.53	1.24	62.06
17	28.51	3.37	32.22	22.18	6.84	33.65	–	–	–
18	40.69	4.96	10.56	34.06	10.27	10.83	30.20	3.01	18.94
19	47.29	6.27	6.28	41.04	12.38	6.49	–	–	–
20	51.89	7.55	4.15	45.61	**13.89**	3.09	**37.29**	**4.63**	6.39
21	56.29	9.41	3.50	50.00	15.56	2.42	–	–	–
22	–	–	–	–	–	–	42.02	6.67	4.32
24	–	–	–	–	–	–	46.14	9.09	3.69
26	–	–	–	–	–	–	50.17	11.95	2.40
28	–	–	–	–	–	–	54.04	15.19	2.07
30	–	–	–	–	–	–	58.20	19.38	1.47

Les trois composantes $\mathscr{X}(m)$, $\mathscr{R}(m)$ et $\mathscr{M}(m)$ sont nécessaires pour l'estimation des erreurs de séquences et la proportion de faux positifs. Nous présentons ici les différents calculs pour SAGE-Illumina® et ChIP-Seq-Illumina® en faisant varier la longueur m des séquences.

3.4.2.2 Estimation avec filtre de *#occ.*

Sur la FIGURE 3.5, nous pouvons observer un impact lorsque nous ne considérons qu'un ensemble de *tags* en filtrant sur un nombre d'occurrences x (*#occ.* > x). À chaque augmentation de

FIGURE 3.5 : Choix du seuil sur *#occ.* pour le calcul des composantes $\mathscr{R}$ et $\mathscr{M}$.



La sélection d'un ensemble de *tags* en fonction de son nombre d'occurrences *#occ.* a une influence sur le pourcentage : i/ de *tags* analysés (gris), ii/ d'occurrences analysées (rose), iii/ de *tags* erronés (noir), iv/ d'occurrences erronées (magenta), v/ de *tags* localisés (bleu). Pour chaque point x en abscisse, on ne conserve que le sous ensemble de *tags* tel que $\#occ. > x$. Ainsi avec un filtre $\#occ. > 1$, seulement 25% des *tags* sont analysés mais il reste encore 85% des séquences de la collection de départ. Pour ce même filtre, le pourcentage des *tags* et des occurrences erronés est très faible. Enfin, le pourcentage de *tags* qui sont localisés sur le génome se stabilise à partir de $\#occ. > 10$; la courbe bleue est une méthode graphique pour fixer le seuil qui nous sert de critère dans la sélection de l'ensemble des *tags* qui sont biologiquement valides dans notre procédure d'estimation des erreurs.

x, le pourcentage d'occurrences et des *tags* erronés diminue. D'ailleurs entre les valeurs $x = 0$ et $x = 1$ le changement est drastique, avec un pourcentage d'erreur qui est divisé par trois : on passe de 41.98% à 15.48% pour des *tags* de longueur 21 (TABLE 3.4). Cela signifie qu'une bonne partie des *tags* qui ont un nombre d'occurrences $\#occ. = 1$ sont erronés. Ce constat est plutôt logique, le taux d'erreur étant globalement faible, il est très peu probable qu'une erreur se reproduise au même endroit et de la même façon dans deux mêmes séquences (sauf si le *tag* est vraiment très abondant). Autrement dit, un *tag* qui contient une erreur génère souvent un nouveau *tag* ($\#occ. = 1$).

Finalement, afin de diminuer très franchement les séquences artefactuelles, il est judicieux de ne considérer que l'ensemble des *tags* avec $\#occ. > 1$. Il est vrai qu'en optant pour cette stratégie un nombre d'informations biologiques pourraient être filtrées. Malgré tout, en pratiquant un tel filtre sur notre échantillon SAGE-Illumina®, seulement 15% des séquences sont filtrées (courbe rose de la FIGURE 3.5). Notons que le pourcentage de *tags* filtrés n'est pas représentatif du nombre d'infor-

TABLE 3.2 : Pourcentage d'occurrence et de nucléotide erronés.

	SAGE-Sanger (6 527 650 occ)		SAGE-Illumina® (2 222 344 occ)		ChIP-Seq-Illumina® (1 339 671 occ)	
m	$\mathscr{S}(m) \pm \alpha(m)$	p	$\mathscr{S}(m) \pm \alpha(m)$	p	$\mathscr{S}(m) \pm \alpha(m)$	p
14	6.02 ± 1.64	0.44	4.22 ± 2.77	0.31	–	–
15	6.25 ± 0.88	0.43	5.31 ± 1.26	0.36	–	–
16	6.10 ± 0.67	0.39	4.85 ± 0.96	0.31	6.89 ± 1.59	0.44
17	7.37 ± 0.46	0.45	5.24 ± 0.71	0.32	–	–
18	8.32 ± 0.38	0.48	6.65 ± 0.65	0.38	7.53 ± 0.99	**0.46**
19	$\mathbf{9.52 \pm 0.38}$	0.53	$\mathbf{8.11 \pm 0.61}$	0.44	–	–
20	10.79 ± 0.33	**0.57**	$\mathbf{9.14 \pm 0.61}$	**0.48**	$\mathbf{8.84 \pm 0.09}$	**0.48**
21	12.49 ± 0.32	0.63	10.57 ± 0.60	0.53	–	–
22	–	–	–	–	10.39 ± 0.09	0.50
24	–	–	–	–	11.99 ± 0.09	0.53
26	–	–	–	–	13.51 ± 0.09	0.56
28	–	–	–	–	15.22 ± 0.09	0.59
30	–	–	–	–	16.83 ± 0.09	**0.61**

Le pourcentage d'occurrences erronées $\mathscr{S}(m)$) est calculé pour les expériences SAGE et ChIP-Seq à différentes longueur de m comprise entre 14 et 21 pour SAGE-{Sanger,Illumina®} et comprise entre 16 et 30 pour ChIP-Seq-Illumina®. $\pm\ \alpha(m)$ correspond à l'erreur standard de $\mathscr{S}(m)$. Le pourcentage d'occurrences erronées augmente avec la longueur. En outre, plus la séquence augmente et plus la probabilité d'avoir un nucléotide erroné augmente (tout du moins jusqu'au $30^{\text{ième}}$ *pb*). Cette constatation suggère que quelle que soit la technologie de séquençage (Sanger ou Illumina®), l'erreur de séquence est plus fréquente en $3'$.

mations pertinentes qui sont perdues. En effet, rappelons qu'un *tag* erroné a de grandes chances de posséder un $\#occ. = 1$, alors en adoptant un filtre $\#occ. > 1$, nous supprimons un *tag* erroné, d'où les 75% de *tags* filtrés dans notre expérience (courbe grise dans la FIGURE 3.5). D'ailleurs, ce filtre impacte aussi le taux de faux positifs. En fait, pour notre collection de *reads*SAGE-Illumina®, en optant un tel filtre sur des *reads* d'une longueur $m = 20$ *pb*, le taux de fausses localisations chute de 2.17 à 0.58% ($\mathscr{V}(20)$ dans la table 3.4).

3.4.2.3 Estimation en fonction de la longueur m

À cause du *bruit de fond*, l'estimation de l'erreur est moins précise pour des séquences de longueur 16 que pour des séquences de longueur 20. Ce constat se voit directement sur l'erreur standard (équation 3.17) de $\mathscr{S}(m)$ dans la table 3.2, pour l'expérience SAGE-Illumina® elle est à ± 2.77 pour des occurrences de longueur 14, alors qu'elle est à ± 0.61 pour des séquences de longueur 20. Malgré cela, pour les trois échantillons (SAGE-Illumina®, ChIP-Seq-Illumina®, SAGE-Sanger), le pourcentage de séquences erronées augmente de façon régulière à partir de la longueur $m = 17$ où le *bruit de fond* est trop faible pour impacter l'estimation des erreurs. Néanmoins, cette constatation est tout à fait logique (et ne démontre rien pour le moment) : si une séquence contient une

TABLE 3.3 : Pourcentage de *tags* erronés et de faux positifs.

	SAGE-Sanger (1 192 500 tags)		SAGE-Illumina® (440 445 tags)		ChIP-Seq-Illumina® (929 165 tags)	
m	$\mathscr{S}(m) \pm \alpha(m)$	$\mathscr{V}(m)$	$\mathscr{S}(m) \pm \alpha(m)$	$\mathscr{V}(m)$	$\mathscr{S}(m) \pm \alpha(m)$	$\mathscr{V}(m)$
14	35.79 ± 2.52	35.27	32.46 ± 2.80	31.92	–	–
15	35.17 ± 1.54	32.58	30.18 ± 1.74	27.84	–	–
16	34.96 ± 1.28	25.79	24.84 ± 1.31	17.65	36.23 ± 1.58	26.31
17	39.03 ± 0.71	17.59	25.77 ± 0.77	11.14	–	–
18	42.29 ± 0.79	7.53	30.15 ± 0.77	4.95	34.84 ± 0.99	9.45
19	$\mathbf{46.91 \pm 0.73}$	5.59	$\mathbf{35.33 \pm 0.74}$	3.89	–	–
20	50.21 ± 0.78	4.33	38.20 ± 0.79	2.17	36.71 ± 0.09	3.74
21	53.83 ± 0.90	4.31	41.98 ± 0.99	2.04	–	–
22	–	–	–	–	39.71 ± 0.09	2.96
24	–	–	–	–	42.48 ± 0.09	2.91
26	–	–	–	–	44.63 ± 0.09	2.15
28	–	–	–	–	46.96 ± 0.09	2.11
30	–	–	–	–	49.06 ± 0.09	1.73

Le pourcentage de *tags* erronés $\mathscr{S}(m)$ et la proportion de faux positifs $\mathscr{V}(m)$ sont calculés pour les expériences SAGE et ChIP-Seq à différente longueur de m comprise entre 14 et 21 pour SAGE-[Sanger,Illumina®] et comprise entre 16 et 30 pour ChIP-Seq-Illumina®. $\pm\,\alpha(t)$ correspond à l'erreur standard de $\mathscr{S}(m)$. Le pourcentage de *tags* erronés augmente logiquement alors que le pourcentage de faux positifs diminue avec la longueur.

TABLE 3.4 : Comparaison des *tags* erronés et des faux positifs pour la librairie SAGE-Illumina® avec et sans filtration.

SAGE-Illumina® private library			$\#occ. > 0$ (440 445 tags)			$\#occ. > 1$ (114 721 tags)		
m	$\mathscr{R}(m)$	$\mathscr{M}(m)$	$\mathscr{X}(m)$	$\mathscr{S}(m) \pm \alpha(m)$	$\mathscr{V}(m)$	$\mathscr{X}(m)$	$\mathscr{S}(m) \pm \alpha(m)$	$\mathscr{V}(m)$
14	0.23	97.36	1.01	32.46 ± 2.80	31.92	0.59	14.77 ± 2.82	14.47
15	0.96	88.42	4.16	30.18 ± 1.74	27.84	2.25	12.22 ± 1.75	11.06
16	2.34	63.35	10.86	24.84 ± 1.31	17.65	5.86	10.27 ± 1.32	6.91
17	6.84	33.65	22.18	25.77 ± 0.77	11.14	13.50	11.19 ± 0.78	4.35
18	10.27	10.83	34.06	30.15 ± 0.77	4.95	20.09	12.45 ± 0.77	1.69
19	12.38	6.49	41.04	35.33 ± 0.74	3.89	23.61	13.85 ± 0.74	1.18
20	13.89	3.09	45.61	38.20 ± 0.79	**2.17**	25.52	14.01 ± 0.79	**0.58**
21	15.56	2.42	50.00	41.98 ± 0.99	2.04	28.23	15.48 ± 0.99	0.57

En fonction de la longueur m, nous mesurons les différentes composantes $\mathscr{X}(m), \mathscr{R}(m), \mathscr{M}(m)$ ainsi que le pourcentage de *tags* erronés $\mathscr{S}(t)$) avec son erreur standard $\alpha(t)$. Nous calculons aussi la proportion de faux positifs $\mathscr{V}(t)$ (le pourcentage de *tags* qui se localisent parmi les erronés). Enfin, les différentes statistiques sont aussi calculées sur un sous ensemble de *tags* pour voir l'influence d'un filtre sur le nombre d'occurrences des *tags*. En d'autres mots, le premier ensemble ne contient aucun filtre ($\#occ. > 0$) alors que le deuxième est filtré avec $\#occ. > 1$.

erreur à une position x, cette séquence restera erronée lorsqu'on se focalisera sur les erreurs en position $x+1$; par conséquent la proportion d'occurrences erronées ne peut qu'augmenter.

Il faut donc passer à l'échelle nucléotidique pour démontrer un éventuel biais sur les erreurs de séquences. D'après l'équation 3.22, le pourcentage p de nucléotides erronés se calcule directement à partir du pourcentage $\mathscr{S}$ d'occurrences erronées (et non pas des *tags* erronés car nous voulons estimer l'erreur produite par le séquenceur sur toutes les séquences générées). Le résultat pour les trois expériences est consultable dans la table 3.2, le niveau d'erreur mesuré pour la technologie Sanger est d'ailleurs en accord avec la littérature [Piquemal *et al.*, 2002; Colinge et Feger, 2001]. Nous constatons tout de même que plus la position est à la fin de la séquence, plus l'erreur est grande. Par exemple pour ChIP-Seq-Illumina®, le ratio d'erreur est à 0.46% à la position 18, puis augmente jusqu'à 0.61% à la position 30 : soit une augmentation de 32%. Le comportement est le même pour les autres échantillons. Il y a donc une augmentation du taux d'erreurs en fonction de la position du nucléotide dans la séquence ce qui nous révèle la tendance qu'ont les SHD Illumina® à produire plus d'erreurs en fin de séquence.

3.4.2.4 Relation entre les erreurs, la technologie Illumina® et la DGE

Maintenant, intéressons nous particulièrement à la technologie Illumina®. D'après la table 3.2 pour SAGE-Illumina®, le pourcentage de séquences erronées baisse lorsque les séquences sont inférieures à 16 *pb*, puis augmente de façon linéaire jusqu'à une longueur de 21 *pb*. Pour ChIP-Seq-Illumina®, le pourcentage augmente aussi linéairement jusqu'à 30 *pb*. Une première remarque à souligner est la très bonne adaptation de la technique digitale SAGE à la technologie Illumina® parce qu'il n'y a pas d'augmentation significative de l'erreur entre notre échantillon génomique et notre échantillon transcriptomique : 8.84% pour ChIP-Seq-Illumina® contre 9.14% pour SAGE-Illumina® pour des séquences de longueur $m = 20$.

La technologie de séquençage Illumina® semble produire moins d'erreurs que son aînée ce qui constitue une seconde remarque tout aussi intéressante : nous pouvons compter 8.11% d'occurrences erronées pour des séquences de 19 *pb* chez SAGE-Illumina® alors que nous atteignions 9.52 % d'erreurs chez SAGE-Sanger pour des séquences de même longueur (table 3.2). Le séquençage haut débit semble donc être très bien adapté aux méthodes digitales pour la transcriptomique, telle que la DGE.

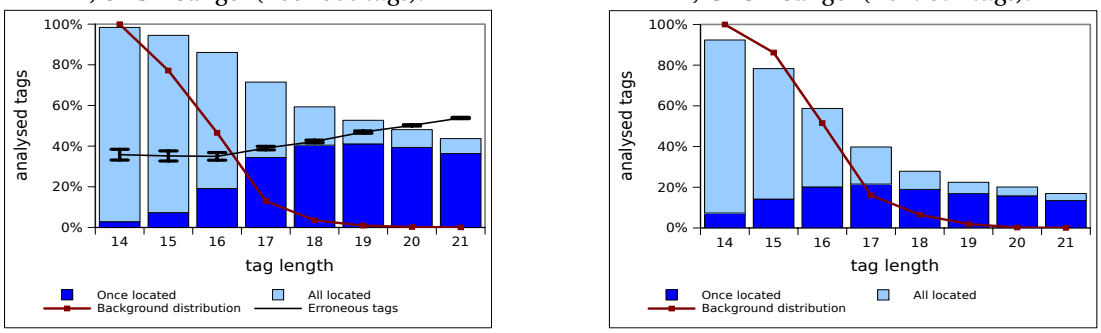
3.4.3 Optimisation de la phase de *mapping*

Après avoir estimé le *bruit de fond* et les erreurs pour une collection de séquences, nous portons notre attention sur la question principale : comment maximiser le nombre de prédictions génomiques d'origines sans se tromper ? Dans la sous-section 3.4.1, nous démontrons que, selon le génome utilisé, il existe une longueur minimale pour positionner les séquences sur ce génome afin d'avoir un *bruit de fond* tolérable. En contrepartie dans la sous-section 3.4.2, nous révélons que le SHD Illumina® a tendance à produire davantage d'erreurs en fin de séquence. Finalement, existe-

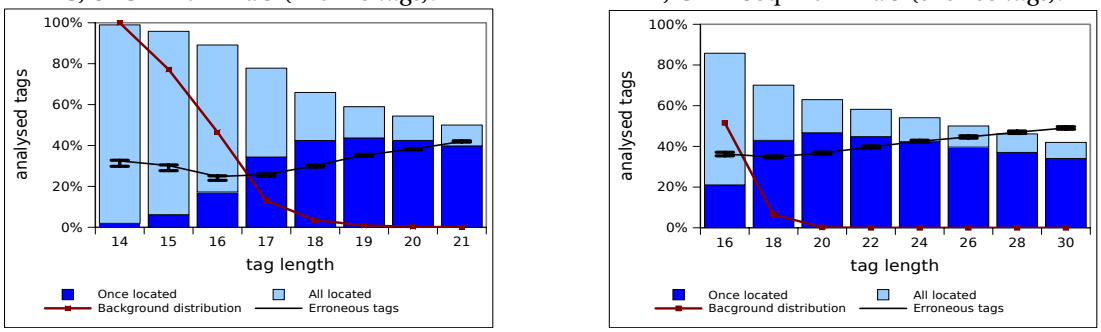
t-il une bonne longueur pour le *mapping* ? Nous traitons cette question à travers nos quatre collections de *reads*.

FIGURE 3.6 : Évaluation du « mapping » en fonction de la longueur des séquences.

A) SAGE-Sanger (1 992 500 *tags*). B) CAGE-Sanger (1 627 871 *tags*).



C) SAGE-Illumina® (440 445 *tags*). D) ChIP-Seq-Illumina® (929 165 *tags*).



Étude de la variation de la localisation pour SAGE-Sanger, CAGE-Sanger, SAGE-Illumina® et ChIP-Seq-Illumina®. Chaque histogramme donne, pour chaque longueur, le pourcentage des *tags* qui sont localisés parmi ceux qui sont analysés (barre bleue ciel), le pourcentage des *tags* qui sont localisés de manière unique (barre bleue foncée), le pourcentage de *tags* erronés (courbe noire) et le bruit de fond (courbe marron). Les histogrammes sont le résultat de la localisation des *tags* sur le génome humain (hg18). Par souci de concision, toutes les longueurs ne sont pas analysées pour la collection ChIP-Seq dans l'histogramme (D). Nous observons un comportement similaire pour toutes les expériences sauf pour les données CAGE. Le pourcentage de localisation unique atteint son maximum à 19-20 *pb* quand le bruit de fond devient négligeable, il diminue ensuite à cause des erreurs de séquences.

Parmi elles, nous avons deux types de données transcriptomiques (CAGE et SAGE) et nous avons un type de données génomiques (ChIP-Seq), puis nous avons deux technologies de séquenceurs (Sanger et Illumina®). Pour chacune de nos expériences, nous proposons en FIGURE 3.6 un histogramme qui rassemble toutes les informations du *mapping* lorsque nous faisons varier la longueur des séquences. Ainsi, nous suggérons d'évaluer l'influence des différents facteurs (*bruit de fond*, erreurs de séquences et longueur des séquences) sur chacun des échantillons lors du *mapping*. Nous les comparons deux à deux afin d'analyser rigoureusement cette influence sur une technologie ou un type précis de donnée.

3.4.3.1 Caractéristiques communes des quatre expériences

Commençons par décrire les points communs des différents histogrammes :

i/ le pourcentage de *tags* localisés (barre bleu ciel) baisse lorsque la longueur augmente,

ii/ le pourcentage de localisations uniques (barre bleu foncé) augmente jusqu'à atteindre un maximum aux alentours de 19-20 *pb* (17 pour CAGE) puis baisse ensuite,

iii/ le *bruit de fond*, que ce soit pour le modèle standard ou adapté, est à $\simeq 100\%$ pour les *tags* de longueur 14, aux alentours de 15% pour une longueur de 17, puis diminue jusqu'à devenir acceptable $< 1\%$ pour les *tags* de 20 *pb*,

iv/ le pourcentage de *tags* erronés (non calculé pour les données CAGE) est difficilement mesurable lorsqu'il y a du *bruit de fond*, c'est-à-dire à l'endroit où l'intervalle de confiance est large (entre 14 et 17). Toutefois lorsque c'est mesurable, le pourcentage d'erreurs augmente graduellement (à partir 17). Notons que si le pourcentage de *tags* erronés est démesuré, il n'est pas représentatif de l'erreur produite par le séquenceur[3]. D'ailleurs la plupart des *tags* erronés ont une pondération $\#occ. = 1$ (table 3.4) et peuvent être facilement exclus dans une analyse.

3.4.3.2 Comparaison des expériences

Globalement, tous les histogrammes (sauf pour CAGE) ont un comportement semblable ce qui nous laisse à penser que les différents facteurs (*bruit de fond*, erreurs, etc...) ont un impact identique sur le *mapping*, quel que soit le type de données ou/et la technologie utilisés.

SAGE-Sanger versus CAGE-Sanger. Il est évident, dans la FIGURE 3.6, que les données CAGE-Sanger diffèrent complètement des autres données (y compris celles de SAGE-Sanger) : le pourcentage des *tags* qui sont localisés dans CAGE-Sanger pour les séquences de 19 *pb* est deux fois plus bas que dans les trois autres échantillons. Ce résultat est dû à la présence d'un biais lié à la technologie CAGE, en fait 88% de tous les *tags* de CAGE-Sanger commencent par une guanine (nucléotide *G*). Nous avons adapté notre protocole en faisant varier la longueur sur les suffixes des *tags* plutôt que les préfixes. Ainsi, nous localisons 46% des suffixes de longueur 18 au lieu de 28% dans la FIGURE 3.6.B avec notre protocole standard. Cela constitue bien la confirmation que des erreurs sont générées en $5'$ des *tags* de la technique CAGE [Harbers et Carninci, 2005] et nous devons donc considérer au préalable un traitement spécifique pour corriger ce biais avant d'éventuelles analyses biologiques.

SAGE-Illumina® versus SAGE-Sanger. Les résultats obtenus avec SAGE-Illumina® sont le produit d'une seule librairie ($\simeq 440K$ *tags*, $2.2M$ occurrences). Afin d'avoir une quantité comparable de données pour SAGE-Sanger nous avons utilisé toutes les librairies Sanger disponibles ($1.9M$ *tags*,

3. voir les explications du paragraphe « Estimation avec filtre de *#occ.* » de la sous-section 3.4.2

6.5M occurrences). Dans les deux cas, la courbe qui représente le *bruit de fond* est la même puisque les données sont ancrées (modèle adapté). Comme l'ancien protocole utilisé dans la technologie Sanger a été quelque peu simplifié avec Illumina® il nous a semblé intéressant de comparer l'influence des effets cumulés (erreurs, profondeur de séquençage, etc...) entre les deux technologies. Cependant, les résultats entre la FIGURE 3.6.A et la FIGURE 3.6.C sont globalement similaires, avec un léger avantage pour SAGE-Illumina® en terme de localisations : le pourcentage de *tags* localisés (barre bleu ciel) varie de 66% à 50% entre les longueurs 18 et 21 *pb* pour Illumina® alors qu'il varie entre 59% et 43% pour Sanger pour les mêmes longueurs. D'ailleurs, cette différence de localisation s'explique par la différence de *tags* erronés : la différence de 59% contre 48% de localisation à la longueur 19 est comblée par la différence de 35% contre 47% de *tags* erronés (table 3.3). Une explication plausible qui justifierait cet écart serait l'accumulation des librairies SAGE-Sanger pour atteindre la même profondeur de séquençage ce qui générerait plus d'erreurs.

En outre pour les deux expériences, le maximum de localisations uniques (barre bleu foncé) est atteint pour la longueur 20 : 44% pour Illumina® contre 41% Sanger, là encore les longueurs 19-20 semblent correspondre à des bonnes longueurs pour le *mapping*.

SAGE-Illumina® versus ChIP-Seq-Illumina®. Nous comparons une expérience génomique avec une expérience transcriptomique pour la même technologie Illumina®. La première regroupe un ensemble de *tags* cibles de mRNA alors que l'autre expérience rassemble des *tags* cibles des régions spécifiques du génome. Comme dans la comparaison précédente, les taux d'erreurs et de *bruit de fond* sont semblables (FIGURE 3.6.C vs FIGURE 3.6.D) avec un meilleur taux de localisations pour ChIP-Seq-Illumina®, bien que les échelles sur les positions ne soient pas les mêmes. De plus pour ChIP-Seq, le taux de localisations uniques atteint son maximum de 47% à 20 *pb* (contre 44% pour LongSAGE) puis décroît linéairement ensuite jusqu'à < 40%. Même si nous choisissons les séquences les plus longues (30 pour ChIP-Seq-Illumina®), toutes ne trouvent pas une localisation unique sur le génome ce qui constitue encore un argument sur le choix de la longueur des séquences pour le *mapping*. Rappelons que l'erreur de séquence augmente avec la position pour la technologie Illumina® (sous-section 3.4.2) ce qui peut expliquer que la capacité à prédire des localisations uniques sur le génome soit plus efficace avec des séquences de longueur 20 qu'avec des séquences de longueur 30.

3.4.3.3 Causes des *tags* non localisés

La comparaison de ces deux expériences nous permet aussi de mesurer le taux des *tags* liés à des phénomènes biologiques qui ne sont pas localisés : $\mathscr{R}(m)$ (sous-section 3.2.2). Pour une expérience de ChIP-Seq, la principale cause de non localisation est liée au polymorphisme inter-individus (ici les *reads* proviennent d'un individu et le génome d'un autre). Nous pouvons raisonnablement supposer que ces variations sont majoritairement dues à des SNP (Single Nucléotide Polymorphism) car les séquences sont courtes et la collection est composée de *reads* qui pro-

viennent de régions conservées du génome. Alors que dans l'échantillon SAGE-Illumina®, des SNP mais aussi des modifications post-transcriptomiques telles que de l'édition d'ARN ou des *tags* à cheval sur la queue polyA peuvent aussi affecter la localisation. Dès lors, à partir d'une certaine longueur comme $m = 20$ (cette longueur semble judicieuse d'après les précédents arguments), si nous considérons que $\mathscr{R}_{chip}(m)$ estime le pourcentage de *tags* non localisés à cause d'un SNP, alors $\mathscr{R}_{sage}(m) - \mathscr{R}_{chip}(m)$ ne mesure que ceux qui ne sont pas localisés à cause d'un phénomène transcriptomique. Ainsi, nous observons dans la table 3.1 que $\mathscr{R}_{chip}(20) = 4.63\%$ des *tags* (*a priori* aussi bien pour ChIP-Seq que pour LongSAGE) sont affectés par un SNP, ce qui veut dire que $\mathscr{R}_{sage}(20) - \mathscr{R}_{chip}(20) = 9.3\%$ des *tags* de LongSAGE sont spécifiques à la transcription. Ce dernier pourcentage est d'ailleurs en accord avec ce qui a été publié pour SAGE-Sanger dans [Keime *et al.*, 2007]. Enfin, le pourcentage d'occurrences erronées est quasiment identique pour les deux expériences à la longueur 20 (8.84 ± 0.09 pour ChIP-Seq contre 9.14 ± 0.61 pour LongSAGE dans la table 3.2). Ce constat signifie que la phase de rétro-transcription de LongSAGE produit très peu d'erreurs.

3.4.4 Annotation et validation de nouvelles régions transcrites

3.4.4.1 Présentation globale des données DGE-StemCell

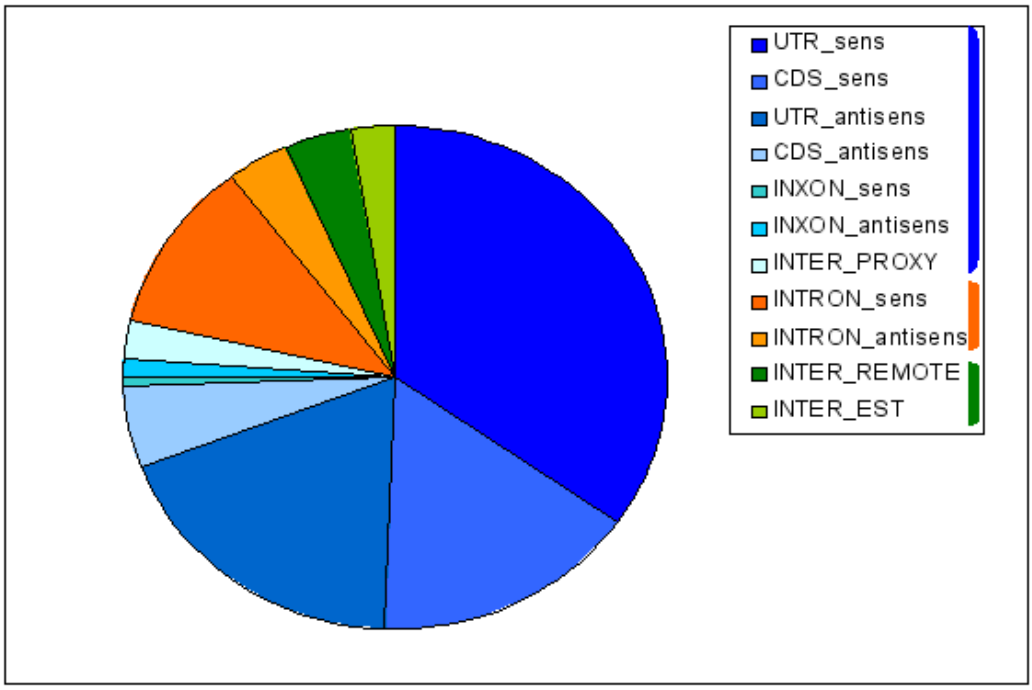
Le pipeline illustré dans la figure 3.2 est utilisé pour traiter les 4 540 466 de *reads* de la banque DGE-StemCell détaillée dans la section 3.4. Cette banque DGE-StemCell contient 527 650 *tags* différents dont 175 812 avec un nombre d'occurrences $\#occ. > 1$. Afin de limiter le nombre de faux positifs (sous-section 3.2.3.2), nous localisons le sous-ensemble des 175 812 *tags* de longueur 21 *pb* sur le génome humain (phases 1 et 2 du pipeline), ce qui nous donne un total de 89 749 localisations uniques ($\simeq 51\%$ des *tags*).

À l'aide de la procédure d'annotation du pipeline (phase 3), nous avons effectué la distribution d'annotation des 89 749 *tags*, soit un total de 2 380 929 occurrences (FIGURE 3.7). La répartition indique que 79 % des *tags* sont annotés dans un exon (UTR ou CDS, sens ou antisens), ce qui correspond à 92 % de l'ensemble des occurrences ; alors que 15 % des *tags* sont annotés dans les introns, ce qui représente 5,4 % des occurrences ; et enfin 6 % des *tags* sont annotés dans les régions intergéniques, c'est-à-dire 2,6 % des occurrences. Ces pourcentages concordent avec ceux de la littérature obtenus en utilisant des données de SHD [van Bakel *et al.*, 2010].

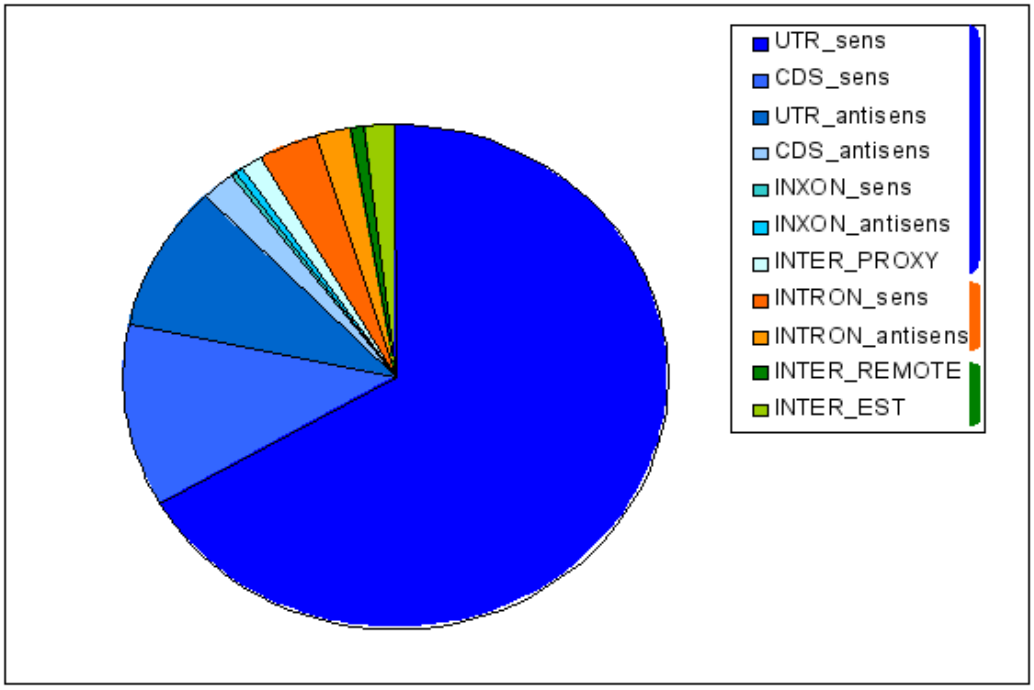
Une remarque immédiate sur cette répartition est la différence d'abondance des *tags* exoniques par rapport aux *tags* introniques ou intergéniques. En effet, il y a 70 860 *tags* exoniques pour un total de 2 192 810, soit une moyenne d'expression de 31 occurrences par *tag*. En revanche, il y a 5 565 *tags* intergéniques pour un total de 60 848, soit une moyenne d'expression de 11 occurrences par *tag*. Au final, les *tags* exoniques sont en moyenne trois fois plus exprimés que les *tags* intergéniques. En fait, la plupart de ces régions intergéniques transcrites ont échappé aux procédures d'annotations

prédictives utilisant des recherches de régions codantes. Une grande proportion de ces transcrits pourrait correspondre en réalité à des ARN non codants [Mercer *et al.*, 2009].

FIGURE 3.7 : Répartition de l'annotation du sous ensemble optimisé des *tags* et occurrences de DGE-StemCell.



(a) Répartition des *tags* de DGE-StemCell



(b) Répartition des *occurrences* de DGE-StemCell

Les *tags* sont annotés suivant la procédure d'annotation décrite en sous-section 3.3.1.3. Dans la figure (a) : 79 % des *tags* sont annotés dans des exons (couleur bleue), 15 % des *tags* sont annotés dans les introns (couleur orange) et 6 % dans les régions intergéniques (couleur verte). Dans la figure (b) : 92 % des occurrences sont annotés dans des exons (couleur bleue), 5,4 % des occurrences sont annotés dans les introns (couleur orange) et 2,6 % dans les régions intergéniques (couleur verte).

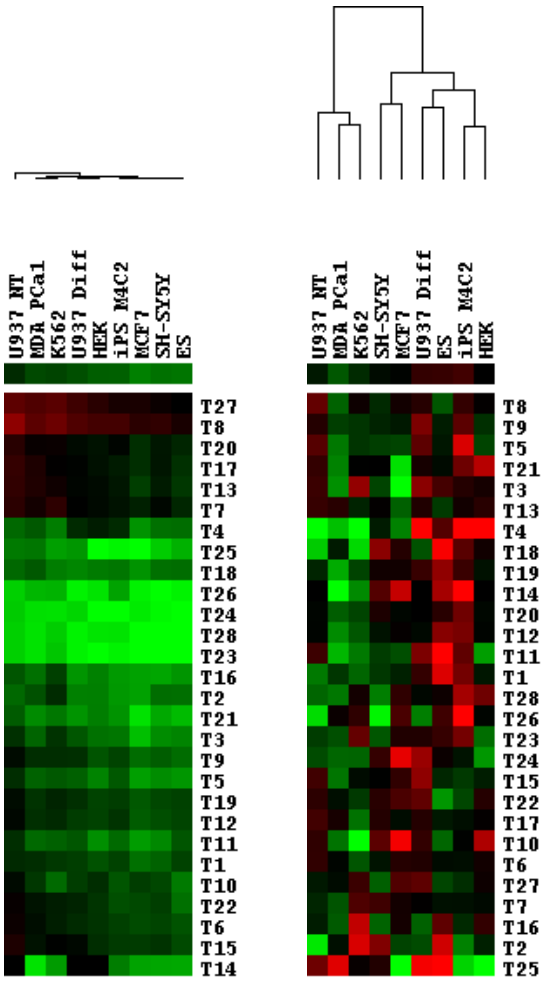
3.4.4.2 Validations biologiques sur les *tags* intergéniques

Parmi les 5565 *tags* intergéniques ci-dessus, nous nous sommes concentrés sur un échantillon de 34 *tags* plus abondants dans les cellules souches humaines (DGE-StemCell) que dans la plupart des autres tissus (base *Sage Genie*) (phases 7 et 8 du pipeline en figure 3.2). Pour chacun d'eux, nous avons conçu les prédictions (phase 5 du pipeline) à l'aide des données K562 de RNA-Seq (voir section 3.4). Ensuite, nous avons effectué des validations biologiques par PCR en temps réel pour confirmer l'expression de ces transcrits potentiels dans une série de cellules normales et tumorales (le protocole est détaillé dans la section 3.3.2).

Comme nous pouvons l'observer dans la figure 3.8, sur les 34 prédictions étudiées, l'expression de 28 transcrits a été validée. La figure 3.8(a) indique que la plupart des transcrits sont faiblement exprimés puisque la comparaison est réalisée avec la p21 (CDKN1A) ; cyclin-dependent kinase inhibitor 1A, qui est un gène généralement peu exprimé dans les cellules, ce qui confirme les observations globales faites en section 3.4.4.1. Cette comparaison qui n'a qu'une valeur indicative permet de remettre ces transcrits dans un contexte cellulaire où un transcrit, même peu abondant comme la p21, peut avoir un rôle cellulaire majeur.

Un des intérêts du pipeline est sa faculté d'extraire rapidement et rigoureusement un ensemble de transcrits spécifiques d'un type cellulaire. A titre d'exemple, nous montrons dans la figure 3.8(b) pour les cellules ES et les cellules IPS.

FIGURE 3.8 : Validations biologiques de 28 *tags* intergéniques.



(a) (b)

transcript id	chr	relative pos
T1	11	47487563
T2	11	65193929
T3	11	88161199
T4	12	17144820
T5	12	27191840
T6	12	46314282
T7	12	64217016
T8	14	43769766
T9	14	76680426
T10	14	89885758
T11	15	41590457
T12	17	3710068
T13	19	15347785
T14	19	23286133
T15	1	168221995
T16	1	224034306
T17	2	230628796
T18	3	139163262
T19	4	103715769
T20	4	103716893
T21	5	87732466
T22	6	52448680
T23	5	178192120
T24	6	26674232
T25	11	94390224
T26	12	51664843
T27	17	51183588
T28	19	35340115

Les figures représentent des clusters d'expression de transcrits validés par PCR temps réel. Nous pouvons observer les différentes *prédictions* (lignes des clusters) et la liste des cellules cibles (colonnes du cluster). Le tableau à droite des clusters indique les différentes *prédictions* ainsi que leur position chromosomique. La figure (a) illustre l'abondance relative dans un tissu donné des 28 transcrits validés par PCR. Leur abondance est mesurée par rapport à un transcrit codant, généralement peu abondant dans les cellules (ici la p21 ; CDKN1A). La figure (b) illustre la spécificité d'expression en comparant les différents tissus. Cette représentation permet de distinguer les transcrits qui ont une expression dans un tissu-spécifique de ceux qui sont ubiquitaires.

3.5 Conclusion et discussions

Les SHD sont maintenant utilisés pour interroger le transcriptome ou l'interactome à l'échelle du génome, avec une spécificité et une sensibilité sans précédent [Velculescu et Kinzier, 2007; Mardis, 2007]. Les séquences transcriptomiques font l'objet d'une profonde investigation tellement la caractérisation et l'annotation des transcriptomes sont complexes. Chez l'humain et la souris, certaines régions du génome restent encore sans annotation et certaines régions sont non-codantes [Bertone *et al.*, 2004; Kapranov *et al.*, 2007; Claverie, 2005]. Pour cerner un transcriptome dans toute sa profondeur, les *reads* séquencés peuvent être positionnés sur le génome de référence afin d'en extraire leur annotation [Saha *et al.*, 2002]. Cependant, une telle procédure est complexe car les SHD produisent des millions de *reads* mais aussi beaucoup d'erreurs. Il semble primordial d'évaluer l'impact de ces erreurs lors du *mapping*, puis de proposer une méthodologie pour annoter les *reads* sur un génome avec le plus de précision possible. Nous nous sommes penchés sur ce problème en faisant une évaluation sur des données de DGE et de ChIP-Seq séquencés avec Illumina®.

3.5.1 Les erreurs de séquences

Pour la technique Sanger, les différentes estimations de l'erreur de séquences (par mesures expérimentales ou sur l'analyse du PHRED score) sont inférieures à 1% [Piquemal *et al.*, 2002; Colinge et Feger, 2001]. Récemment, une estimation a été reportée dans le cadre du re-séquençage du génome avec la technologie haut débit [Dohm *et al.*, 2008a]. De notre côté, nous présentons une méthode précise et originale pour estimer l'erreur de séquences pour des données transcriptomiques de type DGE et des données génomiques de type ChIP-Seq. Nous estimons 0.57% de nucléotides erronés pour des *reads* séquencés avec la technologie Sanger (table 3.2), un taux en accord avec la littérature [Piquemal *et al.*, 2002; Colinge et Feger, 2001], ce qui valide notre approche. Un premier point intéressant dans nos estimations : la technologie Illumina® génère un taux similaire (légèrement plus faible) que la technologie Sanger (0.48 contre 0.57), ce qui montre que cette technologie est appropriée pour des techniques comme la DGE ou le ChIP-Seq. Néanmoins, la différence d'erreurs observées entre Sanger et Illumina® peut s'expliquer par la diversité des plate-formes qui sont utilisées pour extraire les librairies LongSAGE et/ou l'erreur générée pour les étapes de PCR dans le protocole [Khattra *et al.*, 2007].

Depuis la naissance du haut débit et malgré leurs performances, les SHD Illumina® produisent davantage d'erreurs sur la fin des séquences [Kharchenko *et al.*, 2008; Dohm *et al.*, 2008a]. Nos résultats confirment cette tendance, quel que soit le type de données séquencées. Pour les échantillons DGE nous observons un biais conséquent de 0.31% à 0.48% entre les positions 16 et 20, soit une augmentation de plus de 50%. L'augmentation continue de la même façon avec les données ChIP-Seq de 0.48% à 0.61% entre les positions 20 et 30.

Comparée aux autres méthodes, notre approche est capable de mesurer l'influence des erreurs de séquences sur la localisation génomique. Pour des *tags* SAGE-Illumina® d'une longueur de 19

pb, plus de 40% ne sont pas localisés (FIGURE 3.6) et parmi ceux-là, nous estimons que ~ 36% contiennent des erreurs de séquences (table 3.3). D'une manière générale, les *tags* erronés représentent la majorité des non localisés. Cependant, lorsque nous filtrons les *tags* avec $\#occ. = 1$, la proportion des erronés diminue très largement : < 15% pour des séquences de 19 *pb* (table 3.4). D'ailleurs, utiliser un tel filtre peut être un bon compromis pour limiter les erreurs de séquences dans une expérience biologique [Saha *et al.*, 2002]. Toutefois, il reste quand même des *tags* erronés avec une pondération $\#occ. > 1$, particulièrement pour les données transcriptomiques à cause d'une large distribution du nombre d'occurrences. En fait, plus le transcrit va être abondant, plus son *tag* sera séquencé et en conséquence la probabilité de générer une erreur sur l'une de ses occurrences sera grande. Un travail permettant de retrouver et de corriger toutes les occurrences erronées provenant d'un même *tag* est une solution alternative pour filtrer les erreurs (exclusivement possible sur des *reads* courts et ancrés) [Akmaev et Wang, 2004]. Dans notre librairie SAGE-Illumina®, un *tag* avec un nombre d'occurrences $\#occ. = 8\,165$ génère 47 *tags* voisins (seulement 1 *pb* de différence). Parmi ses voisins, 44 ont $\#occ. < 20$ et ne sont pas localisés sur le génome : *tags* erronés ou coïncidence ?

Un argument important dans le fait de filtrer les *reads* avec un faible seuil $\#occ. > 1$ est le taux de faux positifs. Il est divisé par quatre : $\simeq 0.58\%$ contre 2.17% à l'origine ($\mathscr{V}(20)$ dans la table 3.4). Du coup, en ne filtrant qu'une toute petite partie des *tags*, nous nous retrouvons avec très peu de faux positifs ce qui est un point très important, surtout depuis qu'on connaît l'existence de transcrits rares à rôle biologique déterminant [Khattra *et al.*, 2007].

Finalement dans le cas où le *bruit de fond* est quasiment nul, la plupart des *tags* erronés ne sont pas localisés sur le génome. Nous pouvons donc nous aider du génome pour distinguer les *reads* qui contiennent des erreurs de séquences des autres.

3.5.2 Les séquences non localisées

Comme nous l'avons vu dans la sous-section 3.4.3, que ce soit pour des *reads* génomiques ou des *reads* transcriptomiques, les erreurs de séquences sont la principale raison d'une non localisation sur le génome. Bien sûr ce n'est pas la seule, il peut y avoir des erreurs produites par les protocoles biologiques comme lors de la rétrotranscription pour la DGE mais aussi des causes biologiques comme des SNP ou des variants d'épissage. En comparant SAGE-Illumina® et ChIP-Seq-Illumina® dans la FIGURE 3.6, nous avons réussi à apporter des informations sur ces deux points. Tous les pourcentages qui vont suivre sont donnés pour des ensembles de *tags* complets ($\#occ. > 0$).

Pour les données de ChIP-Seq, plus de 62% des *tags* peuvent être localisés sur le génome ($100 - \mathscr{X}_{chip}(20)$ dans la table 3.1), alors que les 38% restants s'expliquent par des erreurs de séquences ou du polymorphisme. Nous en déduisons que 4.6% des *tags* sont affectés par un SNP ($\mathscr{R}_{chip}(20)$ dans la table 3.1). Ce taux a déjà été publié pour des données LongSAGE :

– soit en calculant la fréquence de l'ensemble des *tags* qui se positionnent sur des SNP connus (2% des *tags* avec une fréquence de 1/1000) [Keime *et al.*, 2007],
– soit en positionnant des ARNm (ou des *tags* associés) sur une banque de données de SNP (8.6% pour une base de données de 2020 SNP associés à des *tags*) [Silva *et al.*, 2004].

Dans les deux cas, ces estimations dépendent de la collection de SNP de départ. En plus, les deux approches négligent le fait que cette collection de SNP est construite à partir de plusieurs individus alors qu'un échantillon d'ARN ne provient que d'un seul individu. Notre estimation semble plus réaliste puisque, d'un côté ChIP-Seq-Illumina® est une collection de séquences d'un seul humain ; alors que de l'autre côté nous localisons ces séquences sur le génome d'un autre humain.

Pour les données de transcriptome comme LongSAGE, 54% des *tags* ($100 - \mathscr{X}_{sage}(20)$ dans la table 3.1) peuvent être localisés. Sachant que le pourcentage de *tags* erronés est quasiment le même que pour le ChIP-Seq et que le taux de SNP estimé est de 4.6%, il reste environ 9% de *tags* non localisés dans LongSAGE qui diffèrent de ChIP-Seq. Cette différence peut en partie s'expliquer par des causes transcriptomiques : jonction de deux exons d'ARN et queue polyA [Rivals *et al.*, 2007] ou encore édition d'ARN et variant alternatif [Keime *et al.*, 2007; Li *et al.*, 2008d]. D'ailleurs notre taux global de 9% est le même que celui publié pour des données LongSAGE avec la technologie Sanger (9.6%) [Keime *et al.*, 2007]. À notre connaissance, nous sommes les premiers à faire ces estimations détaillées des *reads* non localisés sur un génome pour une expérience DGE.

3.5.3 La longueur des séquences : ni trop courte, ni trop longue

La technique SAGE (initialement publiée) permet de générer des séquences de 14 *pb* et son but est d'attribuer un *tag* à chaque transcrit (comme une signature) [Velculescu *et al.*, 1995]. Le fait d'annoter de tels *tags* sur le génome humain a nécessité une extension de leur longueur de 14 à 21 *pb* [Saha *et al.*, 2002]. Avec l'apparition des SHD, la nouvelle tendance est d'augmenter de plus en plus la longueur des séquences, en trois ans nous sommes passés de 36 à 100 *pb* avec la technologie Illumina® afin de couvrir une plus large région du transcrit. Pour localiser des *reads* aussi longs sur un génome, nous avons besoin de logiciels de *mapping* particuliers capables de faire de la recherche approchée. Dans ce chapitre, nous expliquons que, chez Illumina®, les erreurs ont tendance à être de plus en plus fréquentes sur la fin des séquences : 0.48% à la position 20 contre 0.61 à la position 30. Avec des séquences plus longues, on augmente les chances d'avoir un SNP et/ou des causes transcriptomiques dans les séquences, ce qui rend la localisation plus compliquée. Par ailleurs, nous avons démontré qu'avec des séquences ni trop longues, ni trop courtes (20 *pb* pour l'humain) nous pouvions : i/ maximiser le nombre de localisations uniques, ii/ avoir un nombre de fausses localisations négligeables. Donc, en utilisant une simple méthode fondée sur la localisation exacte, nous pouvons trouver plus de vraies prédictions génomiques qu'une méthode de recherche approchée, qui donne des résultats moins précis. En effet, en autorisant des substitutions ou/et des gaps dans les *reads*, nous allons certes augmenter le nombre de localisations, mais surtout le

nombre de localisations multiples et le nombre de fausses localisations (le *bruit de fond* est forcément plus fort). D'ailleurs dans le chapitre 5, nous proposons une méthode spécialisée pour le RNA-Seq qui utilise les propriétés de la recherche exacte pour localiser de longues séquences sur un génome. Nous dressons un tableau où nous nous comparons directement aux méthodes de *mapping* approché et calculons les sensibilités et les spécificités des méthodes.

3.5.4 Une stratégie d'annotation : la détection de nouveaux transcrits

L'identification de nouvelles régions transcrites,pour la plupart non-codantes, a explosé depuis l'essor des technologies haut débit et des tiling arrays [Bertone *et al.*, 2004; Kapranov *et al.*, 2007]. Cependant, les tiling arrays ont deux inconvénients : ils ne couvrent que les régions non répétées du génome [Bertone *et al.*, 2004], et ils ne se focalisent que sur les régions hautement transcrites pour que le signal d'hybridation ne soit pas confondu avec le *bruit de fond*. Mais l'accumulation des EST, des librairies SAGE et DGE et maintenant des librairies RNA-Seq mettent en évidence de nombreux transcrits de faible abondance [Khattra *et al.*, 2007]. La technique DGE permet d'accéder à une détection de transcription sans précédent [Kim *et al.*, 2007]. Les SHD produisent des *tags* rares qui sont biologiquement valides, mais cela reste un défi de les distinguer des *tags* erronés. Nous avons montré qu'en filtrant les *tags* qui ont un nombre d'occurrences $\#occ. = 1$, nous éliminons les 2/3 des séquences erronées et le taux de faux positifs devient négligeable (0.6% des séquences). Notre stratégie permet à l'utilisateur de choisir ou non un filtre (table 3.4) pour s'intéresser aux transcrits rares dans les conditions qu'il désire.

Nous avons validé plusieurs transcrits spécifiques des cellules souches embryonnaires humaines qu'il conviendra de caractériser plus en détail. L'approche globale proposée permet de croiser les informations, provenant non seulement de différentes méthodes mais aussi entre de nombreux tissus grâce aux données SAGE et DGE mises en accès public. Elle permet également de croiser les données provenant de différentes espèces pour rechercher des *tags* conservés, c'est-à-dire localisés sur différents génomes. Une des applications futures peut être la recherche de transcrits intergéniques conservés et exprimés entre l'homme et d'autres espèces. Ce travail pourra être facilement réalisé chez la souris puisque nous disposons de nombreuses données d'expression.

La classification des ARN non-codants est une lourde tâche, loin d'être terminée. En dehors des aspects de transcription, abordés dans ce chapitre, la recherche de structures spécifiques pourrait être envisagée et ainsi compléter le pipeline.

Enfin, l'accumulation de données expérimentales dans les génomes Browser pourra rapidement apporter des informations complémentaires concernant la régulation de la transcription dans ces régions peu annotées des génomes (site TSS, liaison de facteurs de transcription, état de la chromatine, etc).

Finalement, la DGE est une technique ouverte qui se présente avec le RNA-Seq [Sultan *et al.*, 2008] comme l'une des solutions les plus appropriées pour explorer en profondeur le transcriptome des mammifères.

Ce pipeline d'annotation est en cours de publication dans une revue internationale. Ce travail a été réalisé en collaboration de Elias Bou-Samra, Florence Ruffle, Anthony Boureux et Thérèse Commes du Centre de Recherche de Biochimie Macromoléculaire (CRBM), ainsi que Eric Rivals du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM).Le reste du travail a déjà été publié dans Nucleic Acids Research en 2009 [Philippe *et al.*, 2009].

CHAPITRE

4

Structure pour indexer et interroger des *reads* : les Gk *arrays*

La comparaison de millions de reads *contre un génome complet exige la construction d'index adaptés aussi bien pour les* reads *que pour le génome. Dans ce chapitre, nous nous focalisons essentiellement au problème d'indexation de millions de* reads *avec toujours la même question : comment les indexer efficacement ? L'étude proposée garantit un bon compromis espace/temps, c'est-à-dire utiliser un minimum de mémoire tout en ayant un accès rapide à la structure lors de requêtes.*

Sommaire

Les SHD actuels permettent de répondre à des questions clés de type biologique à l'échelle du génome grâce à leur profondeur de séquençage sans précédent. Que ce soit déterminer les variations génétiques ou génomiques, cataloguer des transcrits et évaluer leur niveau d'expression, identifier les interactions entre l'ADN et les protéines ou encore évaluer la diversité des espèces dans un échantillon de métagénomes, toutes ces questions sont maintenant abordables grâce au haut débit. Cependant, chaque problème a besoin d' une analyse bioinformatique différente. Pourtant, cette analyse doit être adaptée et évolutive à la gestion d'une quantité de données qui augmente de jour en jour. Par exemple, aux prémices, dans une expérience de RNA-Seq nous avions environ 8 millions de *reads* de 75 *pb* [Maher *et al.*, 2009a] mais la longueur et le nombre de *reads* augmentent avec le temps [Blow, 2009] .

Positionner des *reads* sur un génome de référence permet d'identifier la localisation génomique d'origine des transcrits. Par exemple, avec la technique de RNA-Seq, ces positions permettent de savoir si un gène est exprimé ou non dans une condition donnée. Néanmoins, l'ensemble des *reads* qui sont localisés ne représentent seulement qu'une part de l'information que nous donnent les *reads*, et elle ne peut être obtenue seulement que si le génome est disponible. En fait, d'autres informations pertinentes sont contenues au sein des *reads*, comme déterminer la fréquence des haplotypes à une position correspondante à un SNP. Cela peut être obtenu en considérant, selon une longueur k, les k-mers qui contiennent le SNP et en recherchant tous les *reads* qui partagent ces mêmes k-mers. Cette procédure est applicable, même en l'absence de génome. Puis d'une façon similaire, nous pouvons imaginer une telle procédure pour la recherche de motifs de fixation du ChIP-Seq afin de déterminer si des régions sollicitées par des ARN messagers sont exprimées différemment.

Pour des tâches telles que l'assemblage de séquences, nous avons besoin de savoir quels *reads* se chevauchent (ou s'entrecroisent) entre eux. Pour ce faire, de nombreux algorithmes d'alignements de type *seed and extend* ont été développés (voir section 2.4.2.2 du chapitre 2), mais il est aussi possible d'effectuer ce travail en recherchant les k-mers communs entre deux séquences distinctes [Altschul *et al.*, 1990; Burkhardt *et al.*, 1999].

Il est certain qu'avec cette évolution grandissante de la technologie haut débit, nous avons et aurons besoin de structures d'indexation efficaces pour organiser et interroger des grandes collections de *reads*. Jusqu'à maintenant, la plupart des outils bioinformatiques sont consacrés au problème de localisation (ou « read *mapping* ») et les plus utilisés sont ceux qui indexent le génome à l'aide de structures compressées [Ferragina et Manzini, 2000]. D'un autre côté, la question d'indexer des *reads* reste assez inexplorée, alors que l'amélioration ou la filtration du produit séquencé suggère que de telles structures soient intégrées dans les futurs outils d'analyse. Toutefois, la tendance vient à changer puisqu'il faut augmenter la précision et le rendement dans les expériences de SHD : les outils de maintenant doivent donc indexer, à la fois, le génome et les *reads* pour interroger un maximum d'informations organisées [Hach *et al.*, 2010].

La plupart des structures d'indexation ne permettent d'indexer qu'un seul texte, comme les arbres des suffixes (ST pour *suffix tree*) ou les tables des suffixes (SA pour *suffix array*) [Weiner, 1973; Manber et Myers, 1990] (voir section 2.3 du chapitre 2). Ces structures permettent de trouver très rapidement toutes les occurrences d'un motif donné dans le texte indexé. Elles peuvent être éventuellement adaptées sur un ensemble de textes (où les textes sont tous différents) pour créer respectivement des arbres des suffixes généralisés (gST) ou des tables des suffixes généralisées (gSA). Cette adaptation peut être faite de deux façons : i/ en concaténant tous les textes et en ajoutant un séparateur entre chacun des textes, c'est-à-dire un symbole qui n'apparaît pas dans l'alphabet (par exemple \$ pour l'alphabet de l'ADN) [Gusfield, 1997] ; ii/ en concaténant tous les textes sans ajouts supplémentaires de caractères, mais dans ce cas il est nécessaire d'avoir une structure annexe pour

enregistrer la longueur de chaque texte [Shi, 1996]. Toutefois pour les deux propositions, il n'est pas possible de gérer des collections de textes (où un même texte peut apparaître plusieurs fois) sauf si dans la méthode i/ nous considérons un séparateur différent entre chaque texte. Finalement, il semblerait ne pas y avoir de structure adéquate pour indexer une collection de *reads*.

Pour des textes colossaux, l'utilisation de structures compressées réduit la mémoire nécessaire pour le stocker. Une telle compression est souvent obtenue en échantillonnant la structure non compressée, c'est-à-dire en ne stockant qu'une partie de l'information. En conséquence, le calcul pour retrouver ce qui n'a pas été enregistré prend du temps. Il est vrai que selon l'échantillonnage effectué, il est possible de contrôler ce compromis espace/temps, mais de toute façon la compression aura un impact sur le temps de requête lors d'une interrogation quelconque sur l'index. Ferragina *et al.* montre dans une évaluation expérimentale que les structures compressées sont de 100 à 1 000 fois plus lentes que les structures normales qui sont 5 fois plus gourmandes en mémoire [Ferragina *et al.*, 2009]. Le FM-index (structure compressée) [Ferragina et Manzini, 2000] est utilisé pour indexer tous les chromosomes dans certaines applications de *mapping* [Li et Durbin, 2009]. Néanmoins, que ce soit pour les structures compressées ou non, l'évolutivité n'a pas été étudiée pour des millions de collections de textes.

Dans ce chapitre, nous nous intéressons à la difficulté d'indexer des millions de *reads*. Nous dressons tout d'abord un état de l'art des structures existantes et adaptables à une telle problématique. Ensuite, nous proposons notre propre solution que nous comparons bien évidemment aux existantes, le but étant de trouver une structure évolutive capable de gérer des collections volumineuses de *reads*, tout en conservant un bon compromis espace-temps.

4.1 Description de notre approche

Nous présentons dans cette section une nouvelle structure de données pour indexer des *reads*, un algorithme qui construit les différentes tables nécessaires, ainsi qu'une procédure pour interroger cette nouvelle structure. Cette dernière, que nous avons appelé *Gk arrays*, conserve en mémoire principale différentes tables pour permettre de répondre, à plusieurs reprises, à des requêtes telles que « selon un k-mer donné, donne moi les *reads* qui contiennent ce k-mer (une seule ou plusieurs fois) ». Avec ces *Gk arrays*, nous pouvons récupérer la position des k-mers dans chaque *read* et/ou les *reads* qui le contiennent ce qui peut s'avérer utile selon les applications. Nous focalisons le problème dans un contexte où des millions de requêtes peuvent être demandées simultanément ; clairement, un bon usage de la mémoire et du temps sera la clé du problème.

Une solution alternative est d'adapter aux *reads* des structures de type SA ou encore tables de hachage dédiées à l'origine pour de longs textes [Gusfield, 1997; Homann *et al.*, 2009]. Nous comparons les *Gk arrays* à de telles adaptations et nous montrons par des méthodes expérimentales qu'ils opèrent les requêtes rapidement tout en prenant moins de mémoire que les autres méthodes (entre

2/3 et 1/3 de moins que les gSA). Nous montrons aussi que contrairement aux tables de hachage, le passage à l'échelle sur le nombre de *reads* n'a pas de répercussions sur le temps de réponse des requêtes.

Notons que si en biologie le terme de k-mer est préféré, les informaticiens utiliseront plutôt le terme k-facteur pour ce genre de problème algorithmique ; toutefois nous conserverons le premier terme. Les *Gk arrays* ne permettent uniquement que de répondre à des requêtes sur les k-mers. Avant de rentrer dans le détail des algorithmes, nous nous devons dans un premier temps de lister les applications biologiques qui s'intéressent à des requêtes sur des k-mers, tout particulièrement dans l'analyse de données de SHD. La section des résultats se découpe en deux parties : la première présente l'algorithme principal, la construction des différentes tables, ainsi que la procédure pour répondre à différentes requêtes sur les k-mers ; la deuxième concerne les expériences et la comparaison avec les autres méthodes. Pour conclure, nous discuterons des avantages de notre structure, puis nous parlerons des futurs améliorations. Notons que dans ce chapitre, nous n'aborderons pas la question du *mapping* mais nous nous concentrons plutôt sur un problème d'indexation.

4.1.1 Les requêtes

D'abord, nous devons commencer par présenter le problème. Nous avons une collection de q *reads* de longueur m et nous considérons des facteurs de longueur $k < m$.

Supposons que nous sommes en possession d'un motif f de longueur k. Nous voulons savoir s'il apparaît au moins une fois dans la collection de *reads* ; autrement dit est-ce que f est un facteur de l'un des *reads* ? Dans la section 4.2.1, nous décrivons une structure de données dans laquelle tous les facteurs de longueur k de tous les *reads* sont rangés par ordre lexicographique. Il suffit donc de chercher f parmi ces facteurs par dichotomie qui, dans le pire des cas, est de l'ordre de $O(k\log((m-k+1)q))$ en temps dans la structure (la recherche par dichotomie utilisée est la procédure standard de ce contexte [Manber et Myers, 1990; Gusfield, 1997]), et détermine les éventuelles positions auxquelles le facteur f commence dans les *reads*. Si aucune occurrence de f n'est trouvée, alors la réponse pour toutes les requêtes suivantes est l'ensemble vide ou zéro. Dans le cas contraire, nous savons qu'il existe au moins une occurrence de f dans la collection de *reads*, et nous souhaitons récupérer toutes les informations relatives à f. Nous voulons répondre, par exemple, aux questions suivantes :

Q1 : Dans quels *reads* le facteur f apparaît ?

Q2 : Dans combien de *reads* f est-il présent ?

Q3 : À quelles positions f apparaît-il dans les *reads* ?

Q4 : Quel est le nombre total d'occurrences de f dans la collection ?

Q5 : Dans quels *reads* le facteur f n'apparaît-il qu'une seule fois ?

Q6 : Dans combien de *reads* f n'est-il présent qu'une seule fois ?

Q7 : À quelles positions f n'apparaît-il qu'une seule fois dans les *reads* ?

Q8 : Quel est le nombre d'occurrences total de f dans la collection lorsque f n'est présent qu'une seule fois dans chaque *read* ?

Nous allons d'ores et déjà faire quelques remarques sur les requêtes précédentes.

1. Les requêtes vont par paires : la première calcule un ensemble de positions alors que la deuxième le cardinal de cet ensemble.
2. Il y a une différence sémantique entre Q1/Q2 et Q3/Q4. La réponse à Q1 est l'ensemble des *reads* dans lesquels f apparaît alors que la réponse à Q3 est l'ensemble des positions où l'on trouve f pour chaque *read*. La différence majeure vient du fait que plusieurs occurrences de f peuvent être dans un même *read* (dans les séquences poly-A par exemple). Parfois les positions sont nécessaires et parfois seulement les *reads* (voir plus bas).
3. Les requêtes Q5-Q8 sont les versions de Q1-Q4 assorties de la contrainte d'avoir une seule occurrence de f dans les *reads*. Autrement les *reads* qui possèdent au moins deux occurrences de f ne sont pas pris en compte. Bien sûr, d'autres variantes dans lesquelles le nombre d'occurrences pourrait être limité par un certain seuil, peuvent être calculées. Dans le cas où f est contraint de n'être qu'une seule fois dans les *reads*, Q6 et Q8 sont équivalentes, nous ne mentionnerons que Q6 par la suite.
4. La structure de données que nous suggérons est gardée en mémoire afin d'être interrogée par un certain nombre de requêtes.

4.1.2 Les applications

Bien que ce travail se focalise sur la structure de données en question, sur son efficacité et sur l'algorithme qui permet de traiter des requêtes, il est primordial de mentionner les applications biologiques sur l'analyse des *reads* qui sont liés à ce genre de requêtes. Rappelons que dans notre contexte, k est plus petit que la longueur de n'importe quel *read*. Nous avons vu dans le chapitre 3 qu'en choisissant un $k \geq 19$ ou 20, les k-mers indiquent en moyenne une localisation unique sur le génome humain [Philippe *et al.*, 2009]. Dans le cas général, la valeur de k peut être calculée en fonction de la taille du génome. Ainsi lorsque deux *reads* partagent le même k-mer, il est fort probable qu'ils soient séquencés à partir du même endroit de l'ADN. En d'autres mots, partager un k-mer est un marqueur pour avoir la même position génomique d'origine.

4.1.2.1 La détection de mutation

Les mutations ponctuelles telles que les SNP, les mutations somatiques ou encore les courts indels sont indiqués par des différences entre le *read* et le génome. À partir des *reads* qui sont localisés sur le génome, nous pouvons analyser la sous-collection de ceux qui couvrent une même position génomique et ainsi compter combien d'entre eux supportent une quelconque variation observée

entre *reads* et génome. Si nous considérons deux facteurs de longueur k centrés sur la position de la mutation, un sur le *read* et un sur le génome, alors en répondant à la requête Q2 pour ces deux facteurs nous avons un comptage approximatif des deux haplotypes. Dans le cas où nous voulons récupérer les *reads* associés au facteur, alors Q1 est la requête appropriée. Par conséquent, si un ou seulement un petit nombre de *reads* partagent ce k-mer, alors nous pouvons suspecter une erreur de séquence [Salmela, 2010].

4.1.2.2 La couverture locale

Supposons que nous nous intéressons à une séquence cible : un *read* ou une séquence externe. Pour chacun des k-mers de cette cible, nous appelons la *couverture locale* le nombre de *reads* qui partage ces k-mers (c'est-à-dire la réponse à Q2). Le profil de la couverture locale (c'est-à-dire son histogramme) tout au long de la séquence cible fournit des informations utiles dans des contextes variés. Pour un ARNm connu et une expérience RNA-Seq, la moyenne de la couverture locale de tous les k-mers peut être vue comme le niveau d'expression de la cible, tandis que le profil d'expression de chaque k-mer permet de distinguer les sous-régions exprimées de la cible à différents niveaux [Denoeud *et al.*, 2008; Trapnell *et al.*, 2010]. Dans un tout autre contexte, avec une expérience génomique, nous pouvons faire des requêtes sur les k-mers des *reads*. Le profil de chaque k-mer peut nous aider à détecter les régions répétées ou les éléments transposables, cela peut s'avérer utile d'étudier la distribution et l'évolution de ces éléments dans le génome.

4.1.2.3 Regrouper et assembler des séquences sans génome de référence

Comme pour les EST, il peut être approprié de regrouper et d'assembler des *reads* issus du RNA-Seq pour calculer des variants de transcription qui seraient exprimés dans l'expérience [Denoeud *et al.*, 2008; Trapnell *et al.*, 2010]. Pour assembler, il est nécessaire de trouver le meilleur alignement entre des paires de *reads*, ce qui est souvent effectué efficacement en filtrant par des graines. Dans ce cas, les très efficaces et sensibles graines qui sont utilisées partagent exactement des k-mers [Burkhardt *et al.*, 1999]. Ici, la sous collection de *reads* partageant un k-mer avec un *read* donné (ou les k-mers positions) peut être obtenue avec Q3. Quant à la réponse de Q4, elle peut nous guider pour la procédure de « clustering ».
Des besoins similaires en requête peuvent se manifester pour l'assemblage de *reads* génomiques [Miller *et al.*, 2010; Conway et Bromage, 2011]. Pour savoir quels *reads* peuvent être assemblés avec un *read* donné sans ambiguïté, nous pouvons utiliser Q7 en s'intéressant aux k-mers des extrémités 5' ou 3' du *read*. Ainsi, nous avons des *reads* potentiels et des positions pour choisir le meilleur assemblage entre les *reads*.

Notre liste d'applications fournit des exemples et bien sûr la liste n'est pas exhaustive. Nous pourrions aussi mentionner, par exemple, l'estimation de la longueur du génome cible dans un

contexte d'assemblage, en utilisant un comptage de k-mers [Marcais et Kingsford, 2011]. De plus, tous ces exemples soulignent l'importance de notre structure de données pour l'analyse des séquences haut débit dans des contextes variés d'applications biologiques, elle pourrait être ainsi intégrée au sein de programmes qui analysent les *reads*.

4.2 Implantation de la structure et résultats

Cette section contient notre principale contribution à ce chapitre : une structure de données qui indexe de très grandes collections de *reads*. Nous avons appelé cette structure les *Gk arrays*. Afin de décrire ces derniers le plus clairement possible, nous commençons dans la sous-section 4.2.1 par introduire les notations, formaliser les requêtes, exposer la structure d'indexation, proposer l'algorithme de construction, et donner les différentes procédures pour répondre aux requêtes. Ensuite, dans la sous-section 4.3, nous étudierons son utilité pratique comparée à deux structures alternatives : une table des suffixes généralisée et une table de hachage. Ce travail inclut des comparaisons pratiques et théoriques.

4.2.1 L'algorithme principal

Nous allons détailler dans un premier temps les algorithmes de construction des *Gk arrays* et dans un second temps les réponses aux requêtes. Cependant, nous avons besoin de définir plus formellement les requêtes et d'introduire les notations nécessaires.

4.2.1.1 Notations et définitions des requêtes

Soit Σ un alphabet de taille σ. $\Sigma^\star$ dénote l'ensemble des *mots* ou *séquences* sur Σ et, pour tout entier n, Σ^n représente l'ensemble des mots de longueur n sur Σ. Pour un mot x, $|x|$ est la *longueur* de x. Soient deux mots x et y, nous dénotons par xy la *concaténation* de x et y. Pour tout $0 \leq i \leq j \leq |x|-1$, $x[i]$ est le $(i+1)^{\text{ième}}$ élément de x, et $x[i..j]$ est le *facteur* $x[i]x[i+1]\dots x[j]$. Soit $\leq_L$ l'opérateur de comparaison pour l'ordre lexicographique des mots. Le rang lexicographique commence à zéro et toutes les tables sont indexées à partir de zéro. Pour tout ensemble fini A, son cardinal est représenté par $\#A$.

En entrée, nous avons une liste de q *reads* $R = (r_0, \dots, r_{q-1})$ de longueur m (dans le cas où tous les *reads* sont de la même longueur). Nous parlons ici de liste car les *reads* ne sont pas nécessairement tous distincts. Nous savons que $m, k, q \in \mathbb{N}$ satisfont $m \geq k > 0$.

Un k-facteur d'un mot est appelé dans notre contexte un *k-mer*. Pour tout $u \in \Sigma^\star$, nous appelons $F_k(u)$ l'ensemble des k-mers dans u : $F_k(u) = \{v \in \Sigma^k \mid \exists p \in [0, |u|-k] \text{ tel que } v = u[p..p+k-1]\}$. Soit $f \in \Sigma^k$, l'ensemble des *reads* où f apparaît est alors dénoté par $Ind_k(f) = \{j \in [0, q[\mid f \in F_k(r_j)\}$, et l'ensemble *occurrences positionnées* de f dans tous les *reads* par $Pos_k(f) = \{(j, \ell) \mid r_j[\ell..\ell+k-1] = f\}$, où une occurrence positionnée est donnée par la paire (*read* indexé dans

R, position du début de f dans le *read*). Nous appelons $UInd_k(f)$ (resp. $UPos_k(f)$) les restrictions de $Ind_k(f)$ (resp. $Pos_k(f)$) où nous nous intéressons aux *reads* qui contiennent le facteur f qu'une seule fois. Formellement, $UPos_k(f) = \{(j,\ell) \mid r_j[\ell\,..\,\ell+k-1] = f \text{ et } \forall i \neq \ell, r_j[i\,..\,i+k-1] \neq f\}$, puis $UInd_k(f) = \{j \mid (j,\ell) \in UPos_k(f)\}$.

Soit $i \in [0,q[$, $j'' \in [0,\hat{m}[$, et soit f le k-mer qui commence à la position j'' dans le *read* r_i. Rappelons qu'ici nous devons connaître la paire (i,j''), qui définit le k-mer f. Maintenant, nous avons tout le matériel nécessaire pour définir formellement les requêtes.

Q1 : l'ensemble $Ind_k(f)$ — **Q2 :** $\#Ind_k(f)$, le cardinal de $Ind_k(f)$
Q3 : l'ensemble $Pos_k(f)$ — **Q4 :** $\#Pos_k(f)$, le cardinal de $Pos_k(f)$
Q5 : l'ensemble $UInd_k(f)$ — **Q6 :** $\#UInd_k(f)$, le cardinal de $UInd_k(f)$
Q7 : l'ensemble $UPos_k(f)$

Les algorithmes pour calculer $UInd_k(f)$, respectivement $UPos_k(f)$, pour répondre aux requêtes Q5/Q7, consiste à filtrer $Ind_k(f)$, resp. $Pos_k(f)$, à la volée, et ils sont donc quasiment similaires aux algorithmes pour calculer Q1/Q3. De ce fait, nous ne détaillerons que les solutions des quatre premières requêtes.

4.2.1.2 Définitions des tables

Notre algorithme s'appuie sur quatre tables qui vont stocker et organiser tous les k-mers de tous les *reads* de façon à pouvoir faire des requêtes facilement. Pour commencer, nous définissons un texte qui est la concaténation de tous les *reads* : $C_R = r_0 r_1 \cdots r_{q-1}$ et c'est à partir de C_R que nous allons récupérer tous les k-mers. Cependant, les k-mers dans C_R qui chevauchent deux *reads* consécutifs ne sont pas à indexer. C'est pourquoi, nous introduisons un système de re-numérotation afin de n'indexer que les k-mers qui commencent à des positions intéressantes dans C_R. En procédant de la sorte, nous économisons de l'espace dans les Gk *arrays* en écartant directement les positions des k-mers chevauchants dans C_R. Soit $\hat{r}$ le nombre de k-mers distincts de tous les *reads* et par souci de clarté, nous définissons $\hat{m} := m-k+1$ et $\hat{q} := q\hat{m} = q(m-k+1)$ (respectivement le nombre de positions intéressantes dans un *read* et dans C_R). Nous nommons :

- *P-position* , une position de départ dans C_R d'un k-mer qui ne chevauche pas deux *reads* consécutifs, c'est-à-dire un élément de $P_{\text{pos}} := [0, qm[\setminus \{j \mid (j \bmod m) \geq \hat{m}\}$.
- g, la fonction qui re-numérote P-positions *dans l'ordre* tel que leur index soit consécutif ; g est définie par :

$$\begin{array}{rccl} g: & P_{\text{pos}} & \longrightarrow & Q_{\text{pos}} \\ & j & \longmapsto & \lfloor j/m \rfloor \times \hat{m} + (j \bmod m)\,. \end{array}$$

- *Q-position* , une image de P-position de la fonction $g(.)$, c'est-à-dire un élément de $Q_{\text{pos}} := g(P_{\text{pos}}) = [0, \hat{q}[$. Notons que l'ensemble Q_{pos} n'est pas une requête.

Nous pouvons remarquer que puisque P_{pos} et Q_{pos} ont exactement le même nombre d'éléments $\hat{q}$ et que $(j \neq j')$ implique que $g(j) \neq g(j')$, alors g est bijective. En conséquence, la fonction inverse g^{-1} existe et positionne une Q-position dans C_R à partir d'une P-position. La proposition 4.1 explique la conversion entre une occurrence positionnée et une P-position.

Proposition 4.1. *Soient* (j, ℓ) *avec* $j \in [0, q[$, $\ell \in [0, \hat{m}[$ *la position d'une occurrence d'un* k*-mer dans un* read. *La position* P*-position correspondante dans* C_R *est* $jm + \ell$. *Inversement, soit* j' *une* P*-position, alors la position correspondante de l'occurrence dans le* read *est* $(\lfloor j'/m \rfloor, j' \bmod m)$.

Ce système de numérotation est important pour nous permettre des allers retours entre une occurrence positionnée dans un *read*, sa P-position correspondante dans C_R, puis sa Q-position qui sera stockée dans nos Gk *arrays*.

Soit j une Q-position. Nous dénotons par $s_Q(j)$, resp. $f_Q(j)$, le suffixe, resp. le k-mer, de C_R qui commence à la position P-position de $g^{-1}(j)$, c'est-à-dire $s_Q(j) = C_R[g^{-1}(j)..qm-1]$ et $f_Q(j) = C_R[g^{-1}(j)..g^{-1}(j)+k-1]$. Nous appelons $s_Q(j)$ un P-suffixe. Notons que tous les suffixes qui commencent aux P-positions sont de différentes longueurs et deux à deux distincts. Cela implique que nous avons en tout $\hat{q}$ suffixes qui ont tous un rang lexicographique différent. Cependant, ce n'est pas forcément le cas pour les k-mers, c'est-à-dire $f_Q(j)$. Du coup, nous définissons $\{f_Q(j) \mid j \in Q_{\text{pos}}\}$ par l'ensemble des P_k*-facteurs*, qui a pour cardinalité $\hat{r}$ avec notre notation.

Nous pouvons définir les Gk *arrays* de la façon suivante :

GkSA (Generalized k Suffix Array) est une table des suffixes de C_R modifiée qui range par ordre lexicographique seulement les P-suffixes,

GkIFA (Generalized k Inverse Factor Array) est une table inverse de suffixes modifiée qui enregistre pour chaque Q-position, dans l'ordre d'apparition dans C_R, le rang lexicographique des P_k-facteurs qui commencent à la P-position correspondante,

GkCFA (Generalized k Counting Factor Array) est une table qui associe à un k-mer (plus précisément à son rang) son nombre d'occurrences aux P-positions dans C_R,

GkCFPS (Generalized k Counting Factor Prefix Sum) enregistre la somme des préfixes de *GkCFA*. En fait *GkCFA* et *GkCFPS* sont équivalents seulement la deuxième est plus efficace en pratique.

De manière plus formelle, nous obtenons (voir TABLE 4.2.2.2 pour un exemple) :

– Soit i : un rang lexicographique d'un suffixe, et j : une Q-position (c'est-à-dire $i, j \in Q_{\text{pos}}$), alors on a :
$GkSA[i] = j$ ssi $s_Q(j)$ possède un rang lexicographique i parmi les P-suffixes.

- Soit i : un rang lexicographique d'un k-mer et j : une Q-position (c'est-à-dire $i \in [0, \hat{r}[$ et $j \in Q_{\text{pos}}$), alors on a :
  $GkIFA[j] = i$ ssi $f_Q(j)$ possède un rang lexicographique i parmi les P_k-facteurs.
- Soit i : un rang lexicographique d'un k-mer (c'est-à-dire $i \in [0, \hat{r}[$) alors on a :
  $GkCFA[i] = \#\{j \in Q_{\text{pos}} \mid f_Q(j) = f_Q(GkSA[i])\}$.
- Soit i : un rang lexicographique d'un k-mer (c'est-à-dire $i \in [0, \hat{r}[$) alors la définition de la somme des préfixes de *GkCFA* est :
  $GkCFPS[i] = \sum_{t=0}^{i} GkCFA[t]$ et $GkCFPS[-1] = 0$.

Remarque 8. *La table GkCFPS n'est pas essentielle à l'algorithme : elle est seulement créée afin d'éviter de calculer inutilement la somme des préfixes de GkCFA (voir la définition de GkCFPS plus haut). De plus, les valeurs de GkCFA peuvent être retrouvées en temps constant en utilisant* $GkCFA[i] = GkCFPS[i] - GkCFPS[i-1]$. *Finalement, il est préférable de conserver GkCFPS à la place de GkCFA.*

4.2.1.3 Quelques propriétés sur les Gk *arrays*

Nous vous présentons dans ce paragraphe des propriétés sur les Gk *arrays* qui seront utiles pour la compréhension de la suite du travail.

Proposition 4.2. *Pour tout* $i \in [0, \hat{r}[$, $GkCFPS[i] = \#\{j \in Q_{pos} \mid f_Q(j) \leq_L f_Q(GkSA[i])\}$ *(Preuve par induction).*

En d'autres termes, $GkCFPS[i]$ est le nombre de P_k-facteurs qui ont des rangs lexicographiques plus petits ou égaux à i. D'ailleurs si la table *GkSA* est triée par ordre lexicographique des P-suffixes, elle est aussi triée par ordre lexicographique des P_k-facteurs. Dans ce cas, nous avons la propriété suivante :

Proposition 4.3. *Soit* $f \in \Sigma^k$ *tel que* $Ind_k(f) \neq \emptyset$. *Toutes les occurrences de* f *ont le même rang lexicographique parmi les* P_k*-facteurs et elles sont stockées de façon consécutive dans GkSA.*

4.2.2 L'algorithme de construction par étape

Par souci de clarté, nous divisons le détail des algorithmes en deux parties : la construction de *GkSA*, et les constructions de *GkIFA* et *GkCFPS*.

4.2.2.1 La table *GkSA*

Nous commençons par construire la table SA pour tous les suffixes de C_R, en utilisant un algorithme linéaire en espace et en temps. Sachant que $|C_R| = mq$, cette première étape est faite en $O(mq)$. Par la suite, *GkSA* est obtenue à partir de SA en ne sélectionnant que les P-positions qu'on

re-numérote en Q-position en utilisant la fonction g. Cette deuxième étape est effectuée en $O(mq)$ en temps et en espace. Par ailleurs, *GkSA* est construite à la place de SA : notre algorithme n'utilise pas d'espace en supplément de l'allocation de SA.

Concernant les requêtes Q1/Q2, chaque *read* possédant une occurrence d'un P_k-facteur donné, ne devra être compté qu'une seule fois (même si le P_k-facteur apparaît plus d'une fois dans le *read*). De manière similaire pour les requêtes Q5/Q6, nous comptons seulement les *reads* où P_k-facteur n'est présent qu'une et une seule fois. Afin d'éviter d'utiliser un masque sur les *reads* pour mémoriser les *reads* qui ont déjà été parcourus, nous trions par ordre croissant les valeurs de *GkSA*. Ces dernières correspondent aux P_k-facteurs partageant le même rang lexicographique, comme nous pouvons le voir dans la table 4.2). Les valeurs qui doivent être triées sont les Q-positions, c'est-à-dire des entiers. Par conséquent, le tri peut être effectué en temps linéaire sur les valeurs de *GkSA* en utilisant un tri par base, ou « radix sort » (page 172 de [Cormen *et al.*, 2001]). La procédure complète est donc de l'ordre de $O(mq)$ en temps et en espace.

4.2.2.2 Les tables *GkIFA* et *GkCFA*

L'algorithme 2 nous montre comment construire simultanément *GkIFA* et *GkCFA*.

Algorithme 2 : *GkIFA* and *GkCFA* building.

Data : *GkSA*, C_R, k, $\hat{q}$
Result : *GkIFA* et *GkCFA*

```
1  begin
2      GkIFA[GkSA[0]] ⟵ 0 ;
3      GkCFA[0] ⟵ 1 ;
4      t ⟵ 0 ;
5      foreach i ∈ [1, q̂[ do
6          j ⟵ GkSA[i];
7          j' ⟵ GkSA[i − 1];
8          if f_Q(j) ≠ f_Q(j') then
9              t ⟵ t + 1 ;
10             GkCFA[t] ⟵ 0 ;
11         GkIFA[j] ⟵ t ;
12         GkCFA[t] ⟵ GkCFA[t] + 1;
13     return (GkIFA et GkCFA);
14 end
```

Théorème 4.1. *L'algorithme 2 construit correctement les tables GkIFA et GkCFA (voir preuve plus bas).*

Démonstration du Théorème 4.1 : Nous voulons prouver que l'algorithme 2 construit correctement les tables *GkIFA* et *GkCFA*.

Initialisation de la ligne 2 jusqu'à la ligne 4. *GkSA*[0] est le plus petit *P*-suffixe dans l'ordre lexicographique parmi les *P*-suffixes. En conséquence, son rang lexicographique est initialisé à 0 (ligne 2), puis enregistré dans la variable t (ligne 4). Jusqu'à maintenant, il n'y a qu'un seul P_k-facteur qui a pour rang 0 (ligne 3).

Boucle principale de la ligne 5 jusqu'à la ligne 12. La table *GkSA* qui stocke les *P*-suffixes triés dans l'ordre lexicographique, sont scannés de la position 1 jusqu'à la position $\hat{q}-1$. Le rang du dernier P_k-facteur est sauvegardé dans t lorsqu'on entre dans la boucle. La ligne 8 détermine le moment où le P_k-facteur actuel diffère ou non des précédents. Dans le cas échéant, t est incrémenté et le P_k-facteur actuel devient le nouveau facteur dans l'ordre lexicographique. De plus, son nombre d'occurrences *GkCFA*[t] est initialisé à zéro (ligne 9-10). Maintenant, par induction nous savons que t est le rang lexicographique du P_k-facteur actuel. Dans ce cas, t est enregistré à la position j de *GkIFA* (ligne 11). Finalement, *GkIFA* est correctement construite.

En outre, à chaque fois qu'un P_k-facteur ayant pour rang t est rencontré, son compteur *GkCFA*[t] est incrémenté d'une unité (ligne 12). À la fin de l'algorithme *GkCFA*[t] enregistre correctement le nombre d'occurrences de P_k-facteur de rang t, ce qui correspond bien à la définition.

□

La comparaison entre deux P_k-facteurs (ligne 8) est effectuée de manière naïve en $O(k)$, et c'est la seule instruction à l'intérieur de la boucle qui ne se calcule pas en temps constant. Nous pouvons donc affirmer que la construction des deux tables *GkIFA* et *GkCFA* se fait en $O((m-k)qk)$. Nous mettons l'accent sur la simplicité de l'algorithme afin d'expliquer la rapidité de construction que nous obtenons en pratique.

Remarque 9. *Nous ne parlons pas de la construction de GkCFPS mais ses valeurs sont calculées à la place de celles de GkCFA en* $O((m-k)q)$ *(voir la Remarque 8).*

TABLE 4.1 : L'illustration des *Gk arrays*.

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$C_R[j]$	*a*	*a*	*c*	*a*	*a*	*c*	*t*	*c*	*a*	*a*	*t*	*t*	*c*	*a*	*a*	*a*	*c*	*a*	*a*	*g*	*c*
SA[j]	13	0	14	3	17	8	1	15	4	18	9	20	12	2	16	7	5	19	11	6	10
$g(j)$	0	1	2	3	4			5	6	7	8	9			10	11	12	13	14		

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GkSA[i]	0	10	3	13	6	1	11	4	14	7	2	12	5	9	8
rank	0			1	2	3		4	5	6	7			8	9

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GkIFA[i]	0	3	7	0	4	7	2	6	9	8	0	3	7	1	5

ℓ	-1	0	1	2	3	4	5	6	7	8	9
GkCFA[ℓ]		3	1	1	2	1	1	1	3	1	1
GkCFPS[ℓ]	0	3	4	5	7	8	9	10	13	14	15

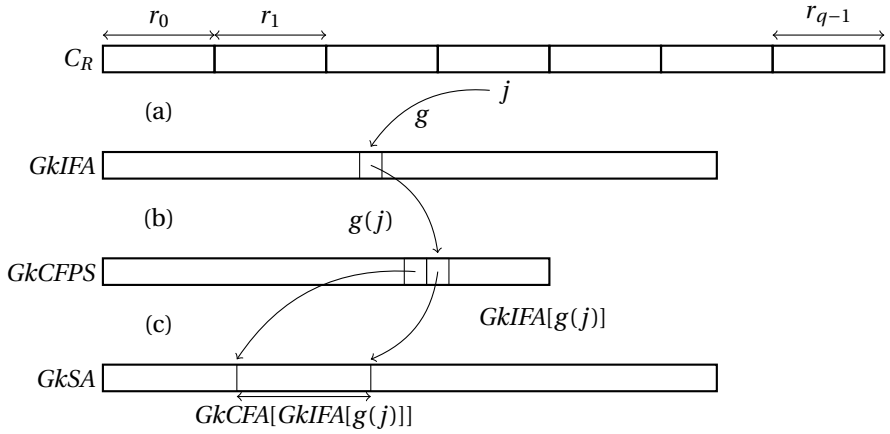
Exemple pour une collection R = (*a*acaact, *c*aattca, *a*acaagc) avec $q = 3$ reads de longueur $m = 7$ et en considérant 3-mers ($k = 3$). L'index est composé de trois tables et en utilise une quatrième pendant la construction (*GkSA*, *GkIFA*, *GkCFA*, et *GkCFPS*). La première table enregistre les indices correspondant au point de départ des k-mers dans le texte C_R, lui même construit par la concaténation de tous les reads. La table des suffixes (SA) de C_R est tout d'abord construite, et la fonction g permet la renumérotation des P-positions de C_R pour les rendre consécutives. Les P-positions de l'exemple sont $\{0,1,2,3,4,7,8,9,10,11,14,15,16,17,18\}$; les autres positions, correspondantes au début de k-mers qui chevauchent deux reads, sont mises en retrait avec un fond gris (lignes j et SA[j]). La ligne SA correspond à la table des suffixes de C_R. Le k-mer *c*aa possède quatre occurrences dans C_R aux positions 2, 7, 12 and 16. Parmi ces positions, seulement 2, 7, et 16 sont des P-positions. Le rang lexicographique de P_k-facteur qui commence en position 16 est donné par $GkIFA[g(16)] = GkIFA[12] = 7$, et le nombre d'occurrences du P_k-facteur *c*aa est donné par $GkCFA[7]$, qui est égale à 3. Les positions des occurrences de *caa* dans C_R sont obtenues par l'ensemble $\{g^{-1}(GkSA[j]) \mid GkCFPS[7-1] \leq j < GkCFPS[7]\} = \{2,7,16\}$. Confère la figure 4.1 qui représente une illustration de cette procédure.

TABLE 4.2 : La table *GkSA* avec des valeurs triées par ordre croissant pour les rangs identiques.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GkSA[i]	**0**	**3**	**10**	13	6	1	11	4	14	7	2	5	12	9	8
rank	0			1	2	3		4	5	6	7			8	9

Dans cette table, comparé à *GkSA* de la table 4.2.2.2, les valeurs qui ont pour rang 0 sont maintenant ordonnées et apparaissent à 0, 3, 10 au lieu de 0, 10, 3.

FIGURE 4.1 : La démarche pour accéder aux occurrences d'un k-mer dans les Gk *arrays*.



Visualisation du cheminement pour accéder aux occurrences d'un k-mer qui commence à la position j dans C_R. Accès à *GkIFA*, *GkCFPS* et *GkSA* : (a) À partir de la position j dans C_R, *GkIFA* : $g(j)$ est la position re-numérotée de la P-position j. (b) Ensuite *GkCFPS* : *GkIFA*[$g(j)$] est le rang lexicographique du P_k-facteur qui commence à la P-position j dans C_R, et *GkCFPS*[*GkIFA*[$g(j)$]] est le nombre d'occurrences dans C_R des P_k-facteurs pour les rangs inférieurs à *GkIFA*[$g(j)$]. (c) De *GkCFPS* à *GkSA* : les positions des occurrences de P_k-facteur sont dans *GkSA* dans l'intervalle [*GkCFPS*[*GkIFA*[$g(j)$] − 1], *GkCFPS*[*GkIFA*[$g(j)$]]].

4.2.3 La procédure pour répondre aux requêtes

Supposons que les Gk *arrays* aient été construits au préalable (voir section 4.2.2). Nous allons dans cette section détailler les algorithmes permettant de répondre aux quatre premières requêtes (les quatre suivantes se déduisant trivialement des premières). En amont, nous allons ajouter quelques précisions supplémentaires sur les requêtes Q4 et Q3 pour que la compréhension des algorithmes soit plus limpide.

Soit $i \in [0, q[$, $j'' \in [0, \hat{m}[$, et soit f un k-mer commençant à la position j'' dans le *read* r_i. L'occurrence de f dans C_R se situe à la P-position $j' := im + j''$ et sa Q-position correspondante est $j := g(j')$.

Q4 : Calcul du cardinal de $Pos_k(f)$. En premier lieu, nous devons trouver le rang lexicographique de f parmi les P_k-facteurs, que nous pouvons directement obtenir avec $t :=$ *GkIFA*[j] (d'après la définition de *GkIFA*). Le cardinal de $Pos_k(f)$ est simplement le nombre d'occurrences qui commencent aux P-positions dans C_R que nous obtenons à l'aide *GkCFA*[t] (par définition de *GkCFA*). Par ailleurs, d'après la remarque 8 nous pouvons poser *GkCFA*[t] = *GkCFPS*[t] − *GkCFPS*[$t-1$].

Q3 : Calcul de $Pos_k(f)$. D'après la proposition 4.3, toutes les occurrences de f qui commencent aux P-positions sont enregistrées de manière consécutive dans *GkSA*. Cela nous suffit à trouver le

plus petit et le plus grand des indices, que nous nommons respectivement ℓ_f et u_f. Dans l'ordre de lecture de *GkSA*, toutes les occurrences des facteurs plus petits que f dans l'ordre lexicographique sont enregistrées avant. D'après la définition de *GkCFPS* et la Proposition 4.2, nous obtenons $u_f = GkCFPS[t]$ et $\ell_f = GkCFPS[t-1]$. Puisque *GkSA* est indexée à partir de 0, les premières Q-positions des occurrences de f sont comprises dans l'intervalle $[\ell_f, u_f[$ de *GkSA*. Les P-positions correspondantes sont obtenues en utilisant la fonction $g^{-1}(.)$ et sont alors transformées en occurrences positionnées grâce à la Proposition 4.1. Cela prouve le Théorème 4.2.

Théorème 4.2. *Soit f un k-mer d'un* read *qui apparaît à la Q-position j dans C_R. Alors, son rang lexicographique parmi les P_k-facteurs est $t := GkIFA[j]$. Si nous avons comme bornes $u_f := GkCFPS[t]$ et $\ell_f := GkCFPS[t-1]$ alors*

1. *les premières P-positions des occurrences de f dans C_R sont $\{g^{-1}(GkSA[\ell]) \mid \ell \in [\ell_f, u_f[\}$,*
2. *$Pos_k(f) = \{(\lfloor g^{-1}(GkSA[\ell])/m \rfloor\ ,\ g^{-1}(GkSA[\ell]) \bmod m) \mid \ell \in [\ell_f, u_f[\}$,*
3. *$\#Pos_k(f) = u_f - \ell_f$.*

En utilisant le Théorème 4.2, les requêtes concernant $Ind_k(f)$ peuvent être traitées de la façon suivante :

Q1 : $Ind_k(f) := \{\lfloor g^{-1}(GkSA[\ell])/m \rfloor \mid \ell \in [\ell_f, u_f[\}$,

Q2 : en comptant les éléments $Ind_k(f)$ qui ont été calculés en 1).

Les algorithmes pour Q1, Q3, et Q4 sont donnés successivement par les algorithmes 3, 4, et 5.

Algorithme 3 : Q1 ($Ind_k(f)$) for *Gk arrays*.

```
Data : f ∈ Σ^k, j ∈ P_pos such that C_R[j..j+k-1] = f
Result : The set Ind_k(f)
1  begin
2      Ind_k ⟵ empty set;
3      t ⟵ GkIFA[j];
4      ℓ_f ⟵ GkCFPS[t-1];
5      u_f ⟵ GkCFPS[t];
6      prev ⟵ −1;
7      foreach i ∈ [ℓ_f, u_f[ do
8          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
9          if readIndex ≠ prev then
10             Add readIndex to Ind_k ; prev ⟵ readIndex;
11     return (Ind_k);
12 end
```

Pour répondre à la requête Q7, nous commençons par calculer $Pos_k(f)$ et nous parcourons cet ensemble à la volée pour enlever les *reads* (ou les occurrences positionnées) qui possèdent au

Algorithme 4 : Q3 ($Pos_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : The set $Pos_k(f)$

```
1  begin
2      Pos_k ⟵ empty set;
3      t ⟵ GkIFA[j];
4      ℓ_f ⟵ GkCFPS[t − 1];
5      u_f ⟵ GkCFPS[t];
6      foreach i ∈ [ℓ_f, u_f[ do
7          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
8          posInRead ⟵ g^{-1}(GkSA[i]) mod m;
9          Add the pair (readIndex, posInRead) to Pos_k;
10     return (Pos_k);
11 end
```

Algorithme 5 : Q4 ($\#Pos_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : The cardinality of $Pos_k(f)$

```
1  begin // GkCFA[t] = GkCFPS[t] − GkCFPS[t − 1]
2      t ⟵ GkIFA[j];
3      return (GkCFA[t]);
4  end
```

moins deux occurrences de f. Une méthode similaire peut appliquée pour résoudre Q5. Les algorithmes pour les autres requêtes sont inclus dans le chapitre annexe en section A.1. En fait, d'autres variantes de requêtes sont envisageables en ajustant un paramètre qui fixerait le nombre d'occurrences tolérées d'un facteur f dans un *read*.

4.2.3.1 Complexité

Afin de répondre aux requêtes Q1-Q3 ou Q5, il est nécessaire de parcourir les valeurs de *GkSA* correspondantes à l'intervalle pour le k-mer f. Le temps requis pour ce travail est de l'ordre de $O(occ\_Reads(f))$, où *occ_Reads(f)* représente le nombre d'occurrences de f dans les *reads*. La requête Q4 s'effectue en temps constant, en utilisant *GkCFPS*.

4.2.4 Les considérations pratiques sur les Gk *arrays*

La valeur de k est un paramètre des *Gk arrays*. Elle détermine la longueur des k-mers qui sont plus tard utilisés pour faire des requêtes sur une collection de *reads*. Pour simplifier la compréhension des algorithmes, nous avons supposé jusqu'à maintenant que tous les *reads* avaient la même

longueur. Cependant, les *Gk arrays* sont flexibles et ils peuvent intégrer une collection de *reads* à longueur variable. D'ailleurs, cette adaptation est nécessaire pour que la structure soit polyvalente aux différentes technologies de séquençage car certaines produisent ce genre de *reads*, comme le séquenceur Roche 454®.

4.2.4.1 Indexation des *reads* à longueur variable

Nous montrons comment notre méthode peut être légèrement adaptée pour résoudre ce problème. Rappelons que les *Gk arrays* considèrent le texte (ou mot) C_R qui est une concaténation des *reads*. Lors de l'indexation des k-mers de C_R, nous économisons de la place en écartant les positions associées aux k-mers qui chevauchent deux *reads* consécutifs. Cette étape peut être faite efficacement en convertissant les P-positions en Q-positions par l'intermédiaire de la fonction g. Jusqu'à maintenant, cette fonction partait de l'hypothèse que la longueur des *reads* était fixe. De ce fait, nous devons modifier la définition de g pour s'accommoder à des longueurs variables. Une solution possible est d'utiliser un vecteur de bits F, aussi long que C_R, pour enregistrer les positions qui correspondent à des P-positions : j est une P-position ssi $F[j] = 1$. Nous avons implanté un vecteur muni des fonctions rank et select afin de le manipuler efficacement [Munro, 1996; Raman *et al.*, 2002]. Nous définissons ces opérations comme :

- $\mathsf{rank}_1(F, i)$ est le nombre de P-positions dans $F[0..i]$.
- $\mathsf{select}_1(F, i)$ est la $i^{ième}$ P-position dans F (ou $|F|$ s'il y a moins de i P-positions dans F).

Ces opérations peuvent être faites en temps constant et F peut être enregistré dans un format compressé pour un coût de $|F|H_0(F) + o(|F|)$ bits, où H_0 est l'entropie empirique d'ordre 0 de F. Du coup, la modification de $g(j)$ et $g^{-1}(j)$ peut être simplement réalisée à l'aide des opérations rank et select. En fait, nous obtenons $g(j) = \mathsf{rank}_1(F, j)$, et $g^{-1}(j) = \mathsf{select}_1(F, j)$. En conclusion, avec un léger surplus de mémoire, les *Gk arrays* peuvent résoudre les problèmes de *reads* à longueur variable.

4.2.4.2 Implantation

Les *Gk arrays* sont disponibles sous la forme d'une librairie C++ sous une licence GPL de type Cecill. Cette librairie prend en entrée une collection de *reads*, sous tout type de formats standards (FASTA, FASTQ, RAW, etc). En fonction du nombre total de k-mers présents dans les *reads*, l'utilisateur peut choisir de compiler la librairie sur 32 ou 64 bits. Cela permet de traiter plus de 2^{31} P-positions. Par défaut, l'option de compilation est à 32 bits. Nous proposons aussi une autre option de compilation permettant de gérer des *reads* de longueur variable. Par défaut cette option n'est pas activée ce qui ne permet de traiter que des collections de *reads* à longueur fixe.

Les algorithmes de construction de la structure de données et des différentes requêtes sont codés en C et C++. La table des suffixe (SA) utilisée pour construire C_R est l'implantation de libdivsufsort (https://code.google.com/p/libdivsufsort/). C'est une table qui consomme le minimum

d'espace et elle est très efficace en pratique même si dans le pire des cas sa complexité en temps n'est pas linéaire sur la longueur du texte à indexer (voir https://code.google.com/p/libdivsufsort/wiki/SACA_Benchmarks pour une comparaison à jour du temps et de l'espace consommés pour les principaux algorithmes des SA). Concernant le tri des valeurs de chaque intervalle de *GkSA* correspondant à un P_k-facteur, l'algorithme du tri rapide (ou quicksort) est utilisé. Une construction linéaire de la table *GkIFA* est possible en utilisant une table LCP (une table qui stocke la longueur du plus long préfixe commun entre deux suffixes consécutifs dans l'ordre lexicographique). Cependant, la construction d'une telle table n'est pas gratuite et nécessite $9mq$ octets en mémoire avec l'algorithme de Manzini [Manzini, 2004].

Nous vous proposons deux versions des *Gk arrays* : une qui indexe seulement les *reads* de longueur fixe, puis une autre qui s'occupe des *reads* avec des longueurs variables. Dans la suite des résultats, lorsque nous ne précisons rien, les *Gk arrays* se réfèrent à la première version. Pour la deuxième version, nous utilisons Sux (http://sux.dsi.unimi.it/), une implantation de vecteur de bits avec des opérations rank et select.

4.3 Description des méthodes alternatives aux Gk *arrays*

Les séquenceurs haut débit continuent de s'améliorer au cours du temps. Prendre en considération l'augmentation et la longueur des *reads* est un enjeu majeur dans le domaine de la bioinformatique. Les *Gk arrays* proposent une solution évolutive pour faire de l'indexation de *reads*. La plupart des structures existantes, compressées ou non, ne sont pas appropriées pour gérer des grandes collections de *reads*. Notre approche se concentre sur ce point et nous cherchons à optimiser la consommation mémoire, le temps de construction, et enfin le temps de réponse aux requêtes. Nous comparons les *Gk arrays* à deux autres approches : une table des suffixes généralisée (gSA) et une table de hachage. Parmi les structures d'indexation non compressées qui ont été généralisées pour indexer un ensemble de textes, la gSA est réputée pour être l'une des plus efficaces en mémoire, et sera préférée aux tables de hachage ou encore aux arbres des suffixes dans d'autres contextes [Kurtz *et al.*, 2004; Gusfield, 1997]. D'un autre point de vue, l'optimisation des moteurs de recherche sur internet a suscité le développement de tables de hachage performantes, comme *Google sparse hash* (http://code.google.com/p/google-sparsehash) ou le module d'extension de SGI (http://www.sgi.com/tech/stl). Comparer ce type de tables aux *Gk arrays* nous a semblé instructif et incontournable. Comme nous l'avons introduit dans la section 2.3.2 du chapitre 2, les structures d'indexations compressées permettent d'économiser de la mémoire, mais en contrepartie le temps de construction de l'index et de calcul des requêtes augmentent considérablement. C'est pourquoi nous avons décidé de les exclure de nos comparaisons. En outre, la conception de telles structures compressées qui seraient efficaces, à la fois en temps et en espace, demeure un véritable défi : la compression des *Gk arrays* pourrait être expérimentée.

4.3.1 Les tables des suffixes généralisées

Nous détaillons dans cette section une solution de table des suffixes généralisée (gSA) pour indexer une collection de *reads*, où ils ont tous la même longueur, nous l'appelons les *Gs arrays*. Cette table indexe le texte C_R représentant la concaténation de tous les *reads*. Pour que cette structure réponde au même besoin que les *Gk arrays*, en plus de construire la gSA de C_R, nous devons aussi construire la table inverse (ISA) et la table des plus longs préfixes communs (LCP) de C_R. La table gSA est construite avec le même algorithme que pour les *Gk arrays* (libdivsufsort). La table ISA est construite à partir de gSA en une seule lecture, c'est-à-dire en $O(mq)$. Puis la table LCP est aussi construite en temps linéaire en utilisant un algorithme performant [Kasai *et al.*, 2001]. Les tables sont créées dans cet ordre pour la création de la structure globale permettant de répondre aux requêtes des *Gk arrays*.

Dans les tables 4.3(a) et 4.3(b), nous comparons les complexités en temps et en espace des *Gs arrays* et des *Gk arrays*. Ces deux structures commencent par construire gSA(C_R) qui est la phase dominante de la complexité en temps avec une complexité de $O(mq)$: l'espace occupé pendant la construction de la table gSA seule est $4.02mq$, alors que le temps requis pour la construire est $4mq$ [Puglisi *et al.*, 2007]. Les trois dernières colonnes des tables 4.3(a) et 4.3(b) montrent l'évolution de la mémoire lors de l'accumulation des autres tables, qui sont nécessaires pour l'élaboration de la structure finale. Nous avons aussi enregistré la mémoire réelle utilisée lors de la construction des deux structures dans les figures 4.2(a) et 4.2(b). Les trois tables requises pour l'élaboration de *Gs arrays* sont directement accumulées en mémoire alors que pour les *Gk arrays* :

1. la table *GkSA* remplace gSA(C_R) en mémoire et utilise seulement $4\hat{m}q$,
2. la table *GkIFA* qui est l'inverse de *GkSA* (et non gSA(C_R)) prend le même espace de $4\hat{m}q$, et la table *GkCFA* occupe $4(\hat{r}+1)$ où $\hat{r}$ est le nombre de P_k-facteurs distincts,
3. pour finir, la table *GkCFPS* remplace directement la table *GkCFA* et n'utilise donc aucun espace supplémentaire.

Au total, les *Gs arrays* occupe $12mq$ octets en mémoire alors que les *Gk arrays* occupe $8\hat{m}q + 4(\hat{r}+1)$ octets (avec des entiers de 32-bits), où $\hat{m} := m-k+1$ est plus petit que m. Sur une expérience réelle, nous observons que les *Gk arrays* utilisent moins de mémoire en pratique que les *Gs arrays* (voir figures 4.2(a) et 4.2(b)) et ce, même en faisant varier la longueur des k-mers (voir figures 4.3(a), 4.2(a) et 4.2(b)). En fait, ce gain en mémoire dépend en grande partie des valeurs de k et de q : plus k est petit et plus un k-mer va apparaître plusieurs fois dans la collection des q-*reads*, ainsi $4(\hat{r}+1) \ll 4mq$. En conséquence, plus $\hat{r}$ est plus petit que mq, plus *GkCFPS* prendra moins d'espace que la table LCP. Par ailleurs, lorsque k est grand $4(m-k+1)q \ll 4mq$ et de ce fait *GkSA* et *GkIFA* occupent moins d'espace que gSA et ISA. Finalement, dans la plupart des cas, les *Gk arrays* permettent d'économiser au moins $12(k-1)q$ octets par rapport aux *Gs arrays*.

Rappelons que la recherche dichotomique est une procédure standard pour rechercher un mot dans un texte [Gusfield, 1997]. Alors, la localisation d'un k-mer dans une collection de *reads* peut

TABLE 4.3 : Comparaison des complexités entre la *Gs arrays* et les *Gk arrays*.

Gs arrays	Temps de construction	Mémoire $gSA(C_R)$	Mémoire cumulée gSA	ISA	LCP
	$O(mq)$	$4.02mq$	$4mq$	$8mq$	$12mq$

(a)

Gk arrays	Temps de construction	Mémoire $gSA(C_R)$	Mémoire cumulée *GkSA*	*GkIFA* & *GkCFA*	*GkCFPS*
	$O(mq)$	$4.02mq$	$4\hat{m}q$	$8\hat{m}q+4(\hat{r}+1)$	$8\hat{m}q+4(\hat{r}+1)$

(b)

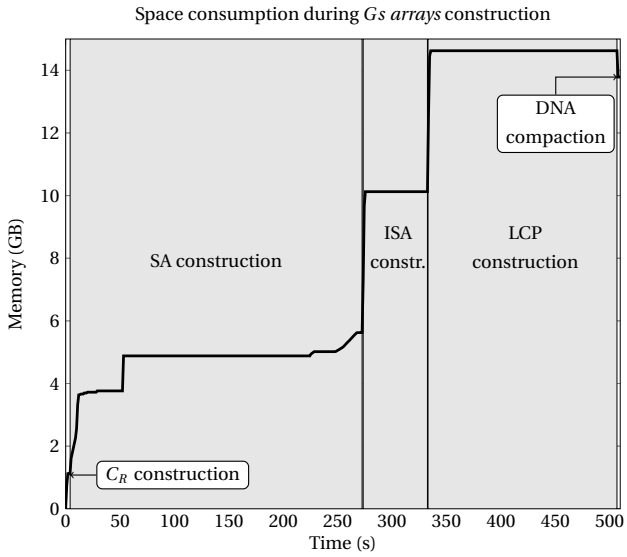
	Gk arrays	*Gs arrays*
Q1	$O(occ\_Reads(f))$	$O(q+occ\_C_R\ (f))$
Q2	$O(occ\_Reads(f))$	$O(q+occ\_C_R\ (f))$
Q3	$O(occ\_Reads(f))$	$O(occ\_C_R\ (f))$
Q4	$O(1)$	$O(occ\_C_R\ (f))$
Q5	$O(occ\_Reads(f))$	$O(q+occ\_C_R\ (f))$
Q6	$O(occ\_Reads(f))$	$O(q+occ\_C_R\ (f))$
Q7	$O(occ\_Reads(f))$	$O(q+occ\_C_R\ (f))$

(c)

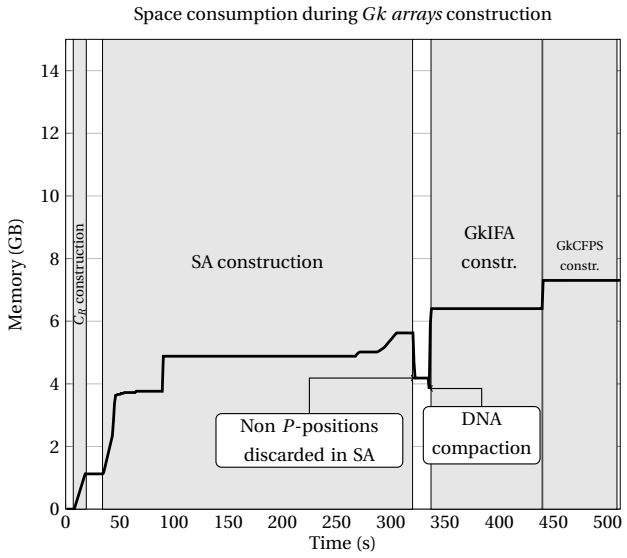
Une complexité est une expression qui évalue le temps de calcul et l'espace mémoire requis en fonction d'une donnée d'entrée. Les complexités en temps et en espace pour indexer q reads de longueur m qui ont exactement $\hat{r}$ k-mers distincts sont données pour la *Gs arrays* d'un coté 4.3(a), et les *Gk arrays* de l'autre 4.3(b). Nous détaillons la mémoire utilisée au fur et à mesure de la construction des deux structures, en commençant par la gSA(C_R) et après les principales étapes de construction des algorithmes (la gSA, la ISA, et la LCP sont construites dans 4.3(a) et la *GkSA*, *GkIFA*, et *GkCFPS* sont construites dans 4.3(b). Pour 4.3 nous donnons les complexités en temps pour les requêtes de Q1 à Q7 avec un k-mer dénoté par f. La procédure pour gSA dépend de *occ_C_R (f)* : le nombre d'occurrences de f dans le texte C_R. Alors que celle pour les *Gk arrays* dépend de *occ_Reads(f)* : le nombre d'occurrences de f dans tous les reads (sans les chevauchements), et nous savons que $occ\_Reads(f) \leq occ\_C_R\ (f)$.

être faite à l'aide d'une recherche dichotomique, en $O(k+\log qm)$ dans le pire des cas pour *Gs arrays* (en utilisant les tables gSA et LCP), puis en $O(k\times\log q\hat{m})$ dans le pire des cas pour *Gk arrays* (en utilisant *GkSA*). Cependant, Manber et Myers [Manber et Myers, 1990] mentionnent dans leur ouvrage qu'une petite amélioration des algorithmes classiques (mémoriser la plus petite longueur parmi tous les plus longs préfixes communs pour chaque sous ensemble, à gauche et à droite, de la recherche par dichotomie) permet, en pratique, de faire la recherche du k-mer en $O(k+\log q\hat{m})$ dans le pire des cas (voir aussi [Gusfield, 1997] Section 7.14.3 page 152).
Pour répondre aux différentes requêtes, il est nécessaire de connaître une occurrence d'un k-mer (ou sa position) dans C_R. Du coup, si cette information n'est pas connue au préalable, il faut recher-

FIGURE 4.2 : Évolution de la mémoire pendant la construction de la *Gs arrays* et des *Gk arrays*.



(a)



(b)

Nous pouvons voir sur ce graphique la mémoire réelle utilisée pendant la construction de *Gs arrays* en 4.2(a) et des *Gk arrays* en 4.2(b) lorsque nous indexons 15 millions de reads de longueur 75 *pb* avec des k-mers de longueur $k = 25$.

cher le k-mer dans C_R. Or, d'après le paragraphe précédent, une telle recherche est similaire dans les deux structures de données *Gs arrays* et *Gk arrays*.

Algorithme 6 : Q1 ($Ind_k(f)$) for *Gs arrays*.

```
   Data : f ∈ Σ^k, j ∈ P_pos such that C_R[j .. j+k−1] = f
   Result : Ind_k(f)
1  begin
2      Ind_k ⟵ empty set;
3      Initialize the whole bit vector, D, to zero;
4      i ⟵ ISA[j] ; // starting position of f occurrences in SA
5      repeat
6          if (SA[i] mod m) ≤ m̂ then
               // the occurrence position does not overlap two reads
7              readIndex ⟵ ⌊SA[i]/m⌋;
8              if D[readIndex] ≠ 1 then
                   // we have not found an occurrence in this read yet
9                  Add readIndex to Ind_k;
10                 D[readIndex] ⟵ 1;
11         i ⟵ i + 1;
12     until (i ≥ qm) or (LCP[SA[i], SA[i+1]] < k) ;
13     return (Ind_k);
14 end
```

Cependant, bien que nous considérons la même entrée (c'est-à-dire une position j d'une occurrence d'un k-mer dans un *read*), la procédure pour répondre aux requêtes diffère entre les deux structures *Gs arrays* et *Gk arrays*,. Pour les *Gs arrays*, puisque les positions des k-mers de C_R sont stockées dans gSA, nous devons filtrer toutes celles qui correspondent à un chevauchement entre deux *reads* pour ne conserver que les P-positions (voir ligne 6 de l'algorithme 6 où l'on vérifie bien que i corresponde à un k-mer sans chevauchement). Alors que pour les *Gk arrays*, les k-mers chevauchants ne sont pas enregistrés, nous obtenons donc directement une position valide. De plus, pour répondre aux requêtes Q1 et Q2, les *Gs arrays* telles qu'elles se présentent ne le permet pas : il faut identifier les k-mers qui sont dupliqués dans un seul *read* afin de ne les compter qu'une seule fois. Pour ce faire, nous devons utiliser un vecteur binaire B de q bits (un bit par *read*). Ainsi, lors du premier passage du k-mer dans un *read*, le bit du vecteur correspondant à la position du *read* passe à 1, puis au deuxième passage le bit sera déjà à 1 ce qui nous permet de savoir que le *read* a déjà été comptabilisé (ligne 8 et 10 de l'algorithme 6).

Supposons que nous voulons interroger un k-mer f représenté par j : la position de l'une de ces occurrences dans C_R. Nous appelons *occ_C_R (f)* le nombre d'occurrences de f dans C_R, y compris celles qui chevauchent deux *reads* (c'est-à-dire les non-P-positions), et appelons *occ_Reads(f)* le nombre de ses occurrences qui sont totalement incluses dans les *reads* (c'est-à-dire les P-positions).

Pour Q1/Q2, Q5-Q7, nous obtenons avec les *Gs arrays* une complexité de $O(q + occ\_C_R\ (f))$ parce que d'un côté il faut initialiser le vecteur B de taille q et de l'autre scanner toutes les occurrences $occ\_C_R\ (f)$. Alors qu'avec les *Gk arrays* la complexité dépend linéairement de *occ_Reads(f)* et nous savons que $occ\_Reads(f) \leq occ\_C_R\ (f)$. Pour Q3/Q4, il n'y a pas besoin du vecteur B pour les *Gs arrays*, alors la complexité est $O(occ\_C_R\ (f))$ parce qu'il scanne les positions dans gSA en utilisant les tables *ISA* et LCP. Néanmoins, les *Gk arrays* offrent une complexité de $O(occ\_Reads(f))$ pour Q3 et $O(1)$ pour Q4. Nous résumons toutes les complexités en temps des requêtes dans la TABLE 4.3.

4.3.2 Les tables de hachage

Une solution alternative aux *Gs arrays* et aux *Gk arrays* est d'indexer tous les k-mers dans une table de hachage et de leur associer la liste des (*reads*, positions) dans lesquels ils apparaissent. Cette liste contient donc des paires d'entiers : l'index (ou numéro) du *read* dans la collection, et la position du début du k-mer dans ce même *read*. L'index du *read* peut être enregistré dans un entier de 32 bits alors qu'un entier de 16 bits suffit pour enregistrer la position du k-mer dans le *read*. Dans une telle structure, il n'y a pas de texte C_R à stocker. Le nombre d'entrées de table de hachage correspond au nombre de k-mers distincts dans la collection de *reads*, c'est-à-dire notre paramètre $\hat{r}$. Généralement, $\hat{r}$ est petit comparé aux 4^k k-mers qui peuvent être théoriquement distincts, pour des valeurs de k dans l'intervalle $[15,60[$. Par conséquent, une table de hachage pour indexer des k-mers n'est pas dense ($\hat{r} \ll 4^k$).

Nous avons essayé plusieurs implantations parmi l'état de l'art des tables de hachage : *Google sparse hash*, *Google dense hash* et l'extension SGI de la librairie standard de C++ (appelée *SGI hash map*). Des expériences préliminaires ont montré que la *Google sparse hash* requiert plus de temps pour se construire que la *SGI hash map* mais en contrepartie elle occupe moins d'espace en mémoire. Pour indexer 20 millions de *reads* de longueur 75 *pb*, *Google sparse hash* n'occupe qu'un tiers de la mémoire que *SGI hash map* utilise pour indexer cette quantité mais elle a besoin de quatre fois plus de temps pour se construire. Quant à la *Google dense hash*, elle occupe deux fois plus de mémoire que la *SGI hash map* pour un temps de construction similaire. Finalement, la *SGI hash map* semble être le meilleur compromis en terme de mémoire occupée et de temps de construction comparée aux propositions de Google. C'est pourquoi, nous avons choisi la *SGI hash* pour se comparer aux *Gk arrays*.

4.4 Comparaisons des structures de données

4.4.1 La description des expériences

Nous avons expérimenté les différentes structures d'indexation sur trois ensembles de données :

1. Une collection de 40 millions de *reads* de longueur 75 *pb* de type RNA-Seq de la technologie Illumina® provenant d'une lignée K562 du génome humain. Les données sont disponibles sur RGASP (numéro d'accession *GM12878* sur www.sanger.ac.uk/PostGenomics/encode/RGASP.html avec la permission de B. Wold).
2. Une collection de 2.8 millions de *reads* génomique de la technologie Roche 454® avec des longueurs comprises entre [40,3000] *pb* et une longueur moyenne de 523 *pb*. Ils ont été séquencés par la plate-forme Roche 454® GS FLX avec la chimie Titanium pour le projet du génome du Khoisan [Schuster *et al.*, 2010].
3. Comme des *reads* de longueur fixe et aussi longue ne sont pas encore disponibles, nous avons découpé/sélectionné des *reads* de 150 *pb* du Khoisan de la collection précédente pour ainsi obtenir une collection de 25 millions de *reads* de longueur 150 *pb* (une telle collection sera bientôt générée par les plates-formes de séquençage à haut débit).

Si nous résumons, pour le premier et troisième ensembles de données, les *reads* sont de longueur fixe, alors que pour le second, les *reads* sont de longueur variable. Toutes les expériences ci-dessous ont été faites sur un Intel Xeon 2.27 GHz équipé avec 48 GO de RAM, et les programmes sont lancés sur un Linux 2.6.18 et compilés avec la version 3.4.6 de `gcc` avec les options `-O2 -funroll-loops`.

4.4.2 Les comparaisons expérimentales

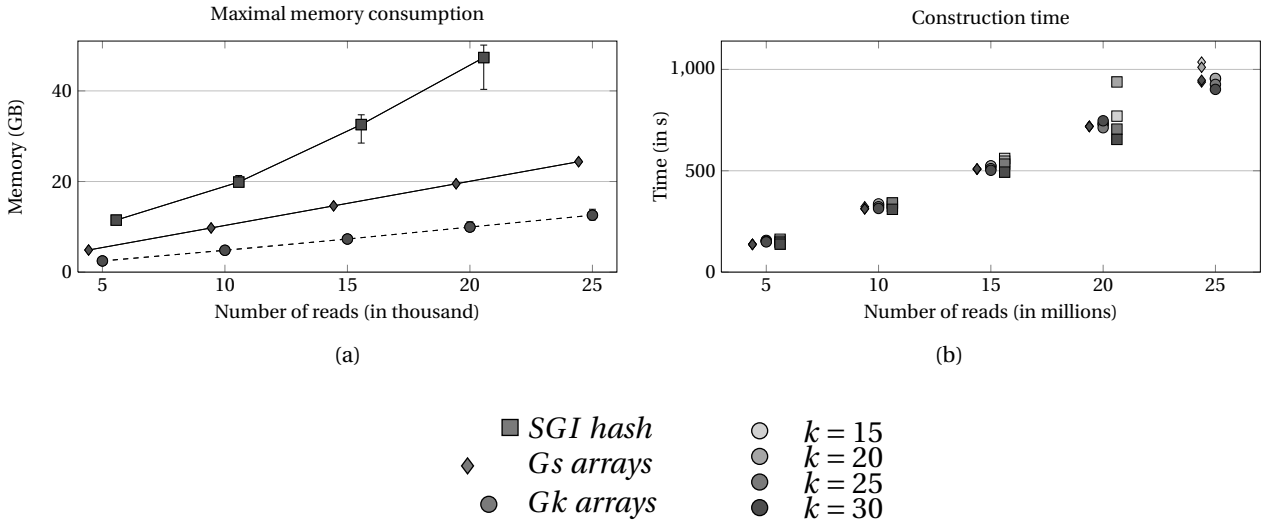
L'utilisation de structures d'indexation pour les *reads* soulève trois questions : quelles sont les ressources matérielles nécessaires pour les gérer ? Est-ce qu'elles sont évolutives ? À quelle vitesse peuvent-elles répondre à un grand nombre de requêtes ? Ostensiblement, les ressources requises vont dépendre du nombre de *reads* (paramètre q) et de la longueur des k-mers (paramètre k). Nous comparons les *Gk arrays* aux deux autres structures détaillées dans la section 4.3 : une table de hachage (*SGI hash*), une table des suffixes généralisée (*Gs arrays*), et notre structure d'indexation (*Gk arrays*).

4.4.2.1 Évolutivité

Nous commençons par mesurer le temps de construction et la mémoire utilisée par les trois structures de données en faisant varier le nombre de *reads* et la longueur des k-mers. La figure 4.3(a) illustre la mémoire maximale utilisée sur l'ensemble de données K562. À cette échelle, la valeur de k n'a d'influence en mémoire que sur les tables de hachage (pas de différences visibles sur le graphe entre les valeurs de k pour les *Gs arrays* ou les *Gk arrays*). En outre, les structures sont classées dans cet ordre croissant en terme de mémoire utilisée (de la moins à la plus gourmande) : les *Gk arrays*, les *Gs arrays* et la *SGI hash*. Cet ordre reste le même quel que soit le nombre de *reads* à indexer. Par exemple, pour $k = 20$, les *Gk arrays* utilisent 10 GO de mémoire, les *Gs arrays* en occupent 20 GO, puis la *SGI hash* en consomme 44 GO. D'ailleurs, les écarts entre les différentes structures ont tendance à augmenter avec le nombre de *reads* et pour n'importe quelle valeur de q, la *SGI hash* est

au moins deux fois plus gourmande que les *Gs arrays* qui ont une consommation en mémoire 70% supérieure à celle des *Gk arrays*. Avec 25 millions de *reads* la *SGI hash* sature, puis avec 30 millions de *reads* c'est les *Gs arrays* qui saturent alors que les *Gk arrays* constituent la seule solution capable d'indexer la collection entière des 40 millions de *reads* sur le même ordinateur. Notons que dans les deux cas, les implantations de 64 bits des *Gs arrays* et des *Gk arrays* ont été utilisées pour pouvoir indexer les 40 millions de *reads*. Pour la collection entière, les *Gk arrays* ont besoin d'au minimum 36 GO de mémoire ($k = 30$) et au maximum 43 GO ($k = 15$).

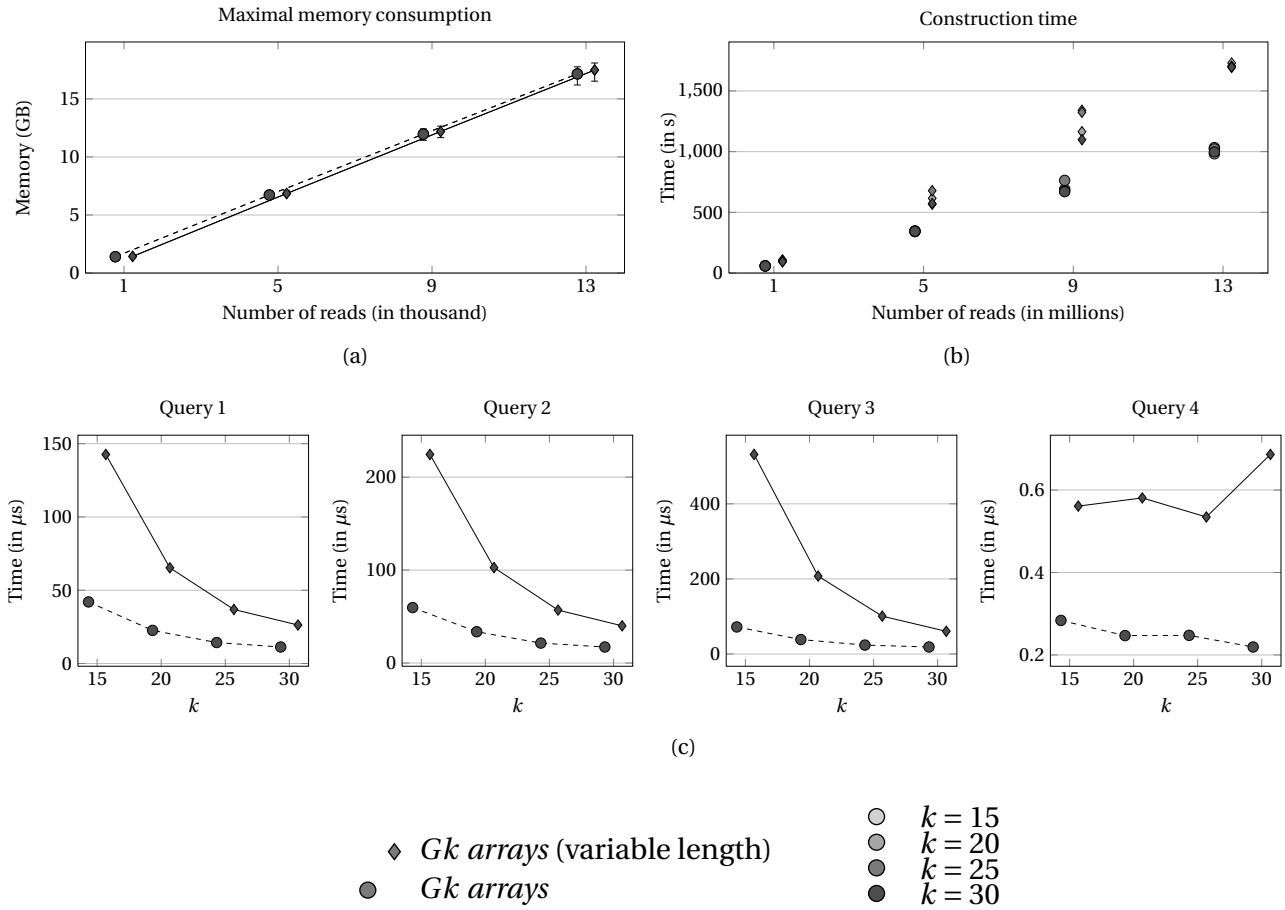
FIGURE 4.3 : Mémoire et temps de construction pour la *Gs arrays*, les *Gk arrays* et la *SGI hash* pour des reads de type RNA-Seq.



Les données de RNA-Seq K562 sont utilisées dans cette expérience, en faisant varier le nombre de reads de 5 millions à 25 millions. La longueur des k-mers varie quant à elle de 15 à 30. Pour les *Gs arrays* (gSA), les points ont été décalés sur la gauche et pour la *SGI hash* (HT), les points ont été décalés sur la droite pour faciliter la lecture. (a) La mémoire maximale utilisée pour la construction de la structure et les requêtes. Les barres d'erreur représentent l'espace consommée en fonction des valeurs de k. (b) Le temps de construction mesuré pour les trois structures correspond au pic de mémoire sur le graphique. Les différents niveaux de gris illustrent les différentes valeurs de k.

Pour les trois structures, le temps de construction augmente de façon linéaire avec le nombre de *reads* comme nous pouvons le voir dans la figure 4.3(b). Les temps sont similaires pour les *Gs arrays* et les *Gk arrays*, quelles que soient les valeurs de k, par exemple elles mettent < 1000 s. à se construire pour 25 millions de *reads*. En revanche, l'influence de k est nettement plus visible pour 20 millions de *reads* pour la *SGI hash* : son temps de construction diminue avec k parce que le paramètre $\hat{r}$ diminue aussi (pour un nombre de *reads* donné). Néanmoins, autant qu'elles puissent tenir en mémoire, les trois structures nous offrent un temps de construction relativement rapide et quasiment équivalent.

FIGURE 4.4 : Comparaison des deux versions des *Gk arrays*: longueurs fixes et longueurs variables.



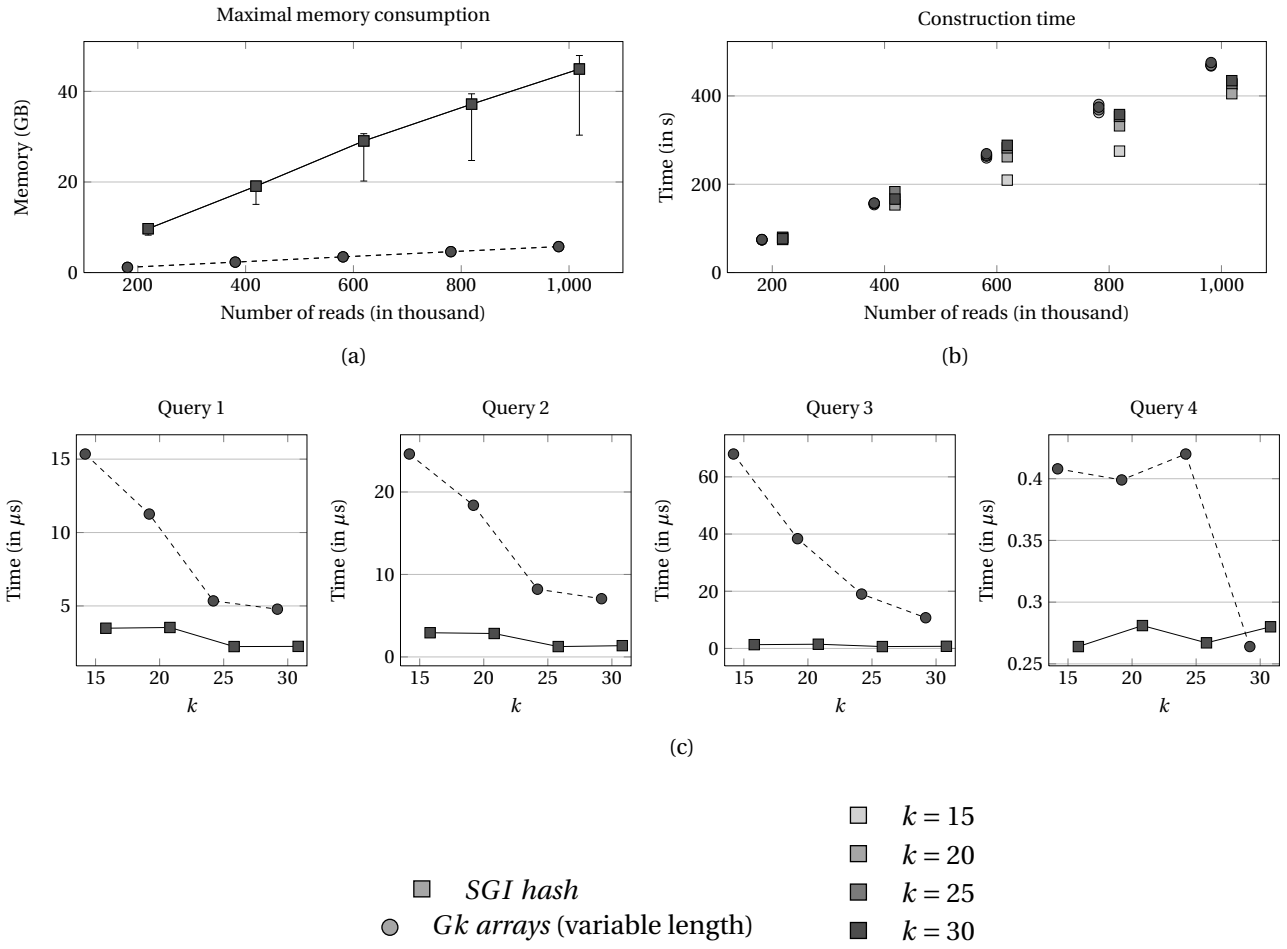
Expériences sur une portion des données du Khoisan Roche 454® (reads de longueurs 150 *pb*) avec les *Gk arrays* (fixed-length reads et variable-length reads). (a) La mémoire maximale utilisée pour chaque structure en faisant varier le nombre de reads à indexer et la valeur de k (les barres d'erreur représentent l'espace consommée en fonction des valeurs de k). (b) Le temps de construction mesuré pour les deux structures correspond au pic de mémoire sur le graphique (les différents niveaux de gris illustrent les différentes valeurs de k). (c) Le temps moyen de calcul des requêtes pour une collection de 13,000,000 reads en fonction des valeurs de k.

Maintenant, nous voulons examiner le comportement des *Gk arrays* sur des *reads* plus longs, 150 *pb*, d'un côté lorsque l'option pour les *reads* à longueur variable est activée, et de l'autre côté lorsqu'elle ne l'est pas. La figure 4.4(a) illustre la mémoire consommée et la figure 4.4(b) enregistre le temps de construction pour les deux options.

D'après la figure 4.4(a) l'ajout supplémentaire d'un vecteur de bits explique la petite différence de mémoire entre les deux méthodes, ce qui nous permet de constater que le vecteur n'est pas consommateur de mémoire. Par exemple, pour 13 millions de *reads* la différence est, au plus, de

300 MO. Dans la figure 4.4(b), nous nous apercevons que l'implantation pour les *reads* à longueurs variables devient plus lente lorsque le nombre de *reads* augmente, comparée à celle pour les longueurs fixes. Bien que la complexité théorique pour les opérations rank et select soit en temps constant, cette dernière constatation montre que la complexité en pratique dépend de la longueur du vecteur de bits. Toutefois, le temps de construction est malgré tout raisonnable pour la version des longueurs variables.

FIGURE 4.5 : Comparaison de la *SGI hash* avec les *Gk arrays* à longueurs variables.



Expériences sur les données du Khoisan Roche 454® (reads à longueurs variables) avec les *SGI hash* et les *Gk arrays* (variable-length reads). (a) La mémoire maximale utilisée pour chaque structure en faisant varier le nombre de reads à indexer et la valeur de k (les barres d'erreur représentent l'espace consommée en fonction des valeurs de k). (b) Le temps de construction mesuré pour les deux structures correspond au pic de mémoire sur le graphique (les différents niveaux de gris illustrent les différentes valeurs de k). (c) Le temps moyen de calcul des requêtes pour une collection de 600,000 reads en fonction des valeurs de k.

Les figures 4.5(a) et 4.5(b) exposent l'espace et le temps mesurés pour la *SGI hash* et les *Gk arrays* (avec l'option activée pour les longueurs variables) sur les données Roche 454® de Khoisan. Les *Gs arrays* n'ont pas de version pour les longueurs variables ; notons que le coût relatif supplémentaire serait similaire à celui observé sur les *Gk arrays* entre longueurs fixes et variables. Nous observons dans la figure 4.5(b) que les *Gk arrays* ont besoin de 470 s. pour se construire contre 428 s. pour la *SGI hash* pour un million de *reads* à indexer. En outre, pour ce temps similaire, la *SGI hash* consomme 8 fois plus de mémoire (46 GO contre 5.5 pour les *Gk arrays*). Cette différence a tendance à augmenter considérablement avec le nombre de *reads*. Cependant, au dessus de un million, la *SGI hash* dépasse la capacité mémoire de l'ordinateur (48 GO), alors que pour les *Gk arrays*, nous indexons la collection complète de 2.8 millions de *reads* pour une consommation < 15.6 GO sur la même machine.

La table de hachage semble nettement plus consommatrice sur les données Khoisan que sur les données k562. La nature des données peut expliquer cette différence : les séquenceurs Roche 454® offrent une moins bonne profondeur de séquençage que ceux d'Illumina®, par conséquent le nombre de k-mers distincts (notre paramètre $\hat{r}$) dans les *reads* est significativement plus grand avec les données de Khoisan.

4.4.2.2 La réponse aux requêtes

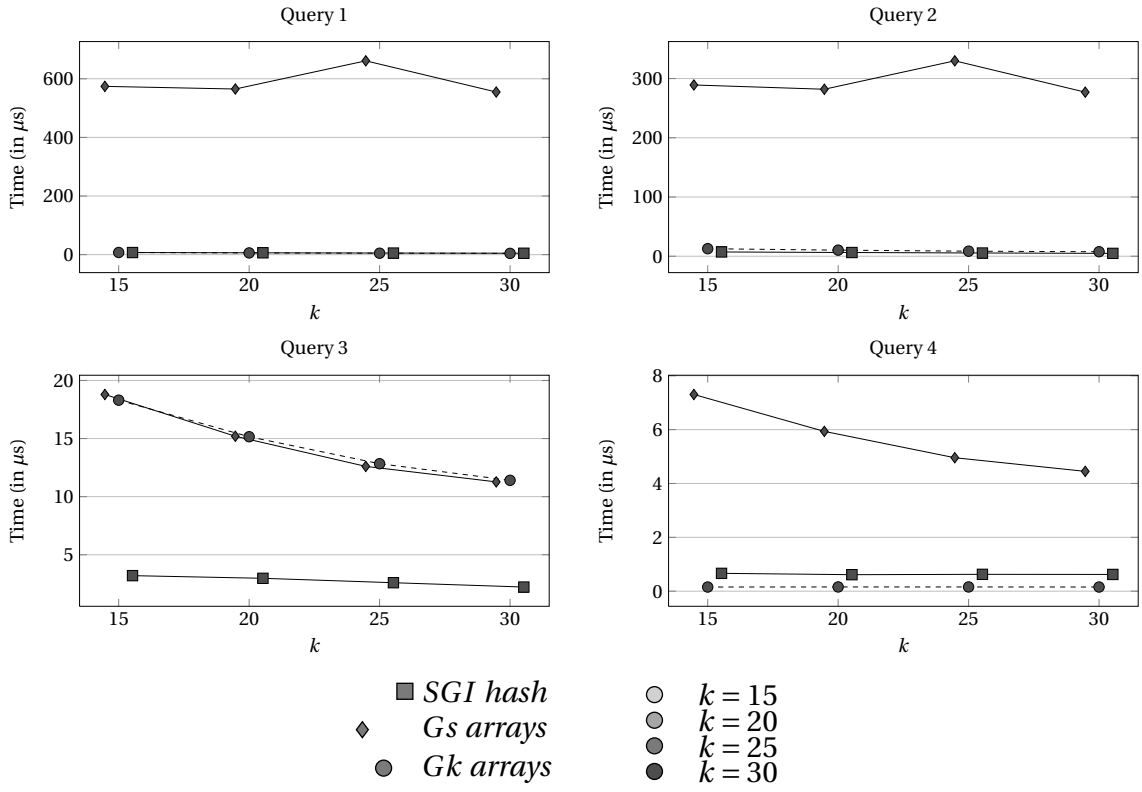
Nous mesurons le temps moyen requis pour un ensemble de 100,000 requêtes aléatoires de types Q1, Q2, Q3 et Q4 (les requêtes Q1-Q7 n'étant que des légères variantes de Q1-Q3, nous ne les traitons pas).

La figure 4.4.2.2 montre pour chaque structure comment le temps moyen varie avec le nombre de *reads* (q) et la longueur des k-mers (k) sur la collection K562. Nous constatons que l'influence de q est similaire pour toutes les structures, et qui, de plus, est petite comparée aux différences observées. Plus généralement, les *Gs arrays* mettent toujours plus de temps que la *SGI hash* quel que soit le type de la requête, puis ils sont aussi plus longs que les *Gk arrays* pour Q1, Q2 et Q4, et avec un temps similaire pour Q3. L'ordre entre les *Gk arrays* et la *SGI hash* dépend du type de la requête : ils sont aussi rapides pour Q1, la *SGI hash* est quelque peu plus rapide pour Q2, clairement meilleure pour Q3, mais elle est à l'inverse nettement moins rapide que les *Gk arrays* pour Q4. Quoiqu'il en soit, pour ces deux structures, le temps moyen de réponse est de l'ordre de 10 microsecondes pour Q1,Q2 et Q3 et autour de 0.1 microseconde pour Q4 avec les *Gk arrays*, ce qui reste très raisonnable en pratique.

Ensuite, pour notre comparaison des *Gk arrays* entre la version à longueurs fixes et celle à longueurs variables, nous constatons que la deuxième répond plus lentement aux requêtes (jusqu'à 7 fois) quand k est petit, c'est-à-dire quand le nombre d'occurrences des k-mers est grand. Avec une plus grande valeur de k, le temps de requête diminue jusqu'à ne devenir que 3 voire 2 fois plus lente que pour la version standard des *Gk arrays*.

Avec des *reads* de longueurs variables (FIGURE 4.5(c)) le temps de requête reste raisonnable en

FIGURE 4.6 : Comparaison des temps de requêtes entre les *Gk arrays*, les *Gs arrays* et la *SGI hash* pour des reads de type RNA-Seq.



Calcul des temps de requêtes de type Q1/Q2/Q3/Q4 sur les données de K562 Illumina® (reads de longueurs 75 *pb*). Les points représentent le temps moyen μs sur les mêmes 100,000 requêtes lancées sur une question donnée (Q1, Q2, Q3, ou Q4). Dans tous les cas, le temps de réponse diminue lorsque la valeur k augmente, c'est-à-dire qu'il y a moins d'occurrences dans les reads d'un 30-mer que d'un 15-mer. Les *Gk arrays* sont toujours plus rapides que les *Gs arrays*; d'ailleurs ils calculent Q4 en temps constant.

pratique, mais la *SGI hash* répond plus rapidement que les *Gk arrays*, avec un facteur qui varie entre 1 et 32 fois en fonction de la requête.

En résumé, dans toutes les conditions, les *Gk arrays* ont un temps de construction qui est équivalent aux *Gs arrays* mais aussi à la *SGI hash*. De plus, toujours en se comparant aux autres structures, les *Gk arrays* offrent un temps de réponse raisonnable aux requêtes quelles que soient les circonstances. En revanche, ils surclassent leurs deux concurrents en terme de mémoire consommée, cela constitue un atout réel pour traiter les millions de données du haut débit.

4.5 Conclusion et discussions

Les SHD sont devenus incontournables pour répondre aux problèmes génomiques et transcriptomiques actuels. Cependant, de part leur quantité et leur complexité, l'exploitation des *reads* reste encore un défi pour les biologistes et les bioinformaticiens. Une telle investigation devient abordable à partir du moment où les données sont déjà bien organisées, à l'aide d'une structure d'indexation par exemple. Cette dernière, appelée aussi index est une structure de données qui, un peu comme un annuaire, permet de trouver facilement les informations qui nous intéressent. Dans le contexte des *reads*, indexer la position des k-mers (pour une valeur de k) de tous les *reads* d'une manière structurée est une façon de minimiser l'usage de la mémoire. Avec une telle structure, pour retrouver tous les *reads* qui sont liés à un k-mer donné, il suffit de chercher le k-mer dans l'index et de lister tous les *reads* qui lui sont associés. Nous n'avons pas besoin de parcourir tous les *reads*, puis de vérifier qu'ils partagent le k-mer recherché, ce qu'on ferait si les données n'étaient pas organisées. En d'autres termes, en indexant les *reads*, nous factorisons le résultat lors d'une quelconque recherche (par requête), ce qui nous permet de gagner un temps précieux, à condition que l'index construit puisse tenir en mémoire.

Notre principale contribution est de créer une telle structure d'indexation pour une grande collection de *reads* : les *Gk arrays*. La structure se construit rapidement et consomme moins de mémoire que les solutions alternatives : 40 millions de *reads* indexés avec les *Gk arrays* contre 20 millions avec des tables de hachages pour la même quantité de mémoire disponible (48 GO). La mémoire est la limite pour ce genre d'applications en pratique.

4.5.1 Les *Gk arrays*, meilleurs que les structures actuelles

Tout en étant aussi efficace, au point de vue du temps de calcul, que la *SGI hash*, seuls les *Gk arrays* sont capables d'indexer une collection complète avec un nombre gigantesque de *reads* (comme l'ensemble de données K562 décrit dans la section 4.4.1) avec les machines de calcul actuelles. De plus, notre structure reste rapide, quelle que soit la taille du k-mer utilisé (valeur du paramètre k). Nous avons aussi montré que les *Gk arrays* sont plus rapides et moins consommateurs que les *Gs arrays*, l'autre structure alternative.

Pour la technologie Roche 454® qui produit des *reads* à longueurs variables, les *Gk arrays* semblent aussi mieux s'adapter. Ils indexent une collection complète de *reads* de type 454 en utilisant moins de 16 GO en mémoire alors que la *SGI hash* ne peut indexer qu'une sous partie de 1 million de *reads* pour les 48 GO disponibles sur la machine.

4.5.2 Les *Gk arrays* : une structure versatile

En plus d'être adaptés pour plusieurs types de technologie, les *Gk arrays* permettent de répondre efficacement à plusieurs types de requêtes. Dans un contexte d'assemblage, la structure

peut lister l'ensemble des *reads* qui partagent un même k-mer, ainsi que la position de l'occurrence dans chaque *reads*. Dans un contexte de quantification, la structure peut donner quasi instantanément le nombre de *reads* qui partagent un k-mer, ce qui est important lorsqu'on veut mesurer le taux d'expression d'un transcrit dans une cellule, par exemple. Les *Gk arrays* peuvent être utilisés pour des *reads* génomiques (assemblage, recherche de marqueurs génomiques) mais aussi pour des techniques transcriptomiques de type RNA-Seq. Notre structure est donc versatile et permet de gérer des grandes quantités de *reads*. Ces deux points sont importants car ils nous ouvrent la possibilité d'envisager une multitude d'applications biologiques dont quelques unes sont mentionnées en introduction de ce chapitre.

Nous avons développé les *Gk arrays* dans une librairie C++ qui contient très peu de paramètres (k la longueur des k-mers, q le nombre de *reads*). Cette librairie ne constitue pas, en effet, un logiciel à part entière, nous l'avons plutôt conçue comme une structure sous-jacente dans l'optique de l'intégrer facilement à d'autres logiciels. D'ailleurs dans le chapitre 5, nous utilisons les *Gk arrays* dans le logiciel CRAC par le biais de cette librairie.

4.5.3 Les limites et les éventuelles perspectives

Les *Gk arrays* constituent une solution appropriée pour faire de l'indexation de *reads* mais nous pouvons néanmoins envisager plusieurs perspectives. L'indexation de k-mers approchés ou de graines espacées permettrait d'autoriser encore plus de types de requêtes, mais il faut mener une étude complexe pour que cette amélioration ne coûte pas trop de temps et d'espace supplémentaires. L'élaboration d'un algorithme dynamique pour la construction des *Gk arrays* pourrait aussi augmenter le panel d'applications. Cependant, notre plus grande perspective est de proposer une version compressée des *Gk arrays* en ne conservant, par exemple, qu'un échantillon des positions et en recalculant les autres à la volée, comme cela a déjà pu être fait avec la transformée de Burrows Wheeler [Ferragina et Manzini, 2000]. Cette amélioration permettrait de gérer les futures collections de *reads* (encore plus grandes) mais surtout de permettre à tout utilisateur d'utiliser les *Gk arrays* sur son ordinateur personnel.

Ce travail a fait l'objet d'une publication dans BMC Bioinformatics en 2011 [Philippe *et al.*, 2011].

CHAPITRE

5 Algorithme spécialisé dans le traitement du RNA-Seq : CRAC

Dans ce chapitre, nous proposons un nouvel algorithme d'analyse de reads*, nommé CRAC. Ce dernier est spécialisé dans le traitement des données transcriptomiques de type RNA-Seq. Il est capable de détecter et de classer le maximum des informations présentes dans les* reads *telles que des jonctions d'épissage, des erreurs de séquences, des SNV, des indels ou encore des chimères. Il a l'avantage d'intégrer son propre algorithme de positionnement lors de l'analyse plutôt que de combiner une phase de* mapping *avec un outil et une phase d'analyse avec un autre outil. Notre logiciel peut être aussi bien utilisé comme un simple outil de* mapping *ou plus précisément dans la recherche d'un phénomène biologique précis. Nous verrons que CRAC est un outil très spécifique qui est notamment capable de détecter le point de cassure des chimères dans une expérience de* single-reads.

Sommaire

L'analyse des *reads* est le procédé qui consiste à répertorier toutes les informations contenues dans les *reads* par rapport à un génome de référence comme les mutations (SNV, indels, etc) ou les jonctions d'épissage (épissages classiques, variants alternatifs, chimères, etc). Pour analyser une collection de *reads* sur un génome de référence, la plupart des analyses commence par une première étape de *mapping* (voir section 2.4 du chapitre 2). Actuellement, le RNA-Seq pose un grand problème à ces outils car les meilleurs ne vont tolérer que des indels plus ou moins longs,

mais jamais de la taille d'un intron (≃ 5 kpb pour le génome humain en moyenne [Sakharkar *et al.*, 2004]). De ce fait, beaucoup de *reads* d'une expérience RNA-Seq ne seront pas localisés, comme ceux qui chevauchent une jonction d'épissage (classique ou chimérique) ou encore les reads qui contiennent un morceau de séquence artefactuelle. Or, la capacité à déterminer des causes biologiques dépend directement de la puissance du *mapping* dans ce type d'approche. Pour la détection des SNV par exemple, les *reads* qui ne sont pas localisés ne sont pas pris en compte [Li *et al.*, 2009a] (ce qui sera certainement le cas des *reads* qui contiennent à la fois une jonction d'épissage et un SNV).

Le fait de séparer la phase de *mapping* de la phase d'analyse rend le processus plus long et plus complexe. Plus long, parce que les *reads* sont traités une première fois avec les outils de *mapping* et une seconde fois avec les outils d'analyse. Plus complexe, parce que les outils de *mapping* possèdent des paramètres intrinsèques qui dépendent de la nature des *reads* (courts ou longs, avec ou sans indels, etc) et peuvent engendrer des artefacts que les outils d'analyse ne prennent pas en considération. D'ailleurs, ils possèdent aussi leurs propres paramètres qui ne sont pas forcément compatibles avec tous les outils de *mapping*.

Cette manière de séparer l'analyse du *mapping* semble donc ne pas être la meilleure solution. Par exemple, prenons les outils de *splicing* (voir section 2.6.1 du chapitre 2) qui procèdent en deux étapes : i/ localisation avec un outil de *mapping* ; ii/ détection des jonctions à partir des *reads* qui ne sont pas localisés [Langmead *et al.*, 2009; Wang *et al.*, 2010]. D'un côté, Bowtie va localiser des *reads* contenant des jonctions alors qu'il ne le devrait pas, et de l'autre il ne va pas localiser des *reads* (s'ils contiennent un indel par exemple) alors qu'il le devrait. Dans les deux cas, cela influe dans la détection des jonctions [Garber *et al.*, 2011]. Un autre exemple, les outils pour détecter les SNV ou d'autres mutations courtes de type biologique, procèdent aussi en plusieurs étapes : i/ localisation avec un outil de *mapping* ; ii/ identification des substitutions et des indels ; iii/ analyses des mutations identifiées en ii/. En conséquence, lorsque l'identification est erronée, l'analyse des mutations devient confuse [Li, 2011].

En bref, les méthodes actuelles d'analyse de *reads* requièrent souvent plusieurs étapes, en commençant par un processus de *mapping* qui autorise souvent quelques substitutions et indels dans l'alignement entre *reads* et génome. Manifestement, ces paramètres sont adaptés aux courtes séquences génomiques mais beaucoup moins aux séquences transcriptomiques (courtes ou longues). L'enchaînement des étapes de manière indépendante engendre une perte du contrôle sur la probabilité de trouver ou non une prédiction fiable [Trapnell et Salzberg, 2009]. En conclusion, une intégration de la phase de *mapping* aux phases d'analyses, devrait augmenter les chances d'être plus spécifique.

Un autre aspect très ouvert dans l'étude du RNA-Seq est la détection des chimères (voir section 2.6.2 du chapitre 2). Algorithmiquement, les méthodes pour identifier les jonctions chimé-

riques sont complexes, tant les formes chimériques sont diverses et les connaissances sur le sujet, faibles. En conséquence, malgré quelques approches sur des données de RNA-Seq *pairé* (*paired reads*) [Li *et al.*, 2011; Sboner *et al.*, 2010] ou sur du *single reads*, en croisant des données génomiques et transcriptomiques de la même lignée cellulaire [McPherson *et al.*, 2011], très peu d'outils sont capables d'identifier précisément la globalité des chimères d'une expérience et encore moins leur point de cassure. De plus, le manque de connaissances biologiques sur le sujet fait que très peu de chimères sont à ce jour caractérisées.

5.1 Description de notre approche

Les SHD produisent des millions, voire des milliards de *reads*. La comparaison de tels ensembles de données avec un génome complet exige l'élaboration de structures spécifiques (voir chapitre 4). D'autre part, les séquences devenant de plus en plus longues ($\simeq$ 100 bp actuellement) et contenant toujours des erreurs [Dohm *et al.*, 2008b], il est avantageux de considérer tous les mots de longueur k d'une séquence (ou k-mer, $k \simeq 22pb$ pour des données humaines [Philippe *et al.*, 2009]) puis de les positionner un par un sur le génome (voir chapitre 3).

Notre outil, nommé CRAC, se base sur une double indexation qui permet de calculer en temps minimal, d'une part le niveau de couverture de chacun des k-mers, d'autre part les positions dans le génome où ils apparaissent. Pour indexer tous les k-mers d'une collection de *reads*, nous utilisons les *Gk arrays* [Philippe *et al.*, 2011]. Quant à l'indexation du génome, nous utilisons une structure compressée qui offre une solution rapide, économique et efficace pour localiser des k-mers sur un génome [Mäkinen et Navarro, 2005].

Cette nouvelle façon de faire du *mapping* est le principal point fort de l'algorithme de CRAC. Plutôt que de trouver la localisation génomique d'origine de chaque *read*, puis de regrouper tous les alignements dans une étape d'analyse, nous intégrons directement le *mapping* dans l'analyse biologique avec d'un côté, un alignement local des k-mers et de l'autre, la couverture de chacun parmi les *reads*. Cette analyse dynamique qui combine les deux informations le long de chaque *read* forme donc l'originalité de notre approche et permet ainsi d'inférer de nombreuses informations, telles que les positions d'erreurs de séquençage ou les mutations biologiques (SNP, indels, etc). De plus, notre approche permet aussi de détecter les différentes formes de jonctions d'épissage (exons d'un même gène ou de gènes distincts) et donc les bords de celles-ci, et par extension la jonction produite par des chimères. Dans la figure 2.3 du chapitre 2, nous avons situé CRAC par rapport aux autres outils de *mapping*.

5.2 L'algorithme de CRAC

CRAC est un programme qui analyse exclusivement des *reads* sur un génome de référence et n'utilise aucune information sur les annotations des gènes ou des transcrits. Rappelons que l'analyse de *reads* est requise pour détecter différents aspects biologiques, et ce dans plusieurs domaines comme la recherche de SNP ou encore la détection des réarrangements chromosomiques sur un génome.

5.2.1 Vue d'ensemble

L'algorithme principal de CRAC est centré sur deux propriétés fondamentales :

P1 Pour un génome d'une certaine longueur, il existe une longueur minimale et suffisante pour localiser des *reads* de manière unique sur le génome sans ambiguïté. Cette longueur, dénotée k, peut être calculée rapidement [Philippe *et al.*, 2009] (voir chapitre 3). En fait, tous les k-mers d'un *read* de longueur m ($m > k$) peuvent être utilisés à bon escient dans une procédure de *mapping*.

P2 Dans une expérience RNA-Seq, les *reads* sont produits au hasard sur toute la largeur des transcrits. Par conséquent, plusieurs *reads* peuvent se chevaucher au même endroit sur une molécule. Ensuite, en procédant à un *mapping* de k-mers (de type **P1**), les *reads* affectés par une cause biologique (SNP, jonction d'épissage, indel, chimère, etc) ou une erreur de séquence ne verront pas tous leurs k-mers se localiser de façon exacte sur le génome de référence. Néanmoins, si un *read* est affecté par une cause biologique alors ses k-mers contenant la cause biologique, c'est-à-dire ceux qui ne se localisent pas sur le génome, seront conservés (de la même façon que les autres k-mers qui se localisent) parmi les *reads* qui chevauchent la même région du transcrit. Au contraire, si un *read* est affecté par une erreur de séquence alors ses k-mers contenant l'erreur ne seront pas conservés (ou très faiblement) car il est très peu probable qu'une erreur se reproduise au même endroit sur d'autre *reads*.

En définitive, l'axiome principal de l'algorithme est d'étudier les k-mers de chaque *read*. Afin d'établir ce processus dans les meilleures conditions possibles, l'ensemble des k-mers de tous les *reads* sont indexés dans les *Gk arrays* [Philippe *et al.*, 2011] (voir chapitre 4). Cette structure de données offre un bon compromis en espace mémoire et en temps de calcul. Elle organise les k-mers de façon à pouvoir les interroger et les relier facilement entre eux : par exemple, étant donné un k-mer, nous pouvons savoir dans combien de *reads* différents il est présent, ce qui nous permet de répondre aux besoins de **P2**. Ensuite, CRAC localise chaque k-mer sur le génome de manière exacte comme l'indique **P1**. Pour réaliser un tel *mapping*, nous utilisons un FM-index basé sur l'implantation de Veli Mäkinen et R. González (http://pizzachili.dcc.uchile.cl/indexes/Succinct_Suffix_Array/) [Mäkinen et Navarro, 2005]. Nous avons construit cet FM-index à l'aide d'une transformée de Burrows-Wheeler que nous avons

développée en utilisant l'algorithme de [Kärkkäinen, 2007]. D'ailleurs ce type d'index compressé a déjà été utilisé pour faire de la localisation de courtes séquences sur un génome, notamment avec les logiciels *bowtie*, *bwa* et *SOAP-v2* [Langmead *et al.*, 2009; Li et Durbin, 2009; Li *et al.*, 2009b]. Cependant, malgré leur efficacité en termes de temps de calcul et de capacité de prédiction, ces différents outils ont des faiblesses (ils ne tolèrent qu'un petit nombre de substitutions, ne savent pas gérer les longs indels). Ces restrictions deviennent problématiques à cause de l'évolution des technologies de SHD (*reads* plus longs et en plus grande quantité) mais aussi à cause des approches transcriptomiques telles que le RNA-Seq (voir section 2.4.3 du chapitre 2). Notre but est donc d'utiliser la puissance du FM-index sans être atteint par les même limites que les autres outils de *mapping* (en supposant que l'on puisse considérer notre approche comme du *mapping*). Pour y parvenir, nous procédons différemment : nous concevons un algorithme à l'aide de cette double indexation (*Gk arrays* et FM-index) afin de récupérer dynamiquement les informations contenues dans les *reads* et le génome, et ainsi intégrer une phase de *mapping* à des analyses biologiques sous-jacentes telles que la détection des mutations ponctuelles (SNV, indels courts), la distinction des erreurs de séquences ou encore la détection des jonctions d'épissage (**P1** + **P2**).

Avant de rentrer plus en détail dans l'algorithme, mettons l'accent sur un point critique de notre approche : les k-mers qui ne sont pas localisés au bon endroit sur le génome. Ces fausses localisations existent et la proportion varie en fonction de k [Philippe *et al.*, 2009]. En outre, pour garder un maximum d'informations, il faut choisir une longueur de k raisonnablement petite quitte à tolérer un petit pourcentage de fausses localisations (voir chapitre 3). Ces dernières perturbent certains aspects de nos méthodes, notamment lors de la détection des chimères. C'est pourquoi, dans le but de faciliter la compréhension, nous supposerons dans un premier temps qu'il n'y a aucune fausse localisation, puis nous reviendrons sur ce point en section 5.2.6.

5.2.2 Formalisation de l'algorithme

Rappelons que nous voulons faire de l'analyse de *reads*, en intégrant la phase de *mapping*, à partir d'une approche sur des k-mers. Pour mener ce travail, nous utilisons deux structures d'indexation : *Gk arrays* et FM-index.

Nous prenons en entrée une collection de *reads* d'une expérience de SHD. L'algorithme de CRAC interroge un par un les *reads* de la collection.
Soit r un *read* de longueur m, k un entier et G un génome de longueur n. Par définition, ce *read* contient un nombre de k-mers égal à $m-k+1$. Considérons toutes les positions i du début de ses k-mers, avec $0 \leq i < m-k+1$, puis notons $f(i)$ le k-mer en position i dans le *read*. Ensuite, nous analysons chaque $f(i)$ du *read* pour tout $0 \leq i < m-k+1$ à l'aide des deux index, puis les informations suivantes sont enregistrées :

1. *L'ensemble des localisations* de $f(i)$ sur le génome de référence ;
2. *le nombre total de localisations* de $f(i)$;
3. *le nombre de* reads qui partagent $f(i)$ (requête **Q2** des *Gk arrays* (section 4.1.1 du chapitre 4)).

Par souci de clarté, nous commençons par introduire quelques définitions qui seront essentielles pour comprendre la suite du travail.

5.2.2.1 Les localisations sur le génome

Définition 5.1. *Le* multi-ensemble des localisations *génomiques correspondant au* read *est définie par le tableau* loc[i] *qui contient l'ensemble des localisations de chaque* $f(i)$ *pour tout* $0 \leq i < m-k+1$. *Chaque* loc[i] *contient des triplets* (chr, pos, strand) *où* chr *correspond au chromosome,* pos *à la position relative sur le chromosome et* strand *à l'orientation du brin d'une localisation de* $f(i)$ *sur* G.

Définition 5.2. *Le* profil de localisations, *noté* P-loc, *est un tableau de* $m-k$ *éléments. L'entier* P-loc[i] *pour tout* $0 \leq i < m-k+1$ *correspond au nombre de localisations de* $f(i)$ *dans* r.

Définition 5.3. *La* continuité *de la localisation est définie s'il existe une localisation de* $f(i+1)$ *dans* G *qui succède directement une localisation de* $f(i)$ *pour tout* $0 \leq i < m-k$. *On dit aussi que la localisation est continue.*

Définition 5.4. *Un* break *de la localisation est défini s'il existe une série de valeurs consécutives de* P-loc *égale à* 0. *Plus formellement, si* j *est le* début *du* break *et* j' *la* fin *du* break, *alors* P-loc[i] $= 0$ *pour tout* $j < i < j'$ *mais* P-loc[j] $\neq 0$ *et* P-loc[j'] $\neq 0$. *Autrement dit,* début *et* fin *sont les points qui encadrent le* break.

5.2.2.2 Les informations partagées entre les *reads*

Définition 5.5. *Le* support *de* $f(i)$ *correspond au nombre de* reads *de la collection contenant le* k*-mer* $f(i)$.

Remarque 10. *Le* support *a une valeur minimale de* 1 *puisqu'on interroge les Gk* arrays *à partir de* $f(i)$ *qui constitue lui même une occurrence.*

Définition 5.6. *Le* profil de supports, *noté* P-support, *est un tableau de* $m-k$ *éléments. L'entier* P-support[i] *pour tout* $0 \leq i < m-k+1$ *correspond au* support *de* $f(i)$.

Définition 5.7. *La* moyenne extérieure du *P-support* *est la moyenne des* P-support[i] *telle que* loc[i] > 0 *(moyenne des* supports *en dehors du* break*) pour tout* $0 \leq i < m-k+1$. *Cette moyenne est notée* P_e-support. *La moyenne des* supports *à l'intérieur du* break *est quant à elle notée* P_i-support.

Définition 5.8. *Une* chute *du* support *est définie s'il existe une série de valeurs consécutives* $0 \leq i < m-k+1$ *telles que* P-support$[i]$ *soient beaucoup plus faibles que* $\widehat{P_e\text{-support}}$. *Plus précisément, on suppose qu'il existe un seuil* t *tel que si* j *est le* début *de la* chute *et* j' *la* fin *de la* chute, *alors* (P-support$[i]+t) < \widehat{P_e\text{-support}}$ *mais* (P-support$[j]+t) \geq \widehat{P_e\text{-support}}$ *et* (P-support$[j']+t) \geq \widehat{P_e\text{-support}}$. *Autrement dit,* début *et* fin *sont les points qui encadrent la* chute.

Corollaire 5.9. *Si une erreur de séquence affecte un* read *à une position* j, *alors le* support chute *à partir du premier* k*-mer qui contient l'erreur, c'est-à-dire en position* $p = j-k+1$ *de* P-support. *De ce fait, si* $p > 0$ *alors le* début *de la* chute *est en position* $p-1$ *et si* $j < m-k$ *alors la* fin *de la* chute *est en position* $j+1$.

Remarque 11. *Le seuil* t *qui distingue une* chute *est calculé en fonction de* $\widehat{P_e\text{-support}}$ *et de* $\widehat{P_i\text{-support}}$. *À l'origine, nous avions défini un seuil fixe à comparer au rapport* $\frac{\widehat{P_e\text{-support}}}{\widehat{P_i\text{-support}}}$ *mais nous avons remarqué que le seuil n'était pas aussi discriminant selon que les valeurs de* $\widehat{P_e\text{-support}}$ *et* $\widehat{P_i\text{-support}}$ *sont hautes ou faibles. C'est pourquoi, nous avons estimé une fonction à partir de données simulées afin de pouvoir associer une chute à une erreur de séquence en fonction de la couverture des* reads *(voir section 5.3.2).*

Exemple 5.1. Les *profils de localisations* et *supports* pour un *read*.
Soient r_1=ACACCGGTTT, r_2=ACACTGGTTT, r_3=AACACCGGTT, r_4=ACCGGTTTGA et r_5=CAACACCGGT une collection de 5 *reads* de longueur $m = 10$ et $k = 3$, alors :

		0 1 2 3 4 5 6 7 8 9			0 1 2 3 4 5 6 7 8 9
r_1	=	A C A C C G G T T T	r_2	=	A C A C T G G T T T
P-loc$_{r_1}$	=	2 2 1 1 1 1 1 1 - -	P-loc$_{r_2}$	=	2 2 0 0 0 1 1 1 - -
P-support$_{r_1}$	=	4 4 4 4 4 5 4 3 - -	P-support$_{r_2}$	=	4 4 1 1 1 5 4 3 - -

Prenons, par exemple, le 3-mer CAC en position 1 de r_1. Nous savons que $P\text{-}loc_{r_1}[1] = 2$ alors supposons que $loc_{r_1}[1] = \{(1,158,-1),(3,1589,1)\}$. Nous avons $P\text{-}loc_{r_1}[2] = 1$ ce qui signifie que ACC est localisé à 1 seul endroit sur le génome. En supposant que $loc_{r_1}[2] = \{(3,1590,1)\}$, alors par *continuité* on considérera seulement le triplet $(3,1589,1)$ de $loc_{r_1}[1]$.

Maintenant, prenons $P\text{-}support_{r_1}[0] = 4$ ce qui signifie que ACA est présent dans 4 *reads* différents de la collection (r_1, r_2, r_3 et r_5). Notons que $\widehat{P\text{-}support}_{e_{r_1}} = \frac{(4+4+4+4+4+5+4+3)}{8} = 4$, il n'y a donc pas de *chute* du *support* de r_1. En revanche, si nous prenons le 3-mer ACT, il n'est présent que dans r_2 et nulle part ailleurs dans la collection. Or, $\widehat{P\text{-}support}_{e_{r_2}} = \frac{(4+4+5+4+3)}{8} = 4$, il y a donc une *chute* du $support_{r_2}$ entre les positions $[2,4]$ de $P\text{-}support_{r_2}$ avec un *début* de *chute* en position 1 et une *fin* en position 5 (probablement une erreur de séquence en position 4 de r_2 puisqu'il y a justement un *break* sur cet intervalle $[2,4]$ avec et $\widehat{P\text{-}support}_{i_{r_2}} = \frac{(1+1+1)}{3} = 1$).

5.2.3 Distinguer les erreurs des causes biologiques

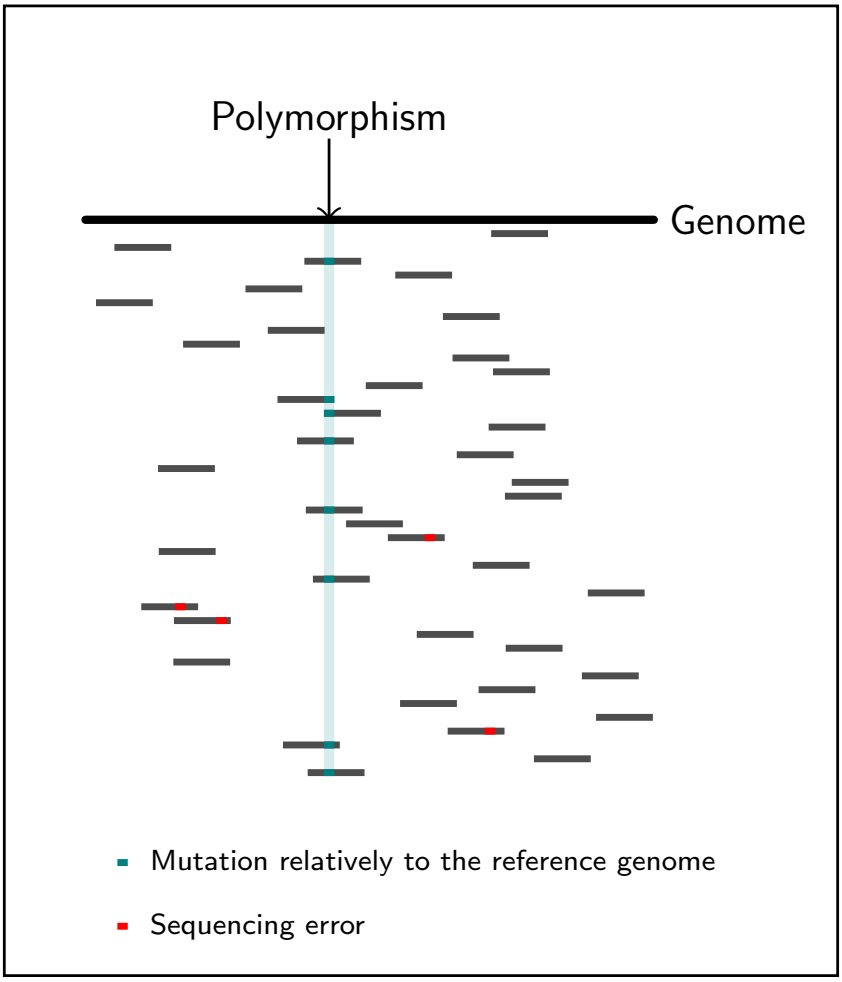
Que ce soit pour une erreur de séquence ou une cause biologique (de type indel ou substitution), la conséquence au niveau de la localisation est la même : nous observons dans *P-loc* une suite de valeurs à 0 qui constitue un *break*. Au niveau du *début* et de la *fin* du *break* se trouvent, respectivement, le dernier k-mer localisé avant la mutation et le premier k-mer localisé après la mutation. En fait, tous les k-mers qui ne sont pas localisés entre *début* et *fin* contiennent cette mutation. Ce phénomène est illustré dans la figure 5.2, où tous les k-mers rouges contiennent la mutation et ne sont pas localisés sur le génome. Dans cet exemple, la mutation est une substitution qui produit un *break* de longueur k sur le *read* et le génome (voir section 5.2.4). Mais à ce stade, rien ne permet d'affirmer que la substitution est un SNV ou une erreur de séquence.

Pour savoir si la source de la mutation est biologique ou artefactuelle, il faut se concentrer sur le *support* des *reads*. L'hypothèse est la suivante : une cause biologique est présente dans tous les *reads* qui proviennent de la même région génomique alors qu'une erreur de séquence n'est, en théorie, partagée par aucun *read*. Dans la figure 5.1, nous pouvons observer la différence de comportement entre des *reads* affectés par une erreur (petites barres rouges) et ceux qui contiennent le SNP (barre verticale bleue). Plus précisément, ce phénomène se voit sur la figure 5.2. On observe que le *support* reste stable pour le SNV ($\simeq 30$ tout le long du *read*) alors qu'il *chute* à 1 au niveau du *break* pour l'erreur. Plus généralement (d'après le corollaire 5.9), s'il existe une *chute* du *support* au niveau du *break* alors la mutation est une erreur de séquence sinon c'est une cause biologique.

Un système de score a été mis en place dans l'algorithme de CRAC afin de décider à partir de quel moment le *support* n'est plus stable, c'est-à-dire le moment où les valeurs de *P-support* sont trop faibles par rapport à la moyenne $\widehat{P_e\text{-}support}$. Nous détaillons cette procédure dans la section 5.3.2.

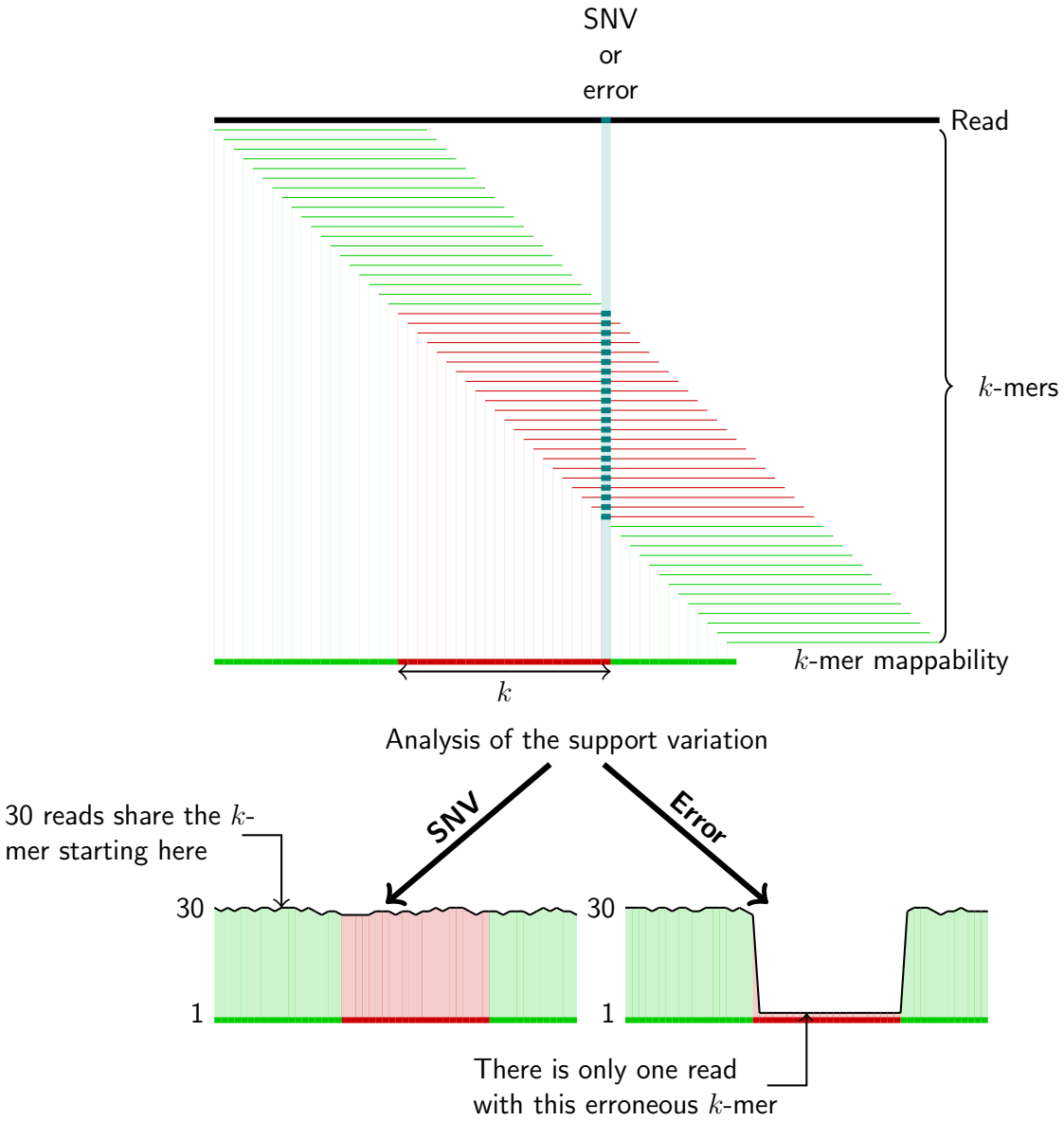
Remarque 12. *Dans tous les cas (cause biologique ou erreur), le début de la mutation se situe à la position (*début $+ k + 1$*) et se termine à la position (*fin $- 1$*) du* read *(*début *et* fin *se référant au* break*).*

FIGURE 5.1 : Détection d'un SNP entre *reads* et génome.



La figure représente le comportement général de la localisation des *reads* sur un génome. Deux catégories de *reads* ne se localisent pas parfaitement : i/ ceux qui contiennent une erreur et ii/ ceux qui contiennent une cause biologique. Dans le cas i/, il est peu probable que la même erreur se reproduise dans d'autres *reads* qui chevauchent la même région du génome (barre rouges), la mutation n'est donc pas partagée. En revanche dans le cas ii/, une mutation biologique affectera tous les k-mers (barre bleue), la mutation est donc partagée.

FIGURE 5.2 : Analyse de *P-support* pour un *read*.



La figure représente la différence entre une erreur et un SNV. Avec les *Gk arrays*, nous connaissons les *reads* qui partagent un même k-mer. Par conséquent lors d'une substitution entre un *read* et la région correspondante du génome, nous pouvons mesurer dans le *P-support* le *support* des k-mers dans le *break* (la portion rouge). S'il y a une *chute* du *support*, la substitution est une erreur, alors que si le *support* est stable, la substitution est un SNV.

5.2.4 Analyse des *breaks*

Un *break* est détecté lorsqu'une succession de k-mers ne se localise pas sur le génome (cf. définition 5.4). Lorsqu'il y a un *break* dans un *read*, il y a nécessairement une explication : une erreur de séquence, une variation génétique entre l'individu séquencé et le génome de référence ou encore une jonction d'épissage. Notre but est de déterminer la nature des *breaks* pour chaque *read* de la collection.

Rappelons que dans les conditions idéales, il n'y a pas de fausses localisations. Ce problème sera abordé en section 5.2.6 où deux procédures sont proposées. De plus, nous supposons qu'un *break* est produit par une seule cause (nous expliquerons en section 5.2.7, comment nous procédons lorsqu'une erreur et un SNV sont contenus dans le même *break*, par exemple). Il est important de mentionner ces deux points dès maintenant car les propositions mathématiques qui vont suivre ne sont plus vraies si l'une des deux conditions n'est pas vérifiée.

5.2.4.1 Localisations adjacentes sur le génome

On définit un *break adjacent* lorsque les localisations à gauche et à droite du *break* sont adjacentes sur le génome, c'est-à-dire qu'elles sont séparées d'une certaine distance positive qui se mesure sur un même chromosome et sur le même brin (en suivant l'orientation du brin).

Considérons les *breaks* induits par une erreur de séquence, un SNV, un indel ou une jonction classique. Dans tous les cas, les *breaks* sont *adjacents*.

Algorithmiquement, il y a donc trois explications possibles : une substitution, une délétion ou une insertion. Le point important est de considérer l'ensemble des localisations *loc* pour les k-mers au niveau du *début* et de *fin* de *P-loc* (les localisations avant et après le *break*).

Proposition 5.1. *Soient* j *et* j' *le* début *et la* fin *d'un* break *dans* P-loc. *Nous savons par définition que* loc$[j]$ *et* loc$[j']$ *correspondent aux deux ensembles des localisations génomiques pour* j *et* j'. *Pour tout couple* $(x,y) \in (loc[j], \text{loc}[j'])$, *si* $x.chr = y.chr$, $x.strand = y.strand$ *et* $x.pos < y.pos$ *alors nous posons les deux distances* $\ell = |j' - j - 1|$ *pour le* read *et* $L = |y.pos - x.pos - 1|$ *pour le génome.*

1. *si* $\ell = k$ *et* $\ell = L$ *alors le* break *est causé par une substitution ;*
2. *si* $\ell = k-1$ *et* $L = k+p+1$ *alors le* break *est causé par une insertion de* p *nucléotides dans le génome ;*
3. *si* $\ell = k+p-1$ *et* $L = k-1$ *alors le* break *est causé par une délétion de* p *nucléotides dans le génome.*

Démonstration de la Proposition 5.1 : Par hypothèse, il n'y a pas de fausses localisations et il n'y a qu'une seule cause par *break*. Par ailleurs, lors d'une substitution entre le *read* et la région correspondante du génome, tous les k-mers qui contiennent la substitution ne sont pas localisés (FI-

GURE 5.2). Il y a donc exactement un *break* de longueur k qui se forme sur le *read*, c'est-à-dire $\ell = k$. Puis, par définition de la substitution, $\ell = L$.

Lors d'une insertion dans le génome, tous les k-mers qui chevauchent la frontière de l'insertion ne sont pas localisés. Il y donc exactement un *break* de longueur $k-1$ qui se forme sur le *read*, c'est-à-dire $\ell = k-1$. Puis, par définition de l'insertion de p nucléotides $L = \ell + p$.

Une délétion sur le génome équivaut à une insertion dans le *read*, la preuve est donc la même que la précédente. Une représentation graphique est aussi proposée en figure 5.3.

□

Exemple 5.2. Les substitutions et les indels entre *read* et génome.
Soit r un *read*, x une séquence génomique et $k = 3$.

1. Cas d'une substitution à la position 5 entre r et x :

$$\begin{array}{lcl} & & \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}{} \\ x & = & \mathrm{ACACTGGTTT} \\ r & = & \mathrm{ACACTAGTTT} \\ \text{P-loc}_r & = & 1\,1\,1\,0\,0\,0\,1\,1\,-\,- \end{array}$$

$$\text{alors} \begin{cases} L = 6-2-1 = 3 \\ \ell = 6-2-1 = 3 \end{cases} \quad \Rightarrow \quad \ell = L = k$$

2. Cas d'une insertion de 3 nucléotides à la position 7 dans x :

$$\begin{array}{lcl} & & \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12}{} \\ x & = & \mathrm{ACACTGGAAATTT} \\ r & = & \mathrm{ACACTGGTTT} \\ \text{P-loc}_r & = & 1\,1\,1\,1\,1\,0\,0\,1\,-\,- \end{array}$$

$$\text{alors} \begin{cases} L = 10-4-1 = 5 \\ \ell = 7-4-1 = 2 \end{cases} \quad \Rightarrow \quad \ell + 3 = L = k-1+3$$

3. Cas d'une délétion de 2 nucléotides à la position 4 dans x :

$$\begin{array}{lcl} & & \overset{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7}{} \\ x & = & \mathrm{ACACGTTT} \\ r & = & \mathrm{ACACTAGTTT} \\ \text{P-loc}_r & = & 1\,1\,0\,0\,0\,0\,1\,1\,-\,- \end{array}$$

$$\text{alors} \begin{cases} L = 4-1-1 = 2 \\ \ell = 6-1-1 = 4 \end{cases} \quad \Rightarrow \quad \ell = L+2 = k-1+2$$

5.2.4.2 Localisations de transcrits chimères

Maintenant, imaginons un transcrit formé à partir de deux séquences non adjacentes sur le génome, les *reads* qui chevauchent la jonction des deux séquences ne sont pas continus. Autre-

ment dit, les localisations à gauche et à droite du *break* ne correspondent pas à la même région du génome et forment ainsi un *break chimérique*. La proposition 5.1 n'est donc pas directement applicable car il est difficile d'évaluer une distance entre deux régions distinctes d'un génome. Cependant, une deuxième proposition s'adapte assez facilement.

Proposition 5.2. *Soient* j *et* j' *le* début *et la* fin *d'un* break *dans* P-loc *et la distance* $\ell = |j' - j - 1|$ *sur le* read. *Nous savons par définition que* loc$[j]$ *et* loc$[j']$ *correspondent aux deux ensembles des localisations génomiques pour* j *et* j'. *Le* break *est causé par une chimère si* $\ell = k - 1$ *et, pour tous les couples* $(x, y) \in (loc[j], \text{loc}[j'])$, *il n'existe pas de distance* $L = |y - x - 1|$ *et* $y > x$ *correspondante à une localisation adjacente sur le génome. Par contraposée, nous devons obtenir : i/* $x.chr \neq y.chr$, *ou ii/* $x.strand \neq y.strand$, *ou iii/* $x.pos > y.pos$.

Démonstration de la Proposition 5.1 : Une chimère est produite à partir de deux régions distinctes d'une séquence génomique (voir section 2.6.2 du chapitre 2). En conséquence, une chimère peut être perçue algorithmiquement comme une insertion génomique, d'où $\ell = k - 1$. Ensuite, pour mesurer une distance d'une localisation adjacente, il faut être sur le même chromosome et le même brin d'un génome avec un sens de lecture qui suit l'orientation du brin, ce qui correspond à une insertion classique. Autrement dit, une chimère est constituée de deux morceaux du génome qui sont : i/ sur des chromosomes différents ($x.chr \neq y.chr$), ou ii/ des brins différents ($x.strand \neq y.strand$), ou iii/ inversés ($x.pos > y.pos$).

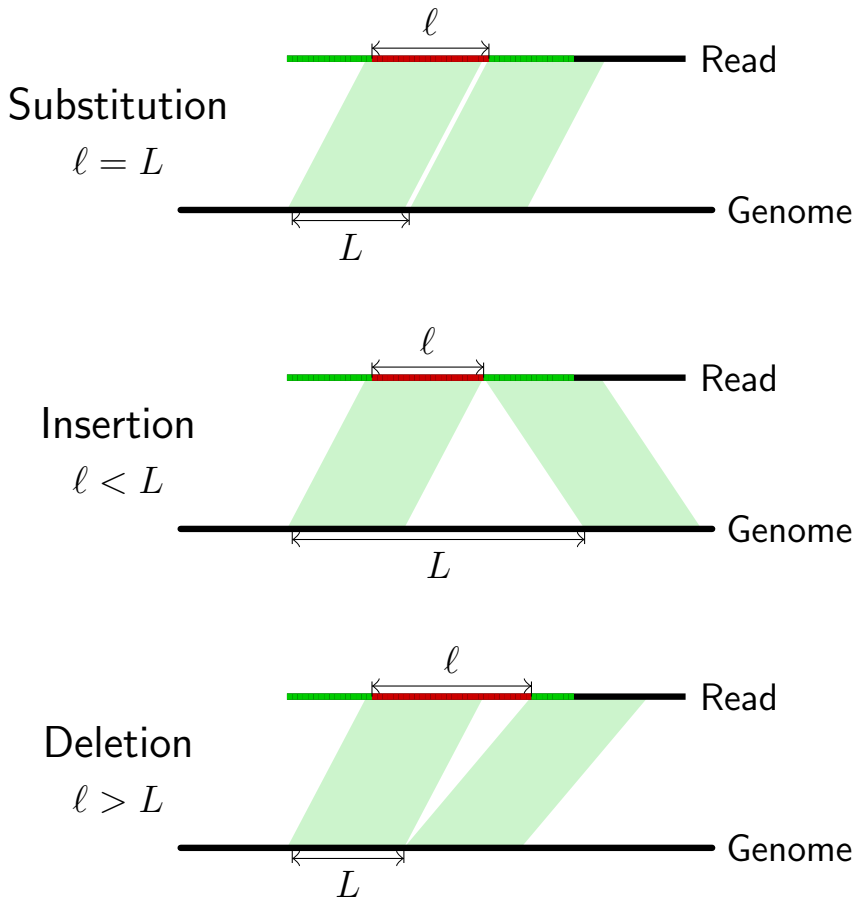
□

5.2.4.3 Quelques remarques importantes

R1 Il est possible qu'une chimère soit composée de deux gènes distincts mais adjacent sur le chromosome [McPherson *et al.*, 2011]. Dans un tel cas, on pourrait confondre ces jonctions chimériques avec des jonctions classiques d'épissage (cf. proposition 5.1). Il y a deux possibilités pour éviter ce problème : soit, par post-traitement en vérifiant l'annotation des deux séquences à gauche et à droite du *break* ; soit, en fixant un seuil sur la distance (en prenant la longueur maximale de l'intron estimé chez une espèce, par exemple entre 400 et 500 kpb chez l'homme [Sakharkar *et al.*, 2004]).

R2 Pour définir précisément la nature d'un *break*, il faut nécessairement un *début* et une *fin*. Dans le cas contraire le *break* est sur un bord du *read*, et nous n'avons seulement que les informations nécessaires pour différencier une erreur de séquences d'une cause biologique. Cependant, nous verrons dans les résultats (section 5.5.1) qu'une expérience de RNA-Seq produit suffisamment de *reads* pour qu'une même cause soit présente sur le bord de certains *reads* mais aussi dans le milieu de certains autres.

FIGURE 5.3 : Analyses des *breaks*.



Cette figure illustre les différents type de *breaks* entre *reads* et génome. Ils sont identifiés dans *P-loc*. Nous regardons la *loc* des k-mers de chaque côté du *break* et nous affirmons (en théorie) : i/ si $\ell = k$ et $\ell = L$ alors il y a une substitution entre *read* et génome, ii/ si $\ell = k-1$ et $\ell < L$ alors il y a insertion d'une séquence dans le génome, iii/ puis si $\ell > L$ et $L = k-1$ alors il y a une suppression d'une séquence dans le génome.

5.2.5 Identifier des informations sur les régions multiples

Dans l'analyse des *reads*, la plupart des programmes ne s'intéressent pas aux localisations multiples car elles impliquent une augmentation du temps de calcul et augmentent la complexité des algorithmes, cependant certaines informations peuvent s'avérer utiles, voire importantes. D'abord, nous distinguons les duplications au sein des localisations multiples. Elles concernent les *reads* qui ne sont localisés qu'une petite quantité de fois sur le génome (≤ 5 dans CRAC). Ensuite, nous distinguons les répétitions. Elles concernent les *reads* qui contiennent un facteur localisé plusieurs fois sur le génome (≥ 20 dans CRAC) mais le reste est localisé de façon unique sur le génome.

5.2.5.1 Identifier les bordures de répétitions

Dans une répétition, *P-support* n'est pas stable (exemple queue polyA) et *P-loc* inexploitable (trop de localisations). En revanche, comme la longueur k est choisie pour optimiser le nombre de localisations uniques des k-mers sur le génome, nous pouvons exploiter les *reads* qui chevauchent une bordure de répétition. Pour un *read* en bordure, les valeurs de *P-loc* identifiants les k-mers en sortie d'une répétition devraient être faibles (moins de localisations). De plus, lorsque le *support* se stabilise, nous pouvons identifier plus précisément le bord de la répétition, même si la localisation des k-mers est faible mais pas unique à cet endroit.

5.2.5.2 Identifier les duplications

La présence de transcrits dans des régions du génome qui sont répétées peut compliquer les analyses biologiques. D'ailleurs, si les répétitions affectent tous les k-mers d'un *read*, alors celui-ci est entièrement répété et il est de ce fait impossible de déterminer sa position génomique d'origine. En pratique, les séquences fortement répétées sont difficiles à exploiter dans une analyse de *reads*, mais celles qui sont faiblement répétées peuvent être informatives sur la variation structurelle des gènes, leur duplication ou d'autres phénomènes biologiques [Bansal et Eulenstein, 2008]. Pour limiter la combinatoire due aux répétitions, nous considérons un seuil sur le nombre de localisations génomiques des k-mers (5 par défaut dans CRAC), ce qui nous permet de séparer en deux classes les hautes et les faibles répétitions (voir section 5.2.7).

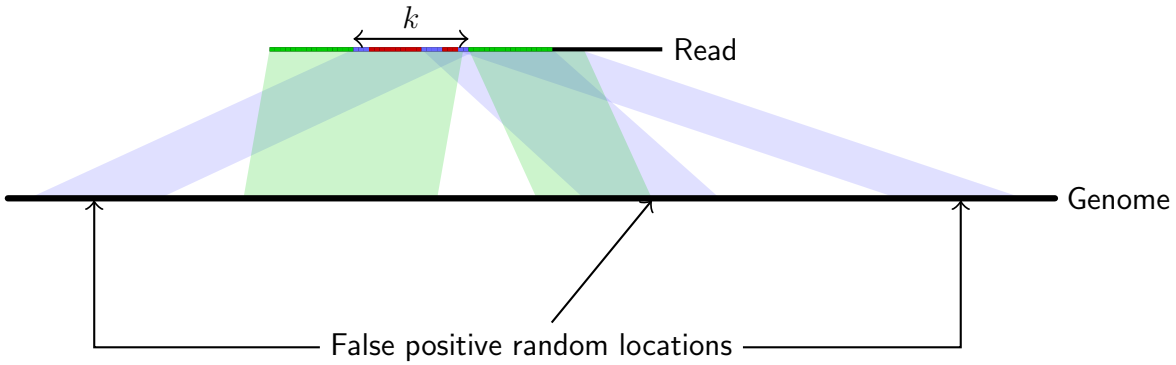
5.2.6 Optimiser l'algorithme à cause des fausses localisations

Les propositions précédentes ne sont plus vérifiées en présence de fausses localisations. Nous devons donc prendre en considération ce problème dans l'algorithme de CRAC.

Tout d'abord, avec l'équation 3.18 du chapitre 3, nous pouvons estimer la proportion de fausses localisations d'une collection de k-mers en fonction de la longueur k, c'est-à-dire le pourcentage des k-mers localisés mais pas au bon endroit sur le génome. En outre, pour garder un maximum d'informations, il faut choisir une longueur de k raisonnable quitte à tolérer un petit pourcentage

de fausses localisations. Par exemple, pour analyser du RNA-Seq sur le génome humain, nous choisissons une longueur $k = 22$ ce qui donne un pourcentage $\simeq 2$ % de fausses localisations. Pour combler cette tolérance, nous devons vérifier la *continuité* de la localisation dans le *read*. Le processus est assez naturel lorsqu'il n'y a pas de *break*, il suffit de vérifier la continuité lors d'une lecture de gauche à droite des k-mers dans le *read*. En revanche dans le cas d'un *break*, une chimère engendre aussi des localisations qui, par définition, ne sont pas continues à gauche et à droite du *break*. Le constat est exactement le même en présence de fausses localisations : elles se propagent sur les bords d'un *break* et coupe la *continuité* de la localisation sur tout le *read* ; pire encore, elles se propagent à l'intérieur d'un *break* et forment ainsi deux nouveaux *breaks* avec des localisations qui ne sont pas continues (FIGURE 5.4). Au final, si nous ne réglons pas ce problème, beaucoup de fausses chimères seront générées et des vraies informations seront perdues (on commence à mieux comprendre la difficulté d'identifier les vraies chimères).

FIGURE 5.4 : Perturbation des *breaks*.



Cette figure illustre le problème des fausses localisations des k-mers. Dans une telle situation, la proposition 5.1 est perturbée par les k-mers qui ne sont pas bien localisés (barres bleues).

C'est pourquoi, nous proposons d'optimiser l'algorithme de CRAC à l'aide de deux procédures : la première pour pallier le problème des fausses localisations sur les bords d'un *break*, la deuxième pour pallier celui des fausses localisations à l'intérieur d'un *break*.

5.2.6.1 La procédure d'*extension*

Cette procédure consiste à étendre de chaque côté d'un *break* afin de vérifier la présence ou non de fausses localisations. Dans le cas positif, le *break* est réajusté et les fausses localisations ne sont pas comptabilisées dans le calcul des *P-loc* et *P-support*.

Par soucis de concision et de compréhension, l'algorithme 7 que nous proposons ici, ne représente qu'une version simplifiée de la procédure d'*extension* implantée dans CRAC. Plus précisément, nous n'allons considérer qu'une seule cause par *break* et nous n'allons arrêter d'étendre

lorsque le *break* devient adjacent (cf. proposition 5.1). Alors qu'en réalité, la procédure d'*extension* doit aussi gérer le comportement de plusieurs causes dans un *break* (la condition de sortie ($posEnd - posStart > k$ en ligne 7 n'est pas aussi simple). De plus, des fausses localisations peuvent engendrer un *break* adjacent (moins probable mais possible) alors nous devons toujours étendre pour vérifier la *continuité*, et ce, même en cas d'adjacence (la condition de sortie *candidateFound* ligne 7 est plus complexe).

Nous présentons ici les différentes étapes de l'algorithme 7 allégé de la procédure d'*extension* implantée dans CRAC.

Initialisation (lignes 2 à 6). Pour commencer la procédure d'*extension*, nous commençons par récupérer le *début* et la *fin* du *break* (lignes 2 et 3 de l'algorithme). Nous interrogeons le FM-index de manière dynamique pour récupérer les ensembles *loc*[*début*] et *loc*[*fin*] (lignes 4 et 5). *CandidatFound* est un booléen qui arrête la procédure dès qu'on a trouvé un bon *break* (avec des localisations adjacentes sur le génome). Il est initialisé à Faux au départ de la procédure (ligne 6).

Boucle principale (lignes 7 à 17). Les deux ensembles *locStart* et *locEnd* contiennent des triplets *(chr, pos, strand)* (cf. définition 5.1). Nous partitionnons ces ensembles en deux sous-ensembles constitués de paires *(chr, pos)* en fonction de *strand*. Ces sous-ensembles sont ensuite triés par chromosome et par position (lignes 8 et 9). À cette étape de la procédure, nous avons quatre sous-ensembles triés avec d'un côté les localisations du brin positif du génome *locStartSense*, *locEndSense*, et de l'autre celles du brin négatif *locStartAntisense*, *locEndAntisense*.

Nous parcourons les ensembles deux à deux : (*locStartSense*, *locEndSense*) d'une part et (*locStartAntisense*, *locEndAntisense*) de l'autre. Nous enregistrons dans les deux variables x et y deux triplets *(chr, pos, strand)* tels que : i/ $x.chr = y.chr$, ii/ $x.strand = y.strand$, et iii/ $L = |y.pos - x.pos|$ où $y > x$ et L est la plus petite des distances (ligne 10).

Si aucun des triplets ne vérifie les conditions i/ et ii/, le couple (x, y) est choisi arbitrairement mais ne correspond pas à un *candidateFound* donc on continue d'étendre (lignes 11 à 17). Dans le cas contraire, (x, y) correspond à une localisation adjacente sur le génome, donc on sort de la boucle. Ensuite, la procédure s'arrête si la distance $|posEnd - posStart| > k$ car nous supposons qu'il n'y a qu'une seule cause par *break* (cf. proposition 5.1). Elle s'arrête aussi lorsqu'on ne peut plus étendre à gauche et à droite (ligne 7).

Étape de réajustement (ligne 18). À chaque étape de la boucle principale, nous étendons des deux côtés du *break*. Il est toutefois possible que des fausses localisations ne soient présentes que d'un côté du *break*. Par conséquent, lorsque nous trouverons un bon candidat, nous aurons trop étendu du côté où il n'y a pas de fausses localisations. Il faut donc revenir en arrière pour réajuster le *break* à ses bonnes bornes. Par ailleurs, dans le cas où le *break* est réellement induit par une chimère, nous allons étendre par vérification. Mais une fois sortie de la boucle, il faut revenir en arrière

pour réajuster les bornes. D'ailleurs, notons qu'il est aussi possible d'avoir des fausses localisations sur un *break* chimérique : elles sont aussi gérées puisque lors du réajustement, nous vérifions la *continuité* de la localisation de chaque côté du *break* (cf définition 5.3).

Algorithme 7 : Extension procedure of CRAC.

```
   Data : loc, start break, end break
   Result : The new (start break, end break)
1  begin
2      posStart = start break ;
3      posEnd = end break ;
4      locStart = loc[posStart];
5      locEnd = loc[posEnd];
6      candidateFound = False ; /* a boolean to stop the process if we have a nice
       break  */
7      while (posStart > 0 or posEnd < m − k) and (posEnd − posStart ≤ k) and (not
       candidateFound) do
8          (locStartSense,locStartAntisense) = sortedByStrand(locStart) ;    /* sorted on chr
           and pos for each strand */
9          (locEndSense,locEndAntisense) = sortedByStrand(locEnd) ; /* sorted on chr and
           pos for each strand */
10         (x, y) =
           getMinimalDistance(locStartSense,locEndSense,locStartAntisense,locEndAntisense) ;
           /* get the best pair of locations according to the absolute
           distance between locStart and locEnd on the genome */
11         if x.chr = y.chr and x.strand = y.strand and x.pos < y.pos then
12             candidateFound = True;
13         else
14             if posStart > 0 then
15                 posStart ← posStart − 1;
16             if posEnd < m − k then
17                 posEnd ← posEnd + 1;
18     (start break, end break) = getContinuedBackTrack(posStart,posEnd) ;    /* We go back
       by checking the continuity for each break independently. We stop when
       there is no more continuity */
19     return (start break, end break);
20 end
```

5.2.6.2 La procédure de *fusion*

Cette procédure consiste à fusionner deux *breaks* produits par des fausses localisations pour n'en créer qu'un seul (celui d'origine). Par exemple, dans la figure 5.4, la présence de fausses localisations dans le *break* (deuxième zone bleue) forme deux « faux » *breaks* (barres horizontales rouges à gauche et à droite de cette zone bleue). Lorsqu'une *fusion* est faite, les fausses localisations ne sont pas comptabilisées dans le calcul des *P-loc* et *P-support*.

Là encore pour que ce soit plus compréhensible, l'algorithme 8 ne représente qu'une version simplifiée de la procédure de *fusion* de CRAC. Nous allons supposer qu'il n'y a qu'un seul « vrai » *break* par *read* (une seule mutation) mais avec des fausses localisations à l'intérieur. Autrement dit, nous nous intéressons aux *reads* qui contiennent exactement deux « faux » *breaks*. En pratique, les *reads* ne contiennent toujours qu'une seule mutation, surtout avec la production de *reads* de plus en plus longs ou de *reads* de type RNA-Seq où souvent nous avons une jonction couplée à une erreur ou un SNV.

En fait, l'algorithme implanté dans CRAC gère un nombre illimité de *breaks*. Il réitère les étapes de l'algorithme 8 sur toutes les combinaisons de *breaks* possibles en sélectionnant les meilleures *fusions*. Voici une explication des différentes étapes de l'algorithme 8 allégé.

Initialisation (lignes 2 à 9). Pour commencer la procédure de *fusion*, nous commençons par récupérer les informations des deux *breaks* à fusionner : $break_1$ (ligne 2) et $break_2$ (ligne 7). Nous interrogeons le FM-index de manière dynamique pour récupérer les ensembles $loc[début]$ et $loc[fin]$ de $break_1$ (ligne 3) et $break_2$ (ligne 8). Les booléens *niceBreak1* et *niceBreak2* permettent de vérifier indépendamment si les deux ($break_1$ et $break_2$) sont adjacents (lignes 4 et 9) : la notion de *nice* est employée lorsqu'un *break* est adjacent (cf. proposition 5.1).

***Fusion* des *breaks* (lignes 12 à 22).** Si aucun des deux *breaks* ($break_1$ et $break_2$) n'est adjacent, autrement dit si chacun représente une chimère (cf. proposition 5.2), alors nous les fusionnons (ligne 15). En revanche, à partir du moment où l'un des deux est adjacent, la *fusion* n'est pas immédiate. Nous commençons par vérifier ce que donnerait cette *fusion* mais sans pour autant la faire (ligne 12). Dans le cas où le *break* issu de la *fusion* de $break_1$ et $break_2$ serait adjacent, nous les fusionnons réellement (ligne 16).

En fait, il n'y a pas de *fusion* lorsque les deux *breaks* ($break_1$ et $break_2$) sont adjacents (c'est-à-dire *niceBreak1* et *niceBreak2*) ou lorsque l'un des deux est adjacent et que le le *break* issu de la *fusion* est chimérique.

5.2.6.3 Remarque importante

Notons qu'une *fusion* est toujours précédée d'une *extension* car pour fusionner deux *breaks*, il faut que les bords soient bien ajustés ce qui n'est pas forcément le cas avant l'*extension*. En effet,

à cause des fausses localisations sur les bords, une *fusion break* engendrerait un *break* chimérique mais en fonction des conditions (ligne 13 de l'algorithme 8), la *fusion* pourrait ne pas se faire. Prenons par exemple la figure 5.4 qui cumule des fausses localisations dans le *break* d'origine et sur ses bords. Si nous essayons de fusionner, nous n'allons pas trouver le *break* avec les bordures vertes (ligne 12 de l'algorithme 8) à causes des fausses localisations sur les bords. En revanche, si dans une première étape, on étend et on réajuste les bornes du *break* (algorithme 7), alors la *fusion* se fera obligatoirement dans une seconde étape.

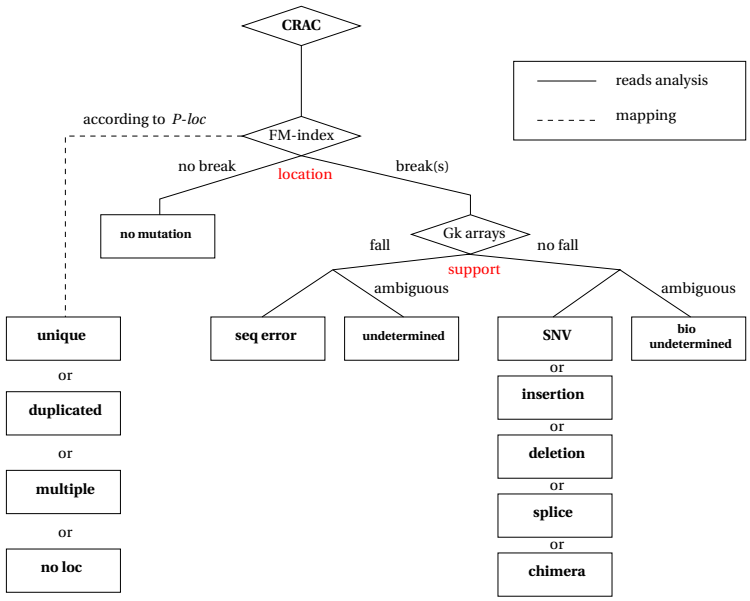
Algorithme 8 : Fusion procedure of CRAC.

```
   Data : loc, break_1, break_2
   Result : The new break if fusion or Null otherwise
1  begin
       /* Initialisation of the first break                              */
2      posStart1 = start break_1, posEnd1 = end break_1;
3      locStart1 = loc[posStart1], locEnd1 = loc[posEnd1];
4      niceBreak1 = ( locStart1.chr = locEnd1.chr  and
5                     locStart1.strand = locEnd1.strand  and
6                     locStart1.pos < locEnd1.pos );
       /* Initialisation of the second break                             */
7      posStart2 = start break_2, posEnd2 = end break_2;
8      locStart2 = loc[posStart2], locEnd2 =loc[posEnd2];
9      niceBreak2 = ( locStart2.chr = locEnd2.chr  and
10                    locStart2.strand = locEnd2.strand  and
11                    locStart2.pos < locEnd2.pos );
       /* The fused condition                                            */
12     niceFusedBreak = ( locStart1.chr = locEnd2.chr  and
13                        locStart1.strand = locEnd2.strand  and
14                        locStart1.pos < locEnd2.pos );
15     if (not niceBreak1  and not niceBreak2)
16     or ((not niceBreak1  or not niceBreak2)  and  niceFusedBreak) then
           /* The new break object is built for the fusion               */
17         buildBreak(break);
18         start break = posStart1;
19         end break = posEnd2;
20         return (break);
21     else
22         return (Null);
23 end
```

5.2.7 Classifier les *reads*

À partir de l'analyse de *P-loc* et *P-support*, nous procédons à plusieurs vérifications sur la *continuité* de la localisation, les *breaks* de localisation, les *chutes* du *support*, etc. À partir de ces critères, nous pouvons classer les *reads* d'une expérience de SHD (FIGURE 5.5).

FIGURE 5.5 : Organigramme de l'analyse des *reads* par CRAC.



Cette figure illustre la façon dont les *reads* sont classés dans CRAC. Nous pouvons distinguer deux niveaux d'organisation, un pour le *mapping* (traits discontinus), l'autre pour l'analyse (traits pleins). Concernant le *mapping* : tous les *k*-mers d'un *read* sont localisés un par un. Ensuite, en respectant la *continuité*, nous calculons la proportion des *k*-mers qui sont localisés de façon unique, dupliquée, multiple ou aucune fois sur le *read* afin de le classer au bon endroit. Quant à l'analyse : nous examinons les deux profils *P-support* et *P-loc* afin de détecter par quelles causes le *read* est affecté (en fonction des *breaks* et des *chutes*). Notons qu'un *read* peut être affecté par plusieurs causes, par exemple erreur+SNV. Dans ce cas, il est classé à un endroit pour chaque cause. Autrement dit, cette classification ne forme pas une partition des *reads* mais des causes suivantes : erreurs, SNV, indels, jonctions d'épissage, etc.

5.2.7.1 Analyses des mutations

Les mutations sont classées en fonction de leur nature. Par exemple, nous distinguons les *reads* qui sont affectés par une jonction, des *reads* affectés par une mutation ponctuelle ou encore des *reads* affectés par une erreur. Comme nous pouvons le voir dans la figure 5.5. Dans chaque *read*, nous commençons par analyser le nombre de *breaks* : le *read* est considéré sans mutations s'il ne contient aucun *break*. À cause des fausses localisations, nous savons que le nombre de *breaks* ne correspond pas nécessairement à un nombre de mutations biologiques, jonctions ou erreurs (cf. section 5.2.6). Une fois les procédures de *fusion* et d'*extension* achevées, nous pouvons associer

chaque *break* à un évènement biologique ou à une erreur de séquence en fonction de *P-support* : la présence d'une *chute* dans le *break* est déterminante (cf. corollaire 5.9). Enfin, une analyse de chaque *break* (cf. section 5.2.4) permet d'identifier précisément la cause biologique (SNV, indels, jonctions d'épissage, chimères). Notons que les résultats sont donnés par événement biologique : si un *read* contient une erreur, une jonction et un SNV, il est présent dans les trois fichiers de résultats associés à chacune des causes. Cependant, quelques cas sont difficiles à classer, en voici la liste.

Mutations peu couvertes et/ou sur les bords. Lorsque très peu de *reads* couvrent la région de la mutation, les valeurs de *P-support* sont faibles avec une moyenne $\widehat{P_e\text{-}support}$ faible elle aussi. Il est donc beaucoup plus difficile de distinguer l'erreur de séquence de la cause biologique (branche du *support* dans la figure 5.5). C'est pourquoi, nous classons, à part, ces mutations peu couvertes afin d'éviter les faux positifs mais aussi de permettre des investigations plus profondes pour éventuellement sauver les mutations les plus rares (classe *bio undetermined* de la figure 5.5 avec le message précisant « no cover », « no end break » ou « no start break » selon la nature de l'ambiguïté). Notons quand même que les SHD produisent de plus en plus de séquences, ce qui augmente les chances d'avoir une couverture minimale sur la quasi totalité des transcrits.

Plusieurs mutations dans un *break*. Bien qu'en théorie quand il y a une différence entre *read* et génome, la longueur du *break* est égale à k, il est possible en pratique que cette longueur soit plus grande lorsqu'il y a plusieurs différences consécutives séparées de moins de k nucléotides sur le *read*. Certains cas sont détectables algorithmiquement. Par exemple, lorsqu'une erreur est suivie d'une mutation biologique, une *chute* du *support* ne va être observée que sur une partie du *break*. Autre exemple, lorsque deux erreurs sont cumulées, la *chute* va durer tout le long du *break* et on saura que le *read* contient plusieurs erreurs sans être sûr précisément des positions des erreurs.

Néanmoins, lorsque des événements biologiques sont cumulés, il est généralement difficile de tirer des conclusions hâtives, surtout lorsqu'il y a une mutation ponctuelle au bord d'une jonction d'épissage. C'est pourquoi nous décidons de les classer à part (classe *bio undetermined* de la figure 5.5 avec le message « multiple biological events in break »).

Causes inconnues. Il y a un certain nombre de *reads* que nous n'arrivons pas à classer parce que les profils *P-loc* et *P-support* sont complexes à analyser, lorsqu'une mutation ou une erreur est présente dans une région répétée du génome par exemple. Ces *reads* sont répertoriés dans la classe *undetermined* (voir figure 5.5) avec un message expliquant pourquoi nous n'arrivons pas à les classer.

5.2.7.2 Possibilité de faire du *mapping*

CRAC donne la possibilité de ne faire exclusivement que du *mapping* bien que notre philosophie soit d'intégrer le *mapping* à des analyses biologiques telles que la détection des SNV ou des

variants d'épissage. Notre stratégie de localisation **P1** (section 5.2.1) semble être appropriée pour des données transcriptomiques (voir résultats en section 5.5.1.1). De ce fait, des outils d'analyse tels que *samtools* peuvent utiliser le *mapping* de CRAC pour optimiser leur recherche sur des données transcriptomiques [Li, 2011]. Notons que les résultats sont donnés au format *SAM* qui est un format standard utilisé par la plupart des outils de *mapping* [Li *et al.*, 2009a].

5.3 Les méthodes expérimentales

Plusieurs outils ont été développés, que ce soit pour le *mapping* avec notamment Bowtie [Langmead *et al.*, 2009], SOAP2 [Li *et al.*, 2009b], BWA [Li et Durbin, 2009], GASSST [Rizk et Lavenier, 2010], GSNAP [Wu et Nacu, 2010], BWASW [Li et Durbin, 2010], la détection des SNV et indels avec samtools [Li *et al.*, 2009a] ou encore la détection des jonctions d'épissage avec TopHat [Trapnell *et al.*, 2009], MapSplice [Wang *et al.*, 2010], GSNAP [Wu et Nacu, 2010]. Afin de comparer CRAC à tous ces logiciels, nous avons utilisé les mêmes jeux de données.

Nous avons calculé leur *sensibilité* : le pourcentage des événements recherchés qui sont retrouvés par l'outil ; puis leur *précision* : le pourcentage des événements précisément retrouvés parmi ceux qui sont détectés par l'outil. Dans le *mapping*, l'événement en question est souvent la localisation unique. Quant à l'analyse de *reads*, les événements sont souvent multiples mais correspondent souvent à la détection des erreurs de séquences, des SNV, des jonctions d'épissage, etc.

Des jeux de données biologiques ne suffisent pas pour mesurer la *sensibilité* et la *précision* d'une méthode, il faut connaître les réponses (où sont les erreurs, les mutations et les jonctions) et le seul moyen est d'utiliser des données simulées. Nous avons conçu de telles données en nous approchant le plus possible du comportement de données réelles.

5.3.1 La simulation de RNA-Seq

CRAC est un logiciel spécialisé dans le traitement du RNA-Seq, nous avons donc simulé des données contenant des jonctions d'épissage.

5.3.1.1 *FluxSimulator*

Afin de simuler des *reads*, nous utilisons *FluxSimulator*, un programme capable de reproduire les différentes étapes d'une expérience RNA-Seq (transcription inverse, fragmentation, filtre, séquençage, etc). Flux est développé par Michael Sammeth http://flux.sammeth.net/simulator.html. En entrée, *FluxSimulator* prend un fichier de configuration dans lequel nous devons préciser le nombre de *reads* à générer, leur longueur, le nombre de molécules d'ARN par transcrit, le type de *primer*, un modèle d'erreur de séquence, un fichier au format *GTF* contenant les annotations des exons des transcrits de gènes.

À partir d'un génome au format *FASTA* et d'un fichier d'annotation *GTF* qui lui est associé, *FluxSimulator* est aussi capable de produire des *reads* qui sont issus de transcrits alternatifs.

Un autre point important de *FluxSimulator* est la possibilité de contrôler le niveau d'expression des transcrits grâce au nombre de molécules d'ARN total à indiquer dans le fichier de configuration. Nous fixerons ce nombre de molécules en fonction du nombre et la longueur des *reads* à générer, afin d'avoir un profil d'expression proche de la réalité (voir section 5.4.1).Enfin, notons que *FluxSimulator* produit du RNA-Seq non orienté (voir section 2.2.2 du chapitre 2).

5.3.1.2 *GenomeSimulator*

Nous voulons intégrer des mutations biologiques entre les *reads* et le génome de référence. Pour ce faire, nous avons utilisé *GenomeSimulator*, un logiciel que nous avons développé. Il génère, à partir d'un génome de référence (dmel5 et hg19 dans notre cas), des mutations ponctuelles mais aussi des indels plus longs et des translocations chromosomiques. Les mutations sont produites à des positions aléatoires sur tout le génome.

Dans le cas d'une substitution, le nucléotide d'origine du génome est remplacé par un des trois autres nucléotides qui composent Σ. Par défaut, les longueurs des insertions et des délétions sont distribuées uniformément sur l'intervalle $[1,14]$. Les insertions sont des fragments d'ADN construits aléatoirement et les délétions sont des fragments supprimés du génome de référence. Quant aux chimères, elles sont construites à partir d'un échange de la séquence nucléotidique de deux gènes différents, sur un même chromosome ou deux chromosomes distincts. Les gènes sont aussi choisis de façon arbitraire mais un même gène ne peut pas être impliqué dans plusieurs chimères à la fois. L'échange ne commence pas nécessairement au début du gène mais toujours au début d'un exon et il se termine toujours à la fin du gène. Pour des raisons de simplicité, nous construisons les chimères à partir d'un seul brin du génome de référence (le brin sens).
En entrée, *GenomeSimulator* requiert les pourcentages de substitutions, insertions, délétions à appliquer au génome de référence, ainsi que la probabilité d'implication d'un exon dans une chimère.

Une fois que toutes les mutations ont été produites sur le génome de référence, un génome muté au format *FASTA* ainsi qu'un fichier *GTF* contenant toutes les annotations du génome muté sont générés. C'est ce fichier *GTF* qu'on utilise dans *FluxSimulator*.

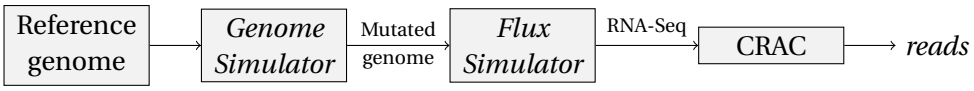


FIGURE 5.6 : Illustration du pipeline de simulation

5.3.1.3 Génération des *reads* et récupération des informations

Lorsque la simulation du génome (*GenomeSimulator*), puis la simulation du RNA-Seq (*FluxSimulator*) sont terminées (FIGURE 5.6), nous avons connaissance de plusieurs informations :

i/ un fichier *ERR* avec les identifiants des *reads* contenant les erreurs de séquences et la position des erreurs dans ceux-ci ;

ii/ un fichier au format *BED* permettant de distinguer les *reads* contenant une jonction d'épissage, ainsi que toutes les informations relatives à cette jonction sur le génome muté (point de cassure sur le *read*, localisations de part et d'autre sur le génome, etc) ;

iii/ un fichier *INFO* avec les identifiants des *reads* affectés par une mutation par rapport au génome de référence (substitution, insertion, délétion, chimère) et les informations relatives à cette mutation (position sur le *read*, localisation sur le génome de référence et le génome muté, longueur de la mutation pour les indels, gènes impliqués pour la chimère, etc).

Cependant, en ii/ le fichier *BED* correspond à l'annotation des jonctions sur le génome muté (*FluxSimulator* génère des *reads* à partir du fichier *GTF* du génome muté). Or, nous allons positionner les *reads* sur le génome de référence, et par conséquent nous voulons le fichier *BED* de celui-ci.

Une routine permettant de passer d'un fichier *BED* (muté) vers un fichier *NotMutatedBED* (non muté) a été conçue, ce qui nous rajoute l'information supplémentaire :

iv/ un fichier au format *NotMutatedBED* permettant de distinguer les *reads* contenant une jonction d'épissage, ainsi que toutes les informations relatives à cette jonction sur le génome de référence ;

En résumé, nous avons la possibilité avec ce pipeline de simulation de mesurer précisément la *sensibilité* et la *précision* des outils d'analyse de *reads* RNA-Seq.

5.3.2 Calcul des scores pour différencier erreurs et causes biologiques

Dans la définition 5.8, nous précisons un seuil permettant de distinguer les erreurs de séquences des phénomènes biologiques. À partir de la génération de données simulées (section 5.3.1), nous pouvons définir ce seuil.

Nous connaissons, à partir des données simulées, les *reads* affectés par des erreurs et ceux affectés par une cause biologique. Notons qu'un *read* peut contenir plusieurs erreurs, plusieurs causes biologiques, ou encore une combinaison d'erreur(s) et de cause(s) biologique(s). Néanmoins, nous avons vu précédemment (section 5.2.7) qu'une série d'événements engendrait une série de *breaks* ou un *break* plus long.

En analysant chaque *break*, nous pouvons différencier les erreurs des causes biologiques. D'après le corollaire 5.9, une erreur se distingue par une *chute* du *support* dans un *break*. Puis, d'après la remarque 11, cette *chute* s'évalue à l'aide des deux moyennes $P_i\widehat{\text{-}support}$ et $P_e\widehat{\text{-}support}$.

La figure 5.7 (A) représente une distribution des *breaks* en fonction de $P_i\widehat{\text{-}support}$ et $P_e\widehat{\text{-}support}$

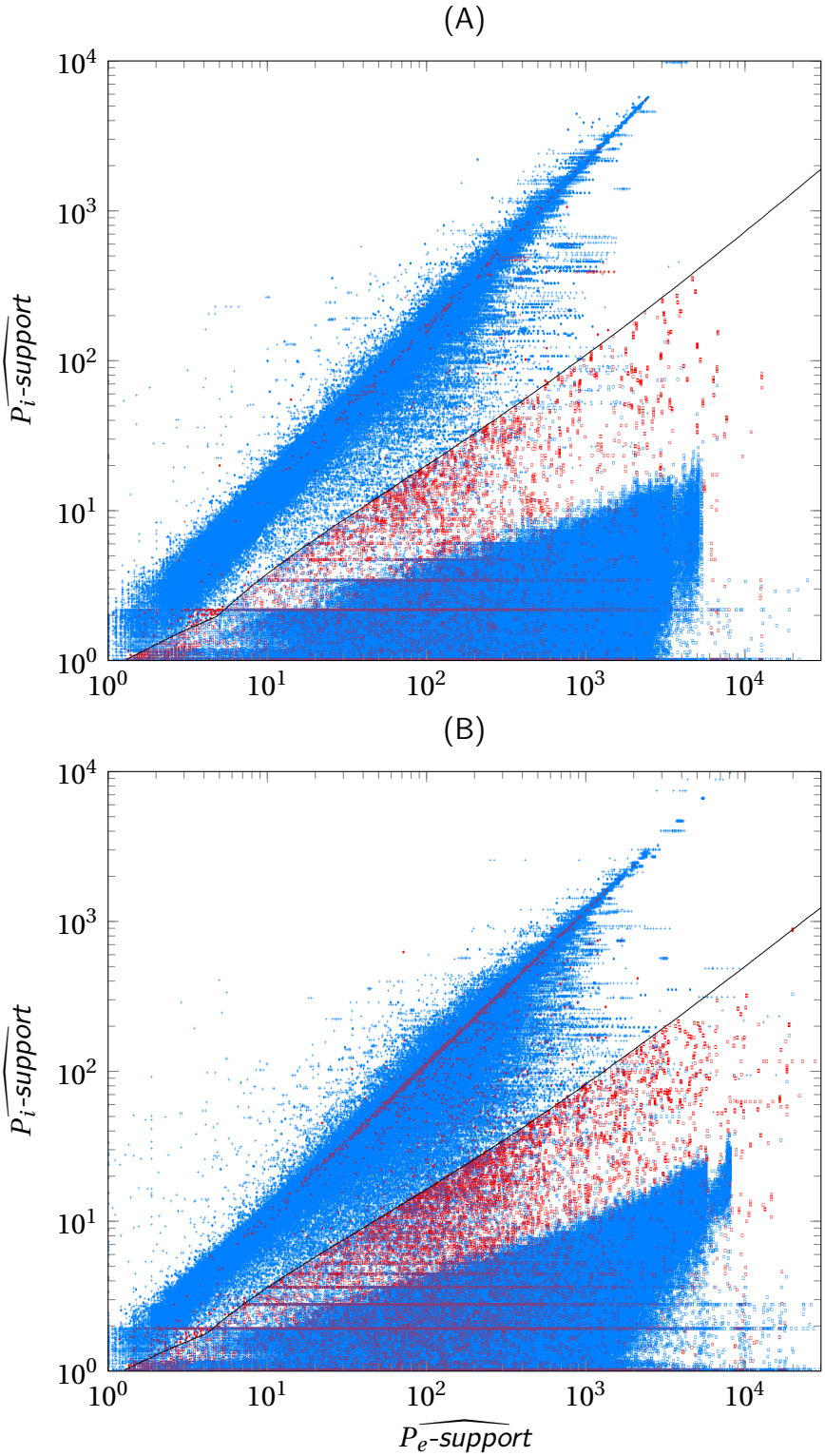
pour des données de 75*pb* et la figure 5.7 (B) présente les mêmes résultats pour des données de 200*pb*. Nous voulons séparer chaque distribution en deux classes : les *breaks* correspondant à des erreurs de séquences et ceux correspondant à des causes biologiques. Sachant que nous connaissons à l'avance les points qui composent chaque classe, nous pouvons séparer ces deux classes à l'aide d'une technique de discrimination. En utilisant dans un premier temps QtiPlot (http://soft.proindependent.com/qtiplot.html) et ensuite une méthode du type *Support Vector Machine* (SVM) implantée dans R (http://www.duclert.org/Aide-memoire-R/Apprentissage/SVM.php), nous avons pu modéliser la frontière entre ces deux classes [Crammer et Singer, 2001]. Cette frontière est représentée par une courbe noire dans la figure 5.7 (A) et la figure 5.7 (B). Par définition de la *chute* du *support* (définition 5.8), les erreurs de séquences se retrouvent en dessous de la courbe et les causes biologiques au dessus. Les points bleus sont les évènements bien classés et les points rouges représentent ceux qui sont mal classés.

Nous visualisons la formation de deux nuages distincts : un de chaque côté de la courbe (pour les deux figures). Entre ces deux nuages, un mélange de points bleus et de points rouges forment une zone ambiguë : les erreurs et les causes biologiques ne sont plus distinguables. Autrement dit, nous ne pouvons pas placer une frontière capable de séparer exclusivement les erreurs des causes biologiques.

Nous aurions pu placer une frontière en plein milieu de cette zone, dite ambiguë, afin d'avoir une équirépartition des faux positifs (points rouges) de chaque côté de la courbe. Cependant, notre but étant de prédire le maximum d'événements biologiques avec précision, nous avons préféré bien séparer les erreurs, quitte à classer quelques causes biologiques parmi les erreurs (points rouges en dessous de la courbe de la figure 5.7 (A) et la figure 5.7 (B)). Notons que cette frontière reste valide malgré une évolution de la taille des *reads* (de 75*pb* à 200*pb*).

Afin de donner un certain degré de validation, nous établissons un score pour chaque événement prédit. Ce score est calculé en fonction de la distance verticale de chaque point par rapport à la frontière. En effet, un point proche de la frontière a moins de chance d'être bien classé qu'un point très éloigné. Prenons par exemple un *read* qui contient un *break* proche de la frontière et assimilé à un SNV avec $P_e\widehat{\text{-}support} = 2,6$ et $P_i\widehat{\text{-}support} = 2$. En projetant verticalement sur la courbe, nous déduisons un score faible de 0,43. En revanche, si nous prenons un SNV très éloigné de la frontière avec un $P_e\widehat{\text{-}support} = 26$ et $P_i\widehat{\text{-}support} = 25$, nous calculons un score de 23,85. Il n'y a aucun doute sur le deuxième cas alors que pour le premier nous observons une chute faible de 0,6 mais le SNV est très peu exprimé (*support* de 2 dans le *break*).

FIGURE 5.7 : Calcul du seuil pour différencier erreurs et causes biologiques.



Cette illustration représente deux graphiques, avec en abscisse $\widehat{P_e\text{-support}}$ et en ordonnée $\widehat{P_i\text{-support}}$. À partir des données simulées, nous savons quels *reads* sont affectés par une erreur et quels *reads* sont affectés par un événement biologique. Lorsqu'une erreur est détectée, il y a une *chute* du *support* à l'intérieur du *break* d'un *read*. Autrement dit, $\widehat{P_i\text{-support}}$ est faible par rapport à $\widehat{P_e\text{-support}}$. La courbe noire est la frontière qui sépare les erreurs des causes biologiques en fonction de $\widehat{P_i\text{-support}}$ et $\widehat{P_e\text{-support}}$. Les causes biologiques sont représentées par les points bleus au dessus de la courbe et les erreurs par les points bleus en dessous. Les points rouges au dessus de la courbe représentent les erreurs classées comme cause biologique et ceux en dessous sont les causes biologiques classées comme erreur. La figure (A) représente le jeu de données *simulatedHuman75bp-42M* et la figure (B) *simulatedHuman200bp-48M* (voir section 5.4.1).

5.3.3 Traitement plus strict de CRAC pour les chimères

La détection des chimères est très complexe. Nous avons vu dans la section 5.2.6, qu'un *break* chimérique peut être la conséquence de fausses localisations.

Afin de maximiser nos chances de prédire des chimères valides, nous avons mis en place un traitement plus strict. Il se fait après une analyse de *reads* standard. Ce traitement consiste à filtrer les chimères les plus ambiguës. Les chimères conservées vérifient ces quatre critères :

1. En partant du *break* qui correspond à la chimère sur le *read*, il doit exister, à gauche et à droite de ce *break*, au moins un k-mer localisé de manière unique sur le génome.
2. Le plus petit *support* des k-mers à l'intérieur du *break* doit être plus grand qu'un seuil α donné en paramètre de CRAC ($\alpha = 5$ par défaut).
3. La chimère ne doit pas être le résultat d'une *fusion* (cf. algorithme 8). Autrement dit, il ne doit pas y avoir de fausses localisations à l'intérieur du *break* de la chimère.
4. Le *break* doit être plus grand que $k - 1 - \beta$ où β est donné en paramètre de CRAC ($\beta = 3$ par défaut).

5.4 Matériels

Nous avons généré plusieurs types de données simulées afin d'évaluer d'un côté, la robustesse de CRAC par rapport aux autres outils et d'un autre côté, sa versatilité et sa capacité à analyser les données du futur. Nous avons produit des données actuelles ($75pb$) et des données plus longues ($200pb$), des données peu couvertes (11M de *reads*) et des grands ensembles de données (48M de *reads*). Nous avons aussi généré des données sur deux types d'espèces : la drosophile et l'humain.

Même si nos données simulées sont proches de la réalité, elles sont produites à partir d'un fichier d'annotation *GTF* et par conséquent ne criblent qu'une partir du génome. C'est pourquoi, nous avons aussi utilisé des données réelles de RNA-Seq, avec un jeu de données orienté et un autre non-orienté.

5.4.1 Les données simulées

À partir du pipeline de simulation présenté dans la section 5.3.1, nous avons généré des données à partir du génome humain (hg19) et de la drosophile (dmel5).

Avec *GenomeSimulator*, pour *H. sapiens* (hg19) et *D. melanogaster* (dmel5), nous avons généré un taux de 0,1 % de substitutions, 0,01 % d'insertions, 0,01 % de délétions. Un gène donné a 0,01 % de chance d'être impliqué dans une chimère.

Concernant *FluxSimulator*, nous avons choisi de produire des *reads* courts (75 *pb*) et des *reads* plus longs (200 *pb*) comme ils le seront dans le futur. Pour le modèle d'erreur spécifié dans *FluxSimulator*, nous utilisons un modèle basé sur une observation empirique de données du séquen-

ceur Illumina GAIIx [Aury *et al.*, 2008]. Pour donner un ordre de grandeur, 25 % des *reads* courts contiennent au moins une erreur alors que plus de 50 % des *reads* longs en contiennent au moins une.

Par la suite, nous considérons un SNV comme l'altération d'un seul nucléotide dans la séquence (voir section 1.4 du chapitre 1). Les tableaux ci-dessous récapitulent les différentes mutations générées et séquencées par *FluxSimulator* pour hg19 (TABLE 5.1) et dmel5 (TABLE A.1 en annexe).

Type	SNV	Insertion	Deletion	Chimera
Genome-wide	3,139,937	287,336	287,502	1,002
Sequenced (75bp)	30,892	2,867	2,898	647
Sequenced (200bp)	61,387	5,562	5,610	914

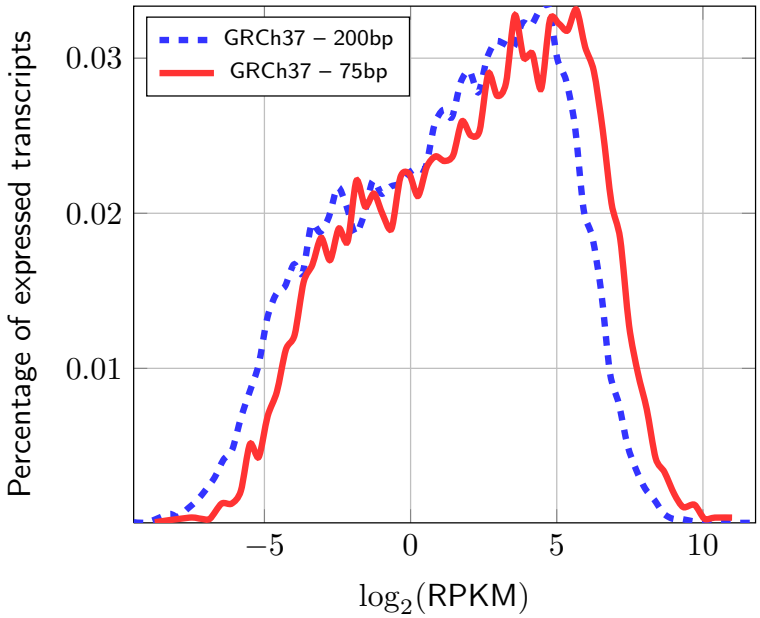
TABLE 5.1 : Nombre de mutations générées aléatoirement et réellement séquencées sur le génome de *H. sapiens*.

Au final, nous avons généré 11 et 42 millions de *reads* de 75*pb*, puis 48 millions de 200*pb* pour *H. sapiens*. Nous appelons ces jeux de données : *simulatedHuman75bp-11M*, *simulatedHuman75bp-42M*, *simulatedHuman200bp-48M*.
Nous avons aussi produit 45 millions de *reads* de 75*pb* et 48 millions de 200*pb* pour *D. melanogaster* : *simulatedDroso75bp-45M*, *simulatedDroso200bp-48M*. Pour tous les jeux de données, nous avons retiré tous les *reads* provenant de la queue polyA des transcrits.

Nous avons paramétré le niveau d'expression de transcription dans *FluxSimulator* pour générer des données de 75 *pb* et 200 *pb*. Nous calculons les niveaux respectifs en mesurant le RPKM de chaque transcrit et en analysant la distribution des RPKM (voir section 2.6.1 du chapitre 2). La figure 5.8 évalue la distribution du RPKM de *simulatedHuman75bp-42M* et *simulatedHuman200bp-48M* parmi les transcrits exprimés du fichier *GTF* de *FluxSimulator*.

Nous remarquons que, pour nos données simulées, cette distribution (à l'échelle logarithmique) a le même comportement qu'une expérience réelle [Mortazavi *et al.*, 2008]. Notons que 50 % des transcrits du fichier d'annotations *GTF* ne sont pas exprimés dans le jeu de données de 75*pb* alors que seulement 0,3 % des transcrits ne sont pas exprimés dans le jeu de 200*pb*.

FIGURE 5.8 : Distribution de $\log_2$(RPKM) pour les *reads* provenant de hg19.



Cette figure illustre la distribution des *reads* par transcrit par kilo et par million pour *simulatedHuman75bp-42M* (en rouge) et *simulatedHuman200bp-48M* (en bleue).

5.4.2 Les données réelles

Concernant les données réelles, nous en utilisons deux types :

1. Des données cancéreuses d'une lignée de K562. Nous avons à notre disposition 3 expériences de 12 – 13 millions de *reads pairés* de 2X75*pb*, soit un total de 35 millions ou 70 millions de *reads* de 75*pb* en combinant les paires. Les données sont disponibles sur RGASP (numéro d'accession *GM12878* sur www.sanger.ac.uk/PostGenomics/encode/RGASP.html avec la permission de B. Wold). Les *reads* proviennent de RNA-Seq non-orienté et sont séquencés avec Illumina®. Nous appelons ce jeu de données K562-75bp-70M.
2. Le jeu de données ERR030856 contenant un mélange de 16 tissus humains. Ce jeu contient un total de 75 millions de *reads* de 100*pb* séquencés avec le séquenceur *HiSeq 2000* de Illumina® (http://trace.ddbj.nig.ac.jp/DRASearch/experiment?acc=ERX011226, http://www.ebi.ac.uk/arrayexpress/experiments/E-MTAB-513). Les *reads* proviennent de RNA-Seq orienté. L'analyse sous-jacente est une comparaison de profils transcriptomiques par haut débit sur des tissus sains de plusieurs individus. Nous appelons ce jeu de données ERR030856-100bp-75M.

5.4.3 Les logiciels utilisés pour les comparaisons

Les différents outils de *mapping* et de *splicing* utilisés sont définis dans la table 5.2 avec leurs paramètres. Tous ces outils sont récents : de Bowtie sorti en mars 2009, jusqu'à GASSST sorti en septembre 2010.

Nous avons essayé d'utiliser la version la plus récente de chaque outil avec les paramètres qui semblaient les mieux adaptés en fonction du nombre et de la longueur des *reads*, ainsi que le génome de l'espèce de référence sur lequel nous analysons les *reads*.

Tool	Version	Parameters
Bowtie	0.12.7	`--best -n 3 -y -t -k 2`
BWA	0.5.9	
BWASW	0.5.9	`-b 5 -q2 -r1 -z10`
GASSST (short)	1.27	`-w 18 -p 94 -s 3 -g 4`
GASSST (long)	1.27	`-w 18 -p 89 -s 3 -g 9`
GSNAP	2011-03-28	`-N 1`
MapSplice	1.15	`--fusion -L 22`
Samtools	0.1.12a	
SOAP2	2.20	
TopHat	1.2.0	

TABLE 5.2 : Paramètres utilisés pour les outils de *mapping* et de *splicing*.

5.5 Résultats

Dans cette partie, nous comparons CRAC aux autres outils de *mapping* et d'analyse. Rappelons que le but du *mapping* est de trouver la position génomique d'origine des *reads*, alors que celui des outils d'analyse est de retrouver tous les événements biologiques qui différencient le génome identifié par les *reads* RNA-Seq du génome de référence.

Notons que mis à part GSNAP, il n'existe pas réellement d'outils de *mapping* vraiment adaptés au traitement du RNA-Seq (MapSplice étant essentiellement un outil de *splicing*). Par conséquent, il est logique qu'ils aient de moins bons résultats mais cela ne présume pas de leur performance sur des données génomiques.

5.5.1 Comparaisons sur les données simulées

Nous avons développé un pipeline de simulation du RNA-Seq afin de mesurer une *sensibilité* et une *précision* proche de la réalité (cf. 5.3.1). À l'aide de ces deux mesures, nous évaluons la capacité et fiabilité intrinsèques de CRAC à prédire certaines causes, puis nous comparons ce dernier avec les autres outils de *mapping* et d'analyse.

5.5.1.1 Le *mapping*

Les outils de *mapping* distinguent trois catégories de localisation de *reads* : uniques, multiples et non-localisés. Comme nous pouvons le voir dans la figure 5.5, CRAC distingue une catégorie supplémentaire : les *reads* dupliqués. Nous regroupons cette catégorie avec les multiples pour se comparer avec les autres outils. L'idée est de concentrer sur les localisations uniques, c'est-à-dire les *reads* qui se localisent sans ambiguïté possible sur le génome.

Avec notre pipeline de simulation (section 5.3.1), nous connaissons la « vraie » localisation des *reads* sur le génome. Par conséquent, nous pouvons déterminer pour chaque outil de *mapping*, la *sensibilité* : pourcentage des *reads* correctement localisés sur le génome, parmi tous les *reads* ; et la *précision* : le pourcentage des *reads* qui sont correctement localisés parmi les *reads* localisés par l'outil, c'est-à-dire ceux qui donnent la même position que celle indiquée dans le fichier *BED* de *FluxSimulator* avec une tolérance de ± 5 *pb*. Lorsque le *read* est épissé, nous considérons que sa localisation est correcte si elle correspond à un des exons. Rappelons que la liste des outils et les paramètres utilisés sont disponibles en sous-section 5.4.3 et table 5.2. Nous calculons la *sensibilité* et la *précision* pour tous les outils dans la figure 5.9 pour l'humain et dans la figure A.1 (en annexe) pour la drosophile.

Analyse du RNA-Seq. BWA, SOAP2, Bowtie et GASSST ne sont pas des outils conçus pour identifier des jonctions d'épissage, nous comparer à eux sur du RNA-Seq peut paraître injuste. Nous les avons quand même choisi pour plusieurs raisons : i/ Bowtie est l'un des outils de *mapping* les plus

connus, il est incontournable de savoir que, malgré tout, il n'est peut être pas le meilleur sur tous les types de données. ii/ CRAC est un outil basé sur une structure d'indexation compressée (BWT), il est intéressant de le comparer à des méthodes proches (Bowtie, BWA, SOAP2). iii/ Mais finalement, il est aussi judicieux de le comparer à des méthodes éloignées, utilisant des graines (GASSST, GSNAP).

La figure 5.9 mesure la *sensibilité* et la *précision* du *mapping* chez l'humain pour les trois jeux de données : *simulatedHuman75bp-11M* (A), *simulatedHuman75bp-42M* (B), *simulatedHuman200bp-48M* (C). La figure 5.10 mesure, pour les jeux de données (B) et (C), les *reads* qui sont localisés correctement dans chaque catégorie : i/ ± 5 *pb* par rapport au fichier *BED* de *FluxSimulator* ; ii/ catégorie indiquée par le fichier *INFO* de *FluxSimulator*.

Pour des *reads* de 75 *pb*, nous observons dans la figure 5.9 (B) et la figure A.1 (A) un écart de *sensibilité* entre les outils de *mapping*. Nous retrouvons d'un côté, CRAC et GSNAP avec une *sensibilité* proche des 95 %, de l'autre BWA, SOAP2, Bowtie et GASSST avec une *sensibilité* variant de 70 % à 79 %. La tendance est confirmée dans la figure 5.10 (B) où CRAC et GSNAP localisent jusqu'à 4 fois plus de *reads* que BWA et GASSST et 8 fois plus que Bowtie et SOAP2 lorsqu'ils sont affectés par un indel, une jonction d'épissage ou une chimère.

Rappelons que SOAP2 et Bowtie ne tolèrent que des substitutions entre *reads* et génome, ils localisent seulement un indel, une jonction ou une chimère lorsque la cause se retrouve sur le bord d'un *read*, de la même façon que BWA et GASSST ne sont pas censés détecter les splices et les chimères. Néanmoins, ces quatre outils sont aussi moins *sensibles* dans la détection des SNV et des erreurs de séquences ($\simeq$ 94 % pour CRAC et GSNAP contre moins de 75 % en moyenne). Cela peut s'expliquer lorsqu'un *read* contient, à la fois, une jonction et un SNV ou une erreur. D'après le tableau 5.9 (A), il y a $\simeq 8,6$M de jonctions d'épissage pour 42M de *reads*, soit 1 jonction tous les 5 *reads*, nous pouvons penser qu'une erreur de séquences ou un SNV puisse se glisser assez souvent dans un *read* contenant déjà une jonction.

Sur des données actuelles de 75 *pb*, tous les outils sont spécifiques à plus de > 99 % : un minimum de 99,13 % pour BWA et un maximum de 99,88 % pour GSNAP sur la figure 5.9 (B).

Premièrement, tous les outils de *mapping* utilisés dans cette analyse sont *précis*. Ils produisent des localisations sûres que nous pouvons réutiliser lors d'une analyse biologique sans risquer d'intégrer des erreurs.

Ensuite, nous montrons que CRAC et GSNAP sont plus *sensibles* car ils sont capables de détecter les jonctions d'épissage. Par conséquent, tous les *reads* contenant une mutation ponctuelle ou courte (SNV, indel, erreur) en plus d'une jonction d'épissage ne sont pas localisés. Or, dans une expérience de RNA-Seq, un *read* contient souvent une jonction d'épissage. Alors, par ce manque de *précision*, les outils non adaptés au RNA-Seq risquent de manquer de nombreuses causes telles que des SNV, des indels, ou encore des erreurs.

Analyse de données du futur. La plupart des outils (y compris GSNAP) ne sont pas adaptés à des données plus longues, que ce soit sur l'humain (FIGURE 5.9 (C)) ou sur la drosophile (FIGURE A.1 (B)). Alors que CRAC conserve une *sensibilité* de 96,10 % sur l'humain, GSNAP plafonne à 84.69 %, BWA-SW à 68,66 % et les autres aux alentours de 55 %.

D'ailleurs, nous remarquons dans la figure 5.10 (C) que GSNAP est perturbé particulièrement de cette élongation sur la longueur du *read*. Alors que CRAC, GASSST et BWASW gagnent en localisation dans la quasi totalité des causes, GSNAP en localise moins partout avec une perte qui varie de 10 à 30 points selon les causes.

Quant à la *précision* sur des données de 200*pb*, presque tous les outils sont toujours très spécifiques avec un minimum de 96,86 % de *précision* pour BWASW (qui remplace BWA sur des *reads* plus longs) et un maximum de 99,92 % pour CRAC. Seul BWASW semble être moins *précis*, avec notamment plus de 3 % d'*imprécision* chez l'humain (figure 5.9 (C)) et presque 3 % chez la drosophile (figure A.1 (B)).

GSNAP versus CRAC. Dans la figure 5.9, nous analysons qu'avec de plus en plus de données, CRAC s'améliore en *sensibilité* : 93,55 % dans (A) contre 94,51 % dans (B). Il s'améliore aussi avec des données plus longues : 94,51 % pour (B) contre 96,10 pour (C). Puis dans tous les cas, il n'y a pas de perte de *précision* pour CRAC : 99,74 (A), 99.72 (B) et 99,92 (C) chez l'humain et dans la figure A.1, une *précision* de 99,99 (A) et 99,99 (B) chez la drosophile. À l'inverse, GSNAP perd de la *précision* et de la *sensibilité* lorsque les données ou les longueurs augmentent : une *sensibilité* qui passe de 93,67 % (A) à 84,69 % et une *précision* qui passe de 99,91 % (A) à 99,33 % (C) dans la figure 5.9.

Dans la figure 5.10 (B), nous remarquons que CRAC localise plus de *reads* que GSNAP dans toutes les catégories sauf pour les erreurs. Pourtant, pour ce même jeu de données (*simulatedHuman75bp-42M*), nous avons calculé une *sensibilité* équivalente pour GSNAP et CRAC dans la figure 5.9) (B). Cela vient du fait que les *reads* ne sont pas équiréparties dans les catégories. Dans le tableau 5.10 (A), représentant le nombre de *reads* affectés par une cause, nous remarquons, par exemple, qu'il y a 12836882 *reads* affectés par une erreur alors qu'il n'y que 314572 *reads* affectés par une insertion biologique, soit 40 fois moins. En fait, GSNAP détecte 94,24 % des *reads* contenant une erreur contre 93,88 pour CRAC : il localise 46213 *reads* en plus de CRAC dans cette catégorie. CRAC comble cette différence dans les autres catégories.

Ainsi, le nombre global de localisation occulte le fait que GSNAP détecte mieux les erreurs que CRAC mais moins de jonctions, d'indels, de SNV et de chimères. Dans la section 5.3.2, nous exposons notre choix de privilégier les causes biologiques, ce qui explique ce résultat.

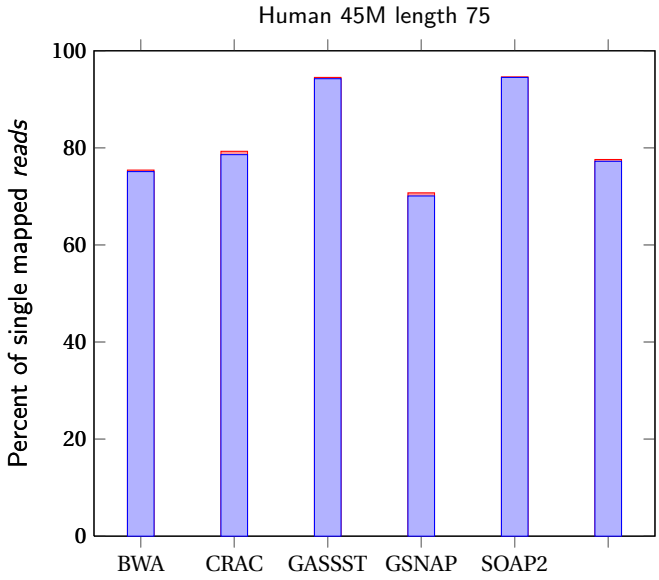
Nous voyons dans la figure 5.10 (B) et figure 5.10 (C) que CRAC localise en moyenne 95 % des *reads* quelque soit la cause et la longueur des *reads*, c'est-à-dire de manière homogène et sans perte entre 75 *pb* et 200 *pb*. Au contraire, GSNAP perd beaucoup de localisation entre (B) et (C), il passe derrière CRAC pour la détection des erreurs (84,36 % contre 95,99 % pour CRAC). D'ailleurs, BWASW s'en tire mieux que lui pour les insertions à 200 *pb* (69,84 % contre 50,67 % pour GSNAP)

et aussi bien sur les délétions. Malgré un algorithme basé sur le même principe de *seed and extend*, nous pouvons noter qu'entre des *reads* de 75 *pb* et 200 *pb*, GASSST gagne en *sensibilité* sur certains points tels que les indels alors que GSNAP perd dans toutes les catégories.

Notons qu'il y a quasiment autant de *reads* dans *simulatedHuman75bp-42M* et *simulatedHuman200bp-48M* alors que dans le tableau 5.10 (A), le nombre de *reads* par cause triple de volume. En effet, du fait que les *reads* soient trois fois plus longs, il y a plus de chance que les causes s'enchaînent dans un même *read*. Nos procédures de *fusion* et d'*extension* décrites en section 5.2.6 semblent donc bien fonctionner.

De plus, à comparaison égale, CRAC est plus *sensible* que GSNAP pour des données de 75 *pb*, et ce, pour toutes les causes biologiques (SNV, indels, splices, chimères) mais il est moins sensible pour les erreurs. Cependant, notre but est de détecter précisément le maximum d'évènements biologiques et non pas de détecter les erreurs (voir section 5.3.2).

FIGURE 5.9 : *Spécificité* et *précision* du *mapping* pour les données simulées de hg19.

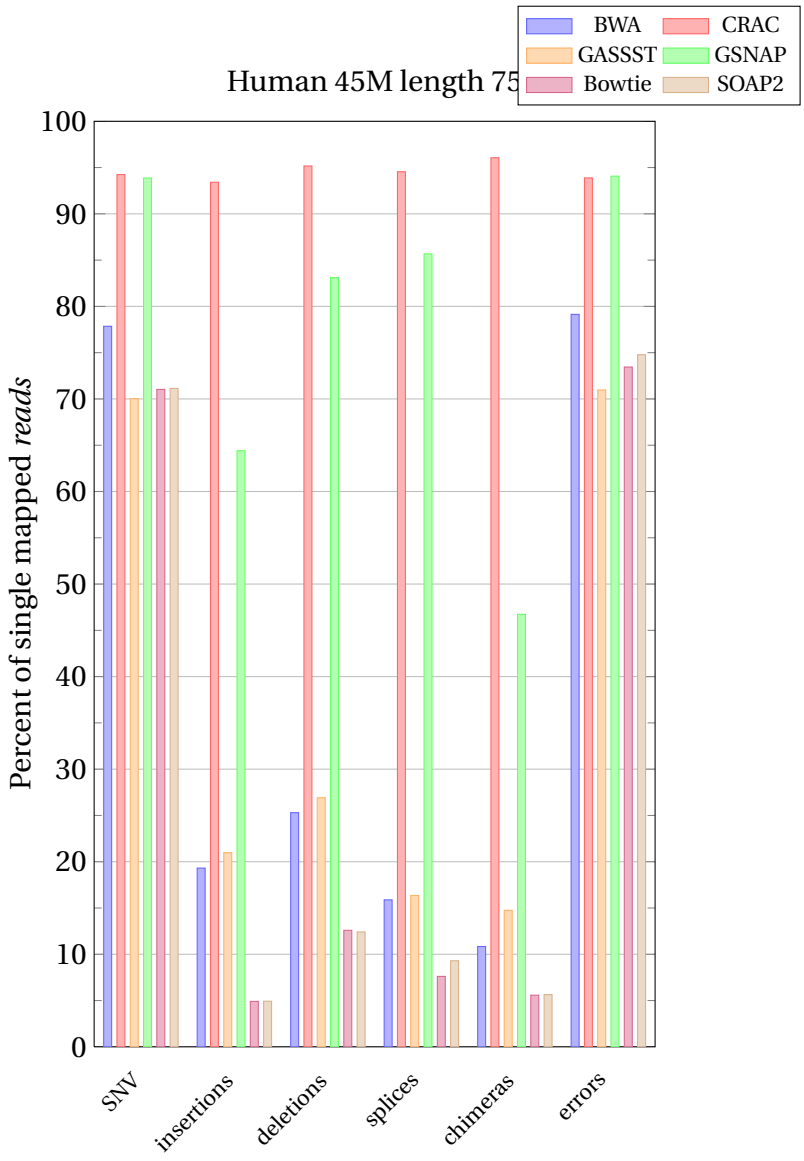


Calcul de la *sensibilité* et de la *précision* du *mapping* pour *simulatedHuman75bp-11M* (A) *simulatedHuman75bp-42M* (B) et *simulatedHuman200bp-48M* (C). La barre bleue correspond au pourcentage de *reads* qui sont correctement localisés parmi ceux qui sont réellement localisés par l'outil (barre rouge). La différence entre barre rouge et barre bleue correspond au pourcentage de fausses localisations données par l'outil.

FIGURE 5.10 : *reads* localisés par catégorie pour les données simulées de hg19.

(A)

category	# *simulatedHuman75bp-42M*	# *simulatedHuman200bp-48M*
SNV	3090951	9074884
insertions	314572	851343
deletions	330228	864634
splices	8567917	25104015
chimeras	180310	696061
errors	12836882	38840045



Calcul du pourcentage de *reads* qui sont localisés de façon unique dans chaque catégorie (SNV, insertions, délétions, *splices*, chimères, erreurs) pour les données *simulatedHuman75bp-42M* (B) et *simulatedHuman200bp-48M* (C). Il y a une barre d'une couleur différente pour chacun des outils. Notons que pour les données de 75*pb* (B), *BWA* est utilisé alors que pour celles de 200*pb* (C) c'est *BWA-SW* qui est utilisé. Le tableau (A) correspond au nombre de *reads* impliqués dans une catégorie.

5.5.1.2 Le *splicing*

Rappelons que le *splicing* est l'action qui consiste à détecter les jonctions d'épissage dans les *reads*. Les outils capables de les détecter fournissent généralement leurs résultats dans un fichier au format *BED* indiquant les localisations de part et d'autre de la jonction, c'est-à-dire la localisation de la fin du premier exon et la localisation du début du deuxième exon identifiés dans le *read*.

Dans le fichier *NotMutatedBed* du pipeline de simulation (cf section 5.3.1.3), nous avons connaissance de ces informations pour tous les *reads* qui contiennent une jonction d'épissage. C'est de cette façon que nous mesurons la *sensibilité* et la *précision* des outils de *splicing*. Nous autorisons une tolérance de $\pm 3 pb$ pour les deux localisations de chaque jonction.

Nous mesurons la *sensibilité* et la *précision* des outils de *splicing* parmi les plus utilisés [Garber *et al.*, 2011]. Nous avons fait l'expérience sur *simulatedHuman75bp-42M* et *simulatedHuman200bp-48M* dans la table 5.3 et sur *simulatedDroso75bp-45M* et *simulatedDroso200bp-48M* dans la table A.2 (en annexe).

TABLE 5.3 : *Sensibilité* et *précision* du *splicing* pour les données simulées de hg19.

(A)	CRAC		GSNAP		MapSplice		TopHat	
	%	nb	%	nb	%	nb	%	nb
sensibilité	79.00	67012	83.55	70865	79.38	67334	84.34	71540
précision	99.61	263	97.74	1656	97.77	1548	91.82	6581

(B)	CRAC		GSNAP		MapSplice		TopHat	
	%	nb	%	nb	%	nb	%	nb
sensibilité	85.15	124249	69.14	100885	82.94	121028	53.11	77494
précision	99.53	591	98.99	1050	99.01	1242	95.25	3956

Calcul de la *sensibilité* et de la *précision* du *splicing* pour *simulatedHuman75bp-42M* (A) et *simulatedHuman200bp-48M* (B) et comparaison entre les différents outils de *splicing*.

Détecter précisément les jonctions d'épissage. Pour des données de 75 *pb*, nous pouvons voir dans la table 5.3 (A) et la table A.2 (A) (en annexe) que TopHat est beaucoup moins précis que ses concurrents : sa *précision* est de 6 à 8 points inférieure à celles des autres pour *simulatedHuman75bp-42M* (91,82 % contre 97,74 % pour GSNAP et 99,61 % pour CRAC), puis sa précision baisse de 2 points sur *simulatedDroso75bp-45M* (97,38 % contre 99,76 % pour CRAC). Même si en terme de pourcentage ce n'est pas évident, CRAC est lui même jusqu'à 6 fois plus précis que MapSplice et GSNAP : avec seulement 263 fausses jonctions prédites, il devance considérablement ses deux concurrents qui génèrent respectivement 1548 et 1656 de fausses jonctions (TABLE 5.3 (A)).

Pour des données de 200 *pb*, nous remarquons dans la table 5.3 une nette progression de la *précision* entre (A) et (B) de 4 points pour TopHat (95,25 % contre 91,82 % sur *simulatedHuman75bp-42M*). Cependant, il reste toujours moins précis que les autres avec notamment 99,53 % de *précision* pour CRAC. Là encore, même si au niveau des pourcentages ce n'est pas manifeste, CRAC ne génère que 591 fausses jonctions contre 1 050 et 1 242 pour GSNAP et MapSplice, c'est-à-dire deux fois plus de faux positifs.

Quel que soit le type de données (humain ou drosophile) et la longueur des données (75 *pb* ou 200 *pb*), CRAC garde une *précision* > 99,5 % alors que les autres outils ont une *précision* plus basse.

Détecter le maximum de jonctions d'épissage. Pour *simulatedHuman75bp-42M*, nous avons vu dans la figure 5.10 (B) que CRAC localisait plus de *reads* épissés que GSNAP (94,55 % contre 85,68 % pour GSNAP). Pourtant la table 5.3 (A) montre le contraire : GSNAP trouve plus de jonctions d'épissage que CRAC (83.55 % contre 79 % pour CRAC). Pour comprendre ce résultat, il faut savoir que plusieurs *reads* peuvent correspondre à une même jonction. Dans la section 5.2.4.3, nous abordons une faiblesse de l'algorithme de CRAC : l'incapacité à détecter précisément une cause lorsque celle-ci est située sur le bord d'un *read* (remarque **R2**). Ainsi, si pour une jonction donnée, aucun *read* ne la contient dans son milieu, elle sera classée dans *bio undetermined* (FIGURE 5.5). Cependant, nous remarquons qu'avec des données plus longues, nous avons moins de chance de manquer une jonction : alors que nous perdons 4 points sur GSNAP pour *simulatedHuman75bp-42M*, nous gagnons 16 points sur lui pour *simulatedHuman200bp-48M* (85.15 % pour CRAC contre 69.14 % pour GSNAP) avec un jeu de données qui contient quasiment le même nombre de *reads* (TABLE 5.3).

De manière générale, tous les outils sont performants pour des données de 75 *pb* avec une *sensibilité* qui varie de 80 à 85 % chez la drosophile et l'humain (TABLE 5.3 (A)) et A.2 (A). C'est en passant au traitement de données plus longues (200 *pb*) que les outils ne se comportent pas de la même façon. D'un côté, MapSplice et CRAC continuent de progresser avec une *sensibilité* qui passe de 79,38 % à 82,94 % pour le premier et de 79,00 % à 85,15 % pour le deuxième. De l'autre, GSNAP et TopHat voient leur *sensibilité* s'effondrer de 83,55 % à 69,14 % pour le premier et de 84,34 à 53,11 % pour le second (TABLE 5.3 (B)).
La tendance est d'autant plus marquante en terme de nombres de jonctions. Alors que GSNAP et TopHat détectent entre 3 000 et 4 000 (ou 5 %) de jonctions en plus que MapSplice et CRAC sur des *reads* de 75 *pb* (TABLE 5.3 (A)), ils en produisent entre 25 000 et 50 000 (ou entre 20 % et 40 %) en moins que CRAC (TABLE 5.3 (B)).

Dans la table A.2, nous pouvons observer que GSNAP est moins bon que CRAC, MapSplice et TophHat en *sensibilité* : 77 % contre 86 % pour TopHat à 75 *pb* et 76 % contre 89 % à 200 *pb* pour CRAC).

Quel que soit le type (drosophile ou humain) et la longueur des données (75 *pb* ou 200 *pb*), CRAC

et MapSplice gardent une *sensibilité* supérieure à 80 % alors que celle de GSNAP et MapSplice sont très affectés par le traitement de *reads* plus longs (inférieure à 55 % pour TopHat sur la table 5.3 (B)). De plus, CRAC est plus spécifique que tous ces concurrents, avec un maximum de 99,61 % sur l'humain (TABLE 5.3) et un maximum de 99,84 % sur la drosophile (TABLE A.2).

5.5.1.3 La détection des chimères

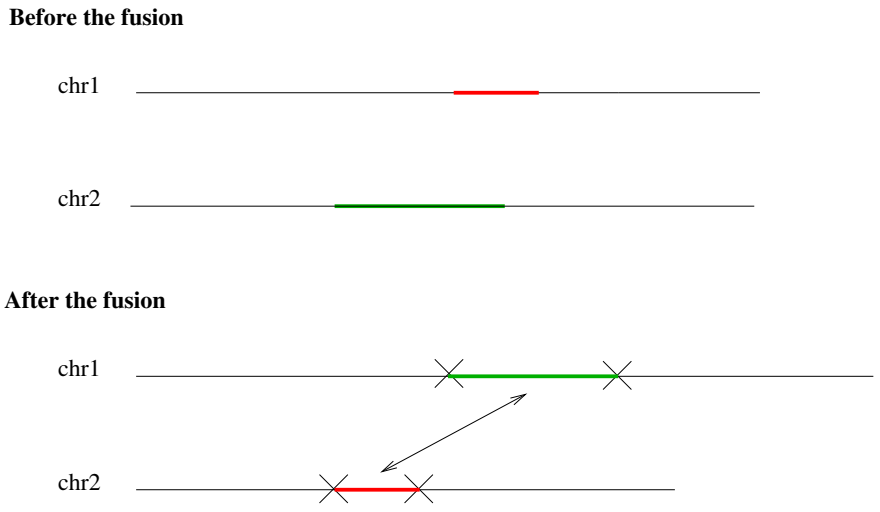
De la même façon que les jonctions classiques d'épissage, nous mesurons la *sensibilité* et la *précision* des chimères avec une tolérance de $\pm 3pb$.

Pour CRAC, nous considérons les chimères à deux moments : après l'analyse des *reads* et après un traitement plus strict (voir section 5.3.3). Le but de cette dernière approche est d'augmenter la *précision* de l'analyse quitte à perdre un peu de *sensibilité*.

Nous avons décidé de ne comparer CRAC qu'avec MapSplice plutôt que d'autres logiciels tels que FusionSeq ou FusionHunter [Sboner *et al.*, 2010; Li *et al.*, 2011] car la stratégie n'est pas la même. Ils analysent des chimères à partir de RNA-Seq pairé, ce qui ne permet pas de détecter le point de cassure précisément [McPherson *et al.*, 2011]. De plus, nous n'avons pas encore pris en charge la simulation de *paired*-reads dans notre pipeline (cf. section 5.3.1).

D'après la table 5.4 (A), nous pouvons voir que CRAC détecte 708 jonctions chimériques alors que d'après la table 5.1, nous avons généré 647 chimères dans *simulatedHuman75bp-42M* lors de la simulation. Cela vient du fait que pour une chimère, il y a 4 jonctions chimériques qui se créent (FIGURE 5.5.1.3).

FIGURE 5.11 : Chimère et jonctions chimériques



Dans cette figure, nous observons que la génération d'une chimère dans *GenomeSimulator* produit 4 jonctions chimériques.

MapSplice versus CRAC. Nous remarquons sans équivoque que MapSplice n'est pas adapté à la détection de chimère pour du RNA-Seq *single*-reads : il ne détecte que 27 jonctions chimériques (TABLE 5.4 (A)) parmi les 1 294 générées dans *simulatedHuman75bp-42M* (TABLE 5.1) en produisant un nombre considérable de fausses chimères (883 635) et il ne détecte que 48 chimères parmi les 914 générées dans *simulatedHuman200bp-48M* (TABLE 5.1) en produisant aussi un nombre colossal de faux positifs (466 023). Le constat est le même sur la drosophile (TABLE A.3).

Sans un traitement strict, CRAC est très sensible avec une détection de 61,14 % des jonctions chimériques pour *simulatedHuman75bp-42M* et 73,à5 % des pour *simulatedHuman200bp-48M* (TABLE 5.4), puis 78,45 % pour *simulatedDroso75bp-45M* et 71,02 % pour *simulatedDroso200bp-48M* (TABLE A.3). Il est aussi raisonnablement précis par rapport à MapSplice avec un minimum de 90 % de *précision* pour la drosophile et un minimum de 47,22 % de *précision* chez l'humain.

Après un traitement strict qui est uniquement appliqué sur des critères algorithmiques de CRAC (cf. sous-section 5.3.3), nous améliorons de façon significative la *précision* en ne perdant que légèrement de la *sensibilité* : nous atteignons plus de 90 % de *précision* chez l'humain tout en conservant au moins 50 % de *sensibilité* (TABLE 5.4) ; nous atteignons plus de 99 % de *précision* chez la drosophile tout au moins 62 % de *sensibilité* (TABLE A.3).

Notons que cette différence de *précision* entre humain et drosophile s'explique par le fait qu'il y a beaucoup plus de duplications chez l'humain augmentant les chances de générer des fausses localisations.

CRAC avec et sans des paramètres stricts. Dans la table 5.4, nous passons de 71,77 % à 93,31 % de *précision* sur *simulatedHuman75bp-42M* (un gain de 22 points) et de 47,22 à 90,85 % sur

TABLE 5.4 : *Sensibilité* et *précision* de la détection des chimères pour les données simulées de hg19.

(A)	CRAC		CRAC-strict		MapSplice	
	%	nb	%	nb	%	nb
sensibilité	61.14	708	46.80	542	2.33	27
précision	71.77	280	93.31	39	0	883 635

(B)	CRAC		CRAC-strict		MapSplice	
	%	nb	%	nb	%	nb
sensibilité	73.05	1 336	55.93	1 023	2.62	48
précision	47.22	1 540	90.85	105	0.05	466 023

Calcul de la *sensibilité* et de la *précision* lors de la détection des chimères pour *simulatedHuman75bp-42M* (A) et *simulatedHuman200bp-48M* (B) et comparaison avec MapSplice qui est capable de détecter des chimères sur du *single reads*.

simulatedHuman200bp-48M (un gain de 44 points). Dans la table A.3, nous passons de 96,96 % à 99,90 % sur *simulatedDroso75bp-45M* (un gain de 3 points) et de 90,33 % à 99,04 % sur *simulatedDroso200bp-48M* (un gain de 9 points).

Chez l'humain et la drosophile, nous remarquons que le gain est plus important sur des longs *reads* (200 *pb*) que sur des *reads* plus courts (75 *pb*). Lorsque la longueur des *reads* augmente, le nombre de causes dans un *read* augmente aussi. Il y a par conséquent plus de *reads* contenant plusieurs *breaks*. La procédure de *fusion* est plus complexe lorsqu'il faut fusionner plusieurs *breaks* chimériques (cf. section 5.2.6.2), il est donc logique d'avoir plus de faux positifs, surtout lors de la détection de *breaks* chimériques qui peuvent être confondus avec des fausses localisations (cf. section 5.2.6).

5.5.1.4 La classification de CRAC

Nous avons vu dans la figure 5.10 que CRAC détectait plus de 90 % des *reads* dans toutes les catégories, que les *reads* soient courts ou longs. Mais nous avons aussi remarqué que le nombre de *reads* n'est pas forcément représentatif du nombre de causes : 9 074 884 contiennent un SNV dans *simulatedHuman200bp-48M* (FIGURE 5.10 (A)) alors qu'il n'y a que 61 387 SVN de générés dans ce jeu de données (TABLE 5.1).

Nous savons qu'il n'est pas possible actuellement pour CRAC de détecter précisément une cause, si tous les *reads* contiennent cette cause sur leurs bords, c'est une conséquence de notre algorithme (cf. section 5.2). Dans cette section, il s'agit donc d'évaluer la capacité de CRAC à détecter une cause et de mesurer l'influence de la longueur du *read* sur cette détection. La figure 5.12 représente les causes sur les données simulées de l'humain (les résultats sur la drosophile étant meilleurs dans les sections précédentes, nous avons choisi de ne pas la représenter). Notons que la mesure des chimères est faite après traitement strict.

Nous pouvons tout d'abord remarquer dans la figure 5.12 (B) et la figure 5.12 (C), une *précision* de plus de 99,5 % pour les erreurs et les jonctions d'épissage, de plus de 98 % pour les délétions, de 97 % pour les SNV et les insertions, et de plus de 90 % pour les chimères. Au niveau de la *sensibilité*, nous mesurons plus de 80 % pour les jonctions, plus de 74 % pour les erreurs, une moyenne de 60 % pour les SNV et les indels, puis une moyenne de 50 % pour les chimères.

Regardons maintenant les données brutes du tableau 5.12 (A). Même si nous produisons 7 % de fausses jonctions chimériques pour *simulatedHuman75bp-42M*, cela n'en représente que 39 pendant que 542 sont détectées correctement. Nous sommes capables de détecter plus de 125 566 jonctions classiques pour *simulatedHuman200bp-48M* avec seulement 1 459 faux positifs.

Nous pouvons observer dans la figure 5.12 une progression de la *sensibilité* entre les données de 75 *pb* et les données de 200 *pb* alors qu'il y a pratiquement le même nombre de *reads* dans les deux jeux de données (42M contre 48M). Nous détectons, par exemple, plus de 85 % des jonctions

dans la figure 5.12 (C) alors que l'on n'en a détecté que 79 % dans la figure 5.12 (C) : soit une progression de 6 points. De même pour les chimères, nous observons une progression de plus de 8 points. La progression est moins forte pour les autres causes biologiques mais elle est en moyenne de 2 à 3 points. En revanche, nous perdons 3 points dans la détection des erreurs. Notons que cette catégorie n'est pas à la même échelle que les autres et nous détectons malgré tout presque 3 millions d'erreurs pour moins de 40 000 faux positifs. De plus, notre but n'est pas de détecter toutes les erreurs mais d'en intégrer le moins possible parmi les autres causes (voir section 5.3.2).

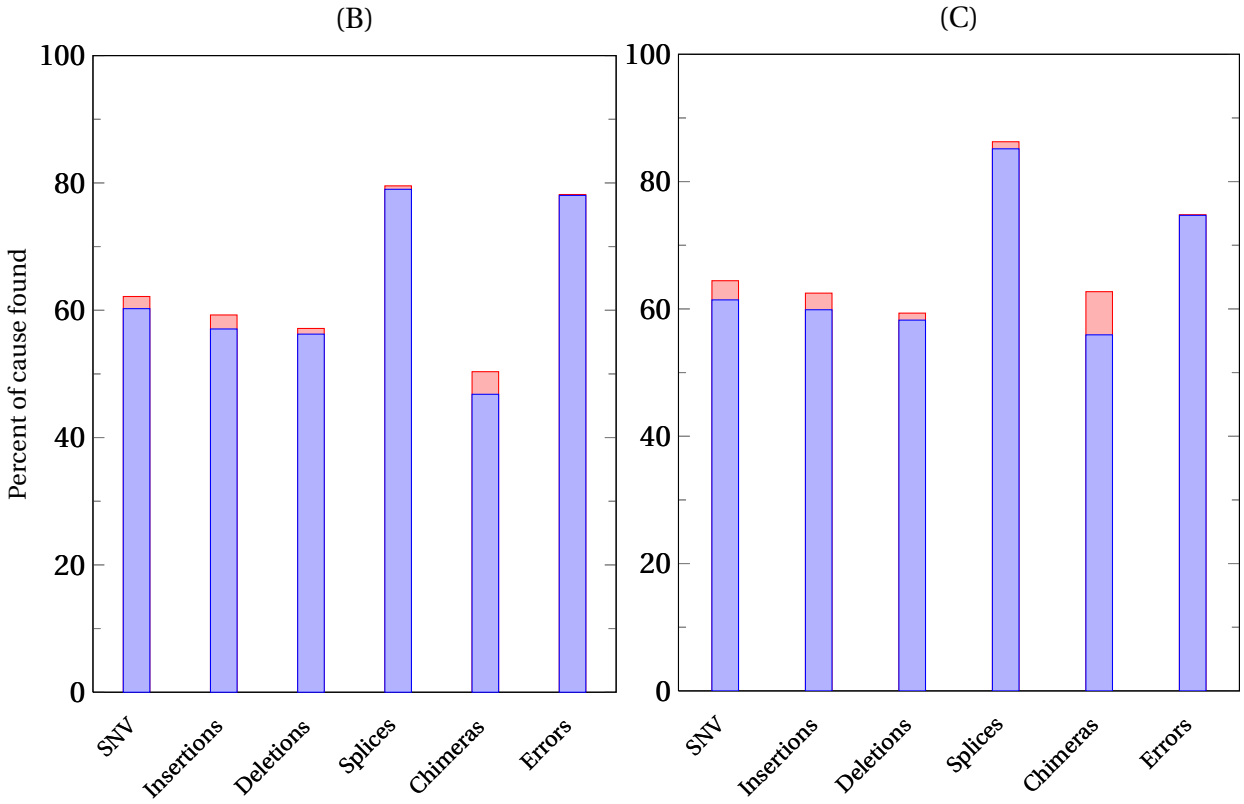
CRAC reste spécifique que les *reads* soient courts ou longs. De plus, il s'améliore en *sensibilité* lorsque la longueur des *reads* augmente. Or, nous avons vu dans la section 5.5.1.1 et la section 5.5.1.2 que cette tendance est inversée pour les autres outils de *mapping* et d'analyse (excepté MapSplice).

En outre, si CRAC ne trouve pas un certain nombre de causes c'est essentiellement lorsqu'elles se trouvent exclusivement sur les bords. À titre d'exemple, si nous prenons les SNV manquant de la figure 5.12 (B), 70 % d'entre eux sont couverts par 3 *reads* ou moins.

FIGURE 5.12 : *Sensibilité* et *précision* par cause des données simulées de hg19.

(A)

simulatedHuman75bp-42M	SNV	Insertions	Deletions	Splices	Chimeras	Errors
true prositives	18612	1636	1630	67012	542	10017293
false positives	590	63	26	263	39	18474
simulatedHuman200bp-48M						
true positives	38094	3366	3301	125566	1023	29013561
false positives	2737	213	135	1459	105	38375



Analyse de la classification de CRAC pour *simulatedHuman75bp-42M* et *simulatedHuman200bp-48M*. La table (A) correspond aux nombres bruts des causes retrouvées et aux fausses causes prédites par CRAC pour les deux jeux de données respectifs. Les histogrammes (B) et (C) mesurent la *sensibilité* et la *précision* dans chaque catégorie (SNV, insertions, délétions, jonctions classiques, chimères, erreurs) pour les deux jeux de données respectifs. La barre bleue correspond au pourcentage des causes qui sont correctes parmi celles détectées par CRAC, c'est-à-dire la barre rouge. La différence entre barre rouge et barre bleue correspond au pourcentage de fausses causes prédites par CRAC.

5.5.1.5 Les performances de CRAC

Dans les sections précédentes, nous comparons CRAC à des outils de *mapping* et de *splicing* et nous obtenons très souvent des meilleurs résultats en terme de *précision* et de *sensibilité*. Cependant, pour que ce soit plus valorisant, il faut que ses temps de calculs soient comparables à ceux des autres outils. Nous avons enregistrer ces différents temps sur le jeu de données *simulatedHuman75bp-42M* (TABLE 5.5.1.5). Nous avons mesurer ces temps avec une version non-parallélisée des logiciels et sur sur un Intel Xeon 2.27 GHz équipé avec 48 GO de RAM. Néanmoins, CRAC offre aussi une version parallèle de son algorithme.

TABLE 5.5 : Temps de calculs pour les différents outils de *mapping* et *splicing*

mapping tools	Bowtie	BWA	CRAC	GSNAP	GASSST	SOAP2
time (dhms)	7h	6h	9h	2d	5h	40m
memory (GB)	3	2	38	5	43	5

splicing tools	CRAC	GSNAP	MapSplice	TopHat
time (dhms)	9h	2d	4h	12h
memory (GB)	38	5	3	2

Mesure du temps de calcul et de la mémoire vive consommée sur le jeu de données *simulatedHuman75bp-42M* pour les outils de *mapping* et les outils de *splicing*. Les temps de calculs sont en jours, heures, minutes, secondes (dhms en anglais). Ces temps de calculs sont arrondis. La mémoire est en Giga Octets (GB en anglais).

Nous remarquons que SOAP2 se démarque en terme d'efficacité parmi les outils de *mapping* standards (Bowtie, BWA, GASSST, SOAP2). Il fait le calcul en moins d'une heure alors que les trois autres présentent des temps de calculs de plusieurs heures.

Pour les outils de *splicing* standards (MapSplice, TopHat), MapSplice est de loin le plus efficace avec seulement 3 heures de calcul contre plusieurs jours pour TopHat.

Pour les outils multifonctions (CRAC, GSNAP), CRAC n'est que 2 à 3 fois plus lent que les outils de *mapping* classiques avec un temps de calcul qui se compte en heures. Il est aussi 3 fois plus lent que MapSplice et beaucoup plus rapide de TopHat. Quant à GSNAP, il est plus lent lorsqu'on rajoute l'option "-N" permettant d'augmenter la *sensibilité* pour la détection des jonctions d'épissage. Son efficacité est du même ordre que TopHat.

Notons que la *sensibilité* et la *précision* de CRAC dépendent directement des *Gk arrays* qui indexent les *reads* (voir chapitre 4). Sur ce type d'expérience, CRAC est plus consommateur de mémoire que les autres outils.

5.5.2 Comparaisons sur les données réelles

L'analyse précédente sur des données simulées constitue une étape obligatoire. Elle nous a permis de mesurer la *sensibilité* et la *précision* de CRAC par rapport aux autres outils de *mapping* et d'analyse. Nous avons constaté que CRAC était plus spécifique que tous les outils, quel que soit le type de données et il s'accommode mieux de données plus longues.

Afin d'évaluer l'impact de cette *sensibilité* et cette *précision* sur des analyses biologiques réelles, nous évaluons la détection des épissages et des chimères sur un jeu de données de RNA-Seq de 75 *pb* non-orienté et un autre de 100*pb* orienté : K562-75bp-70M et ERR030856-100bp-75M.

5.5.2.1 Le *splicing*

Nous voulons comparer les outils de *splicing* (CRAC, GSNAP, MapSplice, TopHat) sur des données réelles. Toutefois, contrairement aux données simulées, nous ne connaissons pas les réponses au préalable, c'est-à-dire les jonctions que l'outil devraient retrouver. Pour vérifier la véracité des méthodes, nous comparons les jonctions prédites à celles qui sont le mieux annotées (*refseq*) sur les génomes hg19 et dmel5, à l'aide de l'outil *BioMart* de l'explorateur de génome *Ensembl* (http://www.ensembl.org/biomart/index.html). Quant à la méthode de comparaison, nous procédons de la même façon que nous l'avons fait pour le *splicing* simulé (section 5.5.1.2) : nous vérifions les localisations de part et d'autre de chaque jonction prédite avec une tolérance de ± 3 *pb*.

La comparaison est quelque peu différente entre K562-75bp-70M et ERR030856-100bp-75M car les premières données ne sont pas orientées. Par conséquent, nous considérons valide la jonction si elle se trouve sur un transcrit qui a une orientation opposée au *read*. En revanche, le deuxième jeu de données étant orienté, nous vérifions les orientations des *reads* par rapport aux *refseq*. Lorsque nous détectons une jonction sur deux exons consécutifs d'un même transcrit, nous parlons d'épissage classique, alors que si nous trouvons une jonction avec des sauts d'un ou plusieurs exons sur un même transcrit, nous parlons d'épissage alternatif. Si la jonction ne remplit aucun de ces deux critères (sur deux gènes différents, région sans annotation, faux positifs), nous la classons à part. Le nombre de gènes correspond au nombre total de gènes qui sont impliqués dans un phénomène d'épissage.

Il est difficile de commenter la table 5.6 et la table 5.7 car les croisements entre les jonctions détectées et la réalité sont faits à partir d'un fichier d'annotation de transcrits bien annotés. Par conséquent, les jonctions qui tombent en dehors de ces gènes ne peuvent pas être considérées, *a priori*, comme fausses. Nous avons donc choisi de constituer une catégorie à part (la catégorie *autre* de la table 5.6) pour regrouper tout ce qui ne s'apparente à une jonction d'épissage (classique ou alternatif) connue : les faux positifs, les jonctions non connues par rapport au fichier d'annotations que nous utilisons, etc.

K562-75bp-70M. C'est une expérience de RNA-Seq non-orienté, ce qui rajoute une difficulté supplémentaire pour analyser les résultats car on ne peut considérer l'orientation des *reads* par rapport à l'orientation des transcrits.

Néanmoins, si on se réfère aux résultats de *splicing* des données simulées de la section 5.5.1.2 pour des *reads* de 75 *pb*, CRAC est plus *précis* que les trois autres mais il est un peu moins *sensible*. La différence de *sensibilité* peut se mesurer dans la catégorie des jonctions classiques où CRAC localise 4 000 jonctions de moins que MapSplice et 8 000 de moins que GSNAP. Quant à la différence de *précision*, elle peut s'expliquer par le fait que CRAC est plus *conservatif* que les autres outils : une plus grande proportion des jonctions qu'il détecte sont classiques, plus sûres car les transcrits alternatifs, les transcrits non-codants,etc sont des événements rares [Dutertre *et al.*, 2010; Sammeth *et al.*, 2008]. Ainsi, CRAC évalue une proportion de 76,22 % de jonctions classiques contre 67,21 % pour MapSplice, 69,56 % pour TopHat et 66,29 % pour GSNAP, c'est-à-dire entre 10 % et 15 % de plus que les trois autres alors que le nombre de gènes identifiés par les quatre outils de *splicing* est stable : il varie entre 11 634 et 11 744 pour CRAC, MapSplice et TopHat, et atteint 12 107 pour GSNAP sachant qu'il produit plus de faux positifs que CRAC et MapSplice à 75 *pb* chez l'humain (TABLE 5.3 (A)).

Rappelons que K562-75bp-70M est une expérience qui ne concerne qu'une lignée cellulaire d'un seul individu, on s'attend donc à maximiser la proportion des épissages classiques par rapport aux épissages alternatifs ou aux nouvelles régions d'épissage.

TABLE 5.6 : Nombre de jonctions détectées pour K562-75bp-70M à partir des refseq.

K562-75bp-70M	CRAC		MapSplice		TopHat		GSNAP	
	%	nb	%	nb	%	nb	%	nb
classical splicing	77.28	97,954	67.23	102,564	62.14	96,309	59.68	106,504
alternative splicing	3.86	4,890	6.25	9,528	5.69	8,823	8.00	14,284
other	18.85	23,903	26.52	40,466	32.17	49,846	32.31	57,664
number of genes		11,634		11,744		11,644		12,107

Nombre de jonctions détectées sur K562-75bp-70M après un croisement avec les refseq du génome humain. Un seuil de $\pm 3 pb$ est toléré pour la précision sur chaque bord de la jonction.

ERR030856-100bp-75M. Ces données de RNA-Seq sont orientées mais elles proviennent d'un mélange de plusieurs tissus cellulaires. L'analyse reste complexe car l'ensemble des transcrits de chaque tissu sont différents, les proportions vont donc se cumuler.

Les *reads* font 100 *pb*, c'est-à-dire plus longs que ceux de K562-75bp-70M. Or, d'après les résultats du *splicing*, lorsque les *reads* sont plus longs, CRAC devient à la fois plus *sensible* et plus *précis*

que les autres outils (TABLE 5.3 (B)) alors que GSNAP s'améliore aussi en *précision*. Nous observons dans la table 5.7 que CRAC, MapSplice et GSNAP se stabilisent sur un même nombre de jonctions classiques ($\simeq$ 138 500 en moyenne) alors que TopHat plafonne à 114 302. Quant à la *précision*, là encore CRAC reste le plus conservatif avec un pourcentage quasiment équivalent aux données K562-75bp-70M (76,22 %), MapSplice se stabilise aussi à 67,21 %, TopHat et GSNAP gagnent 7 points chacun pour passer de 62,14 % à 69,56 % pour le premier et de 59,68 % à 66,29 % pour le deuxième.

Par rapport à K562-75bp-70M, le nombre de jonctions alternatives augmente de 50 % pour CRAC (de 4 890 à 7 200), légèrement pour MapSplice, reste stable pour GSNAP et baisse pour TopHat. CRAC et MapSplice étant les outils les plus *précis* et conservatifs, cette variation est peut-être due au fait que ERR030856-100bp-75M comprend plus de tissus différents que K562-75bp-70M pour un nombre de *reads* presque identique (70M contre 75M). D'ailleurs le nombre de jonctions classiques augmente aussi de 50 % pour CRAC, entre les deux jeux de données (de 97 954 à 138 554), de 35 % pour GSNAP et MapSplice, de 20 % pour TopHat.

Notons que le nombre de gènes est stable pour tous les outils avec un minimum de 15 222 pour TopHat et un maximum de 15 922 pour GSNAP. Le nombre de gènes exprimés, ayant au moins une jonction, est $\simeq$ 40 % plus élevé que dans la banque K562-75bp-70M. On note l'effet du mélange de tissus.

TABLE 5.7 : Nombre de jonctions détectées pour ERR030856-100bp-75M à partir des refseq.

ERR030856-100bp-75M	CRAC		MapSplice		TopHat		GSNAP	
	%	nb	%	nb	%	nb	%	nb
classical splicing	76.22	138,554	67.21	137,873	69.56	114,302	66.29	139,885
alternative splicing	3.96	7200	5.89	12,090	5.15	8,457	7.20	15,195
other	19.83	36,035	26.90	55,183	25.28	41,551	26.52	55,934
nomber of genes		15,861		15,824		15,222		15,922

Nombre de jonctions détectées sur ERR030856-100bp-75M après un croisement avec les refseq de génome humain. Un seuil de $\pm 3pb$ est toléré pour la précision sur chaque bord de la jonction.

5.5.2.2 La détection des chimères

Nous avons exécuté CRAC et son traitement strict spécifique aux chimères (cf. section 5.3.3). Nous avons répertorié 109 chimères potentielles différentes pour K562-75bp-70M et 63 pour ERR030856-100bp-75M. Nous les avons classées en quatre catégories en fonction de plusieurs critères sur les chromosomes, les brins et la distance entre les localisations des deux parties de la chimère.

Les chimères de K562-75bp-70M. Elles sont classées de la façon suivante :

1. 5 sur des chromosomes différents ($x.chr \neq y.chr$).

2. 6 sur le même chromosome et même brin mais avec des localisations trop espacées ($x.pos < y.pos$ et $|y.pos - x.pos >$ max-splice-length | avec max-splice-length=300Kbp).
3. 5 sur le même chromosome et même brin mais avec des localisations inversées ($y.pos < x.pos$).
4. 93 sur le même chromosome mais des brins différents ($x.strand \neq y.strand$).

Pour chaque classe, nous avons annoté les deux parties de la chimère, à l'aide du pipeline d'annotation présenté dans la section 3.3.1 du chapitre 3.

Nous analysons (analyse non montrée) que 3 des 5 chimères de la catégorie 1 semblent correspondre à l'appariement entre pseudo gènes, probablement des faux positifs de CRAC à cause de k-mers dupliqués. Concernant les 6 chimères de la catégorie 2, 5 correspondent à des jonctions classiques d'épissage avec une distance supérieure à la limite de 400 kpb fixée par CRAC. Parmi les 5 chimères de la catégorie 3, 3 sont impliquées dans un même gène. Enfin, nous savons qu'il existe un biais dans la préparation du RNA-Seq non-orienté impliquant des fausses chimères de la catégorie 4 [Houseley et Tollervey, 2010] (voir section 2.6.2.3 du chapitre 2), ce qui pourrait expliquer le nombre de 93 chimères identifiées par CRAC dans cette catégorie. De ce fait, nous avons donc décidé de ne pas les prendre en compte dans le pipeline d'annotation. Nous avons dressé dans la table 5.5.2.2 (A), l'annotation des 5 chimères restantes. Notons que nous détectons les chimères BCR-ABL et NUP114-XKR3, ainsi que leur point de cassure de façon très précise. Ces deux chimères ont été identifiées dans cette lignée cellulaire à de nombreuses reprises [Levin *et al.*, 2009; Maher *et al.*, 2009b].

Les chimères de ERR030856-100bp-75M. Elles sont classées de la façon suivante :

1. 4 sur des chromosomes différents ($x.chr \neq y.chr$).
2. 37 sur le même chromosome et même brin mais avec des localisations trop espacées ($x.pos < y.pos$ et $|y.pos - x.pos >$ max-splice-length | avec max-splice-length=400 kbp).
3. 17 sur le même chromosome et même brin mais avec des localisations inversées ($y.pos < x.pos$).
4. 5 sur le même chromosome mais des brins différents ($x.strand \neq y.strand$).

Pour chaque classe, nous avons annoté les deux parties de la chimère, de la même façon que précédemment.

Nous analysons aussi que 3 des 4 chimères de la catégorie 1 semblent correspondre à l'appariement entre pseudo gènes. Concernant les 37 chimères de la catégorie 2, 33 correspondent à des jonctions classiques d'épissage avec une distance supérieure à la limite de 400 kpb fixée par CRAC. D'ailleurs l'un d'eux correspond à un transcrit alternatif connu du gène RBFOX1, qui est impliqué dans le mécanisme d'épissage. Parmi les 17 chimères de la catégorie 3, 4 correspondent au problème de pseudo gènes (cité plus haut), et 11 sont impliqués dans un même gène. Enfin, parmi les

5 de la catégorie 4, 2 correspondent à des pseudo gènes. Nous avons dressé dans la table 5.5.2.2 (B), l'annotation des 10 chimères restantes.

Comparaison entre K562-75bp-70M et ERR030856-100bp-75M. Le premier jeu de données ne concerne qu'un seul type de tissu et individu alors que le deuxième est un mélange de 16 tissus provenant de 11 humains différents. Nous avons vu que cette diversité semblait augmenter le nombre de jonctions de façon significative (50 % pour CRAC et 35 % pour GSNAP et MapSPlice). Par conséquent, il est logique que le nombre de chimères (et faux positifs) de la catégorie 2 soit plus important chez ERR030856-100bp-75M que chez K562-75bp-70M. Quant à la disproportion de chimère de la catégorie 4 de K562-75bp-70M, elle s'explique par un biais protocolaire de la préparation du RNA-Seq non-orienté [Houseley et Tollervey, 2010].

Finalement, pour plus de 70M de *reads* produits dans chaque jeu de données, nous avons identifier 5 chimères potentielles dans K562-75bp-70M et 10 chimères potentielles dans ERR030856-100bp-75M. Nous confirmons que la présence de chimères dans un tissu constitue un événement rare. En outre, avec une méthode basée sur du *single reads*, nous sommes capables de détecter précisément ces chimères (à la bordure de chaque exon), comme la chimère BCR-ABL et la chimère NUP214-XKR3 des cellules K562.

TABLE 5.8 : Les chimères prédites sur les données réelles.

category	chr1	pos1	strand1	gene	annotation	chr2	pos2	strand2	gene	annotation
1.	22	23632599	1	BCR	CDS	9	133729450	1	ABL	CDS
1.	9	134074401	1	NUP214	CDS	22	17288975	-1	XKR3	CDS
2.	13	94014744	1	GPC6	intron	13	108518029	1	FAM155A	CDS
3.	6	31619432	-1	BAG6	CDS	6	31833563	-1	SLC44A4	CDS
3.	22	42912015	-1	RRP7A	inxon	22	42973104	-1	RP1-222E13.10	inxon

(a)

category	chr1	pos1	strand1	gene	annotation	chr2	pos2	strand2	gene	annotation
1.	2	72358723	-1	CYP26B1	UTR sens	3	113006142	1	BOC	UTR sens
2.	13	99293462	-1	RP11-295B17.4	UTR sens	13	41745254	-1	CALM2P3	UTR sens
2.	5	134262824	-1	CTB-36O1.5	UTR sens	5	99385165	-1	N/A	intergenic
2.	1	569480	1	RP5-857K21.10	UTR sens	1	38077346	1	RSPO1	UTR antisens
2.	5	134262929	-1	CTB-36O1.5	UTR sens	5	99385270	-1	N/A	intergenic
3.	1	144916568	-1	PDE4DIP	inxon sens	1	146528156	-1	RP11-325P15.3	inxon sens
3.	1	16889158	-1	RP5-1182A14.1	UTR sens	1	21749838	-1	N/A	intergenic
4.	17	76207956	-1	N/A	intergenic	17	17394611	1	MED9	UTR sens
4.	15	45361250	1	SORD	inxon sens	15	45120768	-1	N/A	intergenic
4.	1	40218723	1	PPIE	CDS sens	1	39988178	-1	RP11-69E11.4	inxon sens

(b)

Annotation des différentes chimères sélectionnées dans K562-75bp-70M (A) et ERR030856-100bp-75M (B). Les chimères sont classées dans une catégorie en fonction de plusieurs critères sur les chromosomes, les brins et la distance entre les localisations des deux parties de la chimère. Notons que, comparées à ERR030856, les données RNA-Seq de K562 ne sont pas orientées donc nous ne pouvons pas indiquer le sens de l'annotation de chaque gène.

5.6 Conclusion et discussions

Nous avons conçu CRAC dans le cadre du traitement de données RNA-Seq. Notre méthode semble particulièrement adaptée pour les *reads* qui contiennent des longs indels ou des jonctions d'épissage. En théorie, CRAC est adaptable sur toutes les technologies (Illumina®, SOLiD, Roche 454®) et tous types d'approches (RNA-Seq, ChIP-Seq, DGE, WGSS). Néanmoins, sur des données de WGSS nous n'avons pas de jonctions d'épissage, en conséquence les comparaisons avec les autres outils seront certainement plus serrées. En effet, dans le cadre de détections d'indels courts, de SNV et d'erreurs, des méthodes existent et sont spécialisées sur le traitement de données génomiques comme par exemple GASSST ou BWA [Rizk et Lavenier, 2010; Wu et Nacu, 2010; Li et Durbin, 2009]. Notons quand même que CRAC peut apporter un supplément à ces outils pour la détection des réarrangements chromosomiques qui sont assimilés à des longues insertions sur le génome. Il faudrait donc évaluer l'apport de CRAC sur des données génomiques en le comparant aux outils spécialisés.

CRAC est un logiciel qui intègre le *mapping* et l'analyse des *reads* dans une seule étape. En procédant ainsi, nous gagnons en *précision* et en *sensibilité* sur les stratégies actuelles. Notamment pour la détection des jonctions, des outils comme TopHat rencontrent des problèmes à cause de cette séparation entre la phase de *mapping* et d'analyse [Garber *et al.*, 2011] : une intégration est probablement une clé de la qualité de détection des jonctions.

De plus, une telle intégration nous permet d'analyser la globalité des *reads*, puis de les classer en fonction des causes qui les affectent. Nous pouvons en déduire les proportions prises par chaque cause (biologique ou artefactuelle) pour une ou plusieurs expériences SHD.

FluxSimulator est un outil qui permet de simuler du RNA-Seq. Nous avons paramétré cet outil pour qu'il puisse générer des jeux de données tels que le ferait un SHD. Cependant, il ne permet pas d'intégrer des mutations biologiques telles que des SNV, des indels ou des chimères. Nous avons développé *GenomeSimulator* qui intègre ces mutations à *FluxSimulator* pour ainsi produire des données simulées proches de la réalité. Ce pipeline, premier de ce genre à notre connaissance, donne la possibilité de comparer des outils sur tous les aspects d'une expérience RNA-Seq.

À partir de nos simulations, nous avons vu que GSNAP et CRAC sont les seuls outils qui gèrent bien le RNA-Seq. Cependant, lorsque la longueur et le nombre de données augmentent, CRAC est de plus en plus performant alors que GSNAP suit une tendance inverse. Cette constatation est notamment visible sur la détection de jonctions d'épissage : CRAC est plus sensible et plus spécifique sur la drosophile, plus spécifique sur l'humain et il n'est pas du tout affecté par le passage de $75pb$ à $200pb$, ce qui n'est pas le cas de GSNAP. De plus, GSNAP n'est pas capable de traiter les chimères sur du *single reads*.

De manière générale, tous les outils de *mapping* et d'analyse de *reads* que nous avons comparés à CRAC sont perturbés lorsque la longueur des *reads* augmente alors qu'elle améliore les performances de CRAC. Or, dans un futur proche les séquenceurs vont produire des *reads* de plus en plus longs avec des volumes de plus en plus massifs.

CRAC est un logiciel très *sensible* sur un génome « simple » comme la drosophile, mais également sur un génome contenant plus de duplications comme l'humain. Ces duplications sont potentiellement trompeuses et génératrices de fausses localisations.

En intégrant le *mapping* à l'analyse, nous pouvons détecter ces fausses localisations et instaurer des méthodes permettant de les éviter. Par conséquent, si l'intégration est importante pour le *splicing*, elle l'est encore plus pour les chimères : nous devons distinguer les fausses localisations génératrices de fausses chimères.

Outre le fait que CRAC apporte des avancées par son approche intégrée, il peut aussi servir, grâce à sa phase de *mapping*, aux outils d'analyse tels que *samtools*. Ce dernier produit des meilleurs résultats lorsque le *mapping* est précis [Li, 2011].

Enfin, l'intégration du *mapping* à l'analyse permet de traiter plusieurs aspects à la fois (*mapping*, détection de jonctions d'épissage, des SNV, des erreurs, des indels, des répétions, des chimères, etc) avec un temps de calcul du même ordre que les outils de *mapping* ou d'analyse classiques.

J'ai conçu CRAC en collaboration de Mikaël Salson du Laboratoire d'Informatique Fondamentale de Lille (LIFL), Eric Rivals du Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) et Thérèse Commes du Centre de Recherche de Biochimie Macromoléculaire (CRBM). J'ai développé CRAC et le pipeline de simulation avec Mikaël Salson. Un article est en cours de rédaction pour ce projet.

Des améliorations sont prévues comme par exemple l'intégration dans CRAC des critères pour le traitement des chimères. Une autre procédure est en cours de développement pour préserver les SNV présents sur les bords des *reads*. D'autres critères permettant d'augmenter la *précision* et la *sensibilité* de notre outil sont aussi en cours de développement.

Conclusion générale et Perspectives

Le cadre de nos recherches concerne l'annotation des transcriptomes par le biais de nouvelles technologies de séquençage : les SHD. Le but de nos recherches est de détecter le maximum d'informations qui sont exprimées dans un transcriptome à l'échelle des génomes. L'intérêt d'un tel travail est de caractériser des phénomènes de transcription rares, encore inconnus qui pourraient intervenir dans des maladies ou des cancers.

Les SHD produisent des milliards de *reads* dans un temps très court et à coût réduit. Nous pouvons imaginer que tous les centres médicaux seront bientôt équipés de ces technologies afin de séquencer, à n'importe quel moment, les cellules d'un individu pour connaître, comprendre ou suivre l'avancée de sa maladie.

Les SHD sont puissants et atteignent des profondeurs de séquençage sans précédent pour détecter les transcrits les plus rares. C'est pour cette raison que des techniques transcriptomiques de SHD telles que la DGE ou le RNA-Seq sont dites ouvertes car elles permettent, *a priori*, de séquencer des *reads* représentant tous les transcrits d'une cellule.

Pour identifier et caractériser toutes ces informations, il faut pouvoir identifier la position génomique d'origine de tous les *reads*. Cette étape constitue la ligne directrice de mon travail.

La bioinformatique est la seule solution pour positionner des *reads* sur un génome. En effet, les *reads* sont produits en trop grande quantité pour être manipulés manuellement. Cependant, cette tâche reste aussi compliquée dans le domaine informatique car les algorithmes initialement créés pour faire de la recherche de motifs dans un texte ne sont pas prévus pour gérer une telle quantité. Il fallait donc trouver des méthodes robustes permettant de rester dans la logique rapide et efficace des SHD.

De nombreux outils de *mapping* se sont développés permettant de localiser très rapidement des *reads* sur un génome. Cependant, les données séquencées ne cessent d'augmenter en nombre et en taille. Ajoutons à cela, les erreurs produites lors du séquençage et des étapes de préparation des échantillons.

Un simple *mapping* ne suffit pas, et l'interprétation des données exige plus d'informations face à la complexité bioloqique générée : comment distinguer les erreurs des causes biologiques ? avec quel degré de précision peut-on faire confiance au mapping ? quel est la proportion de transcrit qui n'est pas localisée ? quels sont les reads affectés par un SNV, un indel ou un phénomène d'épissage ? comment peut-on détecter un gène de fusion ou d'autres altérations chimériques de ce type ?

Toutes ces questions constituent des défis que seule l'informatique peut relever.

Mes principales contributions

J'ai commencé par me demander s'il y avait une meilleure stratégie de *mapping* à adopter pour positionner des *reads* sur un génome. Cette étape commence par un travail de fond consistant à évaluer l'impact combiné de la longueur des *reads*, de l'attente aléatoire du *mapping* et des erreurs de séquences lors de l'analyse des *reads*. Nous avons démontré qu'il existait une longueur suffisante pour localiser sans ambiguïté les *reads*. Nous avons aussi évalué la proportion d'erreurs générées dans une expérience Illumina® de DGE. Nous avons aussi montré que le séquenceur Illumina® avait tendance à produire plus d'erreurs sur la longueur des *reads*. Ce travail a été publié dans Nucleic Acids Research en 2009.

Nous nous sommes servis de ces estimateurs pour élaborer une stratégie de *mapping* exact pour la DGE. Nous l'avons intégrée dans un pipeline d'annotation pour identifier rapidement l'ensemble des informations et déterminer la proportion des transcrits dans les régions intergéniques du génome humain. À partir de données issues de cellules souches embryonnaires humaines, nous avons pu ainsi valider 28 nouveaux transcrits par PCR. Un article est en cours de rédaction pour ce travail.

À partir de ce premier travail, j'ai cherché à aller plus loin dans l'amélioration des procédures d'analyse des données de transcriptome. Un des aspects que nous avons cherché à exploiter est le fait qu' au delà d'une simple information de *mapping*, un *read* issu d'une expérience de RNA-Seq intègre d'autres informations partagées entre les *reads*. J'ai ainsi pu montrer le besoin d'effectuer des requêtes rapides sur les *reads*. Nous avons créé la structure de données *Gk arrays* permettant d'indexer des milliards de *reads*. Nous montrons que cette quantité de données constitue la limite des structures d'indexations connues telles que les tables des suffixes ou les tables de hachage. Ainsi les *Gk arrays* permettent ce passage à l'échelle tout en étant capable de répondre à un grand nombre de requêtes sur les *reads*. Cette structure d'indexation a été publiée dans BMC Bioinformatics en 2011.

Enfin, j'ai développé, avec l'aide de collaborateurs, le logiciel CRAC spécialisé dans le traitement du RNA-Seq. CRAC est un outil d'analyse de *reads* qui intègre sa propre phase de *mapping* à la détection de causes biologiques telles que les SNV, les indels ou les jonctions d'épissage.

Il est en fait le fruit de tous mes travaux de thèse précédents : i/ nous fixons une longueur k à l'aide des critères que nous avons développés dans l'article du NAR 2009 ; ii/ nous utilisons les *Gk arrays* pour indexer tous les k-mers des *reads* ; iii/ nous procédons par un *mapping* exact pour localiser des k-mers sur le génome en utilisant un FM-index (structure utilisée par Bowtie, BWA et SOPA2 ; elle est efficace pour du *mapping* exact). Avec les deux informations sur le *mapping* et les requêtes des *Gk arrays*, nous pouvons enregistrer toutes les informations pour chaque *read*.

Nous avons aussi développé un pipeline de simulation, en y intégrant notamment *FluxSimulator* (Sammeth, unpublished software) pour simuler du mieux possible une expérience de RNA-Seq intégrant des mutations telles que les SNV, indels, et chimères. À l'aide de cette simulation, nous comparons CRAC à des outils connus et très utilisés dans la communauté scientifique. Nous montrons que CRAC est plus *précis* et plus *sensible* que les autres outils sur du RNA-Seq. Nous montrons surtout que ses performances vont encore s'améliorer avec des données plus longues et plus volumineuses alors que la tendance semble s'inverser pour les autres outils d'analyse. Ces travaux sont en cours de rédaction.

Mes perspectives

Mes perspectives concernent essentiellement trois points : i/ mettre au point un pipeline complet de simulation pour le RNA-Seq ; ii/ intégrer l'assemblage des *reads* dans CRAC ; iii/ caractériser des chimères.

Pipeline de simulation. C'est un manque dans le domaine de la bioinformatique. Dans beaucoup d'outils, des simulations sont proposées pour les comparaisons mais aucun « benchmark » de simulation n'est proposé comme on pourrait le voir pour évaluer des outils de compression, par exemple. L'idée serait de compléter le pipeline combiné de *FluxSimulator* et *GenomeSimulator*, en intégrant par exemple des modèles d'erreurs prédéfinis pour chaque type de séquenceur ou encore de perfectionner la simulation des chimères, pour proposer un outil de simulation d'un RNA-Seq très proche de la réalité.

On pourrait ensuite construire des jeux de données qui serviraient de « benchmark » à tous les outils rendant ainsi une facilité dans le choix de l'outil selon les besoins de l'utilisateur.

Intégrer l'assemblage des *reads*. L'un des atouts de CRAC est sa capacité à détecter précisément toutes les informations (SNV, indel, jonction d'épissage, erreur, chimère, répétition, etc) contenues dans un *read*. Cependant, il serait encore plus intéressant d'utiliser toutes ces informations ensemble afin de caractériser précisément le transcrit d'origine.

L'algorithme principal de CRAC procède sur tous les *k*-mers d'un *read*, et ce, pour chaque *read* d'une collection. Or avec les *Gk arrays*, nous pouvons savoir quels sont les *reads* qui partagent un *k*-mer et plus précisément nous savons à quelles positions le *k*-mer apparaît dans les *reads*. Nous pourrions imaginer assembler des *reads* deux à deux, à partir d'un *k*-mer donné, en choisissant un *read* contenant le *k*-mer sur son extrémité gauche et un *read* contenant le *k*-mer sur son extrémité droite. En procédant ainsi jusqu'à ne plus pouvoir assembler, nous construirons un *métareads* regroupant les informations de chaque *read* détéctées par CRAC. Nous pourrions nous comparer à des méthodes comme cufflinks qui assemblent les transcrits mais d'une façon différente.

Caractériser des chimères. Nous avons essentiellement évalué CRAC sur des chimères simulées. Le résultat est plutôt convaincant mais la simulation des chimères produites dans le pipeline de simulation est bien plus simple que la réalité. Dans ce pipeline, nous ne générons que des transcrits de fusion alors que dans la réalité d'autres phénomènes ont déjà été identifiés. Même si à partir à partir de données expérimentales provenant de la lignée K562 réelles, nous pouvons valider notre capacité à détecter les transcrits de fusion attendus tels que BCR-ABL et NUP214-XKR3, il est beaucoup plus difficile de valider nos autres prédictions.

Notre but serait d'élargir cette étude à partir de données RNA-Seq issues de cellules primaires de patients atteints de leucémies aiguës myéloïdes. On pourrait ainsi identifier de nouvelles chimères, puis de construire une chimérothèque regroupant les différentes formes de chimères identifiées. On pourrait ainsi se consacrer sur la validation précise de quelques candidats. Ce travail aurait deux effets : i/ valider CRAC ; ii/ identifier des marqueurs cliniques pour le diagnostic des cancers.

ANNEXE

A Matériels supplémentaires

A.1 Matériels supplémentaires des *Gk arrays*

Voici une présentation des algorithmes des requêtes Q5 à Q7 pour les *Gk arrays*.

Algorithme 9 : Q2 ($\#Ind_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : $\#Ind_k(f)$, the cardinality of $Ind_k(f)$

```
1  begin
2      t ⟵ GkIFA[j];
3      ℓ_f ⟵ GkCFPS[t − 1];
4      u_f ⟵ GkCFPS[t];
5      prev ⟵ −1;
6      CIndk ⟵ 0;
7      foreach i ∈ [ℓ_f, u_f[ do
8          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
9          if readIndex ≠ prev then
10             CIndk ⟵ CIndk + 1;
11             prev ⟵ readIndex;
12     return (CIndk);
13 end
```

Algorithme 10 : Q5 ($UInd_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : The set $UInd_k(f)$, subset of $Ind_k(f)$ where f occurs only once

```
1  begin
2      UInd_k ⟵ empty set;
3      t ⟵ GkIFA[j];
4      ℓ_f ⟵ GkCFPS[t − 1];
5      u_f ⟵ GkCFPS[t];
6      prev ⟵ ⌊g^{-1}(GkSA[ℓ_f])/m⌋;
7      count ⟵ 1;
8      foreach i ∈ ]ℓ_f, u_f[ do
9          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
10         if readIndex ≠ prev then
11             if count = 1 then
12                 Add prev to UInd_k;
13             count ⟵ 1;
14             prev ⟵ readIndex;
15         else
16             count ⟵ count + 1;
17     if count = 1 then
18         Add prev to UInd_k;
19     return (UInd_k);
20 end
```

Algorithme 11 : Q6 ($\#UInd_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : $\#UInd_k(f)$, the cardinality of $UInd_k(f)$

```
1  begin
2      t ⟵ GkIFA[j];
3      ℓ_f ⟵ GkCFPS[t − 1];
4      u_f ⟵ GkCFPS[t];
5      prev ⟵ ⌊g^{-1}(GkSA[ℓ_f])/m⌋;
6      CUIndk ⟵ 0;
7      count ⟵ 1;
8      foreach i ∈ ]ℓ_f, u_f[ do
9          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
10         if readIndex ≠ prev then
11             if count = 1 then
12                 CUIndk ⟵ CUIndk + 1;
13             count⟵ 1;
14             prev ⟵ readIndex;
15         else
16             count ⟵ count + 1;
17     if count = 1 then
18         CUIndk ⟵ CUIndk + 1;
19     return (CUIndk);
20 end
```

Algorithme 12 : Q7 ($UPos_k(f)$) for *Gk arrays*.

Data : $f \in \Sigma^k$, $j \in P_{\text{pos}}$ such that $C_R[j\,..\,j+k-1] = f$
Result : The set $UPos_k(f)$, subset of $Pos_k(f)$ where f occurs only once

```
1  begin
2      UPos_k ⟵ empty set;
3      t ⟵ GkIFA[j];
4      ℓ_f ⟵ GkCFPS[t − 1];
5      u_f ⟵ GkCFPS[t];
6      prev ⟵ ⌊g^{-1}(GkSA[ℓ_f])/m⌋;
7      posPrev ⟵ g^{-1}(GkSA[ℓ_f]) mod m;
8      foreach i ∈ ]ℓ_f, u_f[ do
9          readIndex ⟵ ⌊g^{-1}(GkSA[i])/m⌋;
10         posInRead ⟵ g^{-1}(GkSA[i]) mod m;
11         if readIndex ≠ prev then
12             Add the pair (prev, posPrev) to UPos_k;
13             prev ⟵ readIndex;
14             posPrev ⟵ posInRead;
15     return (UPos_k);
16 end
```

A.2 Matériels supplémentaires de CRAC

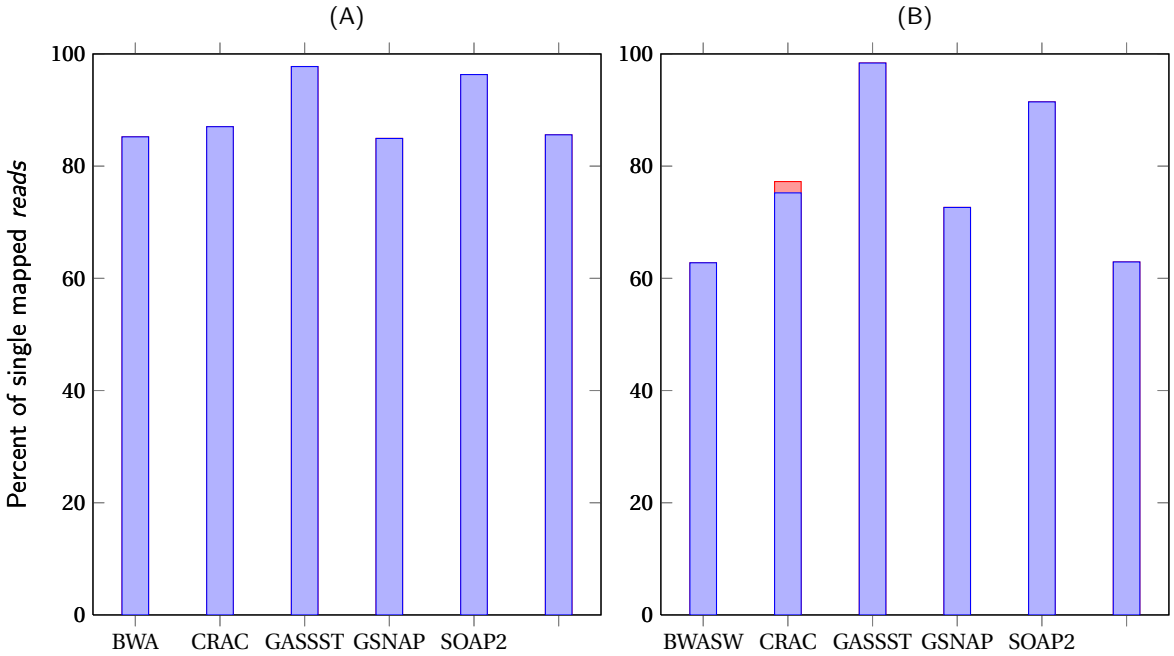
Voici, la distribution des mutations aux différentes étapes de la simulation pour dmel5.

Type	SNV	Insertion	Deletion	Chimera
Genome-wide	132,193	12,146	12,168	676
Sequenced (75bp)	28,535	2,531	2,654	651
Sequenced (200bp)	31,194	2,758	2,915	668

TABLE A.1 : Nombre de mutations générées aléatoirement et réellement séquencées sur le génome de *D. melanogaster*.

Voici la comparaison du *mapping* pour la dmel5.

FIGURE A.1 : *Sensibilité* et *précision* du *mapping* pour les données simulées de dmel5.



Calcul de la *sensibilité* et de la *précision* du *mapping* pour *simulatedDroso75bp-45M* (A) et *simulatedDroso200bp-48M* (B). La barre bleue correspond au pourcentage de *reads* qui sont correctement localisés parmi ceux qui doivent être localisés. La barre rouge correspond au pourcentage de *reads* qui ne sont pas localisés au bon endroit.

Voici la comparaison du *splicing* pour la dmel5.

TABLE A.2 : *Sensibilité* et *précision* du *splicing* pour les données simulées de dmel5.

(A)	CRAC		GSNAP		MapSplice		TopHat	
	%	nb	%	nb	%	nb	%	nb
sensibilité	85.46	38 792	77,34	35 106	82.07	37 254	86.57	39 297
précision	99.84	65	99,35	241	99.58	164	97.38	1 170

(B)	CRAC		GSNAP		MapSplice		TopHat	
	%	nb	%	nb	%	nb	%	nb
sensibilité	89.14	41 883	76,39	35 893	84.73	39 813	89.03	41 832
précision	99.76	105	99,11	337	99.52	201	96.17	1 779

Calcul de la *sensibilité* et de la *précision* du *splicing* pour *simulatedDroso75bp-45M* (A) et *simulatedDroso200bp-48M* (B) et comparaison entre les différents outils de *splicing*.

Voici la comparaison sur la détection des chimères pour la dmel5.

TABLE A.3 : *Sensibilité* et *précision* lors de la détection des chimères pour les données simulées de dmel5.

(A)	CRAC		CRAC-postProcess		MapSplice	
	%	nb	%	nb	%	nb
sensibilité	78.45	1 107	68.25	963	3.61	51
précision	96.96	35	99.90	1	37.14	88

(B)	CRAC		CRAC-postProcess		MapSplice	
	%	nb	%	nb	%	nb
sensibilité	71.02	1 272	62.70	1 123	3.02	57
précision	90.33	138	99.04	11	0.08	29 783

Calcul de la *sensibilité* et de la *précision* lors de la détection des chimères pour *simulatedDroso75bp-45M* (A) et *simulatedDroso200bp-48M* (B) et comparaison entre les différents outils capables de détecter des chimères sur du *single reads*.

Bibliographie

Viatcheslav R. Akmaev et Clarence J. Wang : Correction of sequence-based artifacts in serial analysis of gene expression. *Bioinformatics*, 20:1254–1263, 2004. Cité page 116.

Roger P. Alexander, Gang Fang, Joel Rozowsky, Michael Snyder et Mark B. Gerstein : Annotating non-coding regions of the genome. *Nature Reviews Genetics*, 11(8):559–571, juillet 2010. ISSN 1471-0056. Cité page 24.

SF Altschul, W Gish, W Miller, EW Myers et DJ Lipman : Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990. Cité pages 50 et 122.

Paulo P. Amaral, Marcel E. Dinger, Tim R. Mercer et John S. Mattick : The eukaryotic genome as an RNA machine. *Science (New York, N.Y.)*, 319(5871):1787–1789, mars 2008. ISSN 1095-9203. Cité page 20.

Lucile Amrouche, Raja Bonifay et Dany Anglicheau : MicroARN et maladies rénales - un intérêt grandissant. *médecine/sciences*, 27(4):7, 2011. Cité page 42.

Jean M. Aury, Corinne Cruaud, Valerie Barbe, Odile Rogier, Sophie Mangenot, Gaelle Samson, Julie Poulain, Veronique Anthouard, Claude Scarpelli, Francois Artiguenave et Patrick Wincker : High quality draft sequences for prokaryotic genomes using a mix of new sequencing technologies. *BMC Genomics*, 9(1):603+, décembre 2008. ISSN 1471-2164. Cité page 181.

Mukul S. Bansal et Oliver Eulenstein : The multiple gene duplication problem revisited. *Bioinformatics*, 24(13):i132–i138, 2008. Cité page 167.

A. Barski, S. Cuddapah, K. Cui, T. Y. Roh, D. E. Schones, Z. Wang, G. Wei, I. Chepelev et K. Zhao : High-resolution profiling of histone methylations in the human genome. *Cell*, 129:823–837, 2007. Cité page 77.

P Bertone, V Stolc, TE Royce, JS Rozowsky, AE Urban, X Zhu, JL Rinn, W Tongprasit, M Samanta, S Weissman, M Gerstein et M Snyder : Global identification of human transcribed sequences with genome tiling arrays. *Science*, 306:2242–2246, 2004. Cité pages 20, 24, 77, 115 et 118.

Douglas L. Black : Mechanisms of alternative pre-messenger rna splicing. *Annual Review of Biochemistry*, 72(1):291–336, 2003. ISSN 0066-4154. Cité page 60.

Nathan Blow : Transcriptomics : The digital generation. *Nature*, 458:239–242, 2009. Cité pages 23, 92 et 121.

MS Boguski, CM Tolstoshev et DE Bassett : Gene discovery in dbest. *Science*, 265(5181):1993–1994, 1994. Cité page 32.

Alan P. Boyle, Sean Davis, Hennady P. Shulha, Paul Meltzer, Elliott H. Margulies, Zhiping Weng, Terrence S. Furey et Gregory E. Crawford : High-Resolution Mapping and Characterization of Open Chromatin across the Genome. *Cell*, 132:311–322, 2008. Cité pages 41 et 77.

Yang S. Brooks, Guanghu Wang, Zheqiong Yang, Kimberly K. Smith, Erhard Bieberich et Lan Ko : Functional pre- mrna trans-splicing of coactivator coaa and corepressor rbm4 during stem/progenitor cell differentiation. *Journal of Biological Chemistry*, 284(27):18033–18046, 2009. Cité page 69.

Alexander Johnson Bruce Alberts : How cells read the genome : From DNA to protein, 2002. Cité page 18.

S. Burkhardt, A. Crauser, P. Ferragina, H.-P. Lenhof, E. Rivals et M. Vingron : q-gram Based Database Searching Using a Suffix Array (QUASAR). *In 3rd Annual Int. Conf. on Computational Molecular Biology*, pages 77–83. ACM Press, 1999. Cité pages 52, 122 et 126.

S. Burkhardt et J. Kärkkäinen : Better filtering with gapped q-Grams. *Fundam Inf*, 56:51–70, 2003. Cité pages 52 et 55.

Yangho Chen, Tade Souaiaia et Ting Chen : Perm : efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. *Bioinformatics*, 25(19):2514–2521, October 2009. Cité page 55.

J. Cheng, P. Kapranov, J. Drenkow, S. Dike, S. Brubaker, S. Patel, J. Long, D. Stern, H. Tammana, G. Helt, V. Sementchenko, A. Piccolboni, S. Bekiranov, D. K. Bailey, M. Ganesh, S. Ghosh, I. Bell, D. S. Gerhard et T. R. Gingeras : Transcriptional Maps of 10 Human Chromosomes at 5-Nucleotide Resolution. *Science*, 308:1149–1154, mai 2005. Cité page 24.

JM Claverie : Fewer genes, more noncoding RNA. *Science*, 309:1529–30, 2005. Cité page 115.

Nicole Cloonan, Alistair R. Forrest, Gabriel Kolle, Brooke B. Gardiner, Geoffrey J. Faulkner, Mellissa K. Brown, Darrin F. Taylor, Anita L. Steptoe, Shivangi Wani, Graeme Bethel, Alan J. Robertson, Andrew C. Perkins, Stephen J. Bruce, Clarence C. Lee, Swati S. Ranade, Heather E. Peckham, Jonathan M. Manning, Kevin J. McKernan et Sean M. Grimmond : Stem cell transcriptome profiling via massive-scale mRNA sequencing. *Nature methods*, 5(7):613–619, juillet 2008. Cité pages 37 et 40.

Jacques Colinge et Georg Feger : Detecting the impact of sequencing errors on SAGE data. *Bioinformatics*, 17:840–842, 2001. Cité pages 106 et 115.

Philippe Collas et John Arne A. Dahl : Chop it, ChIP it, check it : the current status of chromatin immunoprecipitation. *Frontiers in bioscience : a journal and virtual library*, 13:929–943, 2008. ISSN 1093-4715. Cité page 41.

IHGS Consortium : Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. Cité page 20.

Human Genome Sequencing ConsortiumInternational : Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945, octobre 2004. ISSN 0028-0836. Cité pages 32 et 41.

Thomas C Conway et Andrew J Bromage : Succinct Data Structures for Assembling Large Genomes. *Bioinformatics*, 27(4):479–486, 2011. Cité page 126.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest et Clifford Stein : *Introduction to Algorithms*. MIT Press, 2nd édition, 2001. Cité page 131.

Koby Crammer et Yoram Singer : On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, décembre 2001. Cité page 178.

F. H. Crick, L. Barnett, S. Brenner et R. J. Watts-Tobin : General nature of the genetic code for proteins. *Nature*, 192:1227–1232, décembre 1961. ISSN 0028-0836. Cité page 13.

F. H. C. Crick : Central dogma of molecular biology. *Nature*, 227(5258):561–563, août 1970. Cité page 18.

Fabio De Bona, Stephan Ossowski, Korbinian Schneeberger et Gunnar Rätsch : Optimal spliced alignments of short sequence reads. *Bioinformatics*, 24(16):i174–i180, août 2008. ISSN 1367-4811. Cité pages 60 et 62.

France Denoeud, Jean-Marc Aury, Corinne Da Silva, Benjamin Noel, Odile Rogier, Massimo Delledonne, Michele Morgante, Giorgio Valle, Patrick Wincker, Claude Scarpelli, Olivier Jaillon et Francois Artiguenave : Annotating genomes with massive-scale rna sequencing. *Genome Biology*, 9 (12):R175, 2008. ISSN 1465-6906. Cité pages 63 et 126.

Thomas Derrien et Roderic Guigó : De longs ARN non codants activateurs de la transcription des gènes. *médecine/sciences*, 27(4):3, 2011. Cité pages 1, 24 et 42.

Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina et Heinz Himmelbauer : Substantial biases in ultra-short read data sets from high-throughput dna sequencing. *Nucleic Acids Res.*, page e105, 2008a. Cité pages 41, 57 et 115.

Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina et Heinz Himmelbauer : Substantial biases in ultra-short read data sets from high-throughput dna sequencing. *Nucl. Acids Res.*, 36(16):e105+, September 2008b. Cité page 155.

Martin Dutertre, Stephan Vagner et Didier Auboeuf : Alternative splicing and breast cancer. *RNA Biology*, 7(4):403–411, novembre 2010. ISSN 1555-8584. Cité pages 1, 42, 61 et 198.

Henrik Edgren, Astrid Murumagi, Sara Kangaspeska, Daniel Nicorici, Vesa Hongisto, Kristine Kleivi, Inga Rye, Sandra Nyberg, Maija Wolf, Anne Lise Borresen Dale et Olli Kallioniemi : Identification of fusion genes in breast cancer by paired-end RNA-sequencing. *Genome Biology*, 12(1):R6+, 2011. ISSN 1465-6906. Cité pages 2 et 68.

Bradley Efron et Gail Gong : A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation. *The American Statistician*, 37:36–48, 1983. Cité page 90.

Scott J. Emrich, W. Brad Barbazuk, Li Li et Patrick S. Schnable : Gene discovery and annotation using LCM-454 transcriptome sequencing. *Genome research*, 17(1):69–73, janvier 2007. Cité page 37.

P. Ferragina et G. Manzini : Opportunistic data structures with applications. *In Proc. of FOCS*, pages 390–398, 2000. Cité pages 122, 123 et 151.

Paolo Ferragina, Rodrigo González, Gonzalo Navarro et Rossano Venturini : Compressed text indexes : From theory to practice. *J. Experimental Algorithmics*, 13:12 :1.12–12 :1.31, 2009. ISSN 1084-6654. Cité pages 48 et 123.

Amanda G. Fisher : Cellular identity and lineage choice. *Nat Rev Immunol*, 2(12):977–982, décembre 2002. ISSN 1474-1733. Cité page 23.

Liliana Florea : Bioinformatics of alternative splicing and its regulation. *Brief Bioinform*, 7(1):55–69, mars 2006. Cité pages 60 et 61.

Ernst Freese : The difference between spontaneous and base-analogue induced mutations of phage t4. *Proceedings of the National Academy of Sciences of the United States of America*, 45(4):622–633, avril 1959a. ISSN 0027-8424. Cité page 28.

Ernst Freese : The specific mutagenic effect of base analogues on phage t4. *Journal of Molecular Biology*, 1(2):87–105, juin 1959b. ISSN 0022-2836. Cité page 28.

M. Y. Galperin : The molecular biology database collection : 2007 update. *Nucleic Acids Res*, 35 (Database issue), janvier 2007. ISSN 1362-4962. Cité page 17.

Manuel Garber, Manfred G. Grabherr, Mitchell Guttman et Cole Trapnell : Computational methods for transcriptome annotation and quantification using RNA-seq. *Nature Methods*, 8(6):469–477, juin 2011. ISSN 1548-7091. Cité pages 62, 63, 154, 189 et 203.

DS Gerhard, L Wagner et EA Feingold : The status, quality, and expansion of the NIH full-length cDNA project : the mammalian gene collection (MGC). *Genome Res*, 14(10B):2121–2127, octobre 2004. Cité page 32.

M. B. Gerstein, C. Bruce, J. S. Rozowsky, D. Zheng, J. Du, J. O. Korbel, O. Emanuelsson, Z. D. Zhang, S. Weissman et M. Snyder : What is a gene, post-ENCODE ? history and updated definition. *Genome Research*, 17(6):669–681, juin 2007. ISSN 1088-9051. Cité pages 16, 20 et 237.

Thomas R. Gingeras : Implications of chimaeric non-co-linear transcripts. *Nature*, 461(7261):206–211, September 2009. Cité pages 65, 71 et 237.

Igor Y. Goryshin, Jerry Jendrisak, Les M. Hoffman, Ronald Meis et William S. Reznikoff : Insertional transposon mutagenesis by electroporation of released tn5 transposition complexes. *Nat Biotech*, 18(1):97–100, janvier 2000. ISSN 1087-0156. Cité page 27.

Gregory R R. Grant, Junmin Liu et Christian J J. Stoeckert : A practical false discovery rate approach to identifying patterns of differential expression in microarray data. *Bioinformatics*, mars 2005. ISSN 1367-4803. Cité page 64.

Rajnish A. Gupta, Nilay Shah, Kevin C. Wang, Jeewon Kim, Hugo M. Horlings, David J. Wong, Miao-Chih Tsai, Tiffany Hung, Pedram Argani, John L. Rinn, Yulei Wang, Pius Brzoska, Benjamin Kong, Rui Li, Robert B. West, Marc J. van de Vijver, Saraswati Sukumar et Howard Y. Chang : Long non-coding RNA HOTAIR reprograms chromatin state to promote cancer metastasis. *Nature*, 464 (7291):1071–1076, avril 2010. ISSN 0028-0836. Cité page 42.

Dan Gusfield : *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997. Cité pages 43, 122, 123, 124, 138, 139 et 140.

Faraz Hach, Fereydoun Hormozdiari, Can Alkan, Farhad Hormozdiari, Inanc Birol, Evan E Eichler et S Cenk Sahinalp : mrsFAST : a cache-oblivious algorithm for short-read mapping. *Nat. Methods*, 7(8):576–577, 2010. Cité page 122.

Matthias Harbers et Piero Carninci : Tag-based approaches for transcriptome research and genome annotation. *Nat Methods*, 2:495–502, 2005. Cité pages 22 et 108.

Arnold Berk Harvey Lodish : Hierarchical structure of proteins, 2000. Cité page 13.

Steve Hoffmann, Christian Otto, Stefan Kurtz, Cynthia M. Sharma, Philipp Khaitovich, Jörg Vogel, Peter F. Stadler et Jörg Hackermüller : Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput Biol*, 5(9):e1000502+, September 2009. Cité page 51.

Robert A. Holt et Steven J. Jones : The new paradigm of flow cell sequencing. *Genome research*, 18 (6):839–846, juin 2008. Cité page 37.

Robert Homann, David Fleer, Robert Giegerich et Marc Rehmsmeier : mkESA : enhanced suffix array construction tool. *Bioinformatics*, 25(8):1084–1085, 2009. Cité page 123.

Jonathan Houseley et David Tollervey : Apparent Non-Canonical Trans-Splicing is generated by reverse transcriptase in vitro. *PLoS ONE*, 5(8):e12271+, août 2010. Cité pages 34, 72, 200 et 201.

International Cancer Genome Consortium, Thomas J. Hudson, Warwick Anderson, Axel Artez, Anna D. Barker, Cindy Bell, Rosa R. Bernabé, M. K. Bhan, Fabien Calvo, Iiro Eerola, Daniela S. Gerhard, Alan Guttmacher, Mark Guyer, Fiona M. Hemsley, Jennifer L. Jennings, David Kerr, Peter Klatt, Patrik Kolar, Jun Kusada, David P. Lane, Frank Laplace et al : International network of cancer genome projects. *Nature*, 464(7291):993–998, avril 2010. ISSN 1476-4687. Cité page 17.

Hui Jiang et Wing H. Wong : Seqmap : mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, 24(20):btn429–2396, August 2008. Cité page 49.

Hui Jiang et Wing H. Wong : Statistical inferences for isoform expression in RNA-seq. *Bioinformatics*, 25(8):1026–1032, avril 2009. ISSN 1367-4811. Cité page 64.

J M Johnson, S Edwards, D Shoemaker et E E Schadt : Dark matter in the genome : evidence of widespread transcription detected by microarray tiling experiments. *Trends Genet*, 21:93–102, 2005. Cité pages 20 et 24.

Petteri Jokinen et Esko Ukkonen : Two algorithms for approximate string matching in static texts. *In* A. Tarlecki, éditeur : *Proc. of the 16th Symposium on Mathematical Foundations of Computer Science*, numéro 520 *in* Lecture Notes in Computer Science, pages 240–248. Springer-Verlag, Berlin, 1991. Cité pages 52 et 54.

Zhengyan Kan, Bijay S. Jaiswal, Jeremy Stinson, Vasantharajan Janakiraman, Deepali Bhatt, Howard M. Stern, Peng Yue, Peter M. Haverty, Richard Bourgon, Jianbiao Zheng, Martin Moorhead, Subhra Chaudhuri, Lynn P. Tomsho, Brock A. Peters, Kanan Pujara, Shaun Cordes, David P. Davis, Victoria E. H. Carlton, Wenlin Yuan, Li Li, Weiru Wang, Charles Eigenbrot, Joshua S. Kaminker, David A. Eberhard, Paul Waring, Stephan C. Schuster, Zora Modrusan, Zemin Zhang, David Stokoe, Frederic J. de Sauvage, Malek Faham et Somasekar Seshagiri : Diverse somatic mutation patterns and pathway alterations in human cancers. *Nature*, 466(7308):869–873, juillet 2010. ISSN 0028-0836. Cité page 27.

P Kapranov, J Cheng, S Dike, DA Nix, R Duttagupta, AT Willingham, PF Stadler, J Hertel, J Hackermuller, IL Hofacker, I Bell, E Cheung, J Drenkow, E Dumais, S Patel, G Helt, M Ganesh, S Ghosh, A Piccolboni, V Sementchenko, H Tammana et TR Gingeras : RNA Maps Reveal New RNA Classes and a Possible Function for Pervasive Transcription. *Science*, 316:1484–1488, 2007. Cité pages 24, 37, 67, 78, 115 et 118.

Toru Kasai, Gunho Lee, Hiroki Arimura, Setsuo Arikawa et Kunsoo Park : Linear-Time Longest-Common-Prefix Computation in Suffix Arrays and Its Applications. *In Proc. of the 12th Symposium on Combinatorial Pattern Matching*, volume 2089 de *Lecture Notes in Computer Science*, pages 181–192. Springer, 2001. ISBN 3-540-42271-4. Cité page 139.

Hideya Kawaji, Takeya Kasukawa, Shiro Fukuda, Shintaro Katayama, Chikatoshi Kai, Jun Kawai, Piero Carninci et Yoshihide Hayashizaki : CAGE Basic/Analysis Databases : the CAGE resource for comprehensive promoter analysis. *Nucleic Acids Research*, 34(suppl_1):D632–636, 2006. Cité pages 35 et 98.

Celine Keime, Marie Semon, Dominique Mouchiroud, Laurent Duret et Olivier Gandrillon : Unexpected observations after mapping LongSAGE tags to the human genome. *BMC Bioinformatics*, 8:154, 2007. Cité pages 57, 78, 110 et 117.

W. James Kent : Blat–the blast-like alignment tool. *Genome research*, 12(4):656–664, April 2002. Cité pages 50 et 52.

Peter V Kharchenko, Michael Y Tolstorukov et Peter J Park : Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nat Biotechnol*, 26:1351–1359, 2008. Cité pages 41 et 115.

Jaswinder Khattra, Allen D. Delaney, Yongjun Zhao, Asim Siddiqui, Jennifer Asano, Helen Mcdonald, Pawan Pandoh, Noreen Dhalla, Anna-Liisa Prabhu, Kevin Ma, Stephanie Lee, Adrian Ally, Angela Tam, Danne Sa, Sean Rogers, David Charest, Jeff Stott, Scott Zuyderduyn, Richard Varhol, Connie Eaves, Steven Jones, Robert Holt, Martin Hirst, Pamela A. Hoodless et Marco A. Marra : Large-scale production of sage libraries from microdissected tissues, flow-sorted cells, and cell lines. *Genome Res*, 17:108–116, 2007. Cité pages 23, 115, 116 et 118.

H. Kim, R. Klein, J. Majewski et J. Ott : Estimating rates of alternative splicing in mammals and invertebrates. *Nature genetics*, 36(9), septembre 2004. ISSN 1061-4036. Cité page 60.

Jae Bum Kim, Gregory J Porreca, Lei Song, Steven C Greenway, Joshua M Gorham, George M Church, Christine E Seidman et J. G Seidman : Polony Multiplex Analysis of Gene Expression (PMAGE) in Mouse Hypertrophic Cardiomyopathy. *Science*, 316:1481–1484, 2007. Cité pages 77, 78 et 118.

Annelies de Klein, Ad Geurts van Kessel, Gerard Grosveld, Claus R. Bartram, Anne Hagemeijer, Dirk Bootsma, Nigel K. Spurr, Nora Heisterkamp, John Groffen et John R. Stephenson : A cellular onco-

gene is translocated to the philadelphia chromosome in chronic myelocytic leukaemia. *Nature*, 300(5894):765–767, décembre 1982. Cité page 67.

Thomas B Kryston, Anastassiya B Georgiev, Polycarpos Pissis et Alexandros G Georgakilas : Role of oxidative stress and dna damage in human carcinogenesis. *Mutation Research*, pages 1–9, 2011. Cité page 27.

Gregory Kucherov, Laurent Noé et Mikhail A. Roytberg : A unifying framework for seed sensitivity and its application to subset seeds. *Journal of Bioinformatics and Computational Biology*, 4(2): 553–569, November 2006. Cité page 55.

Stefan Kurtz : Reducing the space requirement of suffix trees. *Software : Practice and Experience*, 29 (13):1149–1171, novembre 1999. ISSN 1097-024X. Cité page 46.

Stefan Kurtz, Adam Phillippy, Arthur Delcher, Michael Smoot, Martin Shumway, Corina Antonescu et Steven Salzberg : Versatile and open software for comparing large genomes. *Genome Biology*, 5(2):R12, 2004. ISSN 1465-6906. Cité page 138.

Juha Kärkkäinen : Fast bwt in small space by blockwise suffix sorting. *Theoretical Computer Science*, 387(3):249 – 257, 2007. ISSN 0304-3975. The Burrows-Wheeler Transform. Cité page 157.

Ben Langmead, Kasper Hansen et Jeffrey Leek : Cloud-scale RNA-sequencing differential expression analysis with myrna. *Genome Biology*, 11(8):R83+, août 2010. ISSN 1465-6906. Cité page 64.

Ben Langmead, Cole Trapnell, Mihai Pop et Steven Salzberg : Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biology*, 10(3):R25+, 2009. Cité pages 4, 51, 62, 154, 157 et 175.

William Lee, Zhaoshi Jiang, Jinfeng Liu, Peter M. Haverty, Yinghui Guan, Jeremy Stinson, Peng Yue, Yan Zhang, Krishna P. Pant, Deepali Bhatt, Connie Ha, Stephanie Johnson, Michael I. Kennemer, Sankar Mohan, Igor Nazarenko, Colin Watanabe, Andrew B. Sparks, David S. Shames, Robert Gentleman, Frederic J. de Sauvage, Howard Stern, Ajay Pandita, Dennis G. Ballinger, Radoje Drmanac, Zora Modrusan, Somasekar Seshagiri et Zemin Zhang : The mutation spectrum revealed by paired genome sequences from a lung cancer patient. *Nature*, 465(7297):473–477, mai 2010. ISSN 0028-0836. Cité page 27.

Joshua Z. Levin, Michael F. Berger, Xian Adiconis, Peter Rogov, Alexandre Melnikov, Timothy Fennell, Chad Nusbaum, Levi A. Garraway et Andreas Gnirke : Targeted next-generation sequencing of a cancer transcriptome enhances detection of sequence variants and novel fusion transcripts. *Genome biology*, 10(10):R115+, octobre 2009. ISSN 1465-6914. Cité page 200.

H. Li, J. Wang, G. Mor et J. Sklar : A neoplastic gene fusion mimics trans-splicing of rnas in normal human cells. *Science*, 321(5894):1357–1361, September 2008a. Cité pages 67 et 68.

Heng Li : Improving snp discovery by base alignment quality. *Bioinformatics*, 27(8):1157–1158, 2011. Cité pages 33, 58, 59, 154, 175 et 204.

Heng Li et Richard Durbin : Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–1760, July 2009. Cité pages 4, 49, 51, 62, 123, 157, 175 et 203.

Heng Li et Richard Durbin : Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, 26(5):589–595, 2010. Cité pages 5, 51 et 175.

Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin et 1000 Genome Project Data Processing Subgroup : The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009a. Cité pages 59, 154 et 175.

Heng Li et Nils Homer : A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473 –483, 2010. Cité page 59.

Heng Li, Jue Ruan et Richard Durbin : Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, November 2008b. Cité pages 49 et 50.

Ming Li, Bin Ma, Derek Kisman et John Tromp : Patternhunter ii : highly sensitive and fast homology search. *Genome informatics. International Conference on Genome Informatics*, 14:164–175, 2003. Cité page 52.

Ruiqiang Li, Yingrui Li, Karsten Kristiansen et Jun Wang : Soap : short oligonucleotide alignment program. *Bioinformatics*, 24(5):713–714, March 2008c. Cité pages 49 et 50.

Ruiqiang Li, Chang Yu, Yingrui Li, Tak-Wah Lam, Siu-Ming Yiu, Karsten Kristiansen et Jun Wang : Soap2 : an improved ultrafast tool for short read alignment. *Bioinformatics*, 25(15):1966–1967, August 2009b. Cité pages 4, 49, 51, 157 et 175.

Xin Li, Li Zhao, Huifeng Jiang et Wen Wang : Short homologous sequences are strongly associated with the generation of chimeric RNAs in eukaryotes. *Journal of Molecular Evolution*, 68(1):56–65, 2008d. ISSN 0022-2844. Cité pages 1, 65, 69, 117 et 237.

Yang Li, Jeremy Chien, David I. Smith et Jian Ma : FusionHunter : identifying fusion transcripts in cancer using paired-end RNA-seq. *Bioinformatics (Oxford, England)*, mai 2011. ISSN 1367-4811. Cité pages 65, 71, 155 et 191.

Hao Lin, Zefeng Zhang, Michael Q. Zhang, Bin Ma et Ming Li : Zoom ! zillions of oligos mapped. *Bioinformatics*, 24(21):2431–2437, November 2008. Cité page 55.

Bin Ma et Ming Li : On the complexity of spaced seeds. *Journal of Computer and System Sciences*, 73(7):1024–1034, Mar. 2007. Cité page 55.

Bin Ma, John Tromp et Ming Li : Patternhunter : faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, March 2002. Cité page 52.

C.H. Maher, C. Kumar-Sinha, X. Cao, S. Kalyana-Sundaram, B. Han, X. Jing, L. Sam, T. Barrette, N. Palanisamy et A.M. Chinnaiyan : Transcriptome sequencing to detect gene fusions in cancer. *Nature*, 458(7234):97–101, 2009a. Cité page 121.

Christopher A. Maher, Chandan Kumar-Sinha, Xuhong Cao, Shanker Kalyana-Sundaram, Bo Han, Xiaojun Jing, Lee Sam, Terrence Barrette, Nallasivam Palanisamy et Arul M. Chinnaiyan : Transcriptome sequencing to detect gene fusions in cancer. *Nature*, 458(7234):97–101, January 2009b. Cité pages 33, 60, 65, 67 et 200.

Christopher A. Maher, Nallasivam Palanisamy, John C. Brenner, Xuhong Cao, Shanker Kalyana-Sundaram, Shujun Luo, Irina Khrebtukova, Terrence R. Barrette, Catherine Grasso, Jindan Yu, Robert J. Lonigro, Gary Schroth, Chandan Kumar-Sinha et Arul M. Chinnaiyan : Chimeric transcript discovery by paired-end transcriptome sequencing. *Proceedings of the National Academy of Sciences*, 106(30):12353–12358, juillet 2009c. ISSN 1091-6490. Cité pages 1, 67 et 72.

Veli Mäkinen et Gonzalo Navarro : Succinct suffix arrays based on run-length encoding. *Nordic J. of Computing*, 12:40–66, March 2005. ISSN 1236-6064. Cité pages 155 et 156.

Udi Manber et Gene W. Myers : Suffix Arrays : A New Method for On-Line String Searches. *In Proceedings of the first annual ACM-SIAM Symposium on Discrete Algorithms*, pages 319–327, San-Francisco, January 22-24 1990. SIAM. Cité pages 46, 47, 122, 124 et 140.

M. Mancheron : *Extraction de Motifs Communs dans un ensemble de Séquences. Application à l'identification de sites de liaison aux protéines dans les séquences primaires d'ADN.* Thèse de doctorat, Université de Nantes, Sep 2006. Cité page 237.

G. Manzini : Two space saving tricks for linear time LCP array computation. *In Proc. 9th Scandinavian Workshop on Algorithm Theory*, volume 3111, pages 372–383, 2004. Cité page 138.

Guillaume Marcais et Carl Kingsford : A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011. Cité page 127.

Elaine R Mardis : ChIP-seq : welcome to the new frontier. *Nat Methods*, 4:613–614, 2007. Cité pages 25, 41 et 115.

E. Marshall : Getting the noise out of gene arrays. *Science*, 306(5696):630–1, 2004. Cité page 22.

E. McCreight : A space-economical suffix tree construction algorithm. *Journal of the ACM*, 23(2): 262–272, April 1976. Cité page 44.

C. Joel McManus, Michael O. Duff, Jodi Eipper-Mains et Brenton R. Graveley : Global analysis of trans-splicing in drosophila. *Proceedings of the National Academy of Sciences*, 107(29):12975 – 12979, juillet 2010. Cité page 72.

Andrew McPherson, Chunxiao Wu, Iman Hajirasouliha, Fereydoun Hormozdiari, Faraz Hach, Anna Lapuk, Stanislav Volik, Sohrab Shah, Colin Collins et S. Cenk Sahinalp : Comrad : detection of expressed rearrangements by integrated analysis of RNA-seq and low coverage genome sequence data. *Bioinformatics*, 27(11):1481–1488, juin 2011. ISSN 1367-4811. Cité pages 65, 72, 73, 155, 165 et 191.

Tim R. Mercer, Marcel E. Dinger et John S. Mattick : Long non-coding RNAs : insights into functions. *Nature reviews. Genetics*, 10(3):155–159, mars 2009. ISSN 1471-0064. Cité pages 1, 42 et 111.

Michael L. Metzker : Sequencing technologies - the next generation. *Nature reviews. Genetics*, 11 (1):31–46, janvier 2010. ISSN 1471-0064. Cité pages 32, 33 et 48.

Axel Meyer : Genomes evolve, but how ? *Nature*, 51:771–772, février 2008. Cité page 27.

Jason R. Miller, Sergey Koren et Granger Sutton : Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315 – 327, 2010. ISSN 0888-7543. Cité page 126.

Kevin V. Morris : Long antisense non-coding RNAs function to direct epigenetic complexes that regulate transcription in human cells. *Epigenetics : official journal of the DNA Methylation Society*, 4(5):296–301, juillet 2009. ISSN 1559-2294. Cité pages 40, 42 et 94.

A. Sorana Morrissy, Ryan D. Morin, Allen Delaney, Thomas Zeng, Helen McDonald, Steven Jones, Yongjun Zhao, Martin Hirst et Marco A. Marra : Next-generation tag sequencing for cancer gene expression profiling. *Genome research*, 19(10):1825–1835, octobre 2009. Cité pages 1, 35, 60 et 237.

Ali Mortazavi, Brian A. Williams, Kenneth McCue, Lorian Schaeffer et Barbara Wold : Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nature methods*, 5(7):621–628, juillet 2008. Cité pages 39, 42, 64 et 181.

I. Munro : Tables. *In Proc. of Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1180, pages 37–42. Springer, 1996. Cité page 137.

G. Myers : A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM*, 46(3):395–415, 1999. Cité page 54.

Serban Nacu, Wenlin Yuan, Zhengyan Kan, Deepali Bhatt, Celina Rivers, Jeremy Stinson, Brock Peters, Zora Modrusan, Kenneth Jung, Somasekar Seshagiri et Thomas Wu : Deep rna sequencing analysis of readthrough gene fusions in human prostate adenocarcinoma and reference samples. *BMC Medical Genomics*, 4(1):11, 2011. ISSN 1755-8794. Cité pages 2, 65, 68, 69, 71 et 73.

Ugrappa Nagalakshmi, Zhong Wang, Karl Waern, Chong Shou, Debasish Raha, Mark Gerstein et Michael Snyder : The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, 320(5881):1344–1349, juin 2008. Cité pages 39 et 60.

Gonzalo Navarro et Kimmo Fredriksson : Average complexity of exact and approximate multiple string matching. *Theoretical Computer Science*, 321(2-3):283–290, 2004. Cité page 53.

Gonzalo Navarro et Matthieu Raffinot : *Flexible Pattern Matching in Strings - Practical on-line search algorithms for texts and biological sequences*. Cambridge Univ. Press, 2002. Cité page 53.

S. B. Needleman et C. D. Wunsch : A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, mars 1970. ISSN 0022-2836. Cité page 59.

Marius Nicolae et Ion Măndoiu : Accurate estimation of gene expression levels from DGE sequencing data. *In Bioinformatics Research and Applications*, volume 6674, pages 392–403. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21259-8. Cité page 37.

François Nicolas et Eric Rivals : Hardness of optimal spaced seed design. *Journal of Computer and System Sciences*, 74:831–849, 2008. Cité page 55.

Kare L. Nielsen, Annabeth L. Hogh et Jeppe Emmersen : DeepSAGE–digital transcriptomics with high sensitivity, simple experimental protocol and multiplexing of samples. *Nucleic Acids Research*, 34(19):e133–, 2006. Cité page 23.

Z Ning, A J Cox et J C Mullikin : Ssaha : a fast search method for large dna databases. *Genome Res*, 11(10):1725–9, octobre 2001. Cité page 50.

Laurent Noé, Marta Gîrdea et Gregory Kucherov : Designing efficient spaced seeds for SOLiD read mapping. *Advances in bioinformatics*, 2010:1–12, 2010. ISSN 1687-8035. Cité page 55.

O. Owolabi et D. R. McGregor : Fast approximate string matching. *Software Practice and Experience*, 18(4):387–393, avril 1988. Cité pages 52 et 54.

Fatih Ozsolak, Adam R. Platt, Dan R. Jones, Jeffrey G. Reifenberger, Lauryn E. Sass, Peter McInerney, John F. Thompson, Jayson Bowers, Mirna Jarosz et Patrice M. Milos : Direct RNA sequencing. *Nature*, 461(7265):814–818, octobre 2009. ISSN 1476-4687. Cité page 34.

Chandra Shekhar S. Pareek, Rafal Smoczynski et Andrzej Tretyn : Sequencing technologies and genome sequencing. *Journal of applied genetics*, juin 2011. ISSN 2190-3883. Cité page 2.

Dmitri Parkhomchuk, Tatiana Borodina, Vyacheslav Amstislavskiy, Maria Banaru, Linda Hallen, Sylvia Krobitsch, Hans Lehrach et Alexey Soldatov : Transcriptome analysis by strand-specific sequencing of complementary DNA. *Nucleic Acids Research*, 37(18):e123, octobre 2009. Cité page 40.

Genís Parra, Alexandre Reymond, Noura Dabbouseh, Emmanouil T. Dermitzakis, Robert Castelo, Timothy M. Thomson, Stylianos E. Antonarakis et Roderic Guigó : Tandem chimerism as a means to increase protein complexity in the human genome. *Genome Research*, 16(1):37–44, 2006. Cité pages 65 et 67.

P. A. Pevzner et M. S. Waterman : Multiple filtration and approximate pattern matching. *Algorithmica*, 13(1/2):135–154, janvier 1995. Special issue on computational molecular biology. Cité page 52.

Nicolas Philippe, Anthony Boureux, Laurent Brehelin, Jorma Tarhio, Therese Commes et Eric Rivals : Using reads to annotate the genome : influence of length, background distribution, and sequence errors on prediction capacity. *Nucleic Acids Research*, 37(15):e104+, August 2009. Cité pages 4, 119, 125, 155, 156 et 157.

Nicolas Philippe, Mikael Salson, Thierry Lecroq, Martine Leonard, Therese Commes et Eric Rivals : Querying large read collections in main memory : a versatile data structure. *BMC Bioinformatics*, 12(1):242+, 2011. ISSN 1471-2105. Cité pages 4, 151, 155 et 156.

David Piquemal, Thérèse Commes, Laurent Manchon, Mireille Lejeune, Conchita Ferraz, Denis Pugnère, Jacques Demaille, Jean-Marc Elalouf et Jacques Marti : Transcriptome analysis of monocytic leukemia cell differentiation. *Genomics*, 80:361–371, 2002. Cité pages 106 et 115.

S. J. Puglisi, W. F. Smyth et A. Turpin : A taxonomy of suffix array construction algorithms. *ACM Comp. Surv.*, 39(2):1–31, 2007. Cité page 139.

Ronan Quéré, Laurent Manchon, Mireille Lejeune, Oliver Clément, Fabien Pierrat, Béatrice Bonafoux, Thérèse Commes, David Piquemal et Jacques Marti : Mining sage data allows large-scale, sensitive screening of antisense transcript expression. *Nucleic Acids Research*, 32(20):e163, 2004. Cité pages 40 et 94.

R. Raman, V. Raman et S. Rao : Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. *In Proc. of Symposium on Discrete Algorithms (SODA)*, pages 233–242, 2002. Cité page 137.

Kim R. Rasmussen, Jens Stoye et Eugene W. Myers : Efficient q-gram filters for finding all epsilon-matches over a given length. *Journal of computational biology : a journal of computational molecular cell biology*, 13(2):296–308, March 2006. Cité pages 52 et 54.

John L. Rinn, Michael Kertesz, Jordon K. Wang, Sharon L. Squazzo, Xiao Xu, Samantha A. Brugmann, L. Henry Goodnough, Jill A. Helms, Peggy J. Farnham, Eran Segal et Howard Y. Chang : Functional demarcation of active and silent chromatin domains in human HOX loci by noncoding RNAs. *Cell*, 129(7):1311–23, juin 2007. Cité pages 24 et 42.

E Rivals, A Boureux, M Lejeune, F Ottones, OP Perez, J Tarhio, F Pierrat, F Ruffle, T Commes et J Marti : Transcriptome Annotation using Tandem SAGE Tags. *Nucleic Acids Res.*, 35:e108, 2007. Cité pages 37, 77, 78 et 117.

Eric Rivals et Sven Rahmann : Combinatorics of Periods in Strings. *J. of Combinatorial Theory series A*, 104:95–113, 2003. Cité page 86.

Eric Rivals, Leena Salmela, Petri Kalsi, Petteri Kiiskinen et Jorma Tarhio : MPSCAN : fast localisation of multiple reads in genomes. *In Proc. 9th Workshop on Algorithms in Bioinformatics (WABI'09)*, volume 5724 de *LNCS*, pages 246–260, 2009. Cité pages 53, 55, 87 et 98.

Guillaume Rizk et Dominique Lavenier : GASSST : global alignment short sequence search tool. *Bioinformatics*, 26(20):2534–2540, octobre 2010. Cité pages 4, 54, 175 et 203.

Adam Roberts, Cole Trapnell, Julie Donaghey, John L. Rinn et Lior Pachter : Improving RNA-seq expression estimates by correcting for fragment bias. *Genome biology*, 12(3):R22+, mars 2011. ISSN 1465-6914. Cité page 39.

Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D. Jackman, Karen Mungall, Sam Lee, Hisanaga M. Okada, Jenny Q. Qian, Malachi Griffith, Anthony Raymond, Nina Thiessen, Timothee Cezard, Yaron S. Butterfield, Richard Newsome, Simon K. Chan, Rong She, Richard Varhol, Baljit Kamoh, Anna-Liisa Prabhu, Angela Tam, YongJun Zhao, Richard A. Moore, Martin Hirst, Marco A. Marra, Steven J. M. Jones, Pamela A. Hoodless et Inanc Birol : De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7(11):909–912, novembre 2010. ISSN 1548-7091. Cité page 63.

Stéphane Robin, François Rodolphe et Sophie Schbath : *DNA, Words and Models*. Cambridge Univ. Press, 2005. Cité pages 81 et 82.

Stephane Robin, Sophie Schbath et Vincent Vandewalle : Statistical tests to compare motif count exceptionalities. *BMC Bioinformatics*, 8(1):84, 2007. Cité page 79.

Mark D. Robinson, Davis J. McCarthy et Gordon K. Smyth : edgeR : a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, janvier 2010. ISSN 1367-4811. Cité pages 64 et 92.

Joel S. Rozowsky, Daniel Newburger, Fred Sayward, Jiaqian Wu, Greg Jordan, Jan O. Korbel, Ugrappa Nagalakshmi, Jin Yang, Deyou Zheng, Roderic Guigó, Thomas R. Gingeras, Sherman Weissman, Perry Miller, Michael Snyder et Mark B. Gerstein : The DART classification of unannotated transcription within the ENCODE regions : Associating transcription with known and novel loci. *Genome Research*, 17(6):732–745, juin 2007. Cité pages 1, 20, 24, 25 et 42.

Stephen M. Rumble, Phil Lacroute, Adrian V. Dalca, Marc Fiume, Arend Sidow et Michael Brudno : Shrimp : Accurate mapping of short color-space reads. *PLoS Comput Biol*, 5(5):e1000386+, May 2009. Cité page 54.

S Saha, AB Sparks, C Rago, V Akmaev, CJ Wang, B Vogelstein, KW Kinzler et VE Velculescu : Using the transcriptome to annotate the genome. *Nat Biotechnol*, 20:508–12, 2002. Cité pages 22, 37, 78, 115, 116 et 117.

Meena Kishore Sakharkar, Vincent T K Chow et Pandjassarame Kangueane : Distributions of exons and introns in the human genome. *In Silico Biology*, 4(4):387–393, 2004. ISSN 1386-6338. Cité pages 154 et 165.

Leena Salmela : Correction of sequencing errors in a mixed set of reads. *Bioinformatics*, 26(10): 1284–1290, 2010. Cité page 126.

Leena Salmela et Jan Schröder : Correcting errors in short reads by multiple alignments. *Bioinformatics*, 27(11):1455 –1461, juin 2011. Cité page 58.

M. Salson : *Structures d'indexation compressées et dynamiques pour le texte*. Thèse de doctorat, Université de Rouen, Dec 2009. Cité pages 48 et 51.

Michael Sammeth, Sylvain Foissac et Roderic Guigó : A general definition and nomenclature for alternative splicing events. *PLoS computational biology*, 4(8):e1000147+, 2008. ISSN 1553-7358. Cité pages 1, 42, 61, 63 et 198.

Andrea Sboner, Lukas Habegger, Dorothee Pflueger, Stephane Terry, David Chen, Joel Rozowsky, Ashutosh Tewari, Naoki Kitabayashi, Benjamin Moss, Mark Chee, Francesca Demichelis, Mark Rubin et Mark Gerstein : FusionSeq : a modular framework for finding gene fusions by analyzing paired-end RNA-sequencing data. *Genome Biology*, 11(10):R104+, octobre 2010. ISSN 1465-6906. Cité pages 65, 71, 72, 155 et 191.

Jan Schröder, Heiko Schröder, Simon J. Puglisi, Ranjan Sinha et Bertil Schmidt : Shrec : a short-read error correction method. *Bioinformatics*, 25(17):2157–2163, 2009. Cité pages 57 et 58.

Stephan C. Schuster, Webb Miller, Aakrosh Ratan, Lynn P. Tomsho, Belinda Giardine, Lindsay R. Kasson, Robert S. Harris, Desiree C. Petersen, Fangqing Zhao, Ji Qi, Can Alkan, Jeffrey M. Kidd, Yazhou Sun, Daniela I. Drautz, Pascal Bouffard, Donna M. Muzny, Jeffrey G. Reid, Lynne V. Nazareth, Qingyu Wang, Richard Burhans, Cathy Riemer, Nicola E. Wittekindt, Priya Moorjani, Elizabeth A. Tindall, Charles G. Danko, Wee S. Teo, Anne M. Buboltz, Zhenhai Zhang, Qianyi Ma, Arno Oosthuysen, Abraham W. Steenkamp, Hermann Oostuisen, Philippus Venter, John Gajewski, Yu Zhang, B. Franklin Pugh, Kateryna D. Makova, Anton Nekrutenko, Elaine R. Mardis, Nick Patterson, Tom H. Pringle, Francesca Chiaromonte, James C. Mullikin, Evan E. Eichler, Ross C. Hardison, Richard A. Gibbs, Timothy T. Harkins et Vanessa M. Hayes : Complete Khoisan and Bantu genomes from southern Africa. *Nature*, 463(7283):943–947, février 2010. ISSN 0028-0836. Cité pages 33 et 144.

Jay Shendure et Hanlee Ji : Next-generation DNA sequencing. *Nature biotechnology*, 26(10):1135–1145, octobre 2008. ISSN 1546-1696. Cité pages 1 et 33.

Fei Shi : Suffix arrays for multiple strings : A method for on-line multiple string searches. *In* Joxan Jaffar et Roland H. C. Yap, éditeurs : *ASIAN*, volume 1179 de *Lecture Notes in Computer Science*, pages 11–22. Springer, 1996. ISBN 3-540-62031-1. Cité page 123.

L. Shi, LH Reid et WD et al. Jones : The microarray quality control (maqc) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nat Biotech*, 24:1151–61, 2006. Cité page 22.

Michael H Sieweke et Thomas Graf : A transcription factor party during blood cell differentiation. *Current Opinion in Genetics & Development*, 8(5):545–551, octobre 1998. ISSN 0959-437X. Cité page 24.

APM Silva, JES De Souza, PAF Galante, GJ Riggins, SJ De Souza et AA Camargo : The impact of SNPs on the interpretation of SAGE and MPSS experimental data. *Nucleic Acids Res.*, 32:6104–6110, 2004. Cité page 117.

Manabu Soda, Young Lim Choi, Munehiro Enomoto, Shuji Takada, Yoshihiro Yamashita, Shunpei Ishikawa, Shin-ichiro Fujiwara, Hideki Watanabe, Kentaro Kurashina, Hisashi Hatanaka, Masashi Bando, Shoji Ohno, Yuichi Ishikawa, Hiroyuki Aburatani, Toshiro Niki, Yasunori Sohara, Yukihiko Sugiyama et Hiroyuki Mano : Identification of the transforming EML4-ALK fusion gene in non-small-cell lung cancer. *Nature*, 448(7153):561–566, 2007. ISSN 0028-0836. Cité page 67.

P. J. Stephens, D. J. McBride, M. L. Lin, I. Varela, E. D. Pleasance, J. T. Simpson, L. A. Stebbings, C. Leroy, S. Edkins, L. J. Mudie, C. D. Greenman, M. Jia, C. Latimer, J. W. Teague, K. W. Lau, J. Burton, M. A. Quail, H. Swerdlow, C. Churcher, R. Natrajan, A. M. Sieuwerts, J. W. Martens, D. P. Silver, A. Langerod, H. E. Russnes, J. A. Foekens, J. S. Reis-Filho, L. van 't Veer, A. L. Richardson, A. L.

Borresen-Dale, P. J. Campbell, P. A. Futreal et M. R. Stratton : Complex landscapes of somatic rearrangement in human breast cancer genomes. *Nature*, 462(7276):1005–1010, décembre 2009. Cité pages 27 et 63.

Philip J. Stephens, Chris D. Greenman, Beiyuan Fu, Fengtang Yang, Graham R. Bignell, Laura J. Mudie, Erin D. Pleasance, King W. Lau, David Beare, Lucy A. Stebbings, Stuart McLaren, Meng-Lay Lin, David J. McBride, Ignacio Varela, Serena Nik-Zainal, Catherine Leroy, Mingming Jia, Andrew Menzies, Adam P. Butler, Jon W. Teague, Michael A. Quail, John Burton, Harold Swerdlow, Nigel P. Carter, Laura A. Morsberger, Christine Iacobuzio-Donahue, George A. Follows, Anthony R. Green, Adrienne M. Flanagan, Michael R. Stratton, P. Andrew Futreal et Peter J. Campbell : Massive genomic rearrangement acquired in a single catastrophic event during cancer development. *Cell*, 144(1):27–40, janvier 2011. Cité pages 2, 63, 65 et 67.

Marc Sultan, Marcel H. Schulz, Hugues Richard, Alon Magen, Andreas Klingenhoff, Matthias Scherf, Martin Seifert, Tatjana Borodina, Aleksey Soldatov, Dmitri Parkhomchuk, Dominic Schmidt, Sean O'Keeffe, Stefan Haas, Martin Vingron, Hans Lehrach et Marie-Laure Yaspo : A Global View of Gene Activity and Alternative Splicing by Deep Sequencing of the Human Transcriptome. *Science*, 321:956–960, 2008. Cité pages 1, 37 et 119.

Shingo Suzuki, Naoaki Ono, Chikara Furusawa, Bei-Wen W. Ying et Tetsuya Yomo : Comparison of sequence reads obtained from three next-generation sequencing platforms. *PloS one*, 6(5): e19534+, mai 2011. ISSN 1932-6203. Cité page 33.

The ENCODE Project Consortium : Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447:799–816, 2007. Cité pages 1, 17, 20 et 25.

J. Thorne, M. Campbell et B. Turner : Transcription factors, chromatin and cancer. *The International Journal of Biochemistry & Cell Biology*, 41(1):164–175, janvier 2009. ISSN 13572725. Cité page 24.

Scott A. Tomlins, Daniel R. Rhodes, Sven Perner, Saravana M. Dhanasekaran, Rohit Mehra, Xiao-Wei W. Sun, Sooryanarayana Varambally, Xuhong Cao, Joelle Tchinda, Rainer Kuefer, Charles Lee, James E. Montie, Rajal B. Shah, Kenneth J. Pienta, Mark A. Rubin et Arul M. Chinnaiyan : Recurrent fusion of TMPRSS2 and ETS transcription factor genes in prostate cancer. *Science*, 310 (5748):644–648, octobre 2005. ISSN 1095-9203. Cité page 67.

Cole Trapnell, Lior Pachter et Steven L. Salzberg : Tophat : discovering splice junctions with rna-seq. *Bioinformatics (Oxford, England)*, 25(9):1105–1111, May 2009. Cité pages 5, 60, 62 et 175.

Cole Trapnell et Steven L. Salzberg : How to map billions of short reads onto genomes. *Nature biotechnology*, 27(5):455–457, May 2009. Cité pages 2, 35, 57 et 154.

Cole Trapnell, Brian A. Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J. van Baren, Steven L. Salzberg, Barbara J. Wold et Lior Pachter : Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, mai 2010. ISSN 1087-0156. Cité pages 33, 63, 64, 71 et 126.

E. Ukkonen : On-line construction of suffix-trees. *Algorithmica*, 14(3):249–260, 1995. Cité pages 44 et 45.

Harm van Bakel, Corey Nislow, Benjamin J. Blencowe et Timothy R. Hughes : Most "dark matter" transcripts are associated with known genes. *PLoS Biol*, 8(5):e1000371+, mai 2010. ISSN 1545-7885. Cité pages 25, 78, 92 et 110.

VE Velculescu, L Zhang, B Vogelstein et KW Kinzler : Serial analysis of gene expression. *Science*, 270 (5235):484–7, 1995. Cité pages 22 et 117.

Victor E. Velculescu et Kenneth W. Kinzier : Gene expression analysis goes digital. *Nat Biotechnol*, 25:878–880, 2007. Cité pages 35, 42 et 115.

Marc Via, Christopher Gignoux et Esteban González G. Burchard : The 1000 genomes project : new opportunities for research and social challenges. *Genome medicine*, 2(1):3+, janvier 2010. ISSN 1756-994X. Cité pages 1, 17, 33 et 42.

Andreas von Bubnoff : Next-generation sequencing : the race is on. *Cell*, 132(5):721–723, mars 2008. Cité pages 1 et 32.

Kai Wang, Darshan Singh, Zheng Zeng, Stephen J. Coleman, Yan Huang, Gleb L. Savich, Xiaping He, Piotr Mieczkowski, Sara A. Grimm, Charles M. Perou, James N. MacLeod, Derek Y. Chiang, Jan F. Prins et Jinze Liu : MapSplice : accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic acids research*, 38(18):e178, octobre 2010. ISSN 1362-4962. Cité pages 5, 60, 62, 154 et 175.

Zhong Wang, Mark Gerstein et Michael Snyder : RNA-seq : a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, janvier 2009. Cité pages 1, 32, 39, 40 et 237.

Robert H Waterston, Kerstin Lindblad-Toh, Ewan Birney, Jane Rogers, Josep F Abril, Pankaj Agarwal, Richa Agarwala et al : Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562, décembre 2002. ISSN 0028-0836. Cité page 20.

David Weese, Anne-Katrin Emde, Tobias Rausch, Andreas Döring et Knut Reinert : Razers—fast read mapping with sensitivity control. *Genome Research*, 19(9):1646–1654, September 2009. Cité page 54.

P. Weiner : Linear pattern matching algorithms. *In Conf. Record of the 14th Annual Symposium on Swithcing and Automata Theory*, 1973. Cité page 122.

Brian T. Wilhelm, Samuel Marguerat, Ian Goodhead et Jurg Bahler : Defining transcribed regions using RNA-seq. *Nature Protocols*, 5(2):255–266, janvier 2010. Cité page 39.

Barbara Wold et Richard M. Myers : Sequence census methods for functional genomics. *Nature Methods*, 5(1):19–21, décembre 2007. ISSN 1548-7091. Cité page 32.

Thomas D. Wu et Serban Nacu : Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, avril 2010. ISSN 1367-4811. Cité pages 4, 5, 54, 60, 62, 175 et 203.

Zhenhua Jeremy J. Wu, Clifford A. Meyer, Sibgat Choudhury, Michail Shipitsin, Reo Maruyama, Marina Bessarabova, Tatiana Nikolskaya, Saraswati Sukumar, Armin Schwartzman, Jun S. Liu, Kornelia Polyak et X. Shirley Liu : Gene expression profiling of human breast tissue samples using SAGE-seq. *Genome research*, 20(12):1730–1739, décembre 2010. ISSN 1549-5469. Cité page 35.

Table des figures

Crédit photo

Dans le chapitre 1 :

1. La figure 1.1 est extraite de la thèse d'Alban Mancheron [Mancheron, 2006].
2. La figure 1.2 est extraite de la thèse d'Alban Mancheron [Mancheron, 2006].
3. La figure 1.5 est extraite de wikipedia.
4. La figure 1.4 est extraite de la thèse d'Alban Mancheron [Mancheron, 2006].
5. La figure 1.6 est extraite de wikipedia.
6. La figure 1.8 est extraite de wikipedia.
7. La figure 1.7 est extraite de l'article « What is a gene, post-ENCODE ? History and updated definition » [Gerstein *et al.*, 2007].

Dans le chapitre 2 :

1. La figure 2.1 est extraite de l'article « Next-generation *tag* sequencing for cancer gene expression profiling » [Morrissy *et al.*, 2009].
2. La figure 2.2 est extraite de l'article « RNA-Seq : a revolutionary tool for transcriptomics » [Wang *et al.*, 2009].
3. La figure 2.5 est extraite de l'article « Implications of chimaeric non-co-linear transcripts » [Gingeras, 2009].
4. La figure 2.7 est extraite de l'article « Short Homologous Sequences Are Strongly Associated with the Generation of Chimeric RNAs in Eukaryotes » [Li *et al.*, 2008d].

Liste des tableaux

Liste des exemples

Liste des Algorithmes

Abstract

Since their introduction, high-throughput sequencers have revolutionized transcriptomic studies at genome scale. Indeed, they have the ability to generate millions, or even billions of short sequences, called *reads*. New transcriptomic approaches, such as Digital Gene Expression (DGE) and RNA-sequencing (RNA-Seq), enable the identification, quantification, and reconstitution of all transcripts of the cell, even rare ones. Among these transcripts are regulatory non-coding RNAs, alternative splice variants, which code for novel proteins, but also non colinear transcripts termed chimeras (generated by either gene fusion or trans-splicing). The characterization of these transcripts constitutes a sheer algorithmic, but also a biological challenge due to their differences in nature, their diverse implications in physiological and cellular processes, and for some their role in cancer development.

In this work, we focus on algorithms and methods for the characterization and annotation of transcriptomes. First, we proposed a statistical study on DGE to assess the impact of sequence errors on the analysis. Therefrom, we developed a pipeline for the DGE annotation. Through this initial work, we demonstrated that a lot of information is shared between the *reads*. This property led us to design, the *Gk arrays*, an indexing data structure for organizing huge amounts of reads in memory and algorithms to quickly query this structure. Finally, based on the *Gk arrays* we have conceived, CRAC, a software specialised in the RNA-Seq processing. By integrating its own mapping process, CRAC is able to distinguish the biological phenomena from sequence errors. Moreover, it allows to identify chimeric RNAs, which may be weakly expressed in a transcriptome and are inherently complex to detect since their fragments originate from different places on the genome.

Keywords: *algorithmics, bioinformatics, transcriptomics, cancer, chimeras, indexing, splicing junctions,* reads*, RNA-Seq, high-troughput sequencer, stringology, suffix arrays, transcriptome.*

Résumé

Depuis leur apparition, les séquenceurs haut débit ont révolutionné l'étude des transcriptomes à l'échelle du génome. En effet, ils offrent la possibilité de générer des millions, voire des milliards de séquences, appelées *reads*. Des nouvelles approches transcriptomiques, telles que la *Digital Gene Expression* (DGE) et le *RNA-Sequencing* (RNA-Seq), permettent aujourd'hui de répertorier, de quantifier, voire reconstruire tous les transcrits d'une cellule, même les plus rares. Parmi ce type de transcrits se trouvent des ARN non-codants régulateurs ; des variants d'épissages créateurs de protéines ; et aussi des chimères (par fusion de gènes ou trans-épissage). La caractérisation de l'ensemble de ces transcrits représente un réel défi algorithmique, mais suscite aussi un défi biologique car certains peuvent être impliqués dans de nombreux processus cellulaires physiologiques et pathologiques et sont fréquemment décrits dans les cancers.

Dans ce travail, nous proposons des algorithmes et des méthodes pour la caractérisation et l'annotation des transcriptomes. Tout d'abord, nous proposons une étude statistique sur la DGE afin d'évaluer l'impact des erreurs de séquences lors de l'analyse des *reads*. À partir de cette analyse, nous avons développé un pipeline d'annotation pour la DGE. Par le biais de ce premier travail, nous avons pu démontrer que de nombreuses informations étaient partagées entre les reads. Cela nous a amené à concevoir la structure d'indexation *Gk arrays* qui permet d'organiser une quantité massive de *reads* de façon à pouvoir interroger rapidement la structure sous forme de requêtes. Enfin, en s'appuyant sur les *Gk arrays*, nous avons développé CRAC qui est un logiciel spécialisé dans le traitement du RNA-Seq. En intégrant sa propre phase de *mapping*, CRAC est capable de distinguer les phénomènes biologiques des erreurs de séquences. Il permet notamment l'identification de chimères qui sont souvent très faiblement exprimées dans un transcriptome et sont par nature complexe à détecter avec des parties localisées à différents endroits sur le génome.

Mots clefs : *algorithmique du texte, bioinformatique, transcriptomique, cancer, chimères, indexation, jonctions d'épissages,* reads*, RNA-Seq, séquenceurs haut débit, stringology, table des suffixes, transcriptome.*

LIRMM — 161, rue Ada — 34095 Montpellier cedex 5 — France