From reads to transcripts: de novo methods for the analysis of transcriptome second and third generation sequencing.

Camille Marchet

2018

UNIVERSITÉ DE RENNES 1

UNIVERSITE BRETAGNE LOIRE

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale Math-STIC

présentée par

Camille Marchet

préparée à l'unité de recherche n^o 6074 IRISA
Institut de Recherche en Informatique et Systèmes Aléatoires
(Composante universitaire)

From reads to transcripts: *de novo* methods for the analysis of transcriptome second and third generation sequencing

Thèse soutenue à Rennes le 28/09/2018

devant le jury composé de :

Éric Coissac
Maitre de Conférences - Rapporteur

Hélène Touzet
Directrice de Recherche - Rapporteuse

Thomas Derrien
Chargé de Recherche - Examinateur

Dominique Lavenier
Directeur de Recherche - Examinateur

Thierry Lecroq
Professeur - Examinateur

Anne Siegel
Directrice de Recherche - Examinatrice

Hagen Tilgner
Assistant Professor - Examinateur

Pierre Peterlongo
Chargé de Recherche - Directeur

Contents

Remerciements

On n'a pas souvent la chance d'avoir une tribune libre pour se retourner et remercier les gens qui ont avancé avec nous dans la vie, voilà comment je considère le texte qui suit. Si vous vous ennuyez un peu à ma soutenance, je vous propose de tenter de compter le nombre de "merci" qui apparaissent dans ce texte afin que l'on rigole un peu de la faiblesse de ma prose. En échange j'offre un cadeau au première ou à la première à me donner la bonne réponse.

Tous les parcours ne sont pas tout tracés, dans le mien il existe un certain nombre d'heureux malentendus ou de hasards. Je pense à Monsieur Doussot, qui, mon premier jour d'école en petite section me laisse apprendre la comptine des grands à travers la cloison car je m'ennuie. A Monsieur Bedrune au collège, les cours d'arts plastiques sont nuls mais pas sa longue digression sur un bouquin de Sven Ortoli. Après ça, c'est une sorte de petit rêve de faire de la recherche (en physique quantique, bon). Merci à Alex d'avoir conforté mon envie de partir à Lyon. Merci à ce monsieur passionné, en sortie ornitho à Guérande, qui me fait choisir la bioinfo sur un coup de tête en école d'ingénieur, parce que je me dis que j'aimerais bien travailler pour les biologistes.
Je pense à celles et ceux à qui je m'accroche en pensée quand la perspective de faire de la bonne science semble lointaine. Federica et ses cours de bio cell, qui personnalise l'amour de sa disciplince combiné au recul sur sa profession, et qui donne à la biologie beaucoup plus de relief qu'une suite de connaissance à ingurgiter. Dans la même veine, Laurent P et Hedi. Mes enseignants passionés Madame Levret, Monsieur Laloé et Monsieur Pautet, ainsi que Monsieur Leroy qui est finalement mon unique prof d'anglais en quelque sorte.

Un immense merci à Vincent et Marie-France pour m'avoir fait confiance et mise sur les rails à mes débuts. Vous êtes pour moi l'exemple d'une science à visage humain. Merci Vincent de m'avoir accordé ta confiance et de m'avoir aidé à décrocher un financement de thèse malgré mon CV... particulier. Merci d'être un modèle scientifique, même si je mets parfois du temps à prendre conscience des réelles qualités dont on a besoin pour être un bon chercheur. L'aventure a commencé pour moi un jour de réunion ANR Colib'read à Paris, j'ai donc une pensée particulière pour Gustavo, Alice, Blerina, Rayan, Raluca, Hélène, Claire, Clara, Eric, Bastien, et tous les autres membres de ce groupe.
A Lyon je salue tous mes amis et amies, mes co-auteurs et co-autrices et les collègues que j'ai côtoyés à ce moment au labo et à la plateforme de bioinfo. En particulier Guy qui me soutient toujours beaucoup, Christian G, Lilia, Janice, Frank, Vincent N, Clothilde, Amandine, Laurent J, Ricardo, Mariana, Laura, Sheila, Leandro, Philippe, Vincent M, Camille, Magali, Christine et Arnaud, Sylvère ainsi que Mathilde, Susan, Laurent B (et Tigrane), Florence Martin et Emilie C pour les anciens.
Je continue sur les ANR et les collaborations en remerciant mes collègues d'ASTER, ANR qui a été un environnement très enrichissant, voire salvateur, pour moi pendant ma thèse. Merci une seconde fois Leandro d'être un si chouette collègue, Rayan pour tes conseils très avisés, Hélène, Jean-Marc et Corinne pour toute votre aide. Merci aussi aux collègues d'HYDROGEN, Mahendra, Stéphane, Eric P, Olivier. Vous avez été nombreux à m'envoyer un petit mail d'encouragement pour la dernière ligne droite et j'en profite pour remercier Etienne D, Pierre et Claude de Nantes (je regrette qu'on ne se soit pas plus vus !). Je n'oublie pas Thierry, Arnaud et Pierre M et toute l'équipe ELECTOR, Jean-François et toute l'équipe BWISE, Arnaud,

Lucie, Fabrice, Erwan et tous les pros des Radiolaires.

J'en profite pour remercier mon jury de thèse, Hélène, Thierry, Dominique, Eric, Anne et Thomas, ainsi que Christine, Hagen venu de vraiment très loin. Merci d'avoir pris le temps de lire mon manuscrit et de vous être impliqués dans mon travail. Un grand merci Anne d'avoir assuré un remplacement in extremis au pied levé et avec le sourire.
Mes collègues rennais n'étant pas en reste, merci dans un parfait désordre à Maud de m'avoir permis d'enseigner une super UE aux informaticiens de l'INSA avec la plus grande liberté, aux personnes du GT égalité femme/homme que je regrette d'avoir intégré si tard, et à Anne en particulier, à tout le personnel administratif du labo, de l'université et de l'école doctorale et à Elodie Cottrel qui a été une interlocutrice très agréable, ainsi qu'Anne Buzaré.
Bien sûr je salue Symbiose dans son ensemble, les membres passés et présents. Je pense à ses sous-ensembles comme l'éphémère équipe des 4C, tous mes co-bureaux : Cervin, Julie, Patrick (on continue à se montrer les tatouages ok ?), Maël, Arnaud : merci d'avoir une meilleure tête que moi le matin sinon on ne s'en sortirait pas. Mes anciens stagiaires, Benjamin et Lolita je vous souhaite que tout fonctionne selon vos plans dans vos thèses respectives. La Triforce dont j'ai l'honneur de faire partie en compagnie de mes sempaï Gatou et Antoine, est l'entité la plus puissance issue de GenScale, sans aucun doute possible. La preuve, elle essaime en banlieue GenScale nord, je suis contente que l'aventure se poursuive à Lille les garcons ! Dans l'équipe GenScale je remercie chaleureusement Dominique, pour tes encouragements en particulier au début où tout était bien flou, et pour tes nombreux conseils pour la suite, Fabrice et Claire pour vos retours pertinents ("j'ai pas bien compris..!") sur mon travail, Rumen et Seb même si on n'a pas encore transformé l'essai, et bien sûr Jacques, tu t'es jeté sans hésiter dans l'aventure CARNAC-LR et tu apportes ta dose de positif et de zen tous les jours (sans doute grâce à son assidue pratique du yoga). Chez les abonnés au séminaire précaire, j'appelle Hugo, Wesley (tire profit de mon enseignement sur le Hibou stp), Seb (doucement avec l'escalade), Lucas (j'espère qu'on menèra au moins un de nos fameux projets au bout), Arnaud, Maël, Marine, Chloé, Marie, Meziane (on ira en Islande promis j'achète les billets demain), Méline, Maxime, Pierre V, Jérémy, Cervin et Clémence (bon courage vous deux pour vos soutenances !), et Lolita (t'es la meilleure). A toutes et tous je vous souhaite que la suite vous sourie. Merci à tous mes collègues de Genouest, et évidemment à Marie qui assure dans ce job difficile d'organiser la vie de têtes en l'air comme nous.

Enfin, merci à Pierre d'avoir toujours été très positif avec moi, d'avoir mouillé son bison sans peur sur un sujet pas prévu à la base. Il est possible que je te doive des choses dont j'ignore encore l'existence, manque de sagesse oblige. Merci en tout cas de m'avoir laissée très libre quitte à faire des erreurs, comme par exemple et totalement au hasard, faire une thèse sur ce qui aurait pu être trois sujets de thèse différents. Je suis très contente de ce que l'on a accompli et je sais que nous ne sommes qu'à nos premiers succès. Merci surtout d'avoir été compréhensif quand à ma situation familiale, et de m'avoir laissé le large dont j'avais besoin en toute confiance. Tout le monde ne l'aurait pas accepté les yeux fermés, et grâce à ça j'ai pu faire ce qui comptait réellement pour moi sans regrets. Ca me touche beaucoup.
Je ferme cette page scientifique pour en rouvrir une nouvelle, en remerciant Mikael et Rayan qui vont maintenant m'accueillir à Lille pour mon postdoc.

Salut à tous les doctorants, jeunes chercheurs, frères ou meilleurs amis qui voguent ou ont vogué sur la même galère que moi et que je n'ai pas encore cités, Tristan, Muller, Guillaume, Gilles, Kévin, Florent, Victor, Ardi, Joris, Président, Présidente, Romain, Yoann, Bastien. Merci Guillaume d'être venu tout exprès cette semaine pour filer un coup de main et refaire le monde. Vivien, Stéphane, Alex, Xouille, Aurélia, Amandine et Paul, vous qui avez choisi une vie sans doctorat et êtes bien plus zen et dix fois plus riches (?), je vous aime pareil. Ceux de l'IRISA, Simon, Hugo, Tristan et Gus : merci pour le café, les Chanya du vendredi, les conversions aux végétarismes et le soutien mutuel dans l'adversité ! A l'équipe de com chargée de mes affiches de thèse je decerne le prix du résultat le plus hilaro-cryptique. Et puis il y a d'autres rennais et rennaises qui ont participé à ma vie durant ces trois ans et que je n'ai pas encore évoqués : Laurent B, Mathias, Aurore, Benjamin B, Aymeric et Shannon. Je tire mon chapeau à ceux qui m'ont supportée

au quotidien ces trois dernières années, c'est-à-dire essentiellement Gurvan, Gaëtan et Simon. Et Antoine l'inclassable, support indéfectible que je refuse d'écouter une fois sur deux, inutile de résumer ici tout ce que j'ai appris avec toi en venant à Rennes. Tu dois être la personne qui me connait le mieux en tant que collègue grognonne. Tu es un pilier dans ma vie et je te remercie vraiment pour ces derniers mois. On se doit beaucoup et je me réjouis de tous les projets à venir.

Non moins importants, il y a aussi ceux et celles qui tâchent de m'extraire de la recherche et des sciences de temps en temps. Jean Boulat qui m'accueillit en section arts plastiques à l'INSA. Cécile ma prof de yoga qui a donné beaucoup plus d'intensité à ma pratique. Tim qui tous les jours mène sa quête pour être plus libre. La team du jap, championne du monde de spam. Impossible de ne pas citer la bande des Lyonnais, natifs, exilés ou d'adoption, qui me rappelle tous les jours qu'un autre monde que celui de la recherche existe. Ben, Pierre, Thomas, Marina, Antoine, Tristan, Thibaud, Muller, Aurélien, Duff, Eric, Elwing et les autres qui serais-je sans vous ? Une personne qui rit moins. J'ajoute Lisa et Eloïse qui sont mes plus vieilles amies et que je suis fière de garder près de moi, malgré nos chemins si différents; Arnaud le toulousain et le canadien Léo; et mes frères et soeurs, Romane Pierre et Max qui ne se sentent pas obligés de faire comme moi, et tant mieux.
Merci à toute ma famille sur qui je peux toujours compter, Stèphe, Yannick, Isa, Eugénie, Paul, et puis Jean-Marie et papa puisqu'on dirait que j'ai maintenant deux papas. De chacun de vous deux je tire la rigueur (un peu) et l'assurance (un peu aussi, je fais ce que je peux). Vous avez eu des épaules bienvenues surtout ces derniers temps, et vous m'avez toujours fait confiance sans essayer de m'influencer pour choisir ma trajectoire, c'est une grande chance. Je pense aussi à mes tantes et oncles, et cousins-cousines de Paris et de Bourgogne, mais également à l'Invicible Armada au complet et au Renarts avec un t qui tous ensemble sont un grand réservoir de bienveillance à mon égard. Papi et Gracie vous me sauvez la vie chaque été en m'accueillant à la Cabanasse, je pense fort à vous. J'ajoute qu'il m'est impossible d'exister sans mes modèles de femmes dans la vie, moteurs et soutiens : ma grand-mère qui est une des personnes les plus fortes et les plus ouvertes que je connaisse, Isabelle, j'aimerais avoir ton pouvoir d'insuffler la joie de vivre partout où tu passes, Mireille Dominique et Jeanine qui sont encore autant de femmes fortes, drôles et pleines de vie.
Enfin maman. Il y a trois ans je me demandais si tu serais présente le jour de ma soutenance. Il y a six mois je me demandais si tu pourrais au moins la voir, même de loin. Récemment je t'ai rassurée en te disant que ça n'était pas si grave et que c'était au fond une formalité. Qui ce qui comptait c'était tout le reste. Il y a du vrai là dedans, au regard des 29 ans où tu m'as quotidiennement accompagnée, une journée de plus ou de moins ce n'est grand chose. Mais la vérité c'est aussi que tout est plus douloureux que prévu car tu me manques. C'est étrange d'écrire pour quelqu'un qui ne lira pas, mais la force, la constance et l'envie d'aller de l'avant que tu possédais et que tu m'as patiemment inculquées, je les garde pour me souvenir et pour te ressembler un peu.

En résumé j'ai la chance de vous avoir nombreux, nombreuses près de moi et je vous dois beaucoup. Un grand merci.

Preamble

This manuscript is organized in five chapters plus an introduction, a conclusion and perspectives. In the introduction, general concepts pertaining to RNA and key biological notions that will be later invoked are presented. These concepts describe biological realities that are for a large part non fully solved: this is also the occasion to present some fundamental questions in RNA biology. These questions motivate the development of bioinformatics methods, to which the work presented in this dissertation contributes. As nucleic sequence biology is nowadays inseparable from the technologies that give access to sequences, they are presented as well in the introduction. As these technologies deliver imperfect and fragmented information, whose nature have been changing through years, they importantly impacted the development of bioinformatics methods. Sometimes challenges due to these technologies supplant biological questions, but they also open new exciting informatics problems.
The main discipline of this thesis subject is informatics, though being cross-disciplinary it brings readers that belong to a broader scientific audience. We judged useful to provide a summary of indispensable notions in graph and text algorithmics to the reader. They recall the notions encompassed within the whole manuscript that are helpful to globally access the presented work. Introduction ends with a short presentation of the contributions.

This thesis subject was thought as being composed of three main steps, that require various and sometimes quite distant methodologies. The aim of this thesis is to enable the processing of RNA sequencing from the sequencer output to the isoform detection, its central theme being the biological and technological nature of data. Thus we made the choice to treat several methodological steps of a pipeline processing these reads, instead of focusing on only one step. This explains why most of the broached notions do not come with a extensive state of the art. We preferred picking what seemed to us essential matter to correctly address the different encountered issues. It is noteworthy that this thesis focuses mainly on new sequencing technologies producing so-called "long reads" (in comparison to short reads produced by technologies anterior to 2011), which only benefited from a sparse methodological literature for RNA at the beginning of our work.

We made the choice to introduce more precise algorithmic notions as the necessity arises all along the manuscript. We hope this helps easing the reading, since this thesis spans several distinct problems in bioinformatics. Thus each chapter starts with a presentation of its methodological background and issues. Upon needed, a particular biological notion of interest can be developed in a chapter. This is for instance the case in Chapter 1 where the concept of holobiont is presented. In the same spirit, this is meant to avoid overloading or breaking up the introduction's consistency.
Main covered domains are pairwise sequence comparison in Chapter 1, sequence clustering in Chapter 2, multiple sequence alignment and sequence correction in Chapters 3 and 4. Though sometimes going back to DNA applications is needed, all the methods presented in these chapters are meant to be committed to RNA sequences as final goal. Chapters 1 and 2 represent accomplished works, published or in press in conferences proceedings and journals [146, 145, 154], or in revision [144]. They present novel methods and results. Chapter 3 and 4 depict more exploratory works, with ongoing methodological researches and preliminary results. Chapter 4 is also an attempt to methodologically synthesize and link together works presented in previous chapters. Some works on short reads are only briefly mentioned [132, 138, 16, 119] in Chapter 5. We participated in their realization to various extents during this thesis, but they do not constitute the core contribution of our work. Finally a few works are not presented in this document, since they are early projects or are more distant from the central theme.

Conclusion presents a summary of the achievements and discusses their impact. Perspectives section proposes concrete improvements for each main part, and bring prospective works out.

Introduction

1 What is RNA from molecular biology point of view

1.1 Introduction to nucleic acids

1.1.1 DNA: an informational and historical basis

DNA (deoxyribonucleic acid) is the carrier molecule of hereditary information in the majority of the living and is present in its three main domains: Eukarya (organisms with cells that have a nucleus bounded by a membrane), Bacteria and Archaea (both prokaryotes, that do not have nucleus). It is a homopolymer composed of four different nucleic acids called nucleotides. A chain or sequence of these nucleotides is called a DNA strand. DNA nucleotides adenine (A), cytosine (C), thymine (T) and guanine (G) have the property to make pairs. A pairs with T, G with C. For instance the chain ACGT can make a pair with the chain TGCA. These chains, also called strands, are oriented, in the sense that each end is not terminated by the same chemical group. We call the 5' end the one having a terminal phosphate group, conversely the 3' end has a terminal hydroxyl group. This asymmetry gives DNA its directionality. DNA strands thus hybridize by pairs (attached by hydrogen bonds), each strand having the same length, corresponding nucleotides forming pairs (said complements) and running in the opposite direction (they are called antiparallel). One strand is then called the reverse complement of the other. A DNA molecule is composed of these two strands bonded in a helical fashion in three dimensions (right in Figure 1).

DNA is a long molecule which storage in the cell can be extremely optimized in space. In many organisms, it does not only stores sequences called genes, but also regions that regulate the activity of those genes, as well as other sequences of diverse roles. Notions of genes and genomes being models, they can harbor slightly different definitions according to the eye of whom studies. We will call a gene a DNA sub-sequence that

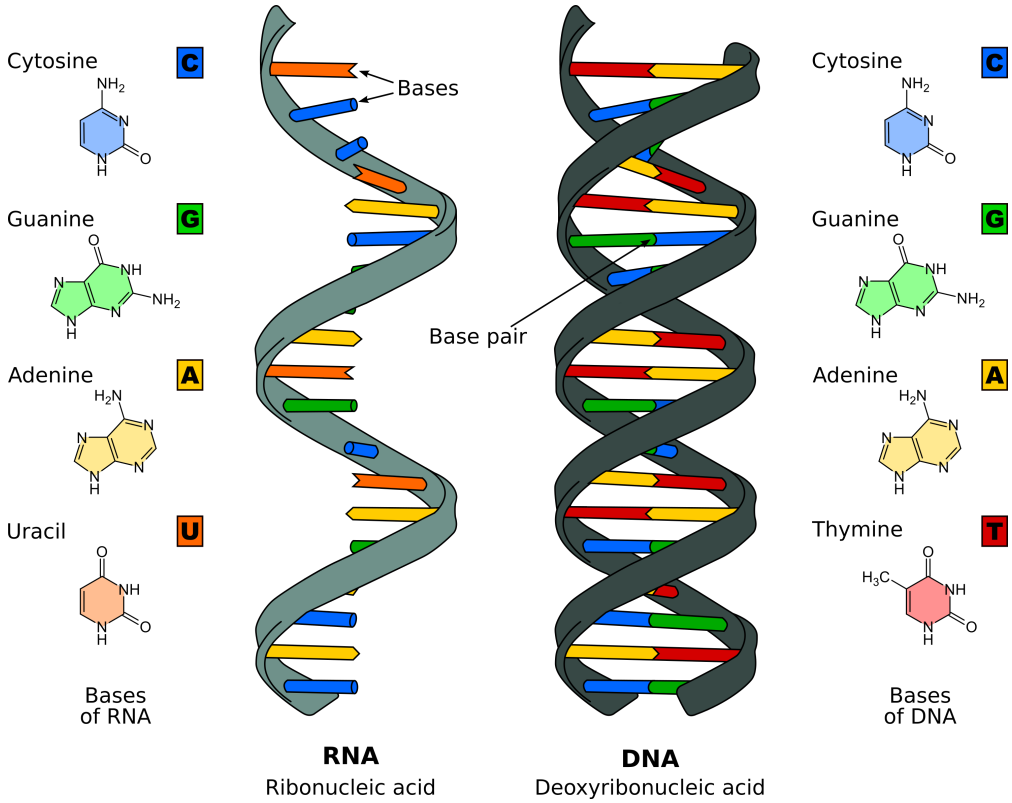

Figure 1: **Molecules of RNA (left) and DNA (right).** Figure adapted from [41].

encodes instruction for any functional process in a cell. In eukaryotes, DNA is stored and well separated from the rest of the cell in the nucleus (it is not the case in prokaryotes). It is supported by one or more structures called chromosomes that imply several proteins to keep the very long DNA molecule condensed. The whole set of genetic material in chromosomes is called the genome. Haploid organisms carry only one copy of each chromosome. Polyploid species have several copies of each chromosomes, for instance humans are diploid: they normally carry pairs of each chromosome. Moreover, genomes content is structured and non-randomly distributed in terms of bases. Several biological phenomenons lead to the presence of repetitions of various sizes, also called repeats in genomes, that are "copy-paste" sequences that have several occurrences in an individual. Such sequences can in fact diverge in their content because each copy accumulates different mutations. Some repeats are inserted in genes, and some genes are themselves copied and form so-called gene families.

It was thought that once the sequences inside genomes would be known and well interpreted, one could access all the complexity of the living. It is then natural that DNA sequences started to be extracted and studied. However when genome sequences started to be obtained for several organisms, the sizes of genomes strikingly did not correlate to the apparent complexity of organisms (examples are reported in Figure 2). It

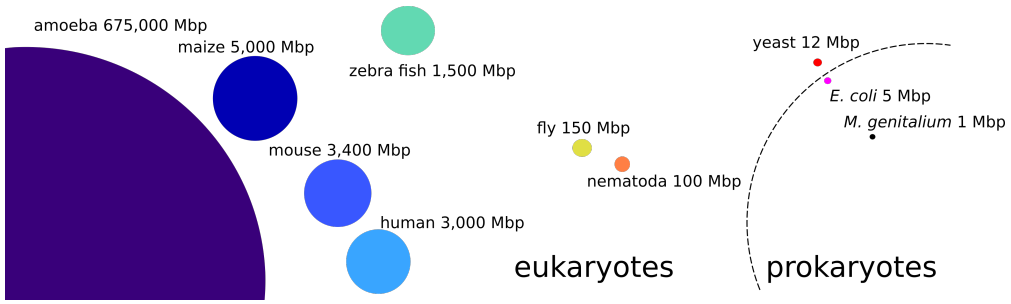

Figure 2: **Several examples of genome approximate sizes in eukaryotes and prokaryotes.**

became more and more clear that the envisioned metaphor of the DNA as the "book of the living" was not accurate enough to represent the biological reality. The diversity of biological processes though existing, one or more keys were gone missing.

1.1.2 From DNA to transcriptomes

The second molecule found in organisms that is composed of nucleic acids is called RNA (ribonucleic acid). In RNA, uracil (U) substitutes to thymine, but RNA properties remain very similar to those of DNA, at the exception that it is usually single stranded (left in Figure 1). We simplified on purpose the universal role of DNA for information support in life, since some viruses only rely on RNA. However a more spread role of RNA across the living is the diffusion of DNA information in the cell. The paired nature of DNA gives it a certain chemical stability, which is a useful property to store information on the long run. The book metaphor stands quite well here, the DNA being a library and RNA being a lent book. RNAs are thus usually orders of magnitude shorter than DNA molecules, and since single stranded, more fragile. As DNA, they have strong affinity with related molecules and tend to hybridize. RNA molecules can basepair with themselves, notably by forming loop structures, which are a category of so-called secondary structure. A whole category of RNA molecules, called messenger RNAs (mRNAs), transport genetic messages from the DNA to the rest of the cell. These molecules have a more temporary lifetime than DNA, since they represent the genetic information that will be used at a given moment of the life of a cell. Messenger RNAs are decoded within the cell to produce proteins.

This path from DNA through messenger RNAs to proteins is stated in the *central dogma of molecular biology*. Contrary to DNA and RNA, proteins are composed of amino acids of 20 different types. The

so-called genetic code is the set of rules that allows translation from the nucleotides alphabet to the amino acid alphabet. In mRNA, triplets of three nucleotides, called codons, encode for one amino acid or a "stop codon" (that is, the end of an amino acid chain). mRNAs serve as intermediary "buffer" molecules, each corresponding to one protein. are processed through cellular molecular machinery to obtain proteins. There are 4^3 codons possibilities for only 20 amino acids, thus some amino acids have several corresponding codons: the code is said "degenerated". Variants of this code exist, however it is highly conserved through the living. The resultant proteins are the cell main effectors that perform functions. They are involved in various tasks such as the transport of molecules, chemical catalyzations, DNA repair or replication, shaping other proteins...

In short, the proteins "blueprints" are encoded and stored in DNA, then this necessary genetic information is extracted and encoded a second time in RNAs before translation to protein. Again, the number of predicted genes after accession to genomes was not in adequacy with the diversity of proteins and functions observed. In other words, despite central dogma, it was difficult for scientists to make correspond genotypes with phenotypes (i.e. expressed characters). Though fundamental, this dogma had to be completed by other principles that govern messenger RNAs production to fit with the actual repertoire of messengers RNA and proteins. This led to study not only genomes, but also transcriptomes (the set of RNA molecules also called transcripts) to both access and understand the molecular content of the pool of RNAs that occur in organisms.

Transcriptomes are not limited to messenger RNAs. Other RNA molecules exist, which different properties and roles impacting the cell dynamics as well have been discovered after messenger RNAs. The full population of RNAs in a cell is called total RNA. For instance, ribosomal RNA is another family of RNA involved in translation mechanism that converts RNA sequences to proteins. Other cellular organelles, such as mitochondria in most eukaryotes, carry their own genome in addition to the nucleic genome and can express RNA. However messenger RNAs remain broadly studied for their necessary and direct role in protein synthesis. In the rest of this document we will generally refer only to messenger RNAs in eukaryotes. So far we have briefly stated RNA's genesis and roles in the eukaryotic cell. We also mentioned that such molecules were sequences, studied by biologists and bioinformaticians. In the following, keys to understand the features of this molecule will be given, as well as current informatic models that allow its study.

1.2 Dynamics of RNA

1.2.1 Production of RNA molecules in the cell

Transcription All steps described hereafter are illustrated in Figure 3. Messenger RNA is produced from DNA subsequences that correspond to genes. According to the properties previously described, RNA molecules are synthesized as complementary strands of templates DNA subsequences [3]. This synthetization reaction is mainly catalyzed by the enzyme called RNA polymerase II. There exist other, more specific, RNA polymerases. Transcription also involves proteins called transcription factors that bind to specific DNA regions called promoters [69]. They participate in the recognition of the gene sequence and to the and well-functioning of the transcription. After the RNA polymerase is placed at the beginning of a gene sequence, it proceeds elongation, in which RNA is synthesized by incorporating complementary nucleotides to the genomic sequence. RNAs are synthesized from the 5' to the 3'. The exact positions of start and end are marked by specific sequences, the 5' transcription start site and the 3' polyadenilation signal (a stretch of A bases). The region within these sites is called the coding DNA sequence (CDS). As a result of transcription a so-called pre-mRNA is released [112]. As well as many processes in the cell, transcription is regulated. Not all genes see themselves transcribed at the same level [91, 196, 208]. Genes transcribed at higher rates are called "highly expressed genes ", on the contrary "lowly expressed genes" are transcribed at low rates.

Modifications to pre-mRNA We mentioned that RNA, because of its single stranded nature, was more chemically unstable than DNA. However a desired property is that it must conserve its message until

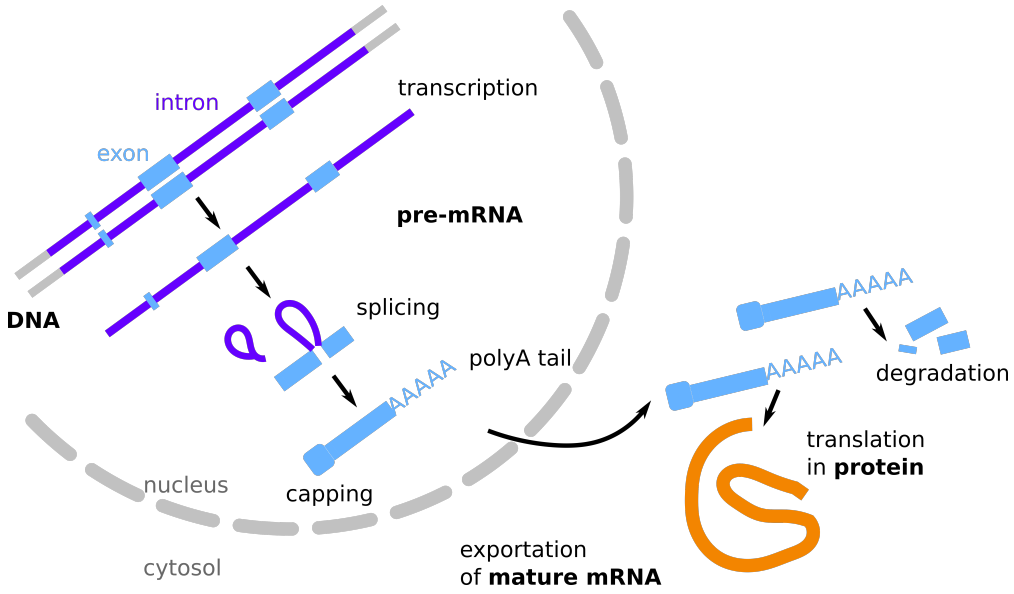

Figure 3: **Simplified view of steps of the central dogma of molecular biology.** Thick sequences are exons and thin sequences are introns. Introns form "lassos" while spliced. We represent the polyA tails and the cap by a thicker 5' end.

it is conveyed to be translated. In fact, pre-mRNA undergoes modifications. A polyAdenosine tail (polyA) tail is added to the 3' end. This tail is typical of messengers RNA among all RNAs. On the 5' end, a nucleotide is also added. The pre-mRNA is then said "capped". They are excluded from the part of the sequence that will be translated to protein. However these modifications play roles in preventing degradation by enzymes, they also help the recognition of the RNA by further molecular players. They are thus realized while within the cell nucleus. While not being a direct modification of the sequence, it must also be noted that pre-mRNAs are marked by the presence of specific proteins (RNA binding proteins), that are removed once ready to be transported out of the nucleus. Altogether, these additional sequences and the CDS are called an open reading frame (ORF). ORFs are looked for in bioinformatics to detect regions prone to code for proteins. Other post-transcriptional modifications can occur, including RNA editing, during which bases within RNA sequences can be modified or added after the transcription; and RNA methylation, in which a methyl chemical group can be linked to a RNA base.

Splicing In eukaryotes, alternating sequences compose a gene, called introns and exons. Exons are protein coding regions while introns are usually not. Thus splicing is the step during which introns are removed from the mRNA sequence that used to contain the whole gene sequence. Splicing was discovered in 1977 by Sharp and Roberts and this contribution was awarded a Nobel Prize in 1993. It involves a large complex of more than a hundred proteins and RNA effectors called the spliceosome. There are two types of spliceosomes: major and minor. The major spliceosome targets the vast majority of introns and is involved in most of the splicing events. It acts by recognizing specific canonical sites called consensus splice site at the beginning and end of introns [149]. Some non-canonical sites exist and are dealt with by the minor spliceosome [237]. Such sites are conserved and occur at the exon-intron and intron-exon junctions. The canonical sites are GU and AG. During splicing, exons are linked while introns form lassos before being removed. As a result of splicing, a mature mRNA is obtained.

1.2.2 Actors of the variability

The key to understanding the gap between the number of proteins and the number of genes, is the production of variability during the process described by the central dogma of molecular biology. In addition to being related to a temporality, RNA content can differ across cells at a same moment in a same individual. Some mechanisms indeed lead to message diversification by producing alternative mRNA [110]: alternative splicing, alternative promoters and alternative polyadenilation. Alternative splicing happens during the splicing process, when certain exons are excluded (spliced) as well as introns. The spliceosome can also recognize alternative 5' or 3' splice sites which leads to different exons starts and ends. Alternatively, some introns are not removed during splicing and remain in the mature mRNA. Thus different mRNA can originate from the same gene. They are called alternative isoforms or mRNA variants. For a given gene, some exons are present in all isoform (constitutive exons), other are called alternative exons. Alternative splicing was first considered as a rare phenomenon [17]. Nowadays we think that a large majority of genes undergo alternative splicing in human (at least 95% of the 90% multi-exonic genes among human genes are expected to be alternatively spliced [45, 173, 12].

Alternative splicing products have been detected in proteins, thus confirming the importance of this mechanism in the variability encountered in the cell [236]. An example of alternative splicing is presented in Figure 4. Alternative promoters and polyadenilation are not splicing events. They can occur because

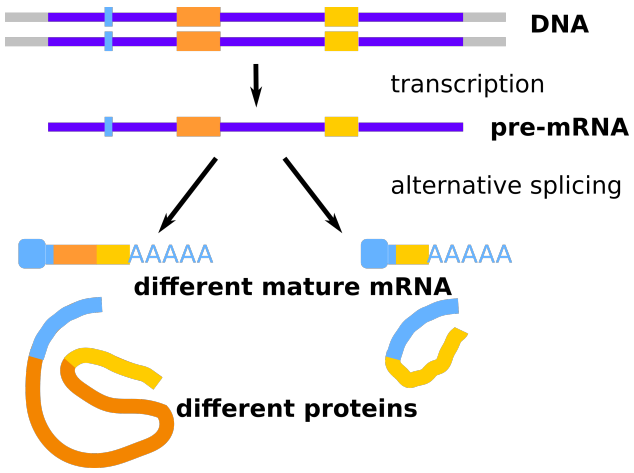

Figure 4: **Schematic view of an example of alternative splicing event.** As in the previous figure, mature transcripts are presented with their cap and tail. Exons are bold parts and introns are thin parts. In this case, the middle exon is skipped in the right mRNA transcript, yielding a different protein from the one obtained through processing the constitutive transcript, that contains all exons.

of a modification of transcription initiation/termination sites, which yields mRNA composed with different first or last exons. Figure 5 summaries the different alternative events. Genomic DNA also carries variants that can appear within an individual (some positions differ in their nucleotidic content between a pair of chromosomes for instance), thus position (locus) of the variants is called heterozygous, or between individuals. Heterozygous genomes include variants of a single nucleotide at given positions (SNPs), insertions and deletions of bases (called indels) and larger variations where large parts of chromosomes are recombined. Some of these variants appear in coding parts of the genome and participate in the expressed phenotype.

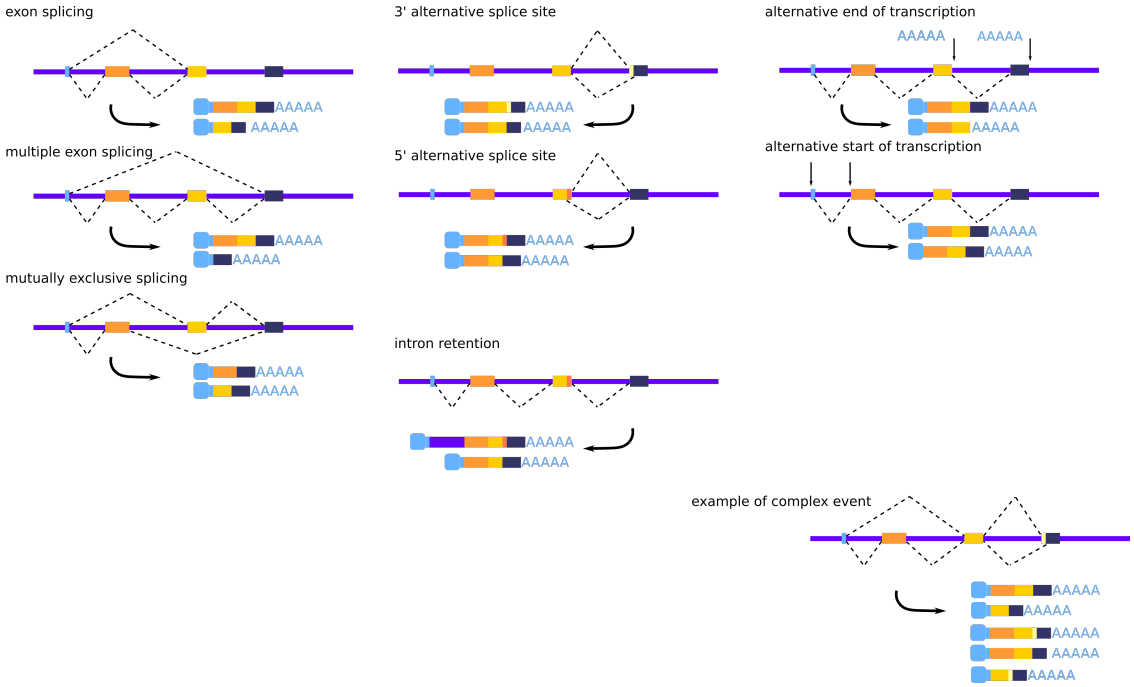

Figure 5: **Summary of splicing and transcriptional alternative events.** For each pre-mRNA, the alternative isoforms are presented. Pairs of junctions involved in the splicing are linked by dotted lines. It is not rare that genes undergo complex splicing events that involve several of the atomic events presented here, such as in the example in the bottom.

1.2.3 Fate of RNAs

Alternative splicing allows a regulation of gene expression via the production of proteins with potentially different roles in the cell, and also plays a role in the quantitative regulation of gene expression. A fraction of alternative transcripts is thus not functional, and will be targeted to be destroyed by cell effectors.

Mature mRNAs are exported from the nucleus to the cell cytosol through nuclear pores. This is the moment where a first filter occurs, since pre-mRNAs cannot pass through the pore [183] if they do not carry proper modifications brought by the previously described steps. Pre-mRNAs that remained in the nucleus are degraded afterwards by a dedicated enzyme complex called exosome. Mature RNAs are then used as templates for translation to proteins. However there exist over mechanisms for the control of error during these processes. If a pre-mRNA is mistakenly exported or if a mRNA contains errors, the NMD (nonsense-mediated decay) is a surveillance mechanism that searches and removes aberrant molecules that contain early stop codons [178]. After translation, all mRNA are degraded, which illustrates well the buffer role of this molecule.

1.2.4 Main characteristics

In eukaryotes, it is common that alternative spliced and transcriptional variants with various exon content (isoforms) occur for a given gene [157]. In human, there is evidence for more than 95% of genes to do alternative splicing [59]. An average of 4 transcripts per gene is depicted, however, this number has to be taken very cautiously, as the quantity of sequences that could reached by the sequencing experiments has a strong influence on the number of alternative transcripts found. DNA and RNA sequences lengths are often given by referring to the quantity of bases they contain (base-pairs or bp, because of the double strand in

DNA). A human gene has around 10 exons, 80% of them have length lower than 200 bp, namely, a range of transcripts size between 1kbp and 5kbp. Human transcripts contain more introns on average than lower eukaryotes. Annotated human introns are frequently longer than 70 bps [201].

2 Access RNA sequences

In order to study DNA, RNA and proteins, there must be a way to pass from biological material to numeric/text format accessible to humans or possible to process with computers. Sequencing is the process through which biological sequences are accessed by reading their bases and encoding them, most of the time in text files such as FASTA that will be described afterwards. Sequencing is therefore a crucial step that puts a bridge between what exist in cells and our understanding. From late 70's to now, several generations of sequencing technologies paved their ways to the retrieval of biological sequences. However even currently speaking, no method ensures that all sequences of an experiment are read full-length. In a simplified view, subsequences called reads are extracted randomly from the genome or transcripts. The task of reading each nucleotide is called basecalling.

The main challenges are to produce fragments as long as possible, to provide fragments distribution that allow correct representation of the whole nucleotidic material, and to ensure the base-wise fidelity to the original sequence. In the following, we present an overview of the principal technologies of interest in our work, with their main features, pros and cons. Sequencing limitations will partly drive the conception of algorithms to process reads. For genomes, the fragmentation in small sequences means that assembly is needed to elongate the signal present in reads, and to order them with respect to one another. Transcriptomes, though containing smaller sequences, are also concerned. Assembly task is described with more details in the following. We will give emphasis on the relative volatility of these technologies. Particularly in the last years, reads properties evolve extremely rapidly.

The intuition of the sequencing "depth" is that it is the fraction of the transcriptome that can be accessed. It can be defined for a transcriptome by the $\text{numberofreads} \times \text{readslength}/\text{transcriptomesize}$. We speak about genome, transcriptome or transcript "coverage" to express how many times a given subsequence must be found repeated in the read set. For instance, in genomics context, a genome coverage of 10X means that the each base of the genome sequence is represented ten times on average in the reads. In transcriptomes, due to gene expression, not all transcripts are not covered at the same level.

2.1 Short reads and former technologies

2.1.1 Sanger and Expressed Sequence Tags

Appeared in 1977, Sanger sequencing was the first to give access to nucleic acid sequences, based on synthesis and electrophoresis [205]. It is still used nowadays and provides read lengths around 1,000 bp, which were involved in first mRNA assemblies and production of EST (expressed sequence tags). EST are reads of complementary DNA (cDNA) clones from cDNA libraries (RNA is retro-transcribed to its complementary DNA sequence) prepared from template mRNAs expressed in an experiment. These reads are typically sized from 500 to 800 nucleotides. The error rate is relatively low (less than 0.1%), however the main drawback of this technology is its low throughput. Indeed, DNA preparation for this technology is long and costly. Since depth is an important to capture transcript diversity in transcriptomes, Sanger technology shows limitations in comparison to more recent sequencing techniques. However it enabled to build high quality reference transcripts.

2.1.2 Massive sequencing with short reads technologies

In the late 2000's a revolution operated in DNA and RNA sequencing. The so-called Next Generation Sequencings (NGS, or Second Generation Sequencing), proposed by companies such as Solexa (that was later acquired by Illumina), allowed the first high throughput sequencing of transcriptomes. The RNA

sequencing protocol RNA-seq was initiated in 2008, principally by Illumina. Other technologies include 454, Solid or Ion Torrent, however Illumina is the most represented [158]. RNA-seq consists in a first step of library preparation, in which RNA is randomly fragmented. Then using transcriptases (RT), RNA is retro-transcribed to its cDNA sequence. This cDNA can undergo an amplification step using a chain reaction called polymerase chain reaction (PCR). As a result, clones of present fragments are created, multiplying the number of copies of a given fragment in order to have more signal. Adaptors (small artificial DNA specific sequences) are then fixed to fragments. At this step, the whole set of cDNA fragments, called the library, is ready to be put on a flowcell. Flowcells are supports where libraries are deposited. Fragments fix on the support by hybridization at one end using the adaptors. Then an amplification step starts using PCR, increasing the number of copies of each fragments and creating spots of similar molecules on the flowcell. From this point, sequencing strictly speaking starts. Each fragment is processed in parallel, and its complementary strand is synthesized nucleotide by nucleotide. Modified nucleotides linked to a fluorescent compound (fluorophores) are introduced in the medium, that fix one at a time on each nucleotide of a fragment. Fluorophores deliver a light signal that is specific to each four type of base, and processed through image capture and processing. Each cluster of similar fragments created by PCR yield a colored spot that corresponds to the reading of one of the A,C,G,T bases. After an image is captured, a new cycle starts that corresponds to the detection of the following nucleotide on each fragment.

RNA-seq possesses well documented biases and flaws. They include bias in the RT-step, due to pseudo random primers used to fix the RT on the RNA molecule, and possible chimeric ends of fragments created at this step. Amplification step in the flowcell is also prone to bias, amplification being expected to be non-uniform across fragments. Errors are also expected to be more numerous at the end of fragments, because the error rate increases with the number of sequencing cycles. Illumina error rates are reported to be below 1% (values vary but 0.1% or less is often mentioned). Finally, all bases are not evenly distributed across the genome sequences, and GC rich and AT rich regions suffer from uneven read coverage [14], an effect known as GC-bias. Final length of reads goes from 50 to $\sim$ 400 bp, but is commonly of 100-150 bp length. Because of the cDNA retro-transcription step, fragments can be in forward or reverse direction but the original strand is usually not known. Despite biases, RNA-seq is broadly used for RNA transcripts identification and quantification.

RNA-seq also includes ways to select RNA molecules that will be sequenced. Since a huge fraction of total RNA is constituted by ribosomal RNA, it can be interesting to filter it out if the matter is messenger RNAs. Ribo0 protocols help removing specifically ribosomal. On the other hand, polyA+ protocols select polyadenilated RNAs. As mentioned previously, most of mRNAs bear a polyA tail.

We will not detail earlier microarrays technologies, that nonetheless represent a very stable and cheap opportunity when whole transcriptome sequencing is not a necessity. There exist many other sequencing protocols that were designed to capture a precise moment of the RNA production, as well as precise loci, or aim at describing particular features of the RNA molecules [6]. Single-cell protocols allow to collect RNAs produced in an isolated cell from an organism. They become more and more popular and are very interesting for clinic investigations, since they allow the capture of transcripts from different subpopulations of cells. Here we will focus on large scale, whole transcriptome sequencing experiments. Finally, protocols that permit to link pairs of reads separated from an insertion distance of a few thousands nucleotides to tens of thousands (paired-end reads and mate-pairs reads) also exist. Still, these reads do not allow to access the same order of magnitude in terms of distance information compared to long reads.

2.2 Focus on long reads

From 2011, new and independent sequencing protocols appeared that again would become game changing in genomics and transcriptomics fields. These platforms have in common the ability to sequence very long reads in comparison to NGS. Fragments can go from a thousand base pairs to hundreds of thousands, these lengths increased with time until today. Even taking into account reads pairs that can be provided by Illumina platforms, the advantage in length is substantial.

We will present two third generation sequencing (TGS) technologies: Pacific Biosciences (PacBio) and

Oxford Nanopore Technologies (ONT). We will not mention at length technologies such as Illumina True-seq (former Moleculo) and 10X Genomics that are sometimes also classified in TGS. They are based on very different methodologies since they rely on short reads, and are recent as well. In their case, long reads are re-created by assembling short reads (thus called synthetic long reads) that are ensured to come from the same region. At the beginning of this work, we were mostly aware of leverage these synthetic long reads had for genomics complex regions assembly. We chose to focus on known applications to RNA, mostly provided by PacBio at the time. During this thesis, ONT technologies became more stable and we shifted to them for several reasons that we explain thereafter.

2.2.1 Pacific Biosciences

PacBio platform relies on biochemical synthesis such as Illumina's for sequencing. However, PacBio manages to get signal from one DNA molecule at a time (it is sometimes referred to as SMRT fot Single Molecule Real Time). The activity of DNA polymerases (one for each DNA templates) is recorded within structures called zero-mode waveguides on a chip. DNA polymerases synthesize the complementary strand of their template, each integrated base emits a fluorescent signal as with Illumina. The reported error rates for PacBio reads are 13 to 15% with a majority of insertions [29]. They adopted a strategy to increase sequences accuracy for not too long fragments. Hairpins adaptors are ligated at each end of a double stranded fragment, enabling the polymerase to realize circles around the read. This way, the polymerase traverses several times a fragment (each repeated sequence is called a subread), then a more accurate consensus (called circular consensus sequence, CCS) can be extracted from these subreads. Reads which sequence was produced without consensus are called CLR (continuous long reads), or more recently RoI (Reads of Insert) [61]. Reads length (and CCS quality) depends on the time when polymerase drops the fragment. PacBio delivers reads that are usually tens of kilo-bases long. PacBio dedicated a protocol, Iso-seq [74], to RNA sequencing. It aims at taking advantage of the relative short size of transcripts to provide as many CCS as possible. In order to obtain transcripts of all lengths in the result, Iso-seq imposes size binning of the initial molecules. The longest transcripts correctness drops, but many 1000-2000 bp long sequences harbor 1-9% error rates that allow easier analysis than with CLR. PacBio protocol is summed up in Figure 6 (left).

2.2.2 Oxford Nanopore Technologies

In ONT, single-stranded DNA molecules pass through pores. Here too biochemical compound inspired a sequencing technology, but this time ONT does not relies on synthesis. A comparison to PacBio is shown in Figure 6. The pores are composed of proteins similar to those found in nucleic pores. They help threading the DNA so that current changes induced by the nucleotides at each end of pores are measured. The current signal is different according to the nucleotides passing a pore, and adequate signal treatment can deduce which group of nucleotide (currently 5-6 bases is the maximal resolution) traversed the pore at a given moment. ONT proposes several platforms, the MinION being broadly used. In the same spirit that what is proposed by PacBio, in MinION libraries a hairpin can be attached to the molecule so that forward and reverse sequences can pass in the pore following each other. Each raw sequence is detected since it is separated by the adaptor sequence, can be separated (thus called 1D reads) or used to propose a consensus (called 2D reads) that are expected to show better quality. For R9.4 ONT chemistry, 1D reads were reported to have a mean 14% errors (with many reads from 8 to 10% errors), 2D around 9%. However this 2D technology is no longer maintained since 2018, because the hairpin used to obtain a consensus was in certain cases producing a secondary structure that would slow the strand traversal inside the pore and therefore have a bad impact on the quality of the sequence. Since 2017, $1D^2$ is meant to replace 1D and 2D protocols. In this protocol, two proteins linked to the strands (forward and reverse) help them passing through the pore but the two strands are no more physically linked. However the two strands sometimes fail to pass after each other and no consensus can be obtained. It seems with the first generation of data that a consensus could be obtained from 30 to 60% of the cases, with percent identity median at 95.7. Genomic reads length can reach currently speaking 200,000 bp on average. One million bp was announced in 2018 by ONT, however such scales are

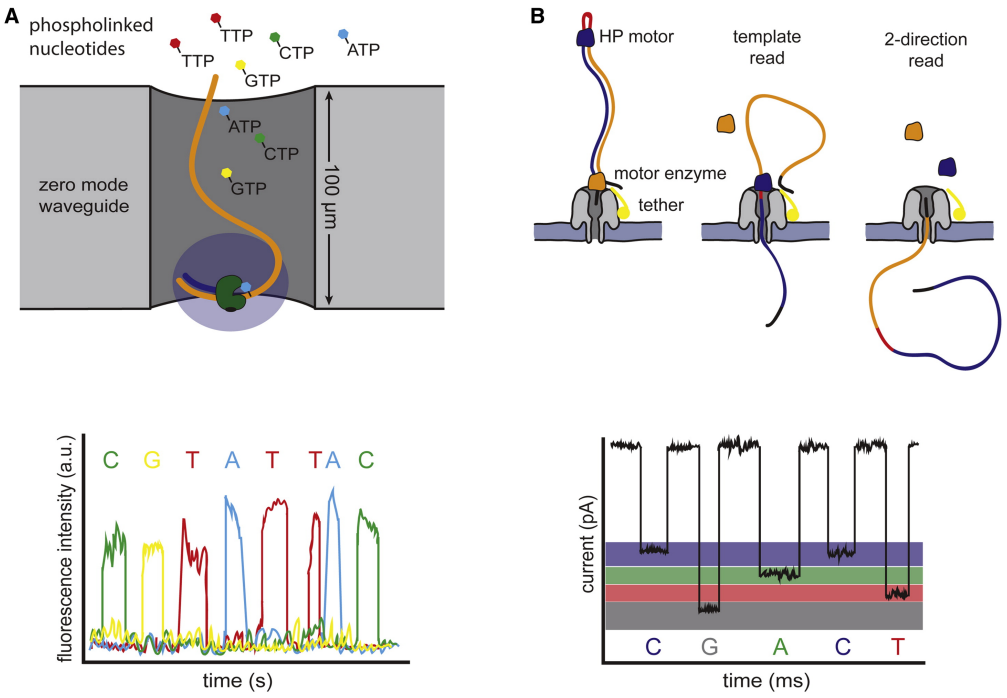

Figure 6: **Differences in sequencing and base calling between Pacific Biosciences on the left and Oxford Nanopore on the right.** Pacific Biosciences relies on synthesis and image processing as Illumina, while Oxford Nanopore works with current measures through a pore. Figure from [189].

still hard to reproduce in labs. For RNA sequencing, ONT proposes sequencing both with and without PCR amplification. Even if less reads are provided, sequencing without PCR allow to lower the biases that were encountered with Illumina. Ribo0 and polyA+ protocols previously described can also be applied. We give a chronology and summarize the main protocols in Figure 7.

A flaw in ONT reads compared to PacBio relies in the homopolymer regions (regions where the same base is repeated, such as polyA tails for instance) that are more error-prone as their length is difficult to assess for this protocol. Errors seem to appear non-randomly in these regions, which make them impossible to be fully corrected without the help of another type of data [137]. It is estimated for R9.4 50% of reads including low complexity subsequences from size 5 contain deletion or insertion errors. Moreover these errors are a trouble when one tries to predict an ORF on the sequences as the frame tends to be shifted (an indel changes the series of triplets that encodes for amino acids). Reads quality is also correlated with the pore throughput. The advents in throughput had a good influence on the sequence quality. The higher the throughput, the less variability for each base to stay in the pore as the reads traverses quicker. Thus, the electric signal is more stable. Basecall requires intensive computing methods, a new panel of works dedicates to build better basecallers. Preparation of flowcells can be optimized for improving either throughput or quality.

Recently, ONT announced that RNA molecules could be directly passed through the nanopore, avoiding the cDNA step (this protocol is called RNA-direct). RNA molecule can be hybridized with a cDNA in order to make it more stable, but only RNA is threaded in the nanopore. Even if we do not have a lot of insight, we can state that such a protocol gets rid of possible bias during cDNA conversion, and must be useful in the future to detect RNA methylations directly by measuring the current in the pore. Moreover, reads can be sequenced stranded. For the moment, basecallers able to report this type of information are at early stages [218]. RNA direct is rather slow (5 times slower than 1D protocols), and a higher error rate is

observed than in cDNA sequencing. A lot of material is required to start a sequencing and thus RNA-direct provides less throughput than other sequencings.

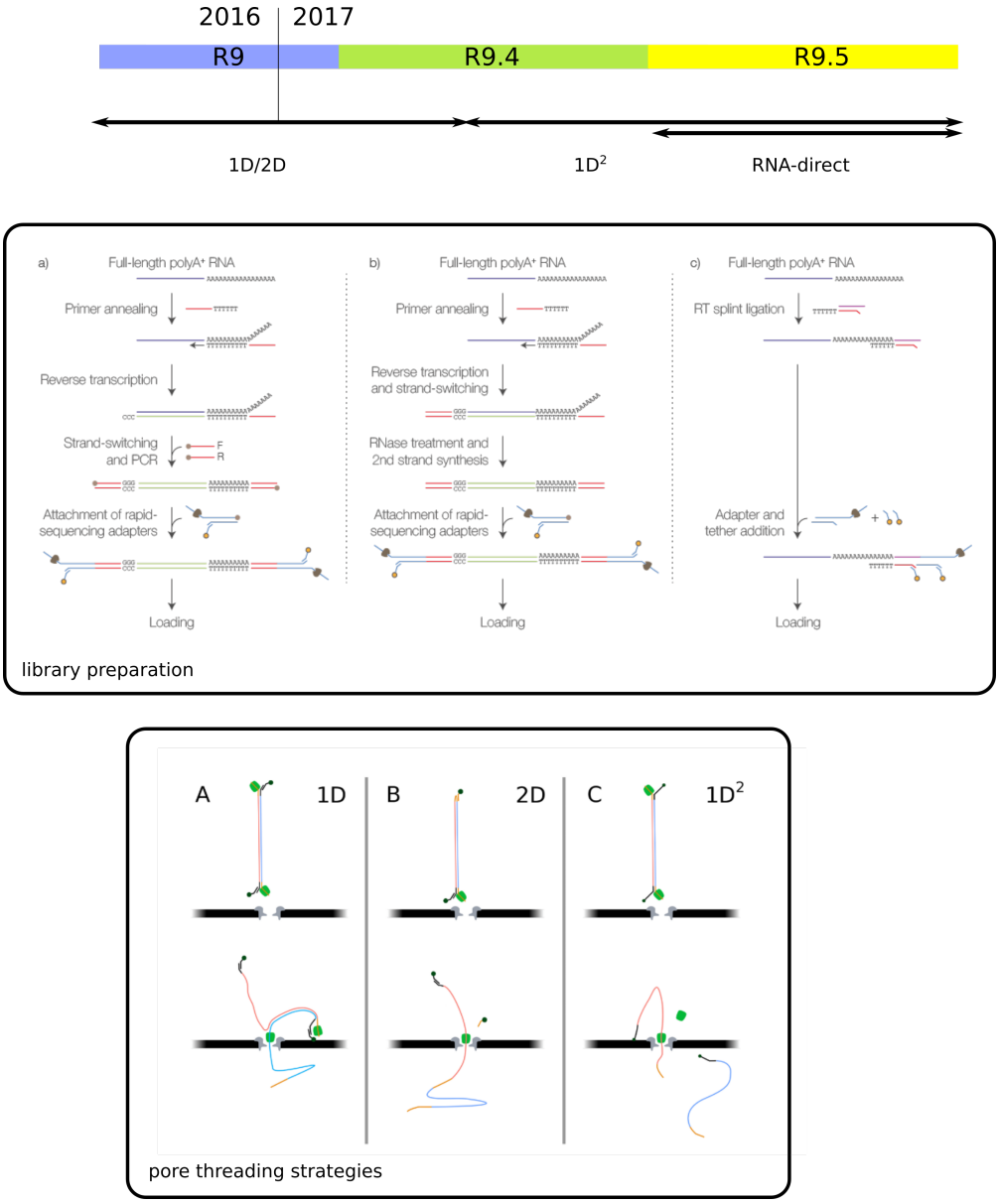

Figure 7: **Chronology and description of different ONT sequencings.** In the left square, we show library preparation a) with PCR amplification b) without PCR amplification c) RNA-direct. On the right square, we show how sequences pass through the pore according to the chosen chemistry (1D/2D/1D^2). With 1D, only the template read in blue is read, the other strand (red) being rejected. With 2D, both strands pass using a hairpin. Using 1D^2, both strands also pass, the second being attached to the membrane while the first one passes, then being released. Figure adapted from [49] and ONT website.

Nowadays RNA-seq is the most mature and delivers more depth than other technologies. ONT platforms are conversely highly "volatile" technologies, prone to many evolutions these last years. Illumina costs remain a critical argument for the communities that study RNA as a high depth allows to recover more variability of the cell. A good coverage is important also for differential expression studies. Thus many low counts are a struggle when starting statistical analysis. Nowadays, Illumina provides 100 to 1000 times more depth than TGS. However latest sequencers by ONT (such as GridION) aim at lowering the cost of a sequencing and we can expect to see a convergence of the costs of each technology in the future. The human cost (time of libraries preparation) does not seem to be in favor of one or the other technology.

When this work started in 2015, PacBio seemed a more assured bet, because its error rate was stabilized and lower than 10% with CCS, while ONT reads could reach 40% errors. However it rapidly appeared than only isoform identification could be performed with PacBio because of the size binning of reads during Iso-seq protocol. Finally, ONT keeps developing extremely handy flowcells that can be brought in the field. Being sized as a USB key they largely reduce the bulk in comparison to other platforms. Basecalling using FPGA (configurable integrated circuits that can be specialized in a task) starts to be developed, which opens the door to full sequencing workflows realizable directly where the RNA is collected and may reduce the processing time. This implies better sequencing conditions for species that cannot be grown in lab. As mentioned above, RNA molecules are unstable and can be degraded in hours, depending on how they are stored. Thus, enabling direct sequencing from the field may have dramatic impacts on the quality of the sequencing [198], in particular in difficult conditions for sequencing (high temperatures...). For all these reason, our choice was to focus on ONT reads.

3 RNA from a computational point of view

Through sequencing, we are given access to sequences in text format that then can be processed in a computational way. Algorithms applied to RNA mostly relate to text algorithms and graph algorithms. Text is a quite natural representation for these sequences. Graphs are convenient to represent sequenced reads and their putative connections (i.e. sharing certain properties or text sequence). They also well render the combinatorial nature of RNA transcripts. Hereafter, we introduce notions that are fundamental to the computational representation and study of RNA. We briefly describe algorithms that are mostly not used as is in current methods, but are bases for most of the algorithms that treat with reconstructing and identifying RNA sequences. Then we level up to the softwares and pipelines and give insight to the main bioinformatics paradigms that are followed for the study of transcriptomes using reads. We made the choice not to introduce hardware aspects and assumed that the readers are familiar with notions of bit, memory (RAM), disk and threads.

3.1 Algorithmic complexity

In the following we will use the notation $\mathcal{O}(f_n)$ which describes the complexity of methods or implementations. A complexity of $\mathcal{O}(f_n)$ means that in the worst case the number of steps required for the algorithm to complete can be bounded by $c \times f_n$ for a constant c. For instance, we will describe methods with quadratic complexity ($\mathcal{O}(n^2)$), we can guarantee that for such algorithms, the number of operations to complete can be bounded by a quadratic function in the worst case.

We call P the class of problems that can be solved in polynomial time. Such problems are usually considered tractable using computers. We call NP-complete problems, problems for which a given solution can be verified (i.e. we can check it is a valid solution to the problem) in polynomial time but no polynomial algorithm is known. We say that a problem is NP-hard if it is at least as difficult to solve as any NP-complete problem. Thus these two classes of problems are currently considered intractable.

For more details, a very good introduction to these notions can be found in the book "Introduction To Algorithms" by Cormen [44].

3.2 Nucleic molecules as text

We will call a *string* or *sequence* S an ordered list of characters issued from an alphabet Σ. Thus the positions of characters are defined in a sequence. DNA alphabet is usually denoted with its 4-sized alphabet $\Sigma = \{A, T, G, C\}$, that can be encoded using 2 bits per nucleotide. It is very common for the uracil nucleotide of RNA to be replaced by a T when data is in numeric format or in implementation. In order to remain coherent in this manuscript, we will systematically replace "U's" by "T's". A *substring* or *word* of S is a sequence where elements are removed without changing the global order in S, such that remaining elements are contiguous. For instance, $ACTG$ is a substring of $ACTGA$, while $ACGA$ is not. A sequence *length* is the number of characters it contains. A *prefix* (respectively suffix) of length p of S is a substring of the p first (respectively last) characters of S. A k-mer is a word of size k in a sequence. Classically, a k-mer of size up to 32 (respectively 16) can be encoded using a 64-bit (respectively 32) integer in efficient implementations.

Edit operations are three possible atomic ways to modify a string. Substitution is the replacement of a character by another. Insertion is the addition of an extra character at some position of the string. Deletion is the removal of one character of the string. These operations also happen in genomes where they can be variations across or within individuals. The edit distance (also called Levenshtein distance) for two string x, y is the minimum of number of edit operation that are needed to transform x to y (this definition is symmetric). The Hamming distance for x, y is the number of positions where characters differ from x to y (the definition is symmetric for two string x, y of equal size).

DNA and RNA sequences can be stored in different formats, the most widely used being FASTA and FASTQ formats [40]. They can contain whole genomes or transcriptomes (thus correspond to references), or smaller sequences coming from various steps from DNA and RNA sequencing experiments and bioinformatics processings. These formats allow to model sequences according to their nucleotide content. The simpler, FASTA format, contains two types of information: sequences headers that always start with the character ">" and are strings that can contain relevant information about a sequence, commonly at least a unique identifier. The other type of lines contain the actual sequence written in a string using the 4 letters alphabet. Sometimes, non coding nucleotides (or repeats) are identified by using lower case as opposed to coding nucleotides in upper case. Headers and sequences finish by an end of line. Thus at least two lines encode for the information of one sequence, the header usually written on one line and the sequence on one line or more. FASTQ format carries in supplementary lines the confidence in each base after sequencing using so-called Phred scores. References are often accompanied by GTF or GFF files that complement information through detailed annotations about known involved genes, sequence strand... However they are not sufficient to represent certain biological properties, such as RNA secondary structures or methylations. They also constitute limitations when it comes to representing RNA's combinatorial nature. Text formats were also proposed to document the exon combinations, such the one used in AStalavista [63].

3.3 Sequences comparison: text algorithms

3.3.1 Sequence alignment

```
1) s1:AGTTGA  2) s1:AGTTGA  3) s1:AGTTG-A
   s2:AGTCGA     s2:A-TTGA     s2:AGTTGCA
```

Figure 8: **Alignment of two sequences s1 and s2.** 1) Alignment with one mismatch. 2) A gap in s2, representing a deletion in s2 or the insertion the base G in s1 3) A gap in s1.

Pairwise sequence alignment consists in finding a way to arrange two sequences so that most similar nucleotides regions are grouped. Usually, an alignment is represented by a trace. Each sequence is written from 5' to 3' on two lines. If two bases are vertically aligned, they are a match or a mismatch. If one or several base miss at a position in any of the sequences in comparison to the other, a dash "-" (called a gap) is inserted at this position, vertically aligned with the corresponding base in the other sequence.

Thus final strings in the alignment have the same length. Examples are presented in Figure 8. Such task is fundamental in bioinformatics since it helps retrieving similarity of sequences. According to the similarity degree, we derive potential common origins for sequences (same gene and same function, organisms that have close common ancestor, ...) or in other applications, structural properties. The first algorithm to propose an optimal solution for pairwise sequence alignment was described by Needleman and Wunsch [163], that relies on dynamic programming. It outputs a similarity measure as well as the corresponding alignment. Its variation, the Smith Waterman algorithm [219], is a local alignment procedure which seeks optimal alignment on subsequences of the original pair of sequences. In both case, the alignment takes place in a matrix, each dimension representing one sequence of the pair to be aligned. The result is obtain in a quadratic time ($\mathcal{O}(MN)$, M and N being the sizes of the sequences).

These algorithms then benefited from optimizations. For instance, computation of alignment only within a diagonal band in the matrix is used to reduce the number of operations. Many optimizations relying on hardware such as vectorization through which several operations are sent simultaneously to the processor. Such an implementation is used in Chapter 3. However they remain extremely costly on real instances, when a lot of sequences and/or long and divergent sequences have to be compared. Finally, in the biological sequences alignment context, it can be useful to relax the alignment constraint on ends of the sequences, for instance because in certain sequencing technologies reads quality drop at the end. Such operation is called clipping.

Some people define differently *alignment* and *mapping* notions, with the assumption that alignment will yield a score and an associated transformation of one of the two sequences, while mapping only provides information of a match between sequences (and positions of the match) without the precise resultant alignment. However this is not a hard rule, mapping can be seen as one pairwise alignment. This is the way it will be used thereafter.

3.3.2 Heuristics

The optimizations of presented sequence comparison algorithms have quadratic complexities and are often non tractable in practice on real instances. Due to this, a majority of methods for sequence pairwise comparison were developed relying on heuristics. Here we will only talk about sequence (meaning read or assembly of reads) versus reference or sequence versus sequence mapping. The tools relying on these methods are called mappers. We use the terms reference or target sequence and query sequence. Most prominent mappers use an index for the reference sequence and map the reads using this index. Indexes can rely on several data structures that we present in this introduction and Chapter 1. The core of most mapping approaches relies on *seed and extend* strategies. First at the seed step, at set of identical or highly similar regions shared between the target and the query are identified. They are used as seeds to extend the alignment, usually with dynamic programming. Classic seeds are chains of contiguous or spaced k-mers (while k-mers are contiguous substrings, spaced-seeds can contain "wildcard" characters that relax the matching), or maximal exact matches (perfect matches that cannot be extended at both ends of the matched string). These methods usually report the best-mapping result. However, all mapping results can be provided with a score over a certain threshold, but this is at the price of a longer computation. Most commonly used tools include BLAST [5], BLAT [103], BWA-MEM [129, 125] or Bowtie [116]. They differ by the data structure employed for indexing and by some fine algorithmic choices during the mapping. Several mapping tools are used in works presented in this manuscript, most of the time for comparison purposes. In Chapter 1 and Chapter 2 as well, widely used mappers are compared to the approaches we developed. *Seed and extend* approach is a powerful paradigm that will be employed in Chapter 3. Moreover, the presented pairwise alignment is in fact a particular case since more than two sequences can be aligned. This case will be presented in Chapter 3 and reviewed in more details in Chapter 4. In terms of format, SAM/BAM [130] established itself as the canonical way to represent alignments. SAM is text based and BAM is the binary equivalent. SAM requests mandatory fields that describe an alignment such as start and stop positions of each sequence in the alignment, and successive edit operations encoded in a string called CIGAR code.

3.4 Graph theory notions

3.4.1 Definitions and notations

Graphs are broadly used in bioinformatics since they are a convenient way to represent sequences and to apply operations on theses sequences. An *undirected graph* $G = (V, E)$ is a set *nodes* $v \in V$ and *edges* $e = \{u, v\} \in E$ with $u, v \in V$. We refer to those graphs in general matter. A *directed graph* $G = (V, E)$ is a set of *arcs* $e = (u, v) \in E$ with $u, v \in V$. In order to differentiate directed and undirected graphs, we will call a *vertex* an element $v \in V$ of a directed graph. Both graph types have different properties. Directed graphs are for instance used for sequence consensus such as in Chapter 3. The number of nodes or vertices in graph is denoted $|V|$, the number edges or arcs is denoted $|E|$. The *degree* of a node v, denoted $deg(v)$ is the number of edges incident to v. Graphs can be visited through following chains of nodes. A *path* of length $p + 1$ in G from u_0 to u_p is a set of nodes $(u_0, \ldots, u_i, \ldots u_p)$ such that $(u_i, u_{i+1}) \in E$. A *cycle* is a path such that $\exists p / u_0 = u_p$. A *walk* is a path that can contain cycles. An undirected *complete graph* is a graph in which any pair of disjoint nodes is connected by an edge.

Sometimes, only a part of the graph is of interest. A *subgraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of G is such that $\mathcal{V} \subseteq V$ and $\mathcal{E} \subseteq E$. In undirected graphs, a *connected component* in G is a maximum set of nodes $\mathcal{V} \subseteq V$ such as for all pairs $u, u' \in \mathcal{V}$, there is a path from u to u' (maximum means that no other node from V can be added to $\mathcal{V}$ without breaking the condition). Connected components delineate connected regions within the graph, they are common objects to work with. Objects more strongly connected within connected component are used for instance to detect genomic variants. A *bi-connected component* is a maximal subgraph that remains a connected component after the removal of any one node. A *clique* in a graph G is a set of nodes $\mathcal{V} \subseteq V$ such as for all pairs $\{u, u'\} \in \mathcal{V}$, there exists from $e = \{u, u'\} \in E$. Cliques have strong properties used to model clustering problems, they appear in Chapter 2. A *star* is a connected component in which all nodes but one have a degree of one. Stars are also discussed in Chapter 2.

A *partition* of a graph G is a set of disjoint subgraphs such that any node of G is in exactly one subgraph of the set. Partitions realize division of graphs into set of nodes. They are used in clustering problems. A *cut* of an undirected graph cut is a partition of the nodes of a graph into two disjoint subsets that are joined by at least one edge. A *minimum cut* of an undirected graph is a cut involving the minimum number of edges among all cuts.

A so-called *directed acyclic graph* (DAG) is another commonly used graph, directed and that contains no cycles.

Some very classic procedures allow graph traversal and graph ordering. The *depth first seach* (DFS) is an algorithm that allows to explore all nodes of a graph. It starts by visiting a given node, said current node. Each visited nodes are marked. From this node, it explores neighbors one by one that become current nodes. For each current node, all neighbors are visited unless they are already marked. If a node has no neighbor or all neighbors are marked, it backtracks to the previous node, until all nodes are marked. The DFS is commonly employed to find the set of connected components of a graph. It is used for several matters in Chapter 2. The *topological sort* is a linear ordering of directed graphs. A graph is topologically sorted if for every arc (u, v), u comes before v in the ordering. Several algorithms exist to realize the sorting, Topological sort is used in alignment heuristics in Chapters 3 and 4. It can be realized using Kahn's algorithm [99] or a DFS.

An emblematic graph of bioinformatics is the De Bruijn graph. It has been used for decades to solve assembly problems. In our context, a De Bruijn graph of dimension k is a directed graph $G = (V, E)$ is created from a set of sequences S by extracting their set of k-mers $v_k(S)$. Then $V = v_k(S)$ and $E = (x, y)$ if and only if $suffix(x, k-1) = prefix(y, k-1)$, that is, the nodes represent a set of $k - mers$ and edges denote $k - 1$ overlaps between nodes. De Bruijn graph are also used in some reads correctors presented in Chapter 3.

In terms of representation, graphs have the advantage to efficiently represent combinatorial objects. Indeed, variants such as alternative exons better deserve graph representations than flattened sequences. Graphs can for instance encode exons in vertices and retain colinearity of exons in transcripts through edges. Such graph representation of RNA isoform is not extremely well spread, however proposition were made

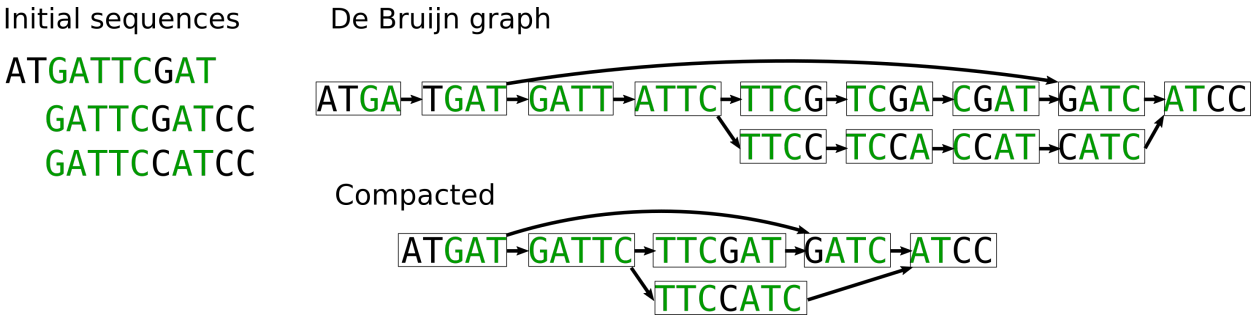

Figure 9: **Three sequences integrated into a De Bruijn graph, with k-mers of size $k = 4$.** Green subsequences are common between the three. Notice how the overlap longer than k between the first sequence and the two others is directly reported in the graph. Also notice that a single nucleotide difference (G/C in the middle) creates a branching pattern called bubble. A compacted De Bruijn graph can be built by concatenating vertices included in simple paths.

towards this idea, often called splicing graphs. Splicing graphs that will be discussed more in detail in Chapter 4, show how exons combine in alternative forms from an expressed gene. In example of Figure 10, we show how alternative events in mRNA are stored in *splicing graphs*.

Splicing graphs are directed graphs that report the alternative events for a given gene. Other types of graphs will be used in our work because they allow the application of relevant algorithms. Sequence similarity graphs are the very first step before clustering in Chapter 2 and particular DAGs are used for read alignment and consensus in Chapters 3 and 4.

3.4.2 RNA transcript reconstitution with fragment assembly

General problematic Up to recently, sequencing technologies that give access to the actual mRNA sequences in cells could only deliver extremely fragmented subsequences, (pseudo) randomly extracted from transcripts in presence. With sufficient coverage, reads overlap, which makes possible to (partly) retrieve how the substring they represent used to be associated in the original sequences. Thus, the task called "assembly" aims at solving the puzzle made of those fragments by ordering and orienting them to reconstitute the original transcripts.

Assembly was first studied in the genomic context, where the goal is to obtain one sequence per chromosome. In order to solve the problem, a class of graphs called assembly graphs is used. These graphs represent strings or substrings from the sequencing experiment in nodes, and their overlaps in edges. The most efficient and most used graph representation for assembly these last years is the previously defined De Bruijn graph. De Bruijn graphs can be efficiently built and cleaned from potential errors. Contiguous nodes are then searched to reconstruct transcripts. The intuition is that simple paths (called unitigs) in the graph do not represent substantially long sequences because either errors or variations break the linearity of the graph. Assembled sequences (called contigs) are paths obtained after simplifications in these graphs. However, certain biological features yield complex patterns in such graphs. Genomic repeats, as well as homozygous regions create "X" patterns were decisions have to be made to return a valid path (see also Figure 11). Genomic repeats longer than the size of the sequencing fragments are the main problem in assembly since we do not dispose from enough information to reconstruct the neighborhood of these regions. A simple metric to describe the contiguity of assembly results is the N50. N50 denotes a length so that half (50%) of the assembly result is composed of sequences larger than the N50. Other indices such as N75 can be defined in the same way.

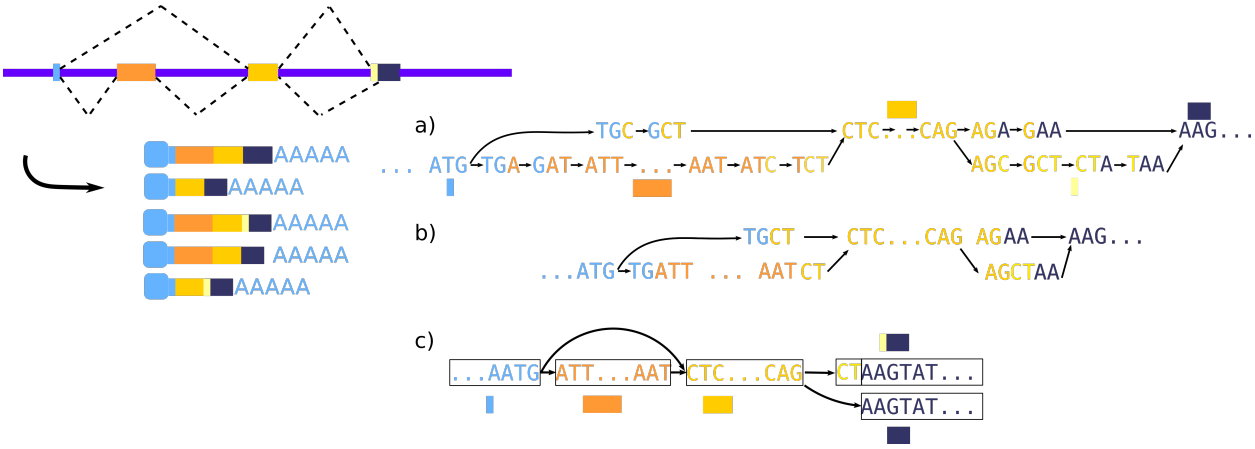

Figure 10: **We take back the complex splicing event example to illustrate splicing graphs.** Several definition were given for these graphs, the first mention is in [86], where they are defined as De Bruijn graphs (a). In such representation, an alternative donor/acceptor cannot *a priori* be identified from a small spliced exon. k-mers in simple paths of the graph can be concatenated in single vertices, resulting in a *compacted* De Bruijn Graph (b). In other papers [31], (c) splicing graphs integrate redundancy in order to better represent the exons. Exons are vertices of the graph, edges represent the juxtaposition of exons seen in transcripts. If two distinct versions of a same exon exist (for instance alternative acceptor), the different versions appear in the graph. The graph's vertices can also retain if they represent the variation of an exon, such as the dark blue vertices. In both case, isoforms are paths in the graph.

RNA and assembly Then the problem was transposed to transcriptomics, where the goal is to find all isoforms in presence. Before NGS, EST assembly relied on other paradigms than the De Bruijn Graph (greedy assemblers such as PHRAP[13] or CAP3[94]), where overlaps were directly computed from the reads sequences (not using k-mers but using alignment), which limitated them in terms of performances. These assemblers considered only local information at each elongating step, which meant that repeats were highly prone to lead to mis-assemblies. The first attempts to describe all possible transcripts from a gene with sequencing data dates from these EST analysis and the notion of splicing graph[86]. Then De Bruijn graphs were preferred with the advent of NGS.

In De Bruijn graphs built from RNA-seq data, the combinatorial nature of transcripts leads to obtain more possible exon combinations in graphs than what is present in the reality (Figure 11 a)). Alternative isoforms, such as genomic variants, generate topological patterns such as bubbles (two flanking nodes linked by disjoint paths) or branching zones in De Bruijn graphs. Moreover, many repeats exist in the expressed regions, thus transcriptomics does not avoid the major issue in assembly. The uneven coverage makes the task to distinguish error from actual variations even harder (Figure 11 c)). Thus alternative splicing events tend to be merged in the graph into complex and dense patterns that make their distinction extremely difficult. The challenge is then to decide which parts of the graph represent real transcripts. Global assembly [210, 78, 194, 253] aims at reconstructing the full-length transcripts. It is expected that the transcripts from a gene will create a connected component in the graph, and that each transcript will be represented by a path in the component. However, all paths do not represent real transcripts present in the original dataset, and genomic repeats make whole transcriptome DBG very complex. That is why choices must be made with heuristics. These (parsimonious) choices may lead to describe less transcripts than what really exists in the

dataset (one strategy is to choose the minimal number of transcripts that can describe all paths in the DBG). Local assembly [200] is more sensitive at the event (expressed SNP, alternative splicing forms) scale. A gene or a gene family corresponds to a bi-connected component in the graph, and variants produce bubbles in which each isoform is represented by a path. Methods tried to take into account read count in order to choose paths, by mapping reads on the components to quantify each variant. Assembly has been a main way to discover new transcripts since sequencing exists. With novel long reads, assembly could be no longer necessary to obtain the different transcripts. Indeed these sequences have the potential to span the whole transcripts sequences. Thus, in the works presented in this dissertation, we will not use assembly to retrieve transcripts. However, assembly paradigms are useful even when twisted from their initial goals. Assembly methods can for instance yield consensus that are interesting for read correction, as will be mentioned in Chapter 3.

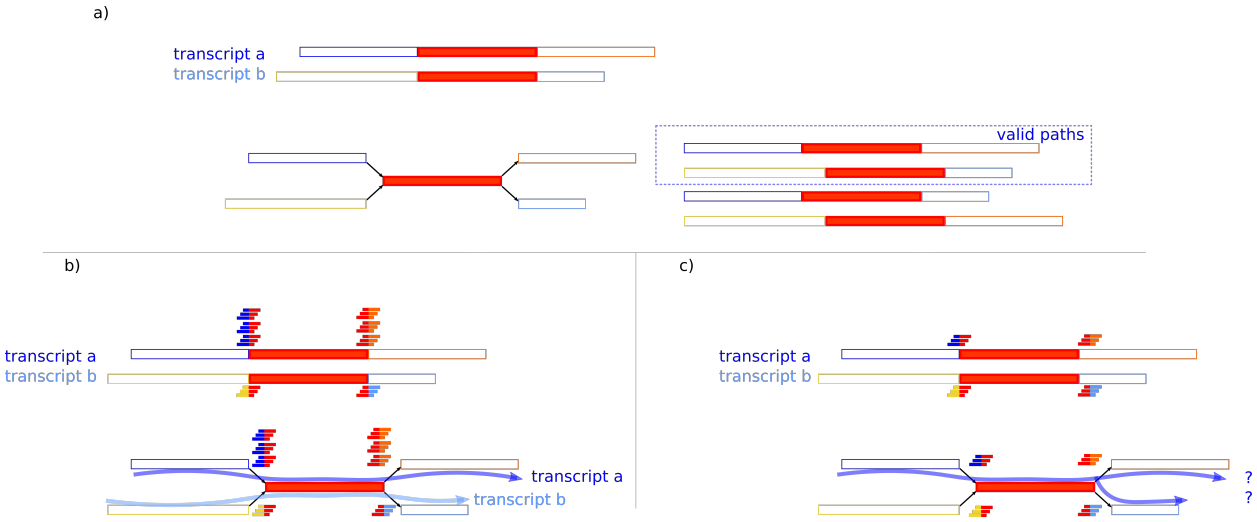

Figure 11: **"X" repetition pattern in De Bruijn graphs.** a) A repeated sequence is represented in red. Its possible origin is not only a repeated region of the genome. It can also be a constitutive exon flanked by alternative sequences. We represent a draft compacted De Bruijn graph. There are more paths in the De Bruijn graph than real transcripts. b) Junctions counts obtained by read mapping help to retrieve the two original transcripts. c) The read counts are not enough contrasted between the two transcript to enable choosing a path. This is a major issue in current assemblies.

3.5 Common indexing schemes

An index is a structure that allows the efficient access to element of an object and/or to associate information to these elements. The "index" can denote both the structure where elements are stored or a given position in the structure. Indexation facilitates access to objects in memory, and represents active fields in computer science and bioinformatics. In bioinformatics, many applications require both texts and graphs to be indexed in order to access their elements to check their presence or modify them. This need was particularly pointed in fields such as sequence alignment and assembly. In both cases, shared subsequences must be rapidly computed and accessed. Since most of the mapping approaches are based on seed and extend, rapidly knowing the presence and positions of words in sequences is crucial. Some substring indexes will be further described in Chapter 1, here we present general associative indexation methods using hashing, that are broadly used and appear in all implementations developed in this work.

A hash function is a function used to associate integers from a given range to indexed elements (called keys). The associated value of a key may be called its hash or its index. Hash functions are used in hash tables, that are arrays that associate an information (called value) to each key. The idea is to get a collection of pairs stored in the array T using a hash function f such that $f(key) = index$ where $index$ is the index in the array, and $T[index] = value$. In most dynamic hash tables implementation, the array is divided into buckets that contain pairs of $(key, value)$. When hash functions do not realize an injection of the initial set of keys, a phenomenon called collision appears. A single hash is then associated to different keys thus several pairs of $(key, value)$ are stored in the same bucket. Mechanisms to handle these collisions are necessary as hash collisions are practically unavoidable. The simplistic mechanism called separate chaining consists in checking each pair of $(key, value)$ of the bucket until the correct pair containing the query key is found. Furthermore in dynamic hash table new pairs of $(key, value)$ can be inserted on the fly and create collisions. To handle this problem most hash table might allocate arrays of higher size than the size of the collection of input keys.

3.6 Main procedures to investigate reads

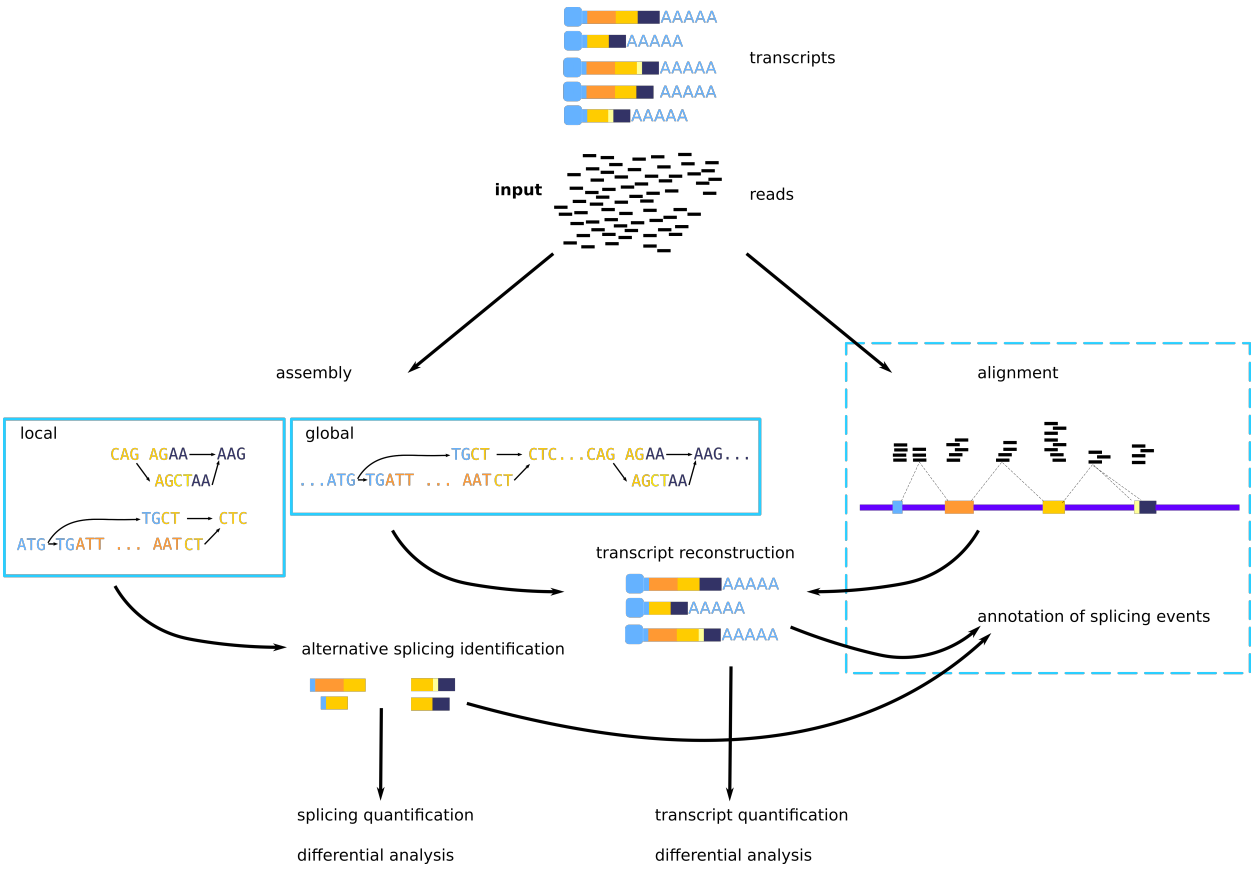

Figure 12: **Summary of pipelines for the study of transcriptomes using short reads.** In the dotted part a reference is obligatory. Assembly-based approaches are on the left and mapping-based approaches on the right. Detection of splicing variants is also possible using mapping but not detailed here.

3.6.1 Correcting reads

Read correction is an essential step to separate sequencing errors from genomic bases. It aims at identifying genomic variants (SNP, indels) from artifacts introduced during the sequencing process. These wrong bases are also limitations to assembly because they represent points where not all reads agree on their sequence, leading to fragmentation of assembly graphs products. Prominent methods rely on static abundance threshold of k-mers, the hypothesis is that k-mers containing errors will appear less frequently in short reads and that these error types are stochastic. These methods are presented in Chapter 3, then long reads that now bring particular issues to read correction that are also thoroughly reviewed in that chapter. Using NGS, read correction is often one of the first step performed in analysis pipelines so that downstream steps do not inherit from noise induced by errors. Such paradigm is directly integrated in De Bruijn graph assembly methods, in which only k-mers seen more times than a certain threshold are considered before starting assembly. It is noteworthy that recently, a shift occurred with long reads in the genomic context, with sequences corrected only once the assembly task is realized [126]. In this work, though avoiding assembly step, we also propose read correction as a late task.

3.6.2 Reference-guided analysis: main goals

In reference-guided studies, reference genomes or transcriptomes are used as bases for the exploitation of new data. Most of the time, reads need to be aligned on these reference sequences beforehand, but some methods avoid mapping. Usually genomes and transcriptomes are not used for the same goals since they do not contain the same information.

Genomes are used to detect known and new isoforms, potentially alternative. First the reads are aligned on the genes sequences, using aligners such as STAR [52], GMAP [250] or HISAT2 [106]. These tools integrate specific features to handle RNA reads that are further detailed in the next chapter. When mapped on reference, reads can be located within an exon sequence, or spanning one (or several, if a small exon is in presence) exon-exon junction. Once reads are mapped, read counts help methods such as the Bowtie-TopHat-Cufflinks suite [116, 234, 235] to identify which transcript is in presence using peaks of read counts on each exon. Reference genomes also provide a baseline for comparison when looking for expressed genomic variants, or RNA editing. GATK [151] is a method for variant calling based on mapping. Exceptions such as CRAC [181] identify variants without aligning the reads, based on their k-mers content. As using references allows easier assessment of results by comparison with well-established biological features, reference-based methods can be preferred in applications where a high precision is required, such as clinical analyses. This is the case for gene fusion detection when linked for instance to cancer. The intuition is that reads from such objects can map on junctions that will link different genes or non colinear exons.

Transcriptomes are often used to determine the abundance of transcripts in presence in a dataset (quantification). This can be done by mapping reads on transcripts, with advantages over genome alignment: easier alignment since no splice-mapping is intended, which provides more simple framework for statistical methods that aim at allocating reads to individual transcripts. In simplest methods, when isoforms have to be identified from one another, only reads covering exon junctions are used, since they are assumed to represent non-ambiguous information. More advanced statistical methods try to assign reads mapping inside an exon to the right transcript [122]. These methods integrate steps to decide which exon-chain is likely according to the observed exon counts. Exactly as in De Bruijn-graph based assembly, such a transcript determination based on selection of certain possibility over a high number of possible combination is error-prone. Again, some methods such as Salmon [175] or Sailfish [176] avoid mapping and use reads' k-mers content to quantify gene expression, in a faster way than alignment.

We will come back to k-mers-based methods for other application than quantification in the next chapter. Quantification bias in long reads is still discussed in the ONT context and benefits from few perspective. It can be forecast that the alignment of full-length reads will allow to dispense with statistical aspects for quantification, that will be done in a more straightforward way. *De novo* mRNA isoform identification and classification by common genic origin will be the main goal in this thesis, thus relating our work to isoform identification. We made the choice not to consider variants at the level of SNPs or indels, given the error rates

and profiles we face. In highly expressed genes though, modules to identify these bases could be developed within the correction step or downstream.

3.6.3 *De novo* studies

De novo applications are the core interest of this thesis work. *De novo* methods are often presented as more "agnostic" with respect to the reference. Thus contrary to reference-based methods, results are not driven by our current knowledge of genomes and transcriptomes since reads are not compared to these sequences before outputting results. They allow to exploit data when no reference/annotation is available, or to avoid the usage of bad quality references. We recall here that a vast majority of species do not benefit from references. Thus *de novo* methods are often presented as allowing to access *the dark matter* of the living. Being reference-free also has the advantage, in case a reference is available, to introduce at later as possible possible biases to results. Results can be computed *de novo* and reference information can be injected afterwards to complete them with additional information that is not present in reads, instead of guiding the whole process. Indeed, reference-free solutions have *a priori* access to less information than reference-based approaches.

Usually, mostly transcript assembly and variant calling are realized *de novo*. For instance, assemblers such as Trinity [78] or Trans-Abyss [194] propose *de novo* assembly of transcripts. Also based on De Bruijn graph but not realizing full-transcript assembly, KisSplice [200] aims ate detecting alternative splicing variants and expressed small genomic variants. Several works based on KisSplice are presented in Chapter 5.

Still, these methods are not fully exempted from biases, and can bring their share of underlying hypothesis, based on our current knowledge of biology. For instance, most *de novo* assembly methods do not report heterozygous variants from expressed regions, or fail to retrieve exhaustive lists of transcripts from a gene. Transcripts assemblers also generally make the hypothesis of colinearity of exons within a transcripts, since this is what has been observed so far. Thus, methods often highlight a fraction of a biological reality and have to be completed by other approaches.

3.6.4 Reads simulation

Read simulation is the production of synthetic reads by computational means, that aims at mimicking sequencing features. This is of interest when developing new methods since the ground truth is perfectly controlled. Simulators embed techniques of various precision to correctly model the expected features of a given sequencing technology, such as read length distribution or error rates and profiles. Since these features can be prone to changes according to protocols, some simulators made the choice to learn sequencing features from real data instead of using direct statistical models. In DNA context, ART [93] will be mentioned again in Chapter 3. The reader can refer to a recent review on these tools [58] for more details.

Many tools for genomic short reads simulation exists, a bit fewer for mRNA, including the Flux Simulator [81] or BEERS [80]. The Flux Simulator is one of the RNA-seq simulator that integrates the most features including fine simulation of gene expression levels. Following new long reads technologies, a generation of simulators reproduces long error-prone sequences from PacBio [170, 225], with only one tool published for ONT [254]. To our knowledge, no published work dedicated to reproduce ONT RNA sequencing experiments. A preliminary work filling this gap is presented in Chapter 4 . For most works presented in this document, we relied either on real data or on straightforward simulations that did not fit all the features of a mRNA sequencing experiment (for instance gene expression).

4 How this work contributes to the study of transcriptomes

4.1 Some current questions on mRNA

Methodological developments in the transcriptomics field follow fundamental and application questions. A vast panel of biological questions nowadays require bioinformatics, because they rely on the processing of data that cannot be processed by hand and/or because they directly pertain to algorithmic questions. Thus current questions motivate works such as presented in this manuscript. Jointly, technologies evolve to propose more and more insight to mRNA sequences. The nature of the proposed data can itself raise new methodological challenges, as well as it gives matter to new biological questions [96]. Sequencing technologies and their problematics will be described in the following section. Here we propose a short, non-comprehensive discussion of current important questions about mRNA. We outline some interesting biological problems that motivate our work in the long run.

4.1.1 Functional transcriptomics

The way transcripts are expressed and their functional roles are widely investigated. Capturing transcripts gives a glance at the expressed phenotypes. In cancer or other diseases, differential gene expression as well as differential splicing are looked for. Mutations can lead to enrichment or depletion of certain exons inclusion in transcripts, or to the inclusion of an intronic region, then to dysfunctioning proteins or other complexes [241, 227, 68, 56]. Given that we can dig deeper and deeper into transcriptome contents, usual questions are to identify which subpopulations of transcript actually contribute to the functions in the cell, and which fraction is noise (or has other functions).

Several projects contributed to build catalogs of transcripts in a functional context. The ENCODE Consortium [43, 42] aimed at building an encyclopedia of DNA elements in human, which would provide a list of functional elements including RNA and the different regulation elements implied in regulation of functional information in cell. The FAANG [7] Consortium extended this goal to several livestock species. The Geuvadis project also produced human transcriptome data, associated with different populations in Europe. Such data was coupled with genomes from the 1000 Genomes Project enables to characterize for instance genomic actors of transcriptome variability such as quantitative trait loci (QTL), that are subparts of the DNA correlated with a variation of the phenotype, and genomic contributors to mRNA expression levels (eQTL) [118].

Expression of family of genes (paralog genes) are also poorly understood because they represent methodological challenges. Family of genes gather several copies of genes that undergone duplication events during evolution. The copies tend to diverge with evolutionary time, each accumulating its own mutations. Copy can express different alternative versions of the transcript, which can be difficult to correctly address to their origin if the genomic loci are not too divergent. Some regions identified as pseudogenes (genes that lost the initial biological function they encoded, and are sometimes not coding for proteins anymore) can in fact express transcripts with differentiated functions [184]. More generally, genomic repeats are a hurdle for transcriptomics analysis since many repeats appear in transcribed regions [51].

4.1.2 Metatranscriptomics

RNA-seq protocol enables to access the expression of mRNA with only a few picograms of RNA material [148]. Thus, it is extremely useful to obtain gene expression information from environmental samples [171, 244]. Such sequencing experiments are called metatranscriptomics (metagenomics when DNA is sequenced). An example of project that produced this type of data is the TARA Oceans expedition [23] that coupled sequencing of total DNA and RNA samples from several stations across oceans to morphological and physico-chemical measures.

RNA-seq paved the way for numerous applications from ecological studies to human diseases (for instance the gut microbiome, i.e. microbial community, is recently at stake concerning many important diseases such as diabetes, obesity or depression) [124, 97]. Species that cannot be grown in lab can be sampled directly from the environment, resulting in a mixture of RNAs from the different individuals that were present. These studies also allow to investigate systems as a whole instead of separating each actors. Such as transcriptomics, one challenge in metatranscriptomics is the contrasted heterogeneity of coverage of the different expressed reads. Biologists are interested in identifying the core functions present in the system, as well as the secondary and more rare functions. They address this question by retrieving transcripts through assembly, then proteins are derived from mRNA sequences using prediction tools. However metatranscriptomics assembly is at least as hard as regular assembly. To this extent, long reads are promising since they no longer require assembly. Short reads are usually preferred since they give access to very high numbers of fragments that capture a lot of the variability present in a sample. However pilot metatranscriptomics projects with long reads exist is TARA Oceans expedition for instance.

4.1.3 Biological models for transcription and splicing

Fundamental questions concerning genomes evolution and splicing apparition are also at stake. Some general rules such as exon definition started to be stated in the 2000's, however there are still many questions. For instance, it remains unclear to which extent exons appear independently in transcripts, or are co-selected to create transcripts [230]. This is mainly due to the lack of exhaustive and reliable reference transcripts list, due to the limitations of short reads.

4.2 Main contributions

Given the general concepts that have been introduced we can present this thesis work as bringing algorithmic and computational solutions to three main problems in the long reads context: 1/detecting similarity between reads; 2/grouping reads that originate from same genes; 3/correcting reads sequences, all of these tasks being realized *de novo* (Figure 13). Finally, the three presented modules can be seen as a three bricks, that once channeled create a pipeline for *de novo* mRNA analysis using long reads. This idea leads the whole manuscript construction. All in all, we aim to propose a group of tools dedicated to describe the isoform content of a transcriptome sequencing, without a reference.

This thesis work invokes methods that belong to quite differentiated and broad fields in bioinformatics and computer science. At the beginning of this thesis, very few methods considered long reads for RNA studies. We anticipated future needs regarding these technologies by looking at how long reads are currently used in the RNA community. We took the stance not to commit to hybrid (i.e. mixed short and long reads) experiments and to rather focus on long read only methods, that were rare at the time we began. Until now, mostly Iso-seq has been involved in the depiction of novel isoforms in transcriptomes of various species, in projects that also involve short reads most of the time [8, 214, 1, 240, 89]. ONT studies have just started to develop and we aimed at pioneering in that direction given the interesting features of ONT platforms [248, 168, 22].

4.2.1 Similarities between reads

We previously identified that one of the fundamental task was identification of pairwise similar sequences. We wanted to propose a method dedicated to finding similarity between pair of long RNA reads. This first work would be the basis of almost any other applications we could propose. It is noteworthy that contrary to short reads, mapping to another read or to a reference makes the difference in long reads. Thus two long reads with more than 10% errors are way more dissimilar than one read and the reference that is supposed to be clean from errors. Studies shown that at the current error rates, seeds no longer than 15-17 should be used [32]. Longer seeds can hardly be found in these sequences. This can have significant impacts on the quality of alignments in species that possess large genomes, since 15-mers are frequently found

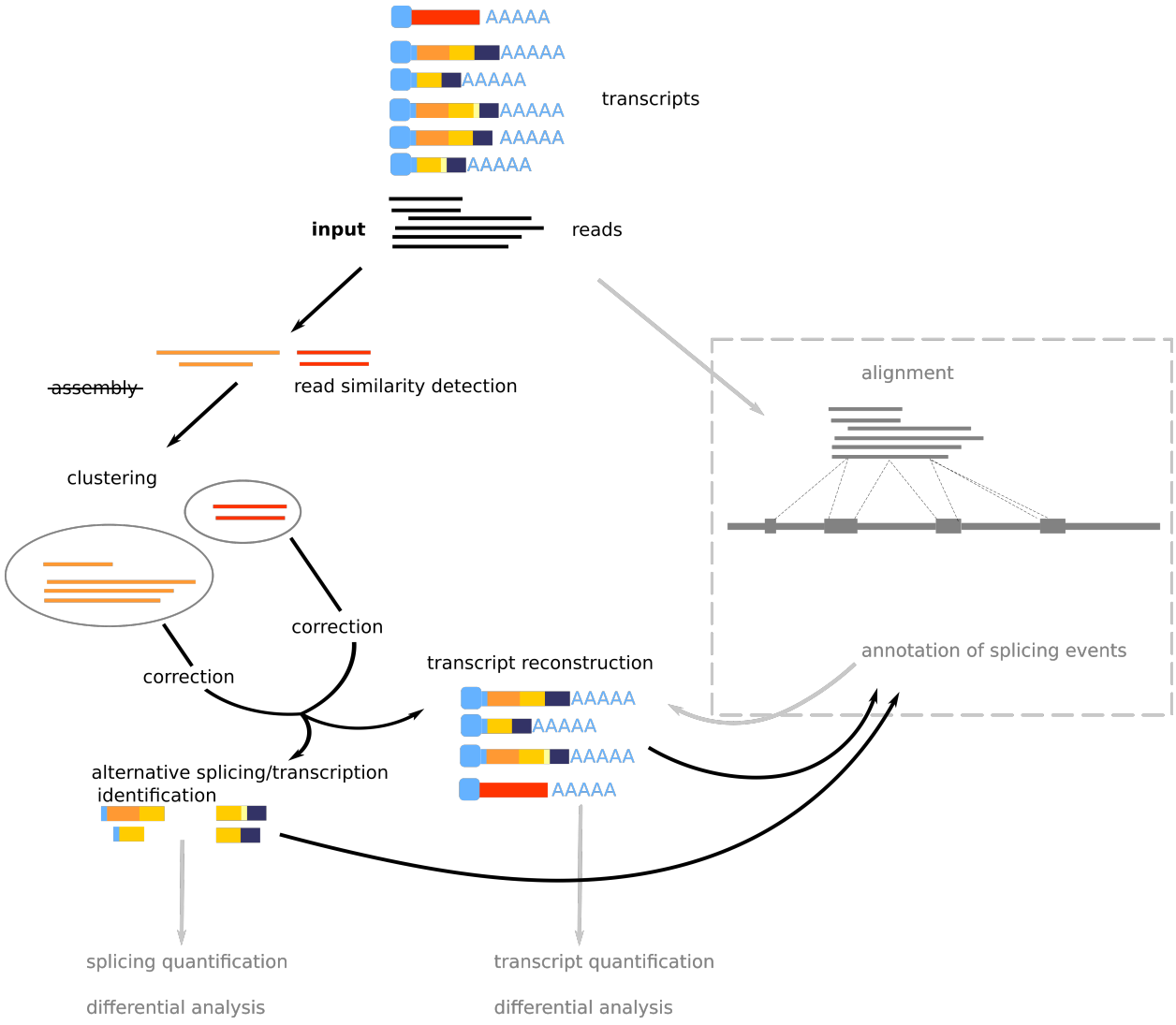

Figure 13: **Main steps of *de novo* transcriptome study using long reads developed during this thesis.** A main particularity is that assembly must no longer be required. However reads need to be grouped per gene, hence clustering is adopted in this pipeline. The gray parts are not covered by our work. See Figure 12 for a comparison with short reads approaches.

repeated across the genome. Moreover, the alignment step is made more difficult by the numerous indels and homopolymer errors. Our work on this subject led us to propose a new structure to index sequences and an alignment-free (thus not relying on paradigms presented before) method for sequence comparison. All related results are presented in Chapter 1. Currently speaking, we only have preliminary results on the application of this method to transcriptomics long reads. However, the structure proposed is highly scalable, we show that it has direct applications with meta transcriptomics short reads. We present a first work that initiates the validation of our tool on biological instances.

4.2.2 Clustering of long reads

In transcriptomics, *de novo* clustering used to be reserved for assembled short reads or single cell experiments. In Chapter 2 we propose a novel method to cluster long reads per gene that takes into account the current error rates and profiles. Such a task can be realized with mapping in cases a reference exists, but were hardly explored with long reads in *de novo* context. This problem is difficult given the intertwined problems of scaling, current noise in data and absence of dedicated mapping tool for RNA. Our method is meant to be a first milestone to investigate transcriptomes with the new leverage of long reads, even for non model species. This work was validated using real mouse transcriptome sequenced on nanopore platform. The algorithm itself pertains to community detection algorithms, and its extension to other bioinformatics applications is still open.

4.2.3 Correction/consensus of long reads

As pointed out in this introduction, the current quality of sequences is not high enough to provide reliable functional annotations. In other words, we predict proteins and their roles with difficulty because ORF predictions suffer from shifts due to indels. Our aim is to correct reads so that the final sequences conserve the exon structure of the original read and have an increased base quality. This correction task is in fact highly linked to several problems: finding common exons or common consecutive exon chains in order to correct these regions altogether, and thus, representing the alternative content in exons of a given gene, and extend this result to the transcriptome. In a work in progress, we will propose a module that proposes corrected consensus for transcripts. Its strategies are detailed in Chapter 4. Such module works on the basis of our clustering method, and additionally brings identified alternative events and their counts in the dataset. Again this work is meant to be applied *de novo*. On the long term, our goal is to provide clean novel isoform using the long reads, in order to help enriching the existing annotations. Since many approaches rely on these annotations (quantification, gene identification, ...), possessing more complete and accurate transcripts annotations is crucial for transcriptome analysis. Works on correction also led us to propose a tool for long read correction assessment to benchmark our method, that is presented in Chapter 3.

Chapter 1

Compare pairwise sequences in (meta)-transcriptomics data

Our work aims to be a channel through which a transcriptomic reads dataset undergo several steps, from the complex and tangled initial dataset in which reads are all mixed up to a final state in which reads are grouped by their initial expressed genes, and separated by isoforms. Such a result requires several treatments performed on reads in order to compare them, decide which are related and understand finer difference between them. The first requirement is to be able to compare nucleotidic reads content despite sequencing errors and taking into account biological variations, in order to spot related reads. Hence this first chapter presents works that pertain to sequence similarity detection.

Many works contributed to this topic, some of them in a particularly seminal way, as illustrated by the software BLAST that has one of the most cited related paper [5]. While its concept is intuitive, similarity detection engages in practice a plethora of heuristics. Thus, we will present a non exhaustive background of the methodological aspects of sequences similarity computation, with its main principles and methods. This field raises many problematics that fall in the different applications of similarity detection. We will introduce problematics that RNA sequences raise, and we will not mention all current general problematics but rather explain those to which we think our work contributes to. We will present novel methods to detect similarity between sequences, with the data structures, implementation choices and algorithms they rely on. These methods enable the comparison of a target sequences dataset to a query dataset, they can be reads sets for instance. They can help identifying reads that are in the intersection of two sets, or to retrieve similar reads by comparing a set to itself. This latter application is the one needed in the global proceeding of these thesis. However, these contributions are firstly designed to work generically on short reads, and remain initial proofs of concept. Indeed, they represent very general frameworks to detect similarity that can be in a second time adapted to more precise goals. In a second part, we wanted to show that we can work through using such proof of concept. We will present a collaborative work where our tool was applied in the context of real biological models and data. We will take time to describe the biological issues of this work. Indeed they are typical motivations for methods such as those presented here. They also are an opportunity to validate the developped approach. Then we will present how our method was used to tackle concrete bioinformatics needs within a pipeline. In a last part we will rely on that framework to draw new ideas that allow the exploitation of long reads, which is the main goal to be achieved all along this document. Straightforward ideas were implemented and we present benchmarks against very recent methodological contributions to the long read field. Other ideas are described but do not benefit from implementation yet.

1 Problem statement and previous works

This section provides general concepts about similarity detection between reads, and how it was applied to RNA molecules, as well as the main approaches to date. It gives the intuition of the difficulty of the problem when applied on the Third Generation Sequencing (TGS) reads, and presents a second challenge that is the scalability of this family of methods. It introduces the aspects of our contribution, that is a method related to pairwise sequence comparison of reads in a set.

1.1 Overview of heuristics applied to pairwise sequence comparison

In this section we provide an overview of standard methods, aside from already presented *seed and extend* mappers, that enable sequence comparison.

1.1.1 Alternatives to seed and extend

The broad applications of sequences alignment include many different problematics in genomics and transcriptomics. Mapping, through its *seed and extend*, has been presented in the introduction such as the main procedure to compare sequences. Tools to map two strings both at the genome scale exist but are out of this scope, thus we will focus on comparisons of shorter sequences such as reads. That being said, other heuristics

for sequence comparison exist. Contrary to mapping algorithms, pseudo-alignment procedures do not involve the alignment of sequence at the base wise level. However, they output sequences similarity but based on matches of words such as k-mers. This is different of alignment since word matches are searched. The similarity is estimated using the abundance of shared words, for instance by computing the number of shared words between datasets. By avoiding the quadratic component of dynamic programming approaches, they are usually more efficient. Both mappers and pseudo-aligners can work *de novo*. However, pseudo-aligners usually work at the data-set level while mappers work at the sequence level. Another difference is that the main data structures for pseudo-alignments methods (frequency vectors) have an optimal behavior in terms of performances for very small k-mer sizes (i.e. less than 15). However, some methods do not compute distances and only rely on the k-mers presence/absence [249]. The set of our contribution methods, Short Reads Connector, falls in the definition of pseudo-alignment. Finally, less methods integrate an alternative to k-mers using spaced-seeds [105]. A concept related to spaced-seeds is presented in the very last part of this chapter.

1.1.2 RNA mapping

Interestingly, RNA mapping shares characteristics with mapping of large structural variants in genomes. When mapping against a reference genome, RNA sequences are split to span each expressed region they come from (often exons). This is, to a lesser extent, the case when mapping two alternative isoforms. Large gaps are formed, thus dedicated methods were developed to deal with this feature. RNA mapping to reference comes in two flavors, *exon-first strategies* and strategies that adapt the *seed and extend* paradigm. Many tools exist, we give a few examples with some of the most used. The first category ([234, 242] for instance) starts by aligning reads on the reference without particular relaxed condition on gaps. In a second pass, unmapped reads are split and fragments are realigned. The second strategy [52, 250] uses seeds and searches for maximal length subsequences that can be aligned without large gaps on the reference. Then these subsequences are clustered when they are not separated by more than a window that represents the size of an intron. Pseudo-alignment is used in the RNA context, but usually it is for quantification purposes [176, 175, 25]. Indeed, these pseudo-alignment methods do not allow precise positioning of reads, which is often required to precisely define exons bound. An exception, though, is CRAC which relies on k-mers to produce mappings [181]. Thus, many splice-mappers use catalogs of splice-sites (annotated or discovered) to map reads with better precision and sensitivity.

1.1.3 Long reads

The first tool to map long reads appeared circa 2012 to map PacBio reads on reference [32]. Since these technologies are only a few years old, a small number of tools exist in the literature to handle the long reads. Due to the error rate, the task of mapping on a reference and the task of finding overlaps between reads became more different. Tools that compute read overlapping are mostly Minimap [126], DALIGNER [161] and MHAP [18]. A second implementation of Minimap (Minimap2) was proposed recently. These tools can also be adapted to map on a reference. Other tools include mostly BLASR, GraphMap [224], and adaptations of NGS mappers such as BWA-MEM. Most of them integrate adaptable parameters to tailor comparisons to ONT or PacBio specifics.

All tools adopted the seed and extend paradigm. However, the tool Minimap is the unique one to allow a mode where only the seed collection is performed. Such feature is interesting for applications that do not require precise read positioning, and will be discussed again in Chapter 3. State of the art tools like Minimap and MHAP make the choice to change the seeding paradigm by using local sensitive hashing using minimizers. Minimizers were first used in [193] but it took around ten years before they spread to recent mapping and pseudo-alignment techniques. Minimizers are smallest k-mers with respect to an order (for instance lexicographically, or with respect to the result of a hash function) within a set of k-mers. Consecutive k-mers tend to share same minimizers, thus this property provides an interesting way to sample the seeds. Seeds are usually sketches of minimizers instead of k-mers. They were first used in [193] but took ten years

to rapidly spread to almost all mapping method recently published. GraphMap does not use minimizers in its first implementation but relaxes the condition of exact matches of k-mers, by using spaced-seed designs that allow errors. A second implementation of GraphMap then integrated minimizers. A last tool that was published in late 2017 uses a novel seed strategy where the mapping quality is ensured by a seed clustering strategy [104].

These tools integrate a second step that allows them to increase the precision of the alignments they predict. Indeed, seed-and-extend using rather small k-mer allows to detect rough alignments. Minimap uses a memoization step to find colinear chains of seeds that give more guarantees that the alignment is correct. MHAP uses a second pass that computes Jaccard distance of the two regions with smaller k-mers. GraphMap also relies on a seed-chaining step.

1.1.4 Data structures

As previously mentioned, up to date mappers rely on specific data structure to index the reference sequences. In order to enable the comparison with the data structure used in our contribution, we recall the main objects used to compute indexes.

Text indexes Two generations of tools proposed hash-table indexing of either the reference or the query sequences. Such tools were intractable on large texts such as eukaryotes genomes and transcriptomes either due to their high memory footprint or because they require a complete scan of genome to align a group of query sequences. Another straightforward method to index such text is the suffix tree. The suffix tree (or trie) is a tree structure where distinct substrings of the reference are represented by different paths from the root of the suffix tree [150]. Query sequences can be aligned against each path from the root up using dynamic programming. However, constructing suffix trees on large instances of the scale of eukaryotes uses too many resources even on best known implementations were 50 GB of memory were reported for a human genome [113]. To compensate such a high memory bottleneck, several compressed data structures were proposed to efficiently index biological sequences, such as suffix arrays, Burrow Wheeler transform (BTW), and FM-index. A suffix array [143] stores the n indices of suffixes of a word of size n ordered lexicographically, and associates to them their starting positions in the word. Both suffix trees and suffix arrays allow to query if a substring is present in the indexed text and to count the occurrences of such substring. Mappers such as STAR are based on suffix arrays and benefit from the quick similarity computation they enable. The usage of the BWT in the FM-index structure allowed to simulate a exploration of the suffix tree without the need to store it explicitly via a process called backward search. Furthermore the advantage of the FM-index over a suffix array is that the reference can be queried while being compressed. The FM-index have been found to be the more efficient for DNA alignment and is integrated in most NGS alignment tools such as BWA or Bowtie [129, 125, 116, 117].

Hash function based indexes Hash tables were described in the introduction chapter. Some mappers are based on hash tables, such as SHRiMP2 [47]. Other structures such as Bloom filters, that are not hash table since they do not enable to associate information to a key, also allow to efficiently store and access a set of keys. However, Bloom filters tend to replace hash tables when possible in recent implementations, thanks to their efficiency in memory. Bloom filters [107] are bit arrays of size b initialized with b zeros. For a given key they are filled with l ones at positions returned by l hash functions applied on the key. They can be queried by applying hash functions to an element and verify the resulting positions are ones. However, because of collisions Bloom Filters provide false positives, i.e. report an absent element as present in the filter during a membership operation. Two methods based on Bloom filters are presented later on. Bloom filters, such as hash tables, are not limited to alignment purposes and are for instance involved in implementation involving assembly [33, 200]. Their design gives them the property to be very memory efficient. For instance, one k-mer can be stored using only 8 bits.

1.2 Current methodological challenges

In this section we present two challenges that were faced with the methods presented in this first chapter. While they are not the only two opened questions, we believe that both matter a lot in many current works. The two axes presented are RNA long reads and scalability. These axes can be thought as parallel since they are not systematically associated in concrete applications. However, RNA reads similarity and highly voluminous datasets sometimes meet, for instance in metatranscriptomics projects. Moreover Nanopore platforms' throughput increases and we believe long reads will be more and more present in high scale sequencing. For instance they started to be used in TARA oceans projects.

1.2.1 Mapping RNA long reads

Currently, no tool was specifically designed to discover overlaps between pairs of RNA long reads, which is our main interest in this chapter. These reads are too long and erroneous to directly apply NGS methods, and too numerous and short (since they only span a gene) to really fit in genome-wide alignment tools such as MUMmer [50]. Long reads mappers were at first designed for genomic assembly purposes. However, very recently some tools started to integrate support for RNA. GMAP was shown to operate quite well for the mapping of these reads on genomes, contrary to other NGS tools [111]. GraphMap was adapted for long read mapping to reference transcriptome.

1.2.2 Strategies for scalable similarity computation

Some tools were conceived specifically to enable the comparison of read datasets at huge scales. They are dedicated to short reads. Compareads [142] indexes k-mers from a read set in a Bloom filter and can query k-mers from a second read set on the indexed set. Bloom filters permit to perform membership operations but they do not allow to store additional information. However, they are the reason why the comparison was made between the queried read set and the set of indexed k-mers as a proxy to the indexed read set, since no information (i.e. read of origin) can be associated to k-mers. Commet [141] generalized Compareads usage to more than comparisons of pairs. Because of the properties of construction of the Bloom Filter, Compareads or Commet are not very efficient on k-mers on rather small size (such as those ≤ 20). It is also impossible to gain knowledge about intra read set similarities (i.e. compare a set to itself) using Compareads or Commet. Simka [15] or MASH [169] are also tools based on k-mers. They use them to estimate different ecological distances such as Jaccard or Bray Curtis distances. However, one difference with Commet-like approaches is that work at the k-mer scale and compare datasets as wholes. Their result is a range of distances that Commet-like approaches do not provide, but there is no information at the read scale. Figure 1 sums up the different methods. All these tools were mainly tested on metagenomic samples. Other tools out of this scope also committed to lowering their memory and time footprint in metagenomics context, and specialize in classification of reads using reference banks or work with specific sequences such as 16s RNA (ribosomal RNA).

1.2.3 Perimeter of this work

Our work is rooted to the Compareads and Commet tools. It does not perform mapping and relies on a different data structure than those presented above. The indexed sequences are reads or assembled sequences and contrary to many pseudo-alignment tools, the range of k-mer sizes it works with is rather high (greater than 15). It was designed to enable pairwise sequence comparison for voluminous datasets, and can work at the read level. It cannot provide mapping of a sequence on a reference genome or transcriptome but it can report similarity retrieved for pairs of sequences based on k-mers content. Its method is explained with short reads and it started to be used by the community on short reads applications, however as it is generic and agnostic about the sequences natures and length, we present a proof a concept that computes similarity for pairs of long and spurious reads.

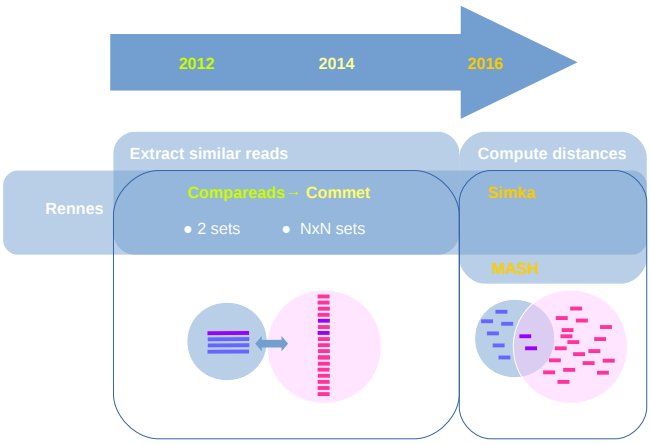

Figure 1: **Tools based on k-mers that help finding similarities between datasets.** GenScale team specializes in producing scalable tools for broad genomic purposes, thus that many of these tools were made in Rennes. Compareads and Commet compare a set of reads to a set of k-mers. They are based on a Bloom filter for indexation. Simka and MASH are shown to illustrate that k-mer based methods not always work at the read level, they compute ecological distances using k-mers content of read sets.

2 Short read connector: two scalable methods to study similarity between sequences sets

In this section we present a first contribution brought by this thesis: a set of two tools called Short Reads Connector-counter and Short Reads Connector-linker that compare pairs of sequences datasets and output similarity information. For the moment, we narrow the context to short reads, as they were primarily implemented and tested with that use case. In the next section we will put at stake other types of sequences. We contributed to the method conception and algorithmic ideas, the implementation of both Short Read Connector tools was mainly realized by Pierre Peterlongo and the whole relies on a data structure conceived during Antoine Limasset's thesis of which Guillaume Rizk provided an optimized implementation dedicated to nucleic sequences application. Short Read Connector tools (Short Read Connector-linker and Short Read Connector-counter) were implemented within GATB library [54], maintained and available[1]. In short, several objects will be presented in the following sections:

- ◇ Short Reads Connector-counter (SRC-c)'s goal is to approximate the number of occurrences of reads from a read set $\mathcal{Q}$ in a read set $\mathcal{T}$.
- ◇ Short Read Connector-linker (SRC-l) links reads from the query set $\mathcal{Q}$ to reads of the indexed set $\mathcal{T}$ using their identifiers.
- ◇ Both rely on a data structure called the Quasi-Dictionary.
- ◇ They take as input a target read set that will be indexed, and a query read set that will be compared to the target set.

[1] `github.com/GATB/short_read_connector`

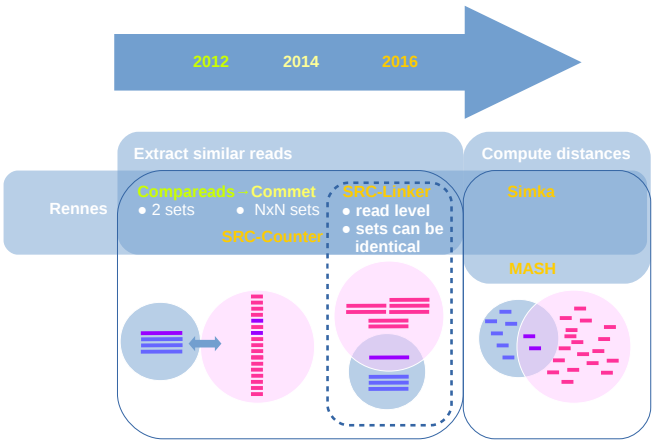

Figure 2: **SRC and related literature.** Figure 1 is completed with Short Read Connector tools: SRC-l and SRC-c. GenScale team specializes in SRC-l compares two reads sets using a MPHF. It can be noted that minor implementation changes could lead SRC-l to compare N vs M datasets in one run, for instance by building one index per indexed set. SRC-c can be seen as performing the same operation than Commet, but relying on a more efficient data structure.

- ◇ They output in a file text the reads that share similarities between the two sets, with different information according to the tool used.

We start by the presentation of the underlying data structure, then we explain both tools.

2.1 Presentation of the data structure

As pointed out in the introduction of this chapter, there is a long list of tools that can perform mapping in the general case and find intersection between datasets, but mapping tools are not adapted to very large (i.e metagenomics or metatranscriptomics) experiments. On the contrary, tools such as presented in 1 scale voluminous experiments but do not perform mapping. Short Reads Connector tools were firstly developed with those challenges in mind. In particular Short Reads Connector-linker can be thought as a next step after Commet, in the sense it allows more resolution than Commet (shown in Figure 2), in the sense that information is retrieved at the read level rather than at the dataset level. This is because the data structure is not the same, Short Reads Connector's data structure allowing more operations than the Bloom filter. This data structure was called *Quasi-Dictionary* (QD), referring to dictionaries such as those in Python that implement hash tables, but with a non-deterministic aspect. The QD is a probabilistic data structure that enables to index elements of a dataset $\mathcal{A}$ and then query elements from a second dataset $\mathcal{B}$, with elements being words from a given alphabet. It is stated as probabilistic because when queried, the QD returns false positives in certain cases. This is detailed in the following.

2.1.1 Hashing scheme of the Quasi-Dictionary

The QD enables to associate to each key a piece of information like classic hashing tables. However, contrary to hash maps such as C++11 `unordered_map`, the QD is a probabilistic data structure, based on a minimal

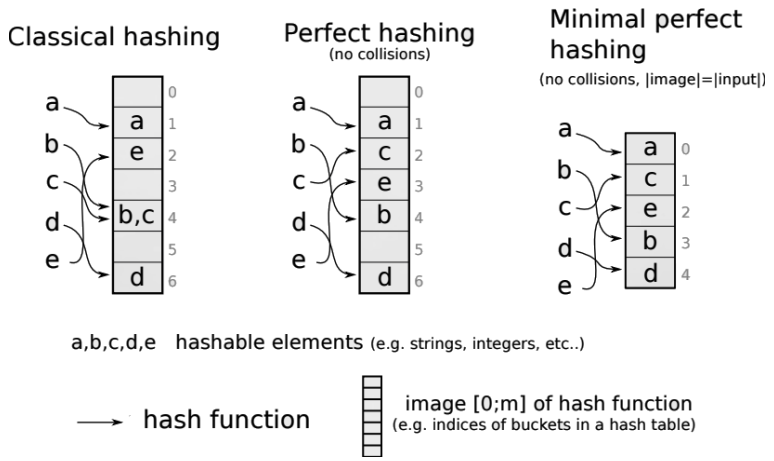

Figure 3: **Indexation using a MPHF**, figure from Rayan Chikhi for presenting the MPHF in [134].

perfect hash function (MPHF). MHPF are hash table that possess the property to be both "perfect" (i.e. no collision occurs for keys) and "minimal" (i.e. it allows to allocate n buckets for a set of n keys). An illustration is proposed in Figure 3. In other words, a MPHF associates any element from a set $\mathcal{A}$ to a unique value in $[0, N-1]$, with $N = |\mathcal{A}|$. It requires less amount of memory than classic dynamic hash tables that rely on pointers to deal with collisions. MPHF are constructed based on static sets. Each MPHF is computed especially for the set to be indexed. A large literature intends to find the best approaches to construct MPHF with lower costs in memory, that is absolutely not bounded to bioinformatics applications. In our case, a method proposed by Limasset et al. does not focus on reaching the theoretical bound for memory, but rather propose an algorithm to construct quickly and in a reasonable amount of time a MPHF, with demonstrations of its scalability to billions of keys. We relied on this implementation as it is quick and practical.

2.1.2 Quasi-Dictionary algorithms

After introducing the underlying core structure of the QD, we state the algorithms that allow to index and query sequences.

Index creation Algorithm 1 shows how QD indexes elements from $\mathcal{A}$ using the MPHF. According to the MPHF properties, QD associates each key a unique value in $[0, N-1]$, with N being the cardinality of $\mathcal{A}$. Any element absent from this initial set of keys $\mathcal{A}$ is called a stranger key. Using a MPHF, a queried stranger keys may be associated to a value in $[0, N-1]$ with a certain probability. The authors of [134] did not provide a way to compute this probability, thus we rely on a simple hypothesis that is: the MPHF will wrongly associate a stranger key to a value in any case. We expect such scenario to happen a lot since the two datasets to compare are not identical. Thus the behavior of QD can be erratic on stranger keys if not controlled, which means providing false positives. In order to limit the probability p of false positives value for our structure, for each indexed element $s \in \mathcal{A}$, we store a fingerprint value associated to s, denoted by $fg(s)$, in an array FG of size N. For each key s, the value of $QD.MPHF[s]$ in $[0, N-1]$ is used as a pointer to an index in the array FG, where are stored the associated fingerprint and pieces of information. The fingerprint of a word s is obtained thanks to a hashing function

$$fg : \Sigma^{|s|} \rightarrow [0, 2^f - 1]$$

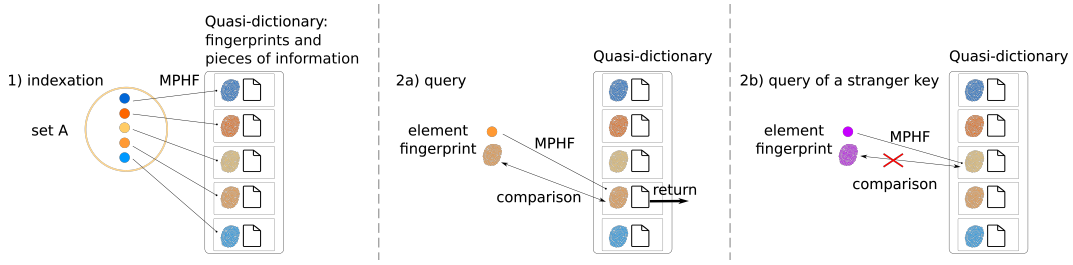

Figure 4: **Indexation and query using the Quasi-Dictionary.** 1) Elements from a set $\mathcal{A}$ are indexed using the MPHF and corresponding information and fingerprint are stored in an array. 2a) Query of a key. 2b) Query of a stranger key (not present in $\mathcal{A}$).

Algorithm 1: Create Quasi-Dictionary

Data: Set $\mathcal{A}$
Result: A Quasi-Dictionary QD indexing elements of $\mathcal{A}$

$QD.MPHF = create\_MPHF(\mathcal{A})$;
foreach s *in* $\mathcal{A}$ **do**
  $index = QD.MPHF(s)$;
  $QD.FG[index] = fg(s)$;
return QD;

. A high f value decreases p and increases the memory usage that is $N \cdot f$ bits for the FG array, and *vice versa*. Thus fingerprints are used as to better ensure a queried element is correctly pointed to by the MPHF (for instance, default f value in our tools is 12). Figure 4 1) summarizes this process. In practice we chose to use a xor-shift [147] hash function to compute fingerprints, for its efficiency in terms of throughput and hash distribution even on large sets of keys.

QD query The querying of the Quasi-Dictionary with a word w is straightforward, as presented in Algorithm 2. The query of an element w is realized by comparing the fingerprint of w, $fg(w)$, to the fingerprint stored at the given index. If the two values are equal, we consider that the queried word w actually corresponds to the element indexed, and that the associated information stored in FG can be returned. We present a first example of query in Figure 4 2a), and an example of stranger key query in Figure 4 2b).

Algorithm 2: Query Quasi-Dictionary

Data: Quasi-Dictionary QD indexing set $\mathcal{A}$, word w
Result: A value in $[0, |\mathcal{S}| - 1]$ or -1 if w detected as non indexed

$index = QD.MPHF(w)$;
if $QD.FG[index] = fg(w)$ **then**
  return $index$;
return -1;

2.1.3 False positives

The probability p described above can be seen as a false positive rate. It means that depending on f, there is a small chance that QD returns a value associated to a stranger key. Thus p $\neq$ 0, unless there is no information loss when computing the fingerprint of an element $fg(w)$, that is, the fingerprint and w are the same size (we refer to this particular case as the "perfect mode" of QD). In practice we enable to keep p value low ($p \approx 2.10^{-4}$) thanks to the fingerprints, since with fingerprints of size g there is a probability $\approx \frac{1}{2^g}$ to obtain a false positive. On the contrary there is no chance of false negative (i.e. the QD always recognized an indexed element).

2.1.4 Performances

2.1.5 Time and memory complexities

QD inherits from the MPHF implementation characteristics. The structure can be constructed in $O(N)$ time and uses ≈ 3 bits by elements. Such a value was carefully chosen during the MPHF design to allow a good trade-off between on one hand speed up of the MPHF construction and query, and on the other hand limiting memory footprint. The fingerprint table is constructed in $O(N)$ time, as the fg function runs in $O(1)$. This table uses exactly $N \times f$ bits. Thus the overall QD size (not taking into account the size of the values associated to each key), with $f = 12$, is ≈ 15 bits per element. The querying of an element is performed in constant time and does not increase memory usage.

2.1.6 Performances of the implementation

We designed a benchmark to compare our implementation to a standard and efficient implementation of hash functions: the `unordered_map` structure provided in the Standard Library in C++11 (QD being implemented in C++11). In this experiment, the index set $\mathcal{A}$ is a set of k-mers from a read dataset. k-mers from the same dataset compose the set $\mathcal{B}$. Here we do not mean to prove that the structure was functioning on a particular type of reads such as RNA-seq, we rather wanted to demonsrate it was scalable on any generic read set. We then chose a metagenomic read set. We performed tests on DNA-seq short reads from a metagenomic *Tara* Oceans [100] read set ERR59928[2], composed of 189,207,003 reads of average size 97 nucleotides. From this read set, we created subsets by selecting first 100K, 1M, 10M, 50M and 100M reads. We asked for k-mers of size 31 to be indexed from these reads. We chose 31 as it is a common value to distinguish k-mers in rather large genomes and metagenomes, that is often used for instance in assembly. The task of extracting all k-mers from each subset to be passed to QD and `unordered_map` was performed using DSK [191]. We assessed wallclock time and memory peak for constructing (i.e indexing k-mers) and for querying (i.e. doing membership operations on the initial set of k-mers) the Quasi-Dictionary using the default fingerprint size $f = 12$ and the C++ *unordered_map*, on a machine that had 252GB of memory. We report the results in Table 1.1. QD is here more performant than the classic hash table to index and query k-mers. As expected, the Quasi-Dictionary data structure size increases when f increases. However, an interesting result is to be noted on the usage of QD's "perfect mode". Here, k-mers are of size 31, which means that using a fingerprint of size $f = 62$ is sufficient to represent exactly any 31-mer, as any base is encoded on 2 bits. The size of QD with $f = 62$ remains on average 4 times smaller than the size of the hash table. Thus, as QD is faster to construct and to query, the usage of the "perfect mode", thought being the most resource-consuming for the QD, still presents only advantages compared to the hash table usage in this example.

[2] `http://www.ebi.ac.uk/ena/data/view/ERR599280`

Indexed Data set (nb indexed k-mers)	Construc. time (s)		Memory (GB)			Query Time(s)	
	QD	Hash	QD	QD62	Hash	QD	Hash
1M (64×10^6)	16	96	0.23	0.61	2.46	11	17
10M (622×10^6)	174	979	1.78	5.40	23.58	11	17
50M ($2{,}813\times10^6$)	538	4,445	7.92	24.29	106.23	11	19
100M ($5{,}191\times10^6$)	1,322	7,995	14.58	44.80	202.88	13	19
Full ($8{,}784\times10^6$)	2,649	-	24.75	75.88	-	15	-

Table 1.1: Wallclock time and memory used for creating and for querying the Quasi-Dictionary using the default fingerprint size $f = 12$ (denoted by "QD") and the C++ *unordered_map*, denoted by "Hash". Tests were performed using $k = 31$ and indexing all k-mers of the set. The query read set was always the 10M set. We additionally provide memory results using the "perfect mode" of the Quasi-Dictionary with a fingerprint size $f = 62$ (denoted by "QD62"). Construction and query time for QD62 are not shown as they are almost identical to the QD ones. On the full dataset, using a classical hash table, the memory exceeded the maximal authorized machine limits (252 GB).

2.2 Short Reads Connector methods

2.2.1 General algorithms and parameters

Short Reads Connector applications use a QD for the data indexation. The main differences between the two tools SRC-c and SRC-l are the information stored in the QD and the operations during the query. In order to explain SRC's methods, we take the example of two reads sets, a target set $\mathcal{T}$ and a query set $\mathcal{Q}$. The two reads sets can be different or identical. We use read sets for the sake of the simplicity of the explanation, however in the following section we will show applications to other types of sequences. The first step, prior to querying, will be to extract a k-mer set $\mathcal{A}$ from $\mathcal{T}$ to be indexed. k-mers are said solid if their count in a given dataset is above or equal to a certain threshold t. Solidity thresholds are used to filter out rare k-mers that are more prone to contain errors [192]. Usual k-mers count distributions contain a peak of k-mers which count is low in comparison to the rest of k-mers. In SRC, filtering of solid k-mers is performed with a first pass of DSK. In addition to deal with noise, removing these k-mers helps reducing the size of the input set passed to QD. Rare genomic k-mers can be lost, thus the choice of the threshold can be particularly impactful for any sequencing data in which coverage is not homogenous. Certain applications or performance purposes motivated to add the possibility to select solid k-mers from the intersection of the k-mers from query and target datasets. Once k-mers from read set $\mathcal{A}$ are indexed (Algorithm 1), read set $\mathcal{Q}$ can be queried. k-mers from reads of $\mathcal{Q}$ form the set $\mathcal{B}$ of keys that are looked up in the QD (Algorithm 2). Outline of both SRC-l and SRC-c procedures is shown in Figure 5.

2.2.2 Short Reads Connector Counter

Overview of the procedure Short Reads Connector-counter approximates the number of occurrences of reads from $\mathcal{Q}$ in $\mathcal{T}$. The whole procedure is presented in the Algorithm 3. During the indexation, each solid k-mers of $\mathcal{T}$ are gathered in $\mathcal{A}$ and recalls its count in $\mathcal{T}$ (obtained with DSK) through the array FG. Then, when querying, for each read q from set $\mathcal{Q}$, the counts of all its k-mers indexed in the Quasi-Dictionary are recovered and stored in a vector. Collected counts from k-mers from q are used to output an estimation of its abundance in read set $\mathcal{T}$. The abundance is approximated using the mean number of occurrences of k-mers from q. Median, minimal and maximal number of occurrences of k-mers from q are

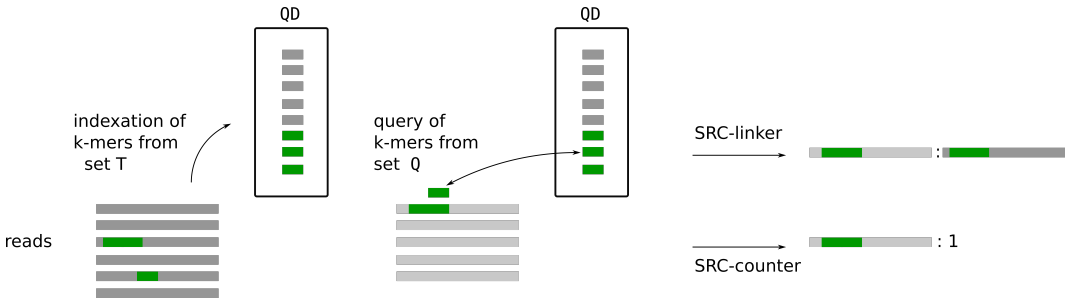

Figure 5: **SRC-l and SRC-c outline.** Indexation with QD, query and output of each tool.

also output.

Algorithm 3: SRC-c: approximating number of occurrences of reads.

Data: Read set $\mathcal{T}$, read set $\mathcal{Q}$, $k \in \mathbb{N}, t \in \mathbb{N}$
Result: For each read from $\mathcal{Q}$, its k-mer similarity with set $\mathcal{T}$

solid k-mer set $\mathcal{R}$ = get_solid_kmers($\mathcal{Q}, k, t$) ;
Quasi-Dictionary $QD = create\_quasidictionary(\mathcal{R})$;
foreach *Solid k-mer* ω *from* $\mathcal{R}$ **do**
    $QD.values[QD.query(\omega)]$ = number of occurrences of ω in $\mathcal{T}$;
foreach *read* q *in* $\mathcal{Q}$ **do**
    create an empty vector *count_q*;
    foreach *k-mer* ω *in* q **do**
        if $QD.query(\omega) \geq 0$ **then**
            insert $QD.values[QD.query(\omega)]$ in *count_q* ;
    Output the q identifier, and (mean, median, min and max values of *count_q*);

Minimal k-mer span It is noteworthy that the mean number of occurrences of k-mers of a read is only a proxy of its abundance. Since k-mers can overlap by at most $k-1$ nucleotides, having for instance two shared k-mers does not mean that $2 \times k$ bases are shared between the reads. The number of shared k-mers is then less informative than the number of bases shared. Thus we introduced an additional parameter to SRC-c, the minimal k-mer span (also s, the "similarity" threshold). The k-mer span is the number of bases from the read query q covered by a k-mer present in the index. If the read contains at least s% of positions where an indexed k-mer spans on its whole length, then it is considered similar to a read in the index and output. Figure 6 illustrates the interest of k-mer span, and Figure 7 shows how it is used during the query of a read.

False positives impact assessment We propose an experiment to assess the impact on result quality when using SRC-c's probabilistic structure instead of a deterministic hash table for estimating read abundances. We used the 100M reads set previously described both for the indexation and the querying, thus providing an estimation of the abundance of each read in its own read set. We made the indexation using $k = 31$ and counted as solid k-mers seen at least twice. In this example only 756,804,245 k-mers are solid among the 5,191,190,377 distinct k-mers present in the read set. This means that 85.4% of queried k-mers are not indexed, this matter of fact enables to measure the impact of the Quasi-Dictionary false

Figure 6: **Bases covered by shared k-mers.** On the left example a read shares two green k-mers that come from the same read in the index, they overlap of $k-1$ bases so the total number of bases shared is $k+1$. In the second example on the right, the two k-mers do not overlap and $2 \times k$ bases are shared. We divide the number of bases covered by a shared k-mer by the length of the read to obtain the span value, and compare it to the threshold s to decide whether reads are similar.

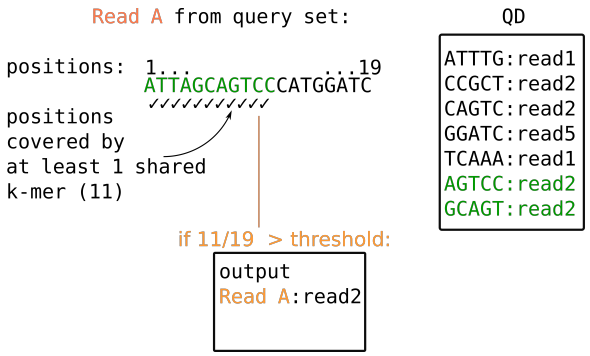

Figure 7: **Query reads with minimal k-mer span.** The k-mers shared between the read A from the query and reads the index are represented in green. An example of shared k-mer GCAGT. The example shows the positions in the read corresponding to nucleotides covered by at least one shared k-mer (positions 1 to 11). The shared k-mer come from a same indexed read (read 2). Read A has a length of 19 bases, thus if $\frac{11}{19}$ is over the similarity threshold s, read A and read 2 are reported in the output.

positives. We applied the SRC-c algorithm, using $f = 12$ or $f = 62$. In the case of $k = 31$, with $f = 12$, the false positive rate is non null while with $f = 62$ SRC-c is in a "perfect mode" with no false positives. These two experiments thus enable to evaluate the impact of false positives when using the Quasi-Dictionary for downstream analyses such as read abundance estimations. False positives lead us to count a k-mer that is not present in the indexed set as present. Thus, because of the QD false positives, counts obtained with $f = 12$ are an over-estimation of the real result obtained with $f = 62$. We computed for each read the observed difference in the counts between results obtained with the two approaches. The max over-approximation is 26.9, and the mean observed over-approximation is 7.27×10^{-3} with a 3.59×10^{-3} standard deviation. Thus, bearing in mind that the average estimated abundance of each read is ≈ 2.22, the average count over-estimation represents $\approx 0.033\%$ of this value.

2.2.3 Short Reads Connector-linker

Method outline Each read r from a given set has a unique identifier denoting its order of appearance in the read file. We start by describing the procedure with a minimal k-mer span of 100%, i.e the whole target read length must be covered by shared k-mers. This time, during indexation, for each k-mer w from the set $\mathcal{A}$ of solid k-mers of $\mathcal{T}$, the FG array stores a list containing the identifiers of reads from $\mathcal{T}$ in which

w occurs. Each k-mer of each input query read q is queried in the Quasi-Dictionary (Algorithm 2). If this k-mer is associated to one or several reads from $\mathcal{T}$, then for each of them, we recall positions on q covered by this shared k-mer using a boolean vector per targeted read in $\mathcal{T}$. Once all k-mers of a read q are treated, the identifier of q is output and for each read r_j from $\mathcal{T}$ its identifier is output together with the k-mer span with q. Again, the k-mer span threshold s can be set to another value. If it is the case, for a given query read q it will be verified for each boolean vector (i.e. each read from $\mathcal{T}$, how many positions are spanned by an indexed k-mer. Results will be output only for r_j from $\mathcal{T}$ for which the number of positions was over $s\%$ over the size of the read. The global algorithm is outlined in Figure 4.

Algorithm 4: SRC_linker: identifying read similarities.

Data: Read set $\mathcal{T}$, read set $\mathcal{Q}$, $k \in \mathbb{N}, t \in \mathbb{N}, f \in \mathbb{N}$
Result: For each read from $\mathcal{Q}$, its similarity with each read from set $\mathcal{B}$

```
solid k-mer set R = get_solid_k-mers(B, k, t) ;
Quasi-Dictionary QD = create_quasidictionary(R) ;
foreach read r in T do
    foreach k-mer ω in r do
        if QD.query(ω) ≥ 0 then
            add identifier of r to QD.values[QD.query(ω)] ;
foreach read q in Q do
    create a hash table targets[a] ;
    foreach position i in q do
        ω = k-mer occurring at position i in q;
        if QD.query(ω) ≥ 0 then
            foreach tg_id in vector QD.values[QD.query(ω)] do
                targets[tg_id][i..i + k − 1] = "True";
    Output for q information about positions covered by shared k-mers with each[b]
    read tg_id from T.
```

[a] *target* keys are read ids, and each *target* value is a boolean vector of size $|q|$ initially filled with "$False$".
[b] In practice only reads whose number of positions covered by a shared k-mers is higher or equal to a user defined threshold are output.

False positives In SRC-l, a false positive of QD means that a k-mer that was not indexed (i.e. not solid in the indexed dataset) is returned as a shared k-mer by the data structure. It can lead to overestimate the number of positions that are covered by a shared k-mer on a queried read. However, this is not always the case. We present different possible cases in Figure 8, showing that large over-estimations are unlikely to happen with a false positive rate.

2.2.4 Performances of SRC-L: comparison to state of the art

We set a benchmark of the SRC-l method with comparisons to state of the art tools that can be used in current pipelines for the read similarity identification. We chose the classical method BLAST [4] (version 2.3.0), as it is able to index big datasets, and consumes a reasonable quantity of memory. We also included two broadly used mappers, Bowtie2 [117] (version 2.2.7), and BWA [128] (version 0.7.10), both based on the BWT principle. By default these two tools only output the best possible alignment found. To enable the comparison with BLAST and our method, we used the "any alignment" mode (`-a` mode in Bowtie2, `-N`

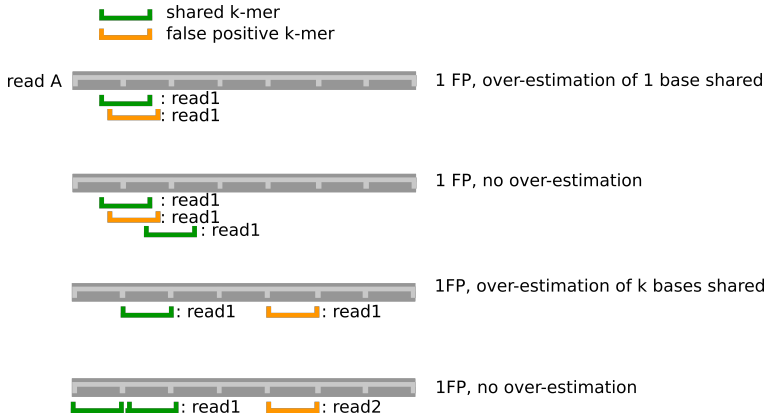

Figure 8: **False positives in SRC-l.** Four scenarios are presented. In the three first scenario, the false positive k-mer comes from the same indexed read than true positives (read 1). In the first one, a false positive k-mer overlaps true positive k-mers and new positions. In the second, the false positive k-mer does not spans a position that is not already covered by true positives k-mers, thus we do not overestimate the span. In the third scenario, a false positive k-mer spans a region totally uncovered by true positive k-mers. This leads to the maximal overestimation per k-mer. In the last scenario, a false positive k-mer from indexed read 2 covers a region of the read A, but the minimal k-mer span is not reached between read A and read 2, thus their similarity is not reported.

for BWA) in order to output all alignments found instead of the best one only. We also compared SRC-l to Starcode (1.0) [260], that clusters DNA sequences by finding all sequences pairs below a certain Levenshtein distance and is particularly designed for large scale comparisons. BLAST, Starcode and SRC-l were used with default parameters. Because of the time or memory limitations we could compare against all methods only up to 1M reads. Results are reported Table 1.2. BLAST suffered from a relatively low throughput and only small datasets were treated within the timeout (10h, wallclock time). BWA and Bowtie shown that the BWT-derived tools are not well suited to index large set of short sequences nor to find all alignments and therefore use considerably more resources than their standard usage. However, BWA performed better than the two other tools in terms of memory, being able to scale up to 10M reads, while Bowtie2 and BLAST could only reach 1M reads comparison. Most likely, Starcode also suffers from the number of pairwise comparisons to process. Importantly, this benchmark is however unfair. All compared tools provides either much more precise distance information between pair of reads or perform additional clustering (Starcode), thus perform more tasks than SRC-l. However, our benchmark highlights the fact that such approaches suffer from intractable number of read comparisons, while the simpler method implemented in SRC-l scales. On this modest size of read set, we see that SRC-l is already ahead both in terms of memory and computation time. The gap between our approach and others increases with the amount of data to process. Dealing with the full dataset reveals the specificity of our approach, being the unique able to scale such dataset.

2.2.5 Limitations

In practice, the current QD implementation works only for an alphabet of nucleotides (note that, on the other hand, the MPHF we use can receive any string or integer as input key). Thus only nucleotidic sequences can be indexed. However, any piece of information can be associated. Obviously, in any case, the larger the size of the information to be stored, the larger the memory footprint of the data structure is. In the case the sequences to be indexed are of size ≤ 15, an open-addressing scheme should be preferred. Thus this

	Data set	100K	1M	10M	100M	Full
	Nb solid k-mers ($\times 10^6$)	0.2	0.6	22	757	1,880
Time (s)	Blast	52	795	-	-	-
	Bowtie2	51	10,644	-	-	-
	BWA	106	3,155	62,912	-	-
	starcode	29	1,103	131,139	-	-
	SRC-linker	**5**	**45**	**587**	**14,748**	**40,828**
Memory (GB)	Blast	18.5	24.5	-	-	-
	Bowtie2	0.77	5.54	-	-	-
	BWA	**0.49**	3.4	5.9	-	-
	starcode	12.06	18.18	73.5	-	-
	SRC-linker	1.07	**1.28**	**3.61**	**44.37**	**110.84**

Table 1.2: CPU time and memory consumption for indexing and querying a dataset versus itself. Tests were performed using $k = 31$ and $t = 2$ (k-mers seen twice or more are solid). We set a timeout of 10h. BLAST crashed for 10M dataset, Bowtie2 reached the timeout we set with more than 200h (CPU) for 10M reads. BWA reached the timeout for 100M reads (more than 200h (CPU) on this dataset). On the 100M dataset, Starcode also reached the timeout. Only SRC-l finished on all datasets.

structure is of limited interest on small k-mers for instance. Finally, a main property that limits the use of SRC to certain applications comes from the MPHF that needs to be computed for a static set of keys. This means that anytime a new element has to be added to the index, the MPHF has to be computed again from scratch for the whole new set. At the moment an online use of SRC is thus impossible.

3 Detection of similarity of sequences in large scale studies

This section presents a validation of our tool SRC-c as a way to compute similarities in large scale sequencing projects, in particular via the *de novo* study of a marine symbiosis. SRC-c was used to better detect biological actors in marine holobionts in an assembly pipeline. We first present the concept of holobiont that we believe can be unfamiliar to the reader. In the meantime we present the interest of holobionts from a biological point of view, as well as demonstrated and expected impacts they have in ecosystems. This allows us to point out difficulties when studying such systems, precisely where SRC-c can be useful. In a second time, we explicit the experiments that were conducted to validate the integration of SRC-c to the holobiont analysis pipeline, then we show an application to a non model holobiont that illustrates both contributions and current limits of the approach. Overall, more details concerning this particular work and holobionts can be found in the preprint manuscript [153].

3.1 Application case: marine holobionts

3.1.1 Holobiont concept and motivation

An ecological paradigm back to trend The observation that not all microorganisms are harmful to their host dates back to many years ago. However, most of them were first investigated for their pathogenic properties. It is through ecology that was proposed the concept of holobiont, that states that the organism is often insufficient to describe an individual (for instance a plant or a fish), and that the organism and all the communities of its symbionts should be seen as a whole. It was first proposed for reef-building corals and their cohorts of bacteria [197] and kindled many interests. These biological models are today a model species to study interaction between micro-organisms and their host. Now such a concept has spread into many aspects of biological research, one of the most impactful those latter years is maybe the gut microbiome. Any example of a plant or of an animal (in fact any macro-organism) that would not have communities of micro-organisms living on its surface or inside would in fact now be the exception rather than the rule.
The holobiont concept is back to trend in ecology, notably because metagenomics and metatranscriptomics allow to set up large scale sequencing experiment from samples that contain material from all actors of the holobiont.

Examples of impacts of holobionts This holobiont concept is then at the center of crucial questions in ecology and evolution. It is becoming more and more clear that many stages such as development, nutrition or metabolism are influenced by the symbionts communities. Popular examples of mutualism are the interaction between fungi and plants that help land plants nutrition [212]. In the ocean, coral holobionts involve unicellular eukaryotes and bacteria, and the loss of such symbionts was shown to provoke coral bleaching [48]. By considering a new paradigm where the organism and its symbionts as an inseparable "meta-organism", we reveal that the fitness (i.e. the selective value according to selection processes) of this whole is the target of natural selection. It implies not only that symbioses (and their perturbations) impact the of species, like in examples such as coral, but also that the microbial "baggage" is (at least partially) acquired and not transmitted the same way than genomic information to the offspring. These communities also harbor a lot of plasticity and can impact on adaptability under stress. Thus holobionts are at the center of many fundamental debates in biology.

Access holobiont's sequences Holobionts remain often hard to study for several reasons. A first point is that a large fraction of the micro-organisms implied cannot be grown in lab. This is why metagenomics and metatranscriptomics are particularly interesting as they allow to sequence sample coming directly from the environment. Secondly structured information must be made from the mixed-up puzzle that is short-fragment sequencing of a metagenome or transcriptome. As already mentioned, these kinds of data suffer from supplementary difficulties for they bioinformatics processing, such as high heterogeneity of coverage that makes many tasks, such as assembly, way harder. However, whole genome information is still extremely hard to recompose. Then, metatranscriptomics from holobiont samples is an interesting alternative that gives access to functional sequences. Indeed, a metatranscriptomics experiment provides the RNA mixture from each actor's expressed genes. However, the application we present is to be distinguished from regular metatranscriptomics, at least because studying metatranscriptomes samples do not *a priori* implies the presence of symbiosis. Holobiont sequencing can be less challenging than large metagenomic experiments, since the number of actors in presence is reduced.

Holobionts' dark matter For many species, no assembled genome is present in databases. We mentioned Bacteria and Archaea, however holobionts can also imply eukaryotes such as fungi or protists (unicellular eukaryotes). Thus many symbioses are poorly understood such as the associations between sponges and bacteria [87], or more recently described associations between two unicellular eukaryotes which belong to plankton. However, recent studies showed that this plankton symbiosis is widely distributed in the ocean and significantly contribute to biomass and carbon export in the open ocean [20, 82]. This is then a real dark matter that is currently understated and just starts to be documented.

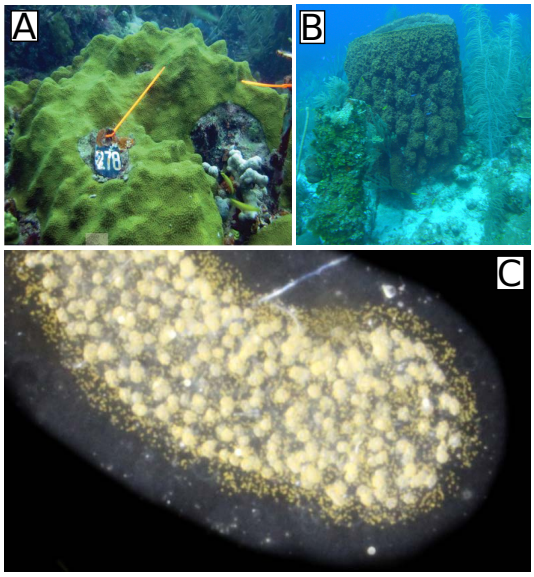

Figure 9: **Studied holobionts models** from [153]. A/ Top left M1 host *Orbicella faveolata* is in symbiosis with Dinophyta symbionts (*Symbiodinium* spp). B/ Top right M2 *Xestospongia muta* hosts a diverse microbial community. C/ Bottom shows model M3, Collozoum sp. as host and Dinophyta symbionts.

3.1.2 Holobiont models in this work

SRC-c was applied on three distinct marine holobionts. The two first models are already known: model 1 (M1) is composed of a coral, Cnidaria host (*Orbicella faveolata*) and Dinophyta symbionts (*Symbiodinium* spp., a unicellular eukaryote belonging to the Alveolata). Species of M1 form a mutualistic association [48, 90]. The coral holobiont also encompass other microorganisms consisting of bacteria, archaea, fungi, viruses [160, 197]. In the second holobiont model (M2), the marine sponge *Xestospongia muta* (Porifera) harbors a dense (around 40% of its volume) and diverse microbial community including marine protists (e.g. fungi), archaea and mainly bacteria [62, 246, 217]. The symbiotic associations between sponges and bacteria are suggested to be commensalism [216]. Both models are associated to publications and possess assemblies and data we could access. They are among best-known models to understand effects of the initiation, maintenance and loss of symbiosis (for instance coral bleaching [90]). The third holobiont dataset (M3) involves two distinct lineages of protists (unicellular eukaryotes): the radiolarian *Collozoum* sp. as host and Dinophyta symbionts belonging to the *Brandtodinium nutricula* species [185]. Only the eukaryote partners of these model are studied here. Recent studies showed that this symbiosis is widely distributed in the ocean and significantly contribute to biomass and carbon export in the ocean [20, 82]. However, this holobiont is yet unpublished. It was sampled in the South Pacific Ocean during the Tara Oceans expedition in 2011 [180]. Pictures for each model are provided in Figure 9.

3.1.3 Goals and place of SRC-c in this work

The goal of this work was to assemble transcripts from holobiont models in order to open the path to a better understanding of their associations. We present two main objectives in this section: 1-validate SRC results on real biological data, 2-increase assembly quality and help identify holobiont actors. The co-occurrence of sequences of different organisms conducts assemblies to provide chimeric transcripts that come from misassemblied contigs [206, 123, 233]. The unavailability of reference for the organisms leads to

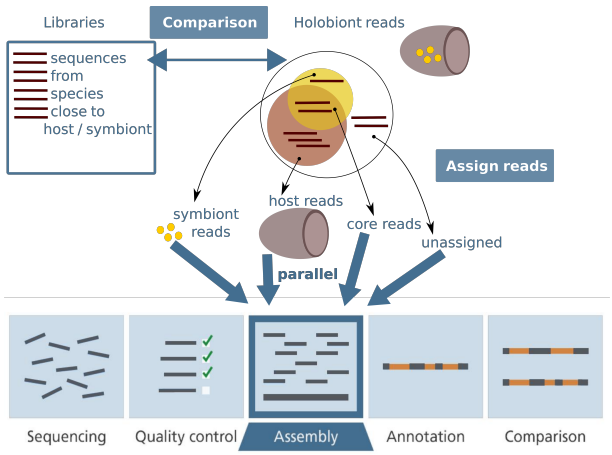

Figure 10: **Outline of the strategy employed to study holobionts.** On the bottom, a standard (meta)transcriptomics pipeline in blue is followed. However, before assembly an assignation of reads from the holobiont is performed with SRC. The reads are dispatched in four groups (symbiont reads, host reads, core reads, unassigned reads) that are assembled in parallel.

integrate those false positives, chimeric, transcripts to the results. Chimeras will then impact downstream results, notably the reconstruction of proteic sequences.

Thus, the idea was to assign reads to one of the actors of the holobiont prior to assembly. Then reads could be assembled in parallel for each actor instead of mixing all reads in a single assembly result. We show how SRC-c is integrated in a classic assembly pipeline in Figure 10. We intended to observe SRC-c impacts on contigs quality and downstream analyses: whether it helps reducing the number of misassemblies, and if its usage is associated with more functional annotation. Models are not all used in the same perspective:

- ◇ M1 and M2 are both associated with previously published results, they serve as proves of concept of the sound results produced using SRC-c.

- ◇ M3 is the main research interest as its actors are poorly known, it is investigated in a quantitative perspective. SRC-c is thus expected to both help identifying actors in a non-model system and to produce better assemblies than a standard pipeline. For this model, we compared SRC-c's results with a classic assembly pipeline.

SRC-c's role will be highlight till the end of this section because it is the main contribution made to this work during this PhD thesis. In the meantime, our collaborator Arnaud Meng also committed to this work and played a main role. We will describe assembly and downstream analyses in order for the reader to fully understand the implications of this work, however we recall that these parts were Arnaud's work. In the same spirit, some details about experiments we did not contributed to (data sequencing protocol, ...) are not provided in this manuscript in order to keep this section synthetic. They are all available in the preprint manuscript [153][3].

[3] `https://www.biorxiv.org/content/early/2017/11/17/221424`

3.2 Experimental design

3.2.1 General pipeline

SRC-c SRC-c is integrated to a metatranscriptomics assembly pipeline. It is used to assign reads from an holobiont transcriptome either to the host or to the symbionts. For each holobiont, libraries were gathered to be indexed in SRC-c (banks). We divided the query step of the holobiont dataset $\mathcal{Q}$ in two parts, one that consists in the comparison of the holobiont sequences to a bank of host sequences, and another that performs the comparison to a bank of symbiont sequences.

Assembly and downstream analyses For all models, after SRC assignation step, each set of reads was assembled using the *de novo* transcriptome assembly program Trinity [78] (v2.4.0) with default parameters. The newly assembled contigs metrics were calculated with the Transrate program [220] (v1.0.3). Protein coding domain prediction and functional annotations were realized using a publicly available pipeline[4]. Taxonomic assignment of the contigs was performed with MEGAN6 [95].

For M3 in particular, chimeras identification for all assembled contigs was realized following the protocol described in [257], comparing contigs to the 7,215 Rhizaria presumed contigs from [9] and 3,494,295 coding domains from de *novo assembled* contigs of 54 dinoflagellates transcriptomes.

3.2.2 SRC assignation before parallel assemblies

Several nature of banks For each of the three holobiont models, we needed banks to identify hosts and symbionts actors. We chose the taxonomically closest organisms available in public datasets. The M1 host bank encompasses 20 assembled transcriptomes from Cnidaria and 2 genome-derived ESTs. The M1 symbiont bank includes 123 RNA-seq reads datasets (including the presumed major symbiont *Symbiodinium spp* [211]) from the MMETSP project [102] [5], which is a project that puts efforts at completing banks of marine protists. The M2 host bank involves 4 RNA-seq reads datasets from distinct Porifera genera. The M2 symbiont bank corresponds to the Tara Oceans metagenomic gene catalog (OM-RGC) assembled from the small plankton corresponding to Bacteria or Archaea [226]. For M3, 7,215 host transcripts were extracted from *de novo* assembled transcripts [9] published so far. The same symbiont bank was used for M3 than for the M1. Thus this selection implied different use cases for SRC-c, with the indexation of reads (M1, M3 symbionts, M2 host), of assembled transcripts (M1 and M3 hosts) and of genes (M2 symbionts) for both eukaryotes and prokaryotes organisms. These datasets sizes vary from 4.5 Mbp to 25 Gbp with sequences length from 100 bp to 84 Kbp.

Parameters choice SRC-c was not always used with default parameters as the sequences natures in bank differed from one another. We adapted the parameters according to the bank type. The different situations encountered and the parameters sets chosen then provide interesting guidelines for users. First, the solidity threshold wads adapted according to the nature of the sequences in the bank dataset. For libraries whose sequences are reads, the default value for the solidity threshold (= 2) was kept. For longer sequences (using libraries of assembled sequences or EST) the threshold was adapted to the non-redundant nature of the datasets and set to 1. We chose a k-mer length of 25 according to the smaller input read length. We set the similarity value s to 50% for models 1 and 3. We first set s to 50% for M2 as well, then decreased it to 40% in order to retrieve more sequences. For this study analyses were performed on a Linux system with 40 cores, with the option -t 0 (maximal number of available threads is used) and 250 GB of memory.

[4] `https://github.com/arnaudmeng/dntap`

[5] `https://www.imicrobe.us/project/view/104`

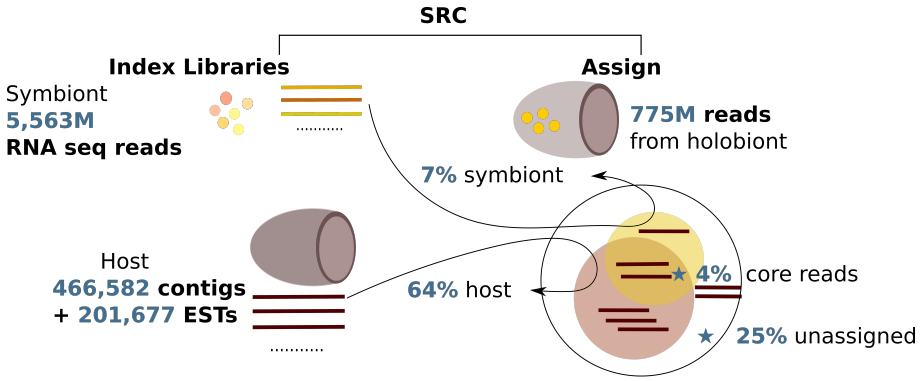

Figure 11: **SRC-c indexation and query strategy for model 1 holobiont.** On the left, we represented banks for each holobiont actor. On the right, reads from the sequenced holobiont. For each assignment category we show the proportion of holobionts reads that could be categorized.

3.3 Validation of SRC using know models

3.3.1 SRC-c allowed comparisons of large scale datasets

The volume of sequences involved in the M1 and M2 reads assignation tasks provide a nice illustration of SRC's capacity to handle large datasets. We present the experiments volumes in Table 1.3. We also illustrate SRC-c strategy applied to M1 in Figure 11

		Indexed sequences number	Queried sequences number	Time (hh:mm:ss)	Memory (Gb)
M1 (Cnidaria/Dinophita)	symbionts library	5,563,498,607	755,025,024	15:40:42	34,2
	host library	668,259		1:06:56	3,9
M2 (Porifera/Bacteria)	symbionts library	40,154,822	33,220,028	21:04:47	58,9
	host library	642,229,224		0:05:57	9,6

Table 1.3: Wallclock runtime and memory usage of SRC-c on M1 and M2.

3.3.2 Comparison to state of the art

We present a summary of qualitative results to highlight conclusions that follow the usage of SRC strategy. For M1, M2, and later M3 as well, the reader can report to this work preprint manuscript for further results details. Both M1 and M2 had assemblies already published, however M2 contigs were not obtained with a similar assembly pipeline, which motivates a qualitative comparison. Our SRC-c approach allowed assembling more contigs than previous studies, however with shorter N50 for both models (N50 of 971 bp for M1 and 719 bp for M2 with our approach, 580 bp shorter than literature for M1 and 219 bp shorter for M2). Conversely those contigs display high remapping rates for M1 (¿80%), and mixed results for m2 (25% ¡ x ¡ 86%)). Differences in the number of contigs as well as contigs metrics could be the results of the use of distinct *de novo* assembly softwares. Contigs difference in size could be explained as well by the assembly, since Trinity contigs are documented to be shorter than those from the other assembly software used (CLC) [228]. More annotated contigs were found for M1 using SRC-c strategy (the comparison could not by made for M2 since this information does not appear in the publication). Again, it is difficult to tell whether this observation can be the consequence of a better suited assembly strategy, and / or the use of a different annotation pipeline, and / or the supplementation of reference annotation databases between 2015 [182] and

2017. Results of assignation using MEGAN6 are very mixed. Depending on the target, MEGAN6 or SRC-c strategy performs better by providing more assignations. However, intersections between MEGAN6 and SRC-c, in particular for M1 host (all assignation found with MEGAN6 were also found with SRC-c, with additional assignations found with SRC-c), encourage to pursue assignation task with our tool since they validate some of the prediction made using our approach.

3.4 Added value of SRC on novel holobiont

3.4.1 Performances

We show SRC's performances on this third Model in Table 1.4.

		Indexed sequences volume	Queried sequences volume	Time (hh:mm:ss)	Memory (Gb)
M3 (Radiolaria/Dinophita)	symbionts library	5,563,498,607	97,957,494	7:05:28	4,10
	host library	7,215		0:05:57	3,9

Table 1.4: Wallclock runtime and memory usage of SRC-c on M3.

3.4.2 Results

Comparison Model 3 does not come with an associated publication like models 1 and 2. In order to have a baseline for comparison, we generated assembled transcripts with a strategy hereafter called no-SRC. This no-SRC strategy consisted in direct production of a *de novo* assembled transcriptome, obtained from holobiont reads considered all together. All metrics were then computed for this set of contigs and for SRC-c strategy contigs and compared. The assembly metrics appear very similar between SRC and no-SRC.

Assemblies and CDS A comparable number of reads were used for the assembly step and a comparable number of assembled contigs were obtained. The N50 value for the no-SRC strategy is slightly longer while the remapping rates are 5% better with the SRC strategy. The SRC strategy showed around 46% less chimeras (418 contigs) than the no-SRC strategy (777 contigs) with most chimeras contained in the unassigned set. Though it seems not significant (0.465% chimeras for no-SRC versus 0.247% for SRC), in terms of number of sequences to be studied afterwards by biologists this is a real gain (i.e. biologists will not waste time to perform PCR or knock out experiments on these false sequences).

CDS, assignation and annotations Again, as well as for M2, M3 libraries may not contain enough quality sequences from the host. Then only 3% of the holobiont reads are assigned to the host. Overall, less unassigned reads were observed when the "correct" actors are involved thus the bank completeness is increased (M1: 24.4% unassigned) compared to the poorly studied models.
We noticed slightly less annotated CDs with the SRC strategy, however regarding functional annotations, the SRC and the no-SRC strategies provided very similar results. The top 5 functional annotations were strictly identical for both approaches. However, the SRC strategy categorized the annotations among 4 subsets, which can be explored independently, allowing group specific interpretations and biological hypothesis building for each partner from the holobiont. For instance, symbiont CDS linked to the photosystem I and II were detected, confirming that SRC-c succeeded to assign reads to photosynthetic actors, as expected here for the symbiotic partner.

3.5 Discussion on holobionts results

Despite the lack of reference, SRC-c enabled to classify a fraction of the reads and helped providing quality transcripts for M3. In addition, our strategy allowed defining a new category of contigs (the shared contigs) in comparison to MEGAN6, we thus access to the information that these contigs may come from the core

expressed transcripts of the holobiont. Thus they give clues to the identification of genes expressed in all the actors. However, results on the three models show that more strategies have to be investigated to make the best of SRC on these instances. Future works on SRC-c parameters settings could include more extensive exploration of the impact of the similarity threshold parameter on the sensitivity of our approach. In this regard, if the reads similarity rate to the libraries could be relaxed, it may decrease the number of unassigned reads in particular for poorly studied models. Assembly stages can also be realized differently, for instance the addition of unassigned reads to other categories before the assembly could lead to longer contigs. However, we will have to verify if the chimeras level remain low. Other perspectives on this work will be presented in the conclusion chapter.

A step forward is to extract sequences related to functions of symbiosis. Sequence similarity networks (SSN) are used to this extent. They are graphs that integrate protein sequences as nodes and record similarity (for instance reached by protein sequence alignment) in the edges. From assembled transcripts, protein sequences are predicted and inserted in the SSN. The SSN should help distinguishing connected components composed exclusively of non symbiotic sequences and connected components linked to symbiotic radiolaria and dinoflagellates. Then by associating the functional annotations linked to these components, hypothesis can be drawn concerning the symbiosis mechanisms. This should also help identify new unannotated sequences linked to symbiosis.

4 Using SRC-linker on long, spurious sequences

This section presents our attempts to tailor SRC to long reads, given the difficulties induced by their spurious nature. We focus on finding overlaps between reads. This time, scaling challenges are intertwined with sensitivity challenges as the sequences are very noisy. A vast majority of k-mers created by sequencing errors in the sequences do not exist in the original DNA. Moreover, as we look for small k-mers, depending on the coverage and on the size of k-mers (around size 15), it can be likely that many spurious k-mers show occurrences above 2 in the dataset [30]. We start by benchmarking a regular SRC-l approach to long reads mappers on simulated data, only downsizing k value to fit error rates. We assess their ability to retrieve read pairs on a simple example. The proof of concept as well as comparisons were published in [145]. This helps us to identify SRC's limitations on long read, then to propose enhancements. We show a new feature in SRC-l's algorithm to better adapt to the chains of exons contained in RNA reads, and we propose an exploratory solution for a new seed scheme for SRC.

4.1 SRC-linker: proof of concept for long reads

4.1.1 Comparison to state of the art

Overview In this comparison, recall, precision and performances are at stake. The recall represents the number of relevant elements retrieved among all relevant elements. It corresponds to the ratio between true positive and all positives. The precision represents the number of relevant element among the retrieved elements. It corresponds to the ratio between true positive over true positive and false positive. Having a good recall while remaining precise is a real hard task. In order to estimate the impact of indexing all k-mers using the Quasi-dictionary in the context of long spurious reads comparisons, we simulated long reads coming from different regions of a genome, and we compared the three described approaches to ours. Our goal here is to demonstrate the potential offered by the Quasi-dictionary data structure. We chose two tools specially dedicated to detect overlap between pairs of long reads (MHAP and Minimap), and one tool specialized in long reads that was presented under the mapping-to-reference aspect. Doing so, we can assess how the two categories of tools behave on this problem. Relying on the Quasi-Dictionary, we argue we can afford to index all (solid) k-mers at a reasonable cost and then benefit from a more complete information about the content of the reads than tools based on local sensitivity hashing. However, as presented above, MHAP, GraphMap and Minimap use a second pass to increase their precision while SRC-l sticks to the single recruitment phase.

Read simulation It must be first noticed that no simulation tool for long RNA reads currently exists. We designed a very simple experiment to be able to distinguish true from false similarities retrieved by the different tools. We chose spots on the *C. elegans* genome (genome version PRJNA13758 from WormBase) separated from hundreds of nucleotides and simulated reads on theses spots of 2000 bases long. This creates the following situation. No reads overlap two different spots, and reads share whole-length overlaps. This simple situation enables us to have access to the ground truth about read similarities without using a third-party mapping tool. We chose *C. elegans* for its known relative simplicity (it contains few repeats within its DNA sequence), which lowers the chance to mistake a region for another, and we preferred real biological sequences to random sequences to be closer to the biological applications we aim at. Two reads sets of 100K and 1M reads were simulated, using an error profile that mimics PacBio reads [170] and 12% and 15% error rates, which represents the expected scenario in that sequencing technology. The ground truth is composed of a set of read id couples. Each couple designs two reads simulated from the same locus. Each tested tool also produces a set of read id couples. Recall and precision measures are given by the following formulas:

$$\text{recall} = \frac{\text{Number of correctly predicted couples}}{\text{Number of ground truth couples}}$$
$$\text{precision} = \frac{\text{Number of correctly predicted couples}}{\text{Number of predicted couples}}$$

The F-measure is provided by the following formula:

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Softwares and command lines We used MHAP version 2.1.1 with default parameters, Minimap version 0.2-r123 with option `-Sw 2` and Graphmap version 0.4.1 with default parameters. We tried increasing the recall of Minimap by allowing to index more minimizers using the option `-Sw` so that its results would be more comparable to GraphMap's and ours. For MHAP, a similar feature could be tuned too but on our simulations it increased non significantly the recall while decreasing the precision. We chose the best set of parameters ($k = 15, s = 8$ for 12% and 15% error) for SRC-l indicated by our simulations. All tools allowed multi-threading and were launched using 20 threads. We extended the timeout of 5 hours from the short reads experiment regarding the longer sequences to process, reaching 15 hours wallclock.

Results We present results on two different error rate that would apply on real data in Tables 1.5 and 1.6

	Minimap	MHAP	GraphMap	SRC-linker
Recall(%)	99.31	86.54	97.77	97.96
Precision(%)	99.83	96.74	99.53	99.58
F-measure	99.57	91.35	99.15	98.76
Memory (GB)	6.37	25.94	11.87	2.55
Time (m:ss)	0:24	3:19	9:00	5:49

Table 1.5: Precision and recall followed by time and memory performances for 100K simulated long reads on 1K distinct regions on the *C. elegans* genome, with 12% error rate.

From those results we can see that the key advantage expected from minimizers methods, time and memory low footprint, is met with Minimap and only half-met with MHAP (which shows a quite high memory consumption). If SRC-l running time if quite high and comparable to GraphMap's, it however has the lowest memory footprint, while indexing more elements than Minimap and MHAP. On 100K long reads

	Minimap	MHAP	GraphMap	SRC-linker
Recall(%)	63.25	14.21	92.08	91.95
Precision(%)	99.89	86.94	99.72	97.89
F-measure	77.46	24.43	95.75	94.83
Memory (GB)	6.38	26.32	12.03	2.65
Time (m:ss)	0:24	3:20	9:09	7:35

Table 1.6: Precision and recall followed by time and memory performances for 100K simulated long reads on 1K distinct regions on the *C. elegans* genome with a 15% error rate.

with 12% error (Table 1.5), with the exception of MHAP recall, all tools present near perfect precision and recall. As expected, all three state of the art tool provide a high precision rate on this first experiment. As we increased the error to a more difficult scenario (Table 1.6), we can see that MHAP and Minimap recall scores decrease while GraphMap maintains very high recall and precision. Our recall outperforms those of other tools, however, as expected, we reach a lower precision than GraphMap. This shows that SRC-l already provides acceptable precision without any post-treatment on the contrary to GraphMap and MHAP that use downstream filters. This also shows that we successfully mimic GraphMap in its ability to adapt to varying error rates. Minimap is presented as an experimental tool and has not the ambition to reach the recalls of the other tools that integrate more developments. Its force relies in its lightweight and fast execution. A third dataset of higher size (1M reads for 10K distinct regions) is generated to show the scalability of each method. GraphMap ran for more than 15 hours on the bigger dataset thus reached the timeout we set. MHAP crashed on this bigger dataset. All in all only Minimap and SRC-l managed to scale on the bigger data volume. They obtained following results: 98.56% recall and 97.95% precision for Minimap, and 98.28% recall and 92.63% precision for SRC-l. In these experiments we chose the parameters of SRC-l to optimize its F-measure, giving results not always in favor of precision. The SRC-l precision could be improved using downstream filters or more stringent parameters. We showed that the tool we provide, while being simple works well as a recruitment tool for highly noisy sequences thanks to its ability to preserve as much information as possible about the sequences. It presents both advantages to be robust to changes over errors or read length, and to be scalable. Two pitfalls for state of the art tools are two choices in their design: first, since thought for rapid assembly (such as Minimap) of long reads, they do not aim at retrieving all overlaps between reads. They must be just enough accurate to elongate contigs. Secondly, the reads we present are particularly small (though realistic for RNA) compared to usual use case for these tools. Their seed recruitment step might expect longer sequences where to find chains of seeds. Then in the case of shorter reads, less valid overlaps might be found. Importantly, we recall that our message here is not to outperform state-of-the-art long read mappers that were designed and optimized specifically for this task. We simply want to show the application potential that our data structure offers. Thus, we can summarize the main disadvantage of SRC regarding those reads:

- ◇ SRC is not very efficient with k-mers of small size, such as explained above.
- ◇ The search of exact matches might not be enough to capture reads similarity since a lot of sequence modifications occur with the errors.
- ◇ SRC's precision can be improved.
- ◇ SRC looks for similarity through the read's whole length while RNA sequences might only share exon subsequences.

In the following, we propose to address the three first points using a novel seeding technique, and the last point by generalizing the minimal k-mer span to defined regions of the reads.

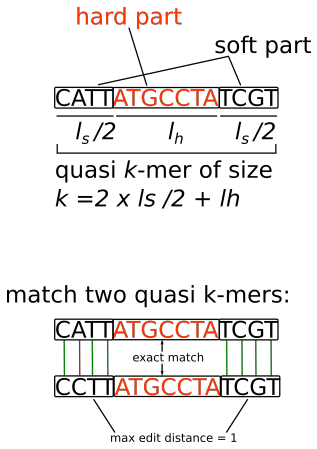

Figure 12: **Quasi k-mers** features and match.

4.2 New features for adapting SRC to long reads

4.2.1 Minimal k-mer span on window

We place ourselves in the SRC-l framework. We generalized the minimal k-mer span algorithm in order to adapt to exon-exon matching. Thus, two reads overlapping by only certain exons is a scenario that can occur. In this case, we should detect local sensitivity within a window of the size of an exon, instead of looking for k-mer span all along the read. Hence a supplementary parameter w is introduced. It determines the length of a window in which computing the minimal k-mer span. Windows are computed by sliding on a given query read length. This time, if at least one window of size w contains at least $s\%$ of positions where an indexed k-mer of an indexed read spans, then these read are considered. The better window is automatically selected by remembering the best matching positions score. This feature is now implemented in SRC.

4.2.2 Seed scheme using quasi k-mers

The idea of quasi k-mers is to allow a (small) threshold of edit distance between the queried and the indexed k-mers.

The intuition is that an exact match is required for a subpart of a k-mer (the *hard* part) and a distance of one is allowed in the rest (the *soft*-part). An example of quasi k-mer and of quasi k-mers match is shown in Figure 12. Quasi k-mers are indexed from the target data-set by associating any seen *hard* part to its soft parts (step 1 of Figure 13). Query reads quasi k-mers are then compared to quasi k-mers of the index, an distance of 1 being authorized between the *soft*-part of both quasi k-mers (step 2 of Figure 13).

Quasi k-mers The idead of quasi k-mers is to allow a (small) threshold of edit distance between the queried and the indexed k-mers. The quasi k-mer is divided into two parts, the *hard* part for which we require an exact match, and the *soft* part for which we authorize an edit distance of 1. This distance can be computed in constant time, which is an advantage for performances. Quasi k-mers differ from spaced seeds since spaced seeds permit wildcard characters (i.e. substitutions) while quasi k-mers allow both indels and substitutions.

Indexation Indexations requires two index. The first index is done usin the QD, where k-mers of the indexed set stored such as in SRC-l. Then all indexed k-mers are screened. For a given k-mer indexed, we compute *hard* part and *soft* part. *Soft* parts length l_s is even, and in practice within a k-mer they are the

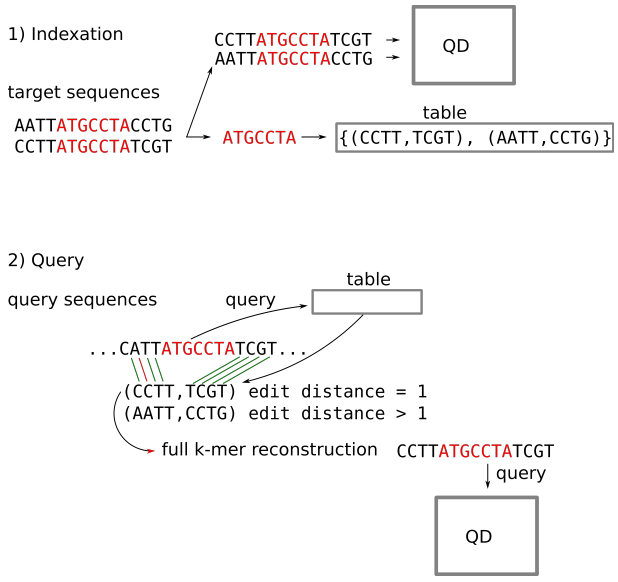

Figure 13: **SRC indexation and query using quasi k-mers.** Hard parts are shown in read, soft parts in black. 1) First, full length quasi k-mers are indexed in the QD. An additional table helps to associate hard parts to soft parts. 2) Hard parts from query reads are queried in the table and quasi k-mers are compared to find matches. When a match is found, the full length quasi k-mer is reconstructed (i.e. the indexed version is taken, not the queried version that can be different from what is indexed) and queried in the QD.

prefix and suffix of size $\frac{l_s}{2}$. We chose to divide *soft* part in two in order to be able to index a quasi k-mer and its reverse complement. *Hard* part is then the middle substring of length $l_h = k - l_s$. A second index $\mathcal{I}$ associates pairs of $(hard, soft)$. Both half soft parts can be concatenated. One *hard* part can be associated to several *soft* parts. Thus any *hard* part is associated to a list of couples (s_1, s_2) of *soft parts*. We consider *hard* parts of size 15 or less, since these are the typical size of conserved-enough words across several long reads sequences [32]. Whole *soft* parts are expected to be around 10 nucleotides long, however this will have to be further tested.

A particular technique called open addressing can be used to handle collisions in hash tables. The solution is to build an array that can contain all possible keys of a set. In this scheme, each possible key has its own bucket. This technique allows to efficiently store sets when the key space is small. Thus if the set of possible keys is too large, the available memory is not enough to allocate the array. In the case of k-mers that are frequently indexed for mapping or storing De Bruijn graphs for instance, associating 32 bits integers to $k = 15$ constitutes the limit since there are 1,073,741,824 possible words of length 15 on the nucleic alphabet, which can fit on most laptop computers. Thus the first indexation of hard parts of small size can be realized using open adressing, however the use of the QD allows to be more adaptive on the length, in a perspective of error rates decreasing in the future years.

Querying In the querying phase, l_h-mers from each read from the query set are screened. Reads are went over $read.size - l_s$ nucleotides, starting at position $\frac{l_s}{2}$. l_h-mers are looked up in the $\mathcal{I}$ index. If a match exists, the $\frac{l_s}{2} + 2$ nucleotides on the left and of the right of the l_h-mer in the query reads are compared to the couples of *soft* parts in the QD. If for a couple the sum of the two edit distance is strictly below 2, the indexed l_h-mer and its the two matching *soft* parts are concatenated to form a k-mer that is queried in the QD. This query of an indexed k-mers allows the the k current positions in the query to be considered

spanned by a shared quasi k-mer using SRC's regular algorithm. The steps involving computing minimal span and other SRC steps remain unchanged. It is noteworthy that the k-mer reconstitution ensures that no false positive is queried in the QD, the whole structure thus does not provide any false positive in this application.

5 Discussion

Quasi k-mer strategy is not yet implemented in SRC and lacks further testing to support the initial ideas. Still, this inexact match scheme is expected to have twofold impact. Firstly, increase the recall by relaxing the hard condition of exact matches. We highlight that our strategy would be particularly well tailored for long reads since it allows indels in addition to substitutions. Secondly, by looking at overall words of size $k \geq 20$, we could increase the precision of our approach. Thus, longer k-mers are less prone to be found repeated across genomes.

In Chapter I, we proposed a new indexation scheme based on a Minimal Perfect Hash Function (MPHF) together with a fingerprint value associated to each indexed element. We have shown experiments on sets containing more than eight billion elements indexed in less than an hour and using less than 25GB RAM.We proposed two applications: SRC-counter (SRC-c) and SRC-linker (SRC-l). The first estimates the abundance of a sequence in a read set. The second detects similarities between pair of reads inter or intra-read sets.

The potential of SRC to scale voluminous datasets is well established. With the work on holobionts, we initiated its validation on biological data and highlighted advantages and current limitations of the method. As for our main interest, long reads pairwise comparison, SRC might not be well-suited in its initial form. Preliminary inputs for SRC's adaptation to RNA long reads shown that SRC could be tuned to gain in speed and precision. We highlight that a tool dedicated to retrieve pairwise similarities between RNA long reads still misses in the literature, however new advances are proposed in Minimap2 paper [127]. We started its evaluation on simulated data and we proposed two new features to better adapt it to the characteristics of long reads. Since these improvements still have to be tested, in the following we will rather rely on the well-established tool Minimap, that performed well on our benchmark. Minimap is though not really adapted for transcriptomics purposes. Future works on SRC should enable to provide a tool that will be tailored for both RNA and long reads. They include the implementation of quasi k-mers, but could it could also be interesting to propose speed-ups for SRC. For instance, the query of a read could be stopped at the first encountered window that completes the minimum k-mer span requirement.

Chapter 2

Cluster sequences in transcriptomics datasets

As presented in the previous section, similarity detection between sequences allows to relate sequences to one another. This deserves many applications, we shown for instance it helps to infer read's origin in a holobiont by assigning groups of reads to species in function of their resemblance to known species sequences. In this application, using SRC, we realized a partition of the reads, meaning that we delineated subsets of reads that share common properties (i.e. in this case, belonging to a certain actor of the holobiont). This task used distance information between queried reads on one hand and a list of reference sequences on which we relied on the other hand. However, certain applications do not imply the presence of reference sequences. Groups can be defined by only comparing reads within a dataset. Different methods allow to divide datasets in groups (clusters) in which elements share common features. Usually, clustering is made on the basis of similarity measures between elements of the set. In the context of sequencing data, clustering is useful because it allows to extract structured information from large reads sets. The sizes of datasets prevent to define groups by hand, as well the quantity of data, and in certain cases noise makes this task non-trivial.

In this chapter our application case is the clustering of reads from a transcriptome sequencing into as many groups as there were expressed genes sequenced. By defining such clusters, we aim at putting together reads that come from isoforms coming from the same gene, even if these isoforms can differ for instance because of alternative splicing. Moreover, the work presented in this chapter allows to cluster reads *de novo*, without mapping on a reference technique and using only the information contained in the initial dataset. With these two points, *de novo* clustering and long reads, in mind, we aim to meet a methodological lack. During this thesis work, several pipelines for RNA long reads were published [28, 1]. However, at the exception of one pipeline [77], all were designed to work with a reference. This means that a large majority of species that do not benefit of reference could not be addressed, limiting the utilization of long reads to applications mostly dealing with model species. *De novo* clustering of reads methods however exist, this problem has been broadly faced in the past with short reads and assembled sequences from short reads. However, it had not been handled in the case of long reads when this thesis work started. As we will show in the following, long reads introduce unprecedented noise problems that need specific developments.

The developments presented in this chapter rely on methods of similarity detection such as those presented in the previous chapter. Being able to define clusters of reads per gene give access a variety of useful information within a transcriptome sequencing experiment:

- ◇ For each gene, a proxy to its expression level. As stated in the introduction, even if a proportion of reads is not full-length, they still correspond to a single transcript molecule expressed by a gene.
- ◇ For each gene, a list of reads that correspond to truncated or full-length isoforms.
- ◇ For the most erroneous reads, a chance to be linked to other reads with lower error rate (less information loss).

Furthermore they allow to process groups of reads at the gene level of detail in downstream analysis. For instance, clusters can be used to:

- ◇ Be corrected independently, offering both only sequences supposed to be close to one another and smaller dataset scales, correction being a bottleneck in big datasets.
- ◇ Be analyzed independently (for instance isoform identification).
- ◇ Build consensus that provide a representative sequence per cluster. Such sequences are meant to be cleared from most of the errors and can be compared to data banks to try to identify the gene reads origin from.

Early attempts to solve such clustering problem can be traced back to before the age of NGS: in the NCBI Unigen database [209] Expressed Sequence Tags (ESTs) are partitioned into clusters that are very likely to represent distinct genes. In fact, clustering has been the basis for gene indexing in major gene catalogs like Unigene, HGI, STACK or the TIGR Gene Indices [24, 186]. Here we update this question to long reads. Thus the study presented in this chapter are meant to allow long read processing with additional information concerning the common genomic origin of reads, and higher resolution by attributing each read a group of related sequences.

In the following, we will discuss existing algorithms and methods that exist to tackle the clustering problem we defined. We introduce definitions and concepts we think are useful to the reader to understand the methodological choices we made. We will show current limitations of published methods with regards to long reads, which led us to propose a novel algorithm and its implementation. Both constitute a main contribution of this thesis work and will be presented in details. Presented results on real instances are intended to help assess the behavior of our clustering method, thus we will not propose biological interpretation of these results, or present novel biological results. This work occurred during the sequencing of novel full transcriptomes datasets from mouse tissues within the ASTER ANR group project, that were used for our study and that are briefly presented.

1 The issue of biological sequences clustering

In this section we introduce methodological concepts that will be useful to understand our contribution. We separate general algorithmic clues for solving our clustering problem and actual implementations applied to sequencing data clustering. We then evaluate how presented algorithms and methods work on long reads.

1.1 Clustering and community detection

1.1.1 Clustering versus community detection

Usually, a matrix of similarity is used to represent connections between elements and to draw groups. This is equivalent to a representation using a complete graph, where nodes are elements and weighted edges report the degree of similarity between each pair of elements. Statistical approaches are then used to retrieve groups. When a threshold is applied on detected similarity, edges carrying similarity below the threshold are removed which leads to an incomplete graph. On this kind of instances, topological approaches can be applied to select relevant groups or classes. Classically, optimization problems look for similarity measures to be minimized (respectively maximized) between (respectively within) clusters. Roughly speaking, these class resolution strategies can be classified into two approaches based on graph study, according to applications and the scientific community of affiliation: a *graph clustering strategy* based on the search for minimal cuts in graphs and a *community finding strategy* based on the search for dense subgraphs (see Figure 1). The first

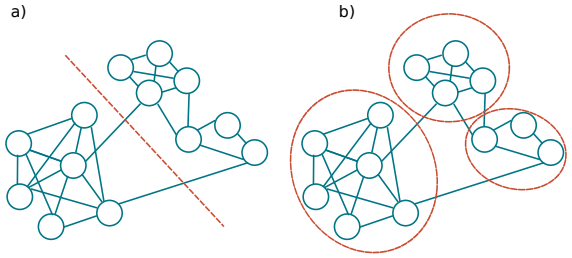

Figure 1: **Clustering paradigms.** a) A graph clustering strategy finds a partition of a graph in a predefined number of clusters, trying to minimizing the number of edges running between clusters. b) A community finding strategy looks for dense subgraphs, such as cliques or quasi cliques within the initial graph.

approach generally searches for a partition into a fixed number of clusters by deleting a minimum number of edges (called the cut size) that are supposed to be incorrect in the graph. The second approach frequently uses a criterion that measures the edge densities in subgraphs and divides the graph in groups of nodes for which the within density is higher than the between density, without *a priori* regarding the number of clusters. Several definitions for edge density of subgraphs have been tried. In an undirected graph $G = (V, E)$

with $n = |V|$ nodes and $v = |E|$ edges, the edge density $d(S)$ of a subgraph S with n_S nodes, v_S internal edges ($v_S = |(n, m)\ m \in S \text{and} n \in S|$) is defined as such: $d(S) = \frac{v_S}{n_S(n_S-1)/2}$ [255]. A simple subgraph that is expected to have a maximal edge density according to any definition is a maximal clique (a clique that cannot be extended by including any more node). γ-cliques can be more adapted to real life data. They are defined as subgraphs with an edge density of at least γ, with γ between 0 and 1. However, the maximal cliques cannot be listed in polynomial time for many graphs, and finding the maximum γ clique for a fixed γ density is also NP-complete [177]. On top of that, incompleteness in real data might lead to loss of maximal cliques that should appear if the information were complete. Thus, simpler measures such as the number of triangles are used instead to measure the subgraphs density. Given that it is difficult to decide on the right number of clusters and to form them solely on the basis of minimizing potentially erroneous links, the main findings and recent developments are based on the community finding strategy. Thus graph clustering techniques work well for specific types of problems (particularly graph bisection, i.e. partition in two sets, or problems with well defined edges similarity measures), but perform poorly in more general cases [165]. Our application falls in the general case since we do not know *a priori* how many clusters are expected, nor we have a precise definition of similarity to be put on the edges (the similarity of two reads is rarely a binary information). We thus choose to model our clustering problem as *community finding* problem, a gene's set of reads being a community in our general case. We therefore introduce more in detail the *community finding* problem and the main associated algorithms.

1.1.2 Quick introduction to community detection

Communities are groups of vertices usually obtained via graph partitioning, however there is no rigorous shared definition. A graph partition is a division of the graph such that any node belongs to a single cluster. Communities are clusters, usually nodes belong to a community because they have something (property related to their nature, role) in common. A common precondition is that the graph must be sparse, i.e. the number of edges in the graph is of the order of the number of nodes, not over. A second one is the "connectedness" of a community. However, objects such as cliques can hardly model properly communities and should have to be relaxed. Apart from this "connectedness", it is frequent that for a given problem, little is assumed about the graph structure of communities. Along with this, the number and sizes of expected communities are often not known *a priori*. Due to the loose setting of the problem definition, a plethora of methods have been proposed. Moreover such problem has appeared among many disciplines, taking many forms and being slightly differently formalized according to the specific problems it was linked with.

Graph partitioning was first formalized in the 1970's. However, algorithms for graph partitioning were not suitable for communities detection as they expected as an input the sizes and/or number of clusters to retrieve. One of the fundamental papers to face this problem was proposed by Girvan and Newman [166] (the proposed algorithm being dubbed after their initials as the GN algorithm). They introduced the idea that a certain measure could assess if subgraphs are eligible to fall in the community definition or not. By maximizing this measure within the graph they partition it into communities. More precisely they proposed to identify edges that connect putative communities in a graph and to remove them via a procedure, in order to isolate each community. This procedure was relying on a measure called *edge-betweenness*. The complexity of this algorithm is cubic ($O(n^3)$ with n the number of edges), on sparse, real instance graphs, which made it computationally intractable on large graphs. Then other measures were proposed, the most popular for this problem being the *modularity* that is described thereafter. We limit the presentation to the most broadly used algorithms, however a very complete review on this topic was proposed in [65].

1.1.3 State of the art algorithms

The first methods for community finding relied on edge-betweeness but were then replaced by more advanced developments. We present algorithms based on *modularity* [166] and clustering coefficient [245] that are two core notions for community finding. The clustering coefficient is used our contribution. We also present briefly some alternative approaches. The most popular algorithms from this section will then be tested for

our specific problem.

Modularity *Modularity* was introduced by Newman as an improvement of the GN algorithm [166]. A first goal was to reduce the limiting complexity of the GN algorithm. Girvan et al. went from a cubic complexity to a second algorithm based on a greedy optimization in $O((m+n)n)$, or $O(n^2)$ on a sparse graph (n edges m nodes) [72]. Then was proposed another optimization that reached $O(nlog2(n))$ on most real instances [39]. This opened the way to communities detection on real life large instances, such as those that can be encountered in bioinformatics.
Modularity Q is the fraction of edges that fall within communities minus the expected value of the same quantity if edges fall at random without regard for the community structure. A high value of Q represents a good community division. The null model (random graph) is the basic concept behind the definition of *modularity*, as a community is expected to have a number of edges that exceeds the number of edges the same subgraph would have in the null model. The idea is to optimize Q over possible divisions of the graph. In practice the optimization is approximated with a greedy algorithm that repeatedly joins communities together in pairs, choosing at each step the join that results in the greatest increase (or smallest decrease) in Q. This algorithm and its further refinements and optimizations helped finding community structure even in very large networks and are maybe the most popular solutions to perform this task. However, they tend to form quickly large communities at the expenses of small ones. A more recent work also based on *modularity* is Louvain algorithm [21]. It is looking for a hierarchy of clusters, by practicing a multi-level optimization that merges the clusters initially reduced to one element as long as the *modularity* increases. This algorithm is fast because it uses a greedy strategy and is quite popular for extracting communities from large networks. However, like the other algorithms based on *modularity*, it suffers from two drawbacks: it has difficulty dealing with small clusters and is unstable in that, depending on the order of application of merges, it can produce very different solutions that are difficult to compare [75]. The clustering coefficient can be related to the average distance between pairs of neighbors of the vertex. This distance can be between one (if neighbors are connected) or two (if they are not). As the distance between two connected nodes is on average greater than one, authors related this to a curvature measure. They proposed a metric representing the distance between nodes and a curvature and defined communities as subgraphs with a large curvature.

Other approaches The clustering coefficient [245] is an alternative to *modularity* for the measure of subgraphs densities. It measures the concentration of triangles in a given subgraph [164]. In [55], the clustering coefficient is used as a local metric on nodes. The clustering coefficient can be related to the average distance between pairs of neighbors of the vertex. Authors proposed a metric representing the distance between nodes and a curvature. They defined communities as subgraphs with a large curvature. An algorithm works with the measure of *information centrality*, based on the concept of *efficiency* [67]. *Efficiency* measures how well information can be disseminated in a graph using the shortest paths between its nodes. The efficiency of a graph is defined as the average of the inverse distances between all pairs of vertices. If the nodes are close then the efficiency is high. It is expected from a community that it maintains a good efficiency within its nodes. However, the complexity of this approach is high ($O(n^4)$, n being the set of nodes in a sparse graph).

The concept of graph-partitioning itself was questioned as, according to definitions, communities can in certain case intersect. In this context, algorithms realizing a cover were proposed. Some applications can justify to change the definition of communities as objects that have potential overlaps. For instance, the reads coming from the expression of two copies of paralog genes can be seen as two overlapping communities. The most popular technique is the Clique Percolation Method (CPM) by Palla et al. [172]. This method defines communities as maximal unions of adjacent cliques of a certain size. It assumes the graph has a large number of cliques and this can be a limitation. On real instances many nodes can be left out, in particular isolated nodes in the graph. Finally, there exist other proposals such as an algorithm based on the generalization of the clustering coefficient [187], or label propagation algorithms [188].

1.2 Method for biological sequences clustering

In this section, we present biological sequences clustering. We start by giving a list of the main generic methods designed for broad purposes (not only RNA sequences), which allows us to illustrate applications of the previously stated community finding algorithms. Then we present clustering specific developments dedicated to RNA sequences.

1.2.1 Background of community detection in the biological sequences context

We mentioned that similarity information was the core information used to build graphs on which clustering methods perform cluster resolution. As they were based on short but reliable sequences, many works applied to sequence clustering put most of their efforts in finding the most efficient way to compute similarity. Indeed, the scale of short reads experiments motivated dedicated developments. These methods essentially tried to avoid all versus all pairwise comparison of sequences, that became a major issue with the advent of NGS and metatranscriptomics. Graphs representing reads similarities could then be draw, with reads as nodes and edges linking pairs of reads with similarity over a given threshold. On the other hand, certain methods remained quite basic in their clustering scheme (e.g. CD-HIT [131], SEED [10], Uclust [57], DNA-CLUST [71]). For instance, connected components could be extracted from the graph of similarity as final clusters. These approaches and the underlying similarity measures were thought for highly similar sequences. Other works that include more specific developments such as OTU clustering (operational taxonomic units, which are groups sequences from organisms from close biological groups according to a similarity threshold that is defined in specific conserved regions of the DNA or RNA) [261, 140]. These methods rely on the general assumption that there exist representative sequences of a cluster, to which other sequences can be linked.

Previously described community finding algorithms also benefited from implementations in this context. In [167] was proposed a method for the characterization of genome-wide repetitive sequences in 454 genomics datasets. The authors described a precise clustering of repeats based on communities detection [166]. This method works on NGS reads and start from pairs of short reads similarities that are way easier to compute and with order of magnitude less errors. The CPM algorithm was used to cluster protein families [98, 2], while Louvain algorithm was used for the clustering of specific RNA subunits [64] or homologous proteins [156].

1.2.2 RNA clustering applications

Several types of clusters can be looked of with RNA sequences. Reads can be grouped by gene of origin, or constitute smaller clusters of similar alternative transcripts. All genes are not expressed at the same level in the cell, which leads to an heterogeneous abundance of reads for the different transcripts in presence. Thus clusters of different sizes including small ones are expected, which is a hurdle for most algorithms, including the prevalent methods based on community detection [65].

A core application for sequence clustering methods when it comes to working with RNA has been the clustering of EST. This type of data, produce and studied in the 2000's, shared the same type of problematics than long RNA reads: being able to cluster and capture alternate transcripts forms that represent a single gene, and obtain a consensus of from the sequences (by assembly). Most of cDNA clones used did not represent full-length mRNA, and ESTs were known to contain errors. ESTs were collected into *gene indices*, which were clusters where each set of transcripts correspond to a single gene. Tools such as STACK [37] (with a clustering phase made by d2_cluster [26]) performed such tasks. d2_cluster [26] was named after the $d2$ distance of dissimilarity between two sequences introduced by Torney [232], based on the comparison of the multiplicity of words between them ($d2 = 0$ meaning that two sequences are identical). After the similarity computation, many of these tools used single-linkage transitive closure algorithms (i.e. finding connected component in the graph) [53, 27]. Such clustering had the advantage to recruit all transcripts of a gene that share at least an exon, and to produce rather big clusters. A counterpart of searching connected components in the graph of EST sequence was that its simplicity could lead to chimeric clusters, for instance

because of repeats. These chimeric clusters would lead unrelated gene families to each other. Other tools such as CD-HIT [131] also proposed a "EST" version to retrieve clusters of similar sequences.

1.3 Clustering RNA long reads

Previous works The issue is to automatically group alternative transcripts in a same cluster (Figure 6), which means retrieve and account for similarity information despite the reads error rates. We presented clustering tools that were proposed for NGS reads. As for TGS long reads, a few recently published pipelines aim at retrieving the expressed gene that correspond to reads, but they rely on reference genome information [28, 1]. A sole tool, Tofu, is designed to work with reads *de novo* and was published in 2015 [77]. Tofu is an iterative pipeline that highly relies on the properties of PacBio sequencing experiment, and requires specific features of long reads from Iso-seq sequencing. It is therefore adapted to PacBio reads, but not to ONT reads. Tofu relies on previous works towards the clustering of PacBio reads, summed up in the core clustering method called ICE. Reads of higher quality (CCS) are mapped against one another and similar isoforms are detected using BLASR [32]. A graph of isoform similarity is then built on this basis. Maximal cliques are looked for in the graph [174]. Once a clique is found, nodes are removed from the graph, and the procedure is repeated until the whole graph is partitioned. Each clique is considered as a cluster. Incidentally, the aim of Tofu differs from our expected result since it retrieves one cluster per isoform rather than one cluster per expressed gene. However, it is a pipeline that goes further than isoform clustering, and we will discuss its interest in the next chapters.

1.3.1 Validation of clustering methods goodness

In order to asses which algorithm performs correctly on long reads from Nanopore experiments, we set up a benchmark used real data from mouse transcriptome. Mouse has the advantage to possess good quality reference genome and transcriptomes that can be used for validation, by comparing any *de novo* result to a mapping result. The reads we used were sequenced at the Genoscope by Jean-Marc Aury's team in the context of the ASTER ANR project, which has intersecting goals with the presented work.

Mapping to obtain reference clusters for validation Among the several generations of sequencing datasets, we used 1Donly Nanopore reads from the mouse brain transcriptome sequenced on a MinION platform (accession number: ERP107503)[1]. 1,256,967 nanopore 1D reads were obtained, representing around 2Gb of data with an average size of 1650bp and a N50 of 1885bp. "Ground truth" clusters were computed at the Genoscope and used here for validation purpose. A Genoscope in-house protocol based on mapping on a reference was performed. Reads from the mouse brain transcriptome were aligned to the masked mouse genome assembly (version GRCm38) using BLAT [103]. For each read, the best matches based on BLAT score (with an identity percent greater than 90%) were selected. Then, those matches were realigned on the unmasked version of the genome using Est2genome [159]. Reads that mapped onto the mitochondrial and ribosomal sequences were discarded. Moreover, one region on chromosome 1 was excluded as it harbors a high number of reads (>10k). Next, reads were clustered according to their genomic positions: two reads were added in a given cluster if sharing at least 10 nucleotides in their exonic regions. For the whole data experiment, all reads that could be mapped on the reference were taken into account (501,787). Due to repeats, some reads were mapped at multiple loci on the reference. When a given read maps on several loci, such loci are gathered in a single expected cluster (12,596 expected clusters, we recall that the total number of genes in mouse is around 27,000). This means that for instance reads from copies of paralog genes that have not diverged to much or reads containing a copy of a transposable elements are expected to be in the same cluster. This set of clusters, or a subset, is used in the following as a "ground truth". not all reads errors due to repeats

[1] https://www.ebi.ac.uk/ena/data/search?query=ERP107503

Clusters' goodness assessment metrics To assess the results, we used recall and precision measures, which are standard measures to assess the relevance of biological sequence clustering [243]. The recall and precision measures are based on reference clusters obtained by mapping for this validation. They are computed based on representative graphs [213]. These measures were already used to assess the relevance of biological sequence clustering [243]. Let $\{\mathcal{C}_1, \dots \mathcal{C}_i\}_{1 \leq i \leq L}$ be the set of clusters found by the clustering method tested, where L is the number of predicted clusters. Let $\{\mathcal{K}_1, \dots \mathcal{K}_j\}_{1 \leq j \leq K}$ be the set of "ground truth" clusters, where K the number of expected clusters. Let R_{ij} be the number of nodes from $\mathcal{C}_i$ that are in "ground truth" cluster $\mathcal{K}_j$. Recall and precision are defined by:

$$Recall = \frac{\sum_{j=1}^{K} max_i(R_{ij})}{\sum_{i=1}^{L} \sum_{j=1}^{K} R_{ij}} \tag{2.1}$$

$$Precision = \frac{\sum_{i=1}^{L} max_j(R_{ij})}{\sum_{i=1}^{L} \sum_{j=1}^{K} R_{ij}} \tag{2.2}$$

Note that the "ground truth" is not really available and that it is estimated from mapping results on the reference genome. Recall expresses the mean over all clusters of the fraction of relevant reads in a result cluster out of the expected read population of this cluster. It presents to which extent clusters are complete. Precision expresses the mean over all clusters of the fraction of relevant reads among the population of a result cluster. It shows the clusters' purity. The F-measure is a summary measure computed as the weighted harmonic mean between precision and recall. Recall and precision are tailored to express the confidence that can be placed in the method, according to its ability to retrieve information and to be precise. We complementary assess the closeness of the computed clusters as compared to mapping approaches. Let $\mathcal{X}_0$ be the reference partition (set of clusters obtained by mapping), and $\mathcal{X}$ the partition obtained using a given clustering method. Then a_{11} is the number of pairs of nodes that are placed in a same cluster in $\mathcal{X}_0$ and $\mathcal{X}_1$. a_{00} indicates the number of pairs for which nodes are placed in different clusters both in $\mathcal{X}_0$ and $\mathcal{X}_1$. a_{10} (respectively a_{01}) is the number of pairs of nodes placed in the same community in the reference $\mathcal{X}_0$ (respectively $\mathcal{X}$) but in different clusters in $\mathcal{X}$ (respectively $\mathcal{X}_0$). Based on those, several indices show the match between the reference and computed partitions described, such as the Jaccard index:

$$J(\mathcal{X}_0, \mathcal{X}) = \frac{a_{11}}{a_{11} + a_{10} + a_{01}} \tag{2.3}$$

In ranges in $[0, 1]$ The closer to 1, the closer the predicted set of clusters is close to the "ground truth" set.

1.3.2 Evaluation of state of the art clustering algorithms on long reads

We first study the behavior of major community finding algorithms on long reads, as we consider they are the basis of implemented methods to cluster sequences. To our knowledge, none of these algorithms had previously been tried for the purpose we describe using TGS reads. We chose to perform the benchmark on a subset of 10K reads (10,183 mouse reads within 207 reference clusters) from the previously described dataset. This allows a quick assessment of the different results. However, such sampling can accentuate the low expression effect in the subset. We have thus checked the methods on a second 10K sample from chromosome 1 only to also account for highly expressed genes. We evaluated four state-of-the-art methods that were previously applied to similar problems of biological clustering: single linkage transitive-closure (which is not properly a community finding algorithm but has been a widely spread solution for EST, used for instance in [53, 27, 37]), a *modularity*-based algorithm [73] (used in [152, 247, 167]),the Clique Percolation Method by Palla et al [172] (used in [98, 2]) and Louvain algorithm [21] (used in [64, 156]). Any of these

methods require a common input similarity graph. Such graph was computed using Minimap in the same fashion than in last section of the previous chapter. The reads constituted the set of nodes of the input graph, and we draw edges between nodes when Minimap detected similarity between a pair of reads. More details on the use of Minimap in this case is given later in this chapter. Both benchmarks are presented in Tables 2.1 and 2.2. Trends presented in both Tables lead to the same conclusions. On a rather small instance already,

Table 2.1: Comparison with state of the art methods. The benchmark was realized on a 10K reads dataset from the mouse brain transcriptome. Recall precision and Jaccard Index are presented (see Equation 2.3, 2.1 and 2.2) to assess for the goodness of communities detection. CPM5 (respectively CPM50) designates the CPM algorithm using $k = 5$ (respectively $k = 50$).

	Recall (%)	Precision (%)	F-measure (%)	Jaccard index
Transitive closure	75.74	5.614	13.62	$7.3E^{-4}$
Modularity	60.70	71.16	65.51	$9.7E^{-2}$
CPM5	79.00	69.35	73.86	$3.5E^{-1}$
CPM50	49.21	89.92	63.60	$7.6E^{-2}$
Louvain	**88.58**	14.91	25.53	$1.1E^{-3}$

Table 2.2: Comparison with state of the art methods on a chromosome 1 10K reads sample. Reads were selected randomly from the brain transcriptome dataset. Recall precision and Jaccard Index are presented (see Equations (3,4,5). CPM5 (respectively CPM50) designates the CPM algorithm using $k = 5$ (respectively $k = 50$).

	Recall (%)	Precision (%)	F-measure (%)	Jaccard index
Connected comp.	69.04	21.76	33.09	$2.3E^{-1}$
Modularity	34.30	12.0	17.77	$2.1E^{-1}$
CPM5	55.29	29.27	38.27	$3.4E^{-1}$
CPM50	12.52	89.63	21.96	$5.2E^{-1}$
Louvain	**87.50**	1.191	2.350	$3.1E^{-2}$

all approaches show globally a lack of precision. As expected, the transitive closure approach suffers from low precision, as it tends to aggregate too many reads. Thus, low precision conveys the idea that clusters were mixed up. Low precision combined to high recall indicates that most of the reads are retrieved in big clusters, that represent several "ground truth clusters" mistakenly put together. On the contrary high precision combined to low recall tends to show that the data was over-clustered, leading to small, hence precise clusters that do not gather a lot of information. The *modularity*-based approach fails to find good clusters for this graph, with both low recall and precision. Louvain is also based on *modularity* and shows even more contrasted patterns, where it seems that most of the reads are recruited in a same cluster. Despite showing the best recall, Louvain's precision is too low to reach a high F-measure or Jaccard index. Louvain being an multi-level algorithm, we also tested results after each iteration, with similar trends (presented results being for the last iteration of their algorithm). In the introduction to community finding methods, we anticipated that the heterogeneous size distribution of the clusters could be a cause of failure, as well as of overlapping effects due to the repeats. CPM was tested with values for its input parameter k ranging from 3 to 50 (no community found for greater values). k represents Results are presented for k=5 and k=50

and summarize the behavior of this approach on our input graph. For low values of k, CPM outputs more clusters and has better recall than for high values. However, its precision is globally low. For higher values of k, the results are strongly enhanced but represent only a small fraction of the input. By concentrating its results in a small subset of clusters, it obtains a poor recall and not all its predicted clusters can be trusted. Thus, this study demonstrates that the current community finding approaches have limited power on a long read dataset. In order to assess whether the implementations specifically designed for sequence clustering help in the treatment of long reads, we present a second study in the following.

1.3.3 Evaluation of sequence clustering tools on long reads

To set up this second benchmark, we reproduced the study designed in one of the latest of sequence clustering tools' paper [261]. Zorita et al. selected the most widely used clustering tools to compare their method with: CD-HIT [131], SEED [10] and Rainbow [36]. We added the only other *de novo* clustering tool that, to our knowledge, is designed to work with long reads, Tofu [77]. We used the same mouse dataset as in the previous section, and present our result in Table 2.3. Again, a second way to sample data is also presented in a second Table 2.4.

	Recall (%)	Precision (%)	F-measure (%)	Status
CD-HIT	26.60	**99.27**	41.96	run
SEED	0	0	0	run
Starcode	-	-	-	error
Rainbow	-	-	-	could not be run
Tofu	-	-	-	could no be run

Table 2.3: Benchmark of nucleic acid sequence clustering tools. Each tool embeds its own strategy to compute similarity between sequences, our pipeline integrates Minimap. The three first columns present the same metrics than in previous benchmark. The *status* column indicates whether the tool could be used on ONT reads. Wallclock time and peak memory are reported in the last columns. Starcode finished with error message "does not work with sequences longer than max sequence length exceeded (1023)". SEED returned an empty output. Rainbow could not be run because it explicitly asks for paired reads typical from RAD-seq short reads sequencing. Tofu pipeline specifically involves sequences from PB Iso-seq sequencing that cannot be replaced with ONT reads.

All these tools counter-perform, as only two could actually be launched and give poor results (F-measure up to 41.96% due to low recall). SEED is designed to create clusters with sequences that differ from at most 3 mismatches, thus finds no clusters. Starcode is not adapted to the size of ONT sequences and terminates with an error message. Moreover, as well as SEEDS, Starcode authorizes a limited distance between pairs of sequences (a maximum Levenshtein distance of 8) which is not adapted to long reads error rate. We modified Starcode source code to increase the maximum length size for reads. This led Starcode to either output singleton clusters or segfaulting for large maximum values. Rainbow only accepts paired reads such

	Recall (%)	Precision (%)	F-measure (%)	Status	time (hh:mm:ss)	Memory (Mb)
CD-HIT	11.70	**99.87**	20.95	run	03:33:2	3.11
SEED	0	0	0	run	ends immediately	-
Starcode	-	-	-	error	-	-
Rainbow	-	-	-	could not be run	-	-
Tofu	-	-	-	could no be run	-	-

Table 2.4: **Benchmark of nucleic acid sequence clustering tools (read sampling from mouse chromosome 1).**

as those sequenced in RAD-seq short reads experiments (sequencing of isolated specific genome regions). CD-HIT, that maybe the most broadly used sequence clustering tool, was used in its "est" version. We tested different values for sequence identity threshold (`-c`), that can be decreased down to 0.8. We report only the best result, reached for `-c 0.8`. Finally, Tofu highly relies on the specificity of Pacific Bioscience RNA protocol (Iso-seq) sequences, and cannot be run with ONT reads. However, we recall that none of these tools was designed to work with ONT reads, which explains the results. At the exception of Tofu, they are anterior to TGS and suppose high similarity between sequences. This study, jointly with the previous section, shows that there is room for improvement for *de novo* clustering of nanopore reads. Notably, any proposed solution lacks of precision (or are very precise at the scale of a few clusters), and many output extremely large or extremely small clusters according to the benchmarks. This conclusion motivates our own developments, that aim at proposing a new clustering method adapted to ONT long reads.

2 Novel algorithm for gene expression clustering in long reads

This section describes an algorithm dedicated to the clustering of long RNA reads in clusters by gene, that we designed during this thesis. Within a long read dataset, our goal is to *de novo* identify for each expressed gene the associated subset of TGS reads. Our method was designed with in mind general long reads properties, before being tested on ONT reads in the next section. Here we state our methodological choices and explain the algorithm.

2.0.4 General principles

While being largely inspired by community finding strategies, our method intend to take the best of both worlds and include some feature from the graph clustering strategies, by trying to minimize a cut size. Details of these methods are detailed in the following. As any general clustering purpose, we seek to maximize intra-cluster similarity and minimize inter-cluster similarity. As mentioned previously, the expected subgraph signature of a gene in the graph of reads is a community, that is, a cluster of similar reads. Our method makes no assumption on the number of expressed genes (i.e. communities), since we want it to work on non model species. We do not make assumption either on the size distribution of such communities. As we want to realize a partition of the graph, there are no intersecting communities (no read belongs to several gene families) and every node belongs to a community (each read is assigned to a gene).

2.0.5 Description of input graph object

Our method starts from a set of long reads and a graph of similarity between them. We define a similarity graph as an undirected graph in which nodes are reads and there is an edge between two nodes if the computed similarity between these nodes exceeds a fixed threshold. This kind of graph can be constructed using mapping methods dedicated to long reads, that are able to detect overlaps between long reads. We presented and compared main methods in the previous chapter. In the graph, reads from a same gene are expected to be connected with one another because they are likely to share exons. In the ideal case, all reads from a gene are connected with one another. It is therefore a clique. We shown in the end of the previous chapter that the spurious nature of data imposes the use of heuristics to detect read overlaps. As previously presented, this leads to failing to retrieve all links between reads (recall of mapping methods is not of 100%), and to the presence of spurious links (wrong connections between unrelated reads, since precision is not of 100% either). This leads to the expectation of a graph with both missing edges and spurious edges. A toy illustration of the situation is presented in Figure 2. These caveats have to be taken into account. In

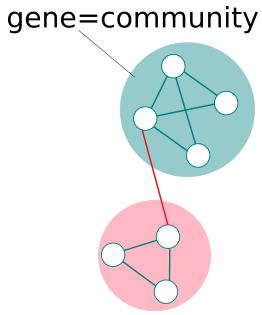

Figure 2: **A small example of read similarity graph.** Each read is a node, detected similarity between reads is materialized by an edge if it is over a given threshold. The expected signature of a gene in such a graph is a clique and a connected component. Communities in this example are colored zones (green and rose). Noise in data leads to missing information (missing edge in the green subgraph) as well as spurious edges between unrelated genes (red edge).

particular, we want to obtain a method which clusters can be trusted, thus makes good choices in removing the spurious links.

2.0.6 Formalization

As previously presented, community finding algorithms rely on measures of the density of edges in subgraphs. To measure the density within subgraphs, we choose to use the clustering coefficient (*ClCo*) [164] rather than *modularity*. Indeed, we assume that a gene should be represented by a complete subgraph (clique) in a perfect similarity graph. As previously stated in its definition, this coefficient is more directly connected to the notion of clique than *modularity*. An example of *ClCo* computation is given in Figure 3. Working with a possibly heterogeneous graphs, *ClCo* computations also allow us to compute local densities rather than trying to optimize a global metric. Naturally, subgraphs which cannot increase their clustering coefficient by the inclusion of new nodes are good candidates for community clusters. The foundation of our method is then a problem depending on two parameters, the number k of clusters and the cutoff θ on the *ClCo* value. Specifically, the original problem is formalized as follows:

Definition 1 *A community is a connected component in the graph of similarity having a clustering coefficient above a fixed cutoff θ. An optimal clustering in k communities is a minimal k-cut, that is, a partition*

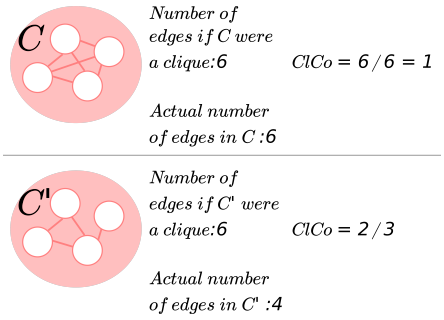

Figure 3: **An example of $ClCo$ computation.** $ClCo$'s value is the highest on cliques. It can be computed by dividing the number of edges of the observed graph by the number of expected edges if this graph were a clique. This leads to a value of 1 for cliques, and between 0 and 1 for any other graph.

of the graph nodes in k subsets, that minimizes the total number of edges between two different subsets (the cut-set).

Relying on results from previous chapter, we assume that the overlap detection procedure has good specificity (it produces a low rate of spurious overlaps), thus our input similarity graph contains mostly biologically sound edges. The logic behind the search for a minimum cut in the graph is that most of the edges of the initial graph should therefore be kept during clustering. This problem is known to be NP-hard for $k \geq 3$ [46]. Another source of complexity is that we don't know in advance the number of communities, so we have to guess the value of k. One should thus compute the k-cut for each possible value between 1 and the maximum, which is the number of reads. Solving this problem is not feasible for the large number of reads that have to be managed. We are thus looking for an approximation of the solution by using an efficient heuristic approach exploring a restricted space of values for k. Finally, the second parameter, the cutoff θ, is not known either. The algorithm has thus to loop over all possible values, that is, all $ClCo$ values for a given connected component. In practice, we will show it is sufficient to sample a restricted space of possible k values. In short, our method means to find communities such as:

1. A community is a connected component having a clustering coefficient above a fixed cutoff θ ¿ 0.
2. Communities are disjoint sets. These two first properties are illustrated in Figure 4.
3. An optimal clustering in k communities is a minimal k-cut of the graph. Figure 5 shows the usage of this third property.

2.1 Implementation details

2.1.1 Outline

Our community detection algorithm is composed of two main steps. The first one looks for an upper bound of the number of clusters k. To this aim, we relax the condition of disjoint communities and look initially for star subgraphs (a read connected to all reads similar to it) having a clustering coefficient above a certain cutoff. This corresponds to detecting well-connected reads, called seed reads, using $ClCo$ and node degrees. They form the basis of communities with their neighborhood.

The main challenge is then to refine the boundaries of each community in order to fulfill the partition condition. During this process, the value of k is progressively refined by possibly merging clusters whose combination produces a better community (greater $ClCo$ value). The other possibility of refinement is to assign nodes to a community and remove them from another. If x edges between a node and its previous

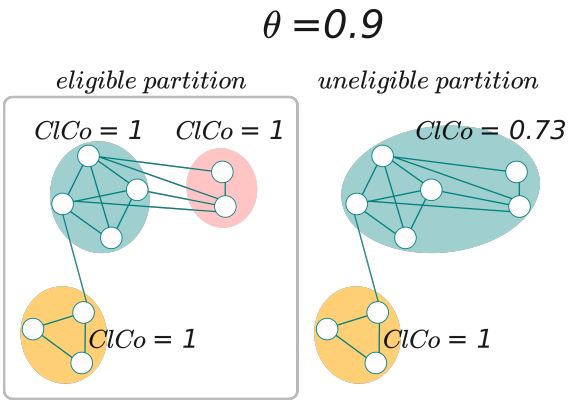

Figure 4: **Here we choose an example of partitions with cutoff θ fixed to 0.9.** Two possible ways to partition the graph are presented on the left and on the right. Communities are colored subgraphs, there are three communities on the left partition and two on the right one. We computed the *ClCo* of each community. The left set of communities is eligible according to the properties looked for in our algorithm, because 1-all communities' *ClCo* are above or equal to the cutoff, and 2- it is a partition. The right community is a partition as well but is not eligible because one *ClCo* is below the threshold.

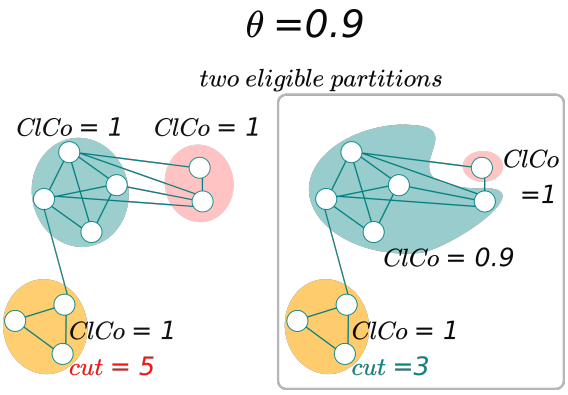

Figure 5: **We take again the eligible partition of the previous figure on the left, we keep θ fixed to 0.9.** The cut to realize the left set of communities is of 5, because separating the graph in three communities leads to cut 1+4 edges. On the other, in another set presented on the right, that respects the conditions of the graph partition and of the cutoff, the cut is only of 3. Thus we will chose this right set over the left one in order to minimize the cut.

community are removed, the cut size of the partition is increased by x. This core algorithm is run for different cutoff values to obtain different partitions that we compare. We keep the partition that is associated to the minimal cut (i.e. number of edges removed when computing this partition). Figure 7 sums up the different steps and the pseudo code of the main algorithm is detailed in Appendix. In the following, the different steps of the implementation are detailed.

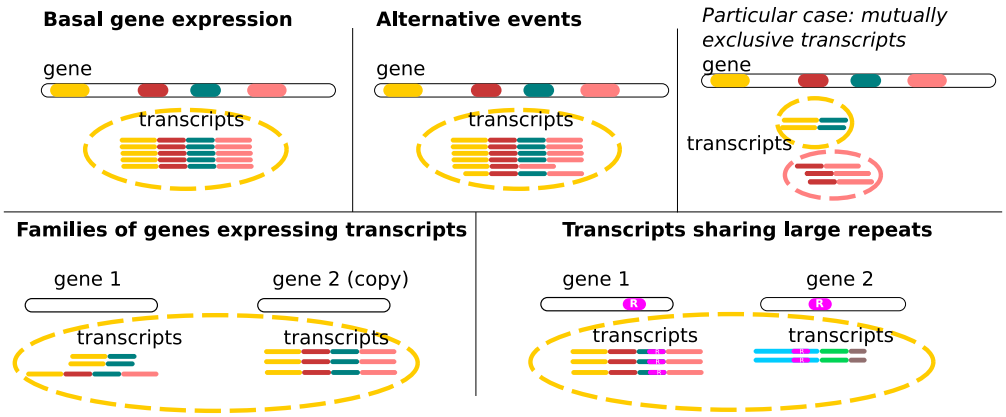

Figure 6: **Clustering scenarios.** In the case of basal gene expression and alternative events (described below), with the exception of mutually exclusive transcripts, it is expected that all transcripts of a gene will be grouped together in a single cluster. Very small exons or very long retained introns (not shown) can also be limitations according to the mapping tool strategies. In the more complex case of families of genes, two or more copies of paralog genes can express transcripts at the same time. If these transcripts share a common exonic content and if the gene sequences have not diverged too much (to allow overlap detection), transcripts from this family of genes are clustered together, despite coming from different loci. Although this is an algorithmic limitation, it can be interesting to group these sequences together, as they likely share similar functions. A like scenario occurs for transcripts sharing genomic repeats (such as transposable elements).

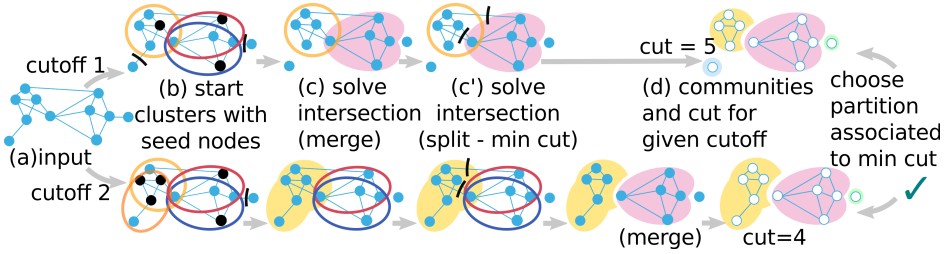

Figure 7: **Summary of the algorithm.** (a) All *ClCo* and degrees are computed. Each *ClCo* value is a cutoff. For a given cutoff, (b) different cutoffs yield different seed nodes (black stroke) that initiate clusters with their neighborhood. (c, c') Boundaries of each cluster are then refined. Intersection between clusters are solved either by (c) merging them or by (c') splitting. (d) The communities at different cutoffs evolve in different partitions. In the end we keep only the best partition according to our criterion, i.e. minimizing the cut.

2.1.2 Generation of partitions

In order to generate and compare different partitions for the graph, we define cutoffs that rule the generation and refinement of communities. The cutoffs can be seen as the level of connectivity at which a community can be generated ((a,b) steps and (c) merge step in Figure 7). In the basic algorithm, for each connected

component, all C_lC_o are computed in the first place, and partitions are built for each non-zero $ClCo$ value as a cutoff. In the end, only one partition is retained, associated to the minimal cut (step (d) in Figure 7). However, we have reduced the number of possible cutoff values for the sake of scalability. To this end, for all node in the graph, a $ClCo$ centered the node (Equation 2.4) is computed in a first pass. The resulting set of $ClCo$ is used as the range of cutoffs. An example is presented in Figure 8. In the following, each step

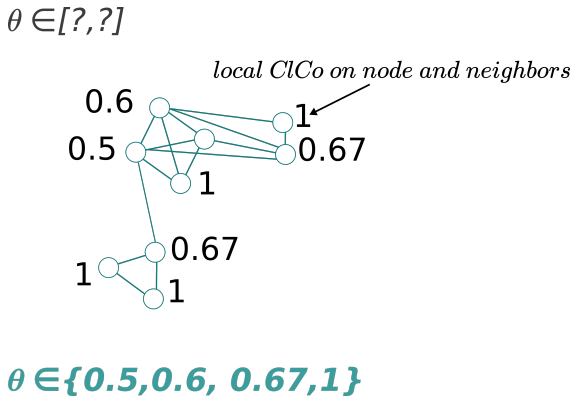

Figure 8: **In this figure we show how we avoid to compute $ClCo$ values for each subgraph in order to have a range of cutoff θ values.** Local $ClCo$ are computed centered on nodes. This to a restraint set of four $ClCo$ values as several nodes share the same values. This set will be used in our algorithm for cutoffs on this connected component. It is noteworthy that on another connected component of the graph, this set might change.

is described for a given cutoff value.

2.1.3 Selection of community founding nodes

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be an undirected graph of reads. Let n_i be a node (read) from $\mathcal{N}$ and $N_i \subset \mathcal{N}$ its direct neighborhood. Let $deg(n_i)$ be the number of edges connecting n_i to its direct neighbors (similar reads), i.e. $deg(n_i) = |N_i|$. For each node $n_i \in N$ with degree $deg(n_i) > 1$, we first compute the *local clustering coefficient*:

$$ClCo_i = \frac{2 \ |\{(n_j, n_k) \in \mathcal{E} : n_j, n_k \in N_i\}|}{deg(n_i) \times (deg(n_i) - 1)} \tag{2.4}$$

Nodes of degree 0 and 1 have a $ClCo$ of 1. This local coefficient represents the *cliqueness* of the $N_i \cup n_i$ set of nodes. The closer to 1, the more the set of nodes is inter-connected, which witnesses a group a reads that potentially come from the same gene. By contrast, the subgraph induced by a node with a $ClCo$ of 0 and its neighbors is a star (i.e. a tree whose leaves are all the neighbors). If the coefficient is close to 0, the nodes are weakly connected and are unlikely to come from the same gene. In order to prevent unwanted star patterns we add an auxiliary condition for nodes to be eligible seeds. At a given cutoff value, the seed reads are primarily nodes which $ClCo$ is above or equal to this value. We add a statistical precaution to prevent star-like patterns (with a very low $ClCo$ with respect to the degree of the seed node) to initiate communities. We recall the star pattern case and a more likely case of degenerated star and its impact on the graph in Figure 9. We state the following auxiliary condition for seeds:

$$\forall n_i, ClCo_i \in]cutoff, \theta_2[\Rightarrow deg(n_i) \leq \theta_1 \tag{2.5}$$

θ_1 and θ_2 are the values such that 1% of the observed degrees are greater than θ_1 and 1% of the observed $ClCo$ are lower than θ_2 (1st and 99th percentiles). The selected seeds with their direct neighbors form the

initial communities. This second condition was motivated by empiric observation as well as the particular case of degenerated star patterns in the graph (Figure 9). These nodes are then prevented to form clusters. We show examples of such nodes in the mouse dataset in Appendix. We recall that at this point, we relaxed

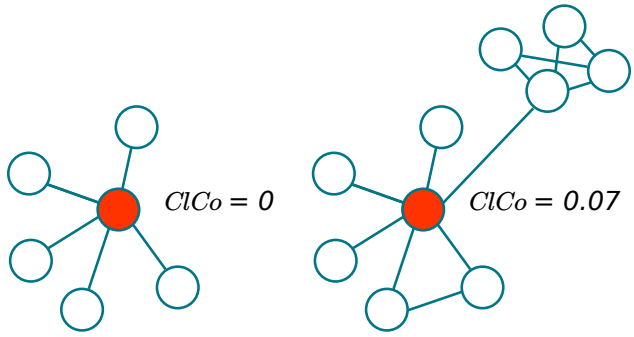

Figure 9: **Examples of stars in graph.** On the left we present an example of a star with its central node in red which $ClCo$ is of 0. On the right we present a more realistic example where a degenerated star pattern creates a very low $ClCo$ value that will also be a cutoff value. In a real example this node might be highly more connected.

the condition that community must realize a partition. It is then possible that two or more communities intersect.

2.1.4 Refinement of community boundaries

Community refinement aims at solving the conflicts of intersecting communities. Communities intersection happen because of spurious connections in the graph due to the creation of edges between unrelated reads in the first step.
The intersecting communities are looked up pairwise in order to assign nodes of the intersection to only one community. In fact two cases have to be distinguished.

Split and merge procedures Either the edges between two communities are estimated spurious and these communities must be seen separated (*split*, (c') step in Figure 7, or the edges have sufficient support and the two communities have to be merged to obtain the full gene expression (*merge*, (c) step in Figure 7). In order to decide between the two, we use again the *cliqueness* notion. This time we introduce an *aggregated clustering coefficient* of the union of two nodes n_i and n_j :

$$ClCo_{ij} = \frac{2\ |\{(n_k, n_l) \in \mathcal{E} : n_k, n_l \in N_i \cup N_j\}|}{|N_i \cup N_j| \times (|N_i \cup N_j| - 1)} \tag{2.6}$$

If the value of $ClCo_{ij}$ is greater than or equal to the current cutoff, we consider that there is a gain in connectivity when looking at the union of the two communities and they are merged. In the other case, the nodes of the intersection are reported to only one of the two communities. We call it split procedure, and report pseudo-code for this part in algorithm of Figure 10. We remove the edges connecting these nodes to one or the other cluster according to which realizes the minimal cut. In case of ties for the cut, the algorithm uses a second criterion. During the split procedure, nodes are assigned to one of the two observed clusters, depending on the cut value (in order to minimize it). In the case of a tie, we add a second criterion: the maximization of the sum over all communities of their clustering coefficient values. For two sets N_1, $N_{1'}$, $N_{1'} \subseteq N_1$, this difference is defined as :

$$\Delta ClCo_{N_1, N_{1'}} = ClCo_{N_1} - ClCo_{N_{1'}}, \tag{2.7}$$

with CC calculated as in Equation 2.6. In some cases, the split procedure leads to breaking connected component. We ensure that for clusters which nodes were removed from, the connectivity is still present. If it is not the case, we create as many new (smaller) clusters as there are connected components after the split.

It must be noted that this last tie procedure could have been the generic procedure. Instead of generally minimizing the cut during a split, we could rather have maximized the ΔCC. However, this alternative was tested in early versions of our algorithms using mouse data, and resulted in a slight decrease of the performances of cluster resolution. Based on those empirical results we preferred this version of the procedure.

Details on the clusters comparison The global result depends on the order in which pairs of clusters are compared. This order is carefully designed. First the communities associated to the two nodes of greatest degree (and secondly maximal $ClCo$) are chosen, the intersection is resolved and the first community is updated. Then, it is compared to the third best community that intersected it if it exists, and so on until all intersections are solved. This way, we start the comparison by the most promising communities that combine reliability (they are well-connected subgraphs) with a high potential of resolution (they likely are the biggest communities, thus solve intersections for many nodes). On the contrary, communities associated to small subgraphs and relatively low $ClCo$ are only resolved afterwards.

2.1.5 Final clusters

Several initial $ClCo$ threshold on connected components give rise to several partitions. For each connected component, we keep the partition associated with the minimal cut. Then we output as many disjoint final partitions as there are connected components as our result clusters. The fact that our algorithm authorizes communities to merge to increase their connectivity leads to wonder how it ensures that it is not always a whole connected component that is chosen as a final cluster, since the cut would be of 0. We show a counterexample in Figure 11 to illustrate that according to the properties we expect from communities, even if some partitions lead to a cut of 0, they are not selected as the final cluster because of their $ClCo$. We also experimentally verified that any output cluster had a $ClCo$ value above or equal to the retained threshold for each connected component, using mouse dataset.

3 CARNAC-LR and long read clustering pipeline

In this section we present the choices we made to implement the previous algorithm into a tool a tool dubbed CARNAC-LR (**C**lustering coefficient-based **A**cquisition of **RNA** **C**ommunities in **L**ong **R**eads). This tool's goal is to output clusters that correspond to genes' expressed reads in a long read data. In particular we propose solutions to make CARNAC-LR scalable. We also present the pipeline in which CARNAC-LR is inserted in, that takes long reads from a transcriptome sequencing and clusters them by genes.

3.1 Pipeline overview

3.1.1 Input/Output

The pipeline's input is a FASTA file of reads. The output is a text file with one line per cluster, each cluster containing the read indexes. Each read is represented by its index in the original FASTA file during CARNAC-LR computation. Then using indexes, each cluster can easily be converted to a FASTA file where each read's sequence is retrieved from the original file (a script is proposed for doing this task). An overview of the pipeline is presented in Figure 12 A visual example of our pipeline's output is provided in Figure 17. We have selected a cluster of sufficient size to be able to present a variety of isoforms. It corresponds to a gene for which mapping retrieved 120 reads. In this example, our approach recovered 93% of the predicted gene's reads while including no unrelated read in the cluster. Two types of missed reads can be distinguished: 1) A

Algorithm: Split

Data: Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$.
$\mathcal{C}_i$ and $\mathcal{C}_j$ two clusters of nodes $\subset \mathcal{N}$ with a non null intersection $\mathcal{I}_k = \mathcal{C}_i \cap \mathcal{C}_j$.
Cl is the set of clusters.

$cut_{\mathcal{C}_i} \leftarrow \{e_{lm} : n_l \in \mathcal{C}_i \setminus \mathcal{I}_k, n_m \in I_k, e_{lm} \in \mathcal{E}\}$;
$cut_{\mathcal{C}_j} \leftarrow \{e_{lm} : n_l \in \mathcal{C}_j \setminus \mathcal{I}_k, n_m \in I_k, e_{lm} \in \mathcal{E}\}$;
switch $\mathbf{cut}_{\mathcal{C}_i}$ **do**
    $> \mathbf{cut}_{\mathcal{C}_j}$: Remove nodes of $\mathcal{I}_k$ from C_j;
    if $\mathcal{C}_j$ *is not connected anymore* **then**
        Split($\mathcal{C}_j$) using steps 11, 12, and 13 of main algorithm in appendix;
        Append new clusters in Cl;
    $< \mathbf{cut}_{\mathcal{C}_j}$:
    Remove nodes of $\mathcal{I}_k$ from $\mathcal{C}_i$;
    if $\mathcal{C}_i$ *is not connected anymore* **then**
        Split($\mathcal{C}_i$) using steps 11, 12, and 13 of main algorithm in appenix;
        Append new clusters in Cl;
    $== \mathbf{cut}_{\mathcal{C}_j}$:
    Compute $\Delta CC_{\mathcal{C}_i, \mathcal{C}_i \setminus \mathcal{I}}$ and $\Delta CC_{\mathcal{C}_j, \mathcal{C}_j \setminus \mathcal{I}}$ (equation 2.7);
    if $\Delta CC_{\mathcal{C}_i, \mathcal{C}_i \setminus \mathcal{I}} \leq \Delta CC_{\mathcal{C}_j, \mathcal{C}_j \setminus \mathcal{I}}$ **then**
        Do steps 5 to 8.
    else
        Do steps 10 to 13.

Figure 10: **Node removal from one of two intersecting sets.** This procedure chooses the set to shrink by keeping the minimal cut between the two. In case of ties, this procedure attributes the nodes of the intersection to the set that has the greatest gain or the lowest loss of connectivity when keeping the nodes of the intersection.

group of black reads that share no genomic sequence with the majority of the gene's transcript, because they come from an intronic region. These reads could not be linked to the others by Minimap and thus cannot be clustered with them, as shown in the particular case described in Figure 6. 2) Two other reads for which local connectivity was not detected by Minimap were discarded from the cluster. The plot shows different exon usage in transcripts, which reveals alternative splicing in this cluster. Thus different alternative isoforms were gathered in a single cluster as expected.

3.1.2 First step: computing similarity between long reads

Our pipeline starts with the computation of long read similarities using Minimap [126] and then produces the clusters. Minimap was chosen according to results of the previous chapter, for its efficiency and its very high level of precision on ONT and PB [38], among other recent methods that can compute similarity or overlaps between long reads despite their error rate [162, 19, 224, 32]. To generate the similarity graph for CARNAC-LR, Minimap version 0.2 was launched with parameters tuned to improve recall (`-Sw2 -L100 -t10`). It produces a file of read overlaps in .paf format.

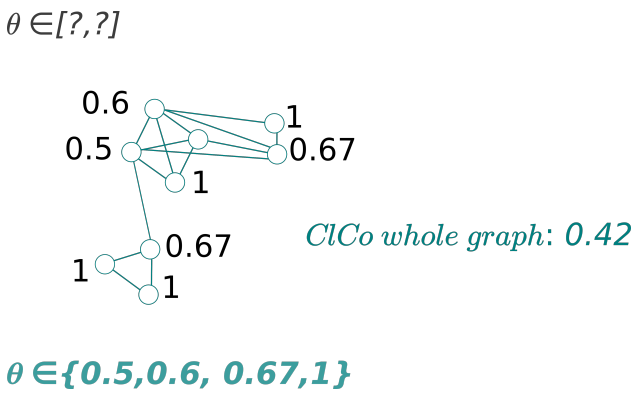

Figure 11: **We compute local *ClCo* in the connected component to obtain a set of values to set the cutoff θ to.** Let's assume that by iterative merges of communities, on ends up with the whole graph. Even by looking at the lowest theta value (0.5), the *ClCo* of the whole graph (0.42) makes it ineligible to be a community as it is lower than θ. Thus, even if the whole graph as cluster would lead to a cut of 0, it is never selected as a final community.

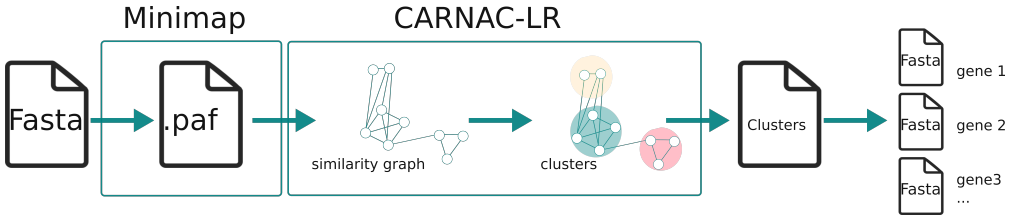

Figure 12: **CARNAC-LR and the long reads clustering pipeline.**

3.1.3 Second step: clustering

Minimap's output is converted into a graph of similarity, where each node represents a read and an edge a sequence similarity between two reads above a certain threshold (see [126]). Such graph is then passed to CARNAC-LR, that is parameter-free and retrieves and outputs the gene clusters. CARNAC-LR benefits from parallelization. A thread can be assigned to the treatment of a single connected component, thus many connected component can be computed in parallel.

3.2 Implementation choices

In order to cope with noise in the input graph, we introduce features to simplify the graph (disconnect loosely connected nodes) and to control the space of research of the possible partitions. In practice these features are also key to reduce the complexity of our approach. Our experiments showed that the running time is reasonable, clustering millions of reads in a few hours. Two key ideas to obtain this result have been to reduce the number of cutoffs and to disconnect the articulation points [92] to reduce the size of connected components in the graph. Indeed, the most costly phase relies on the treatment of the largest connected components. In these components, many clustering coefficients values are very close and their variation is mainly an effect of noise. Introducing a rounding factor when computing the *ClCo* results in a neat decrease in the number of different values observed, which drastically limits the number of iterations required for the main loop, while providing a very good approximation of the minimal cut. In addition, an upper bound is

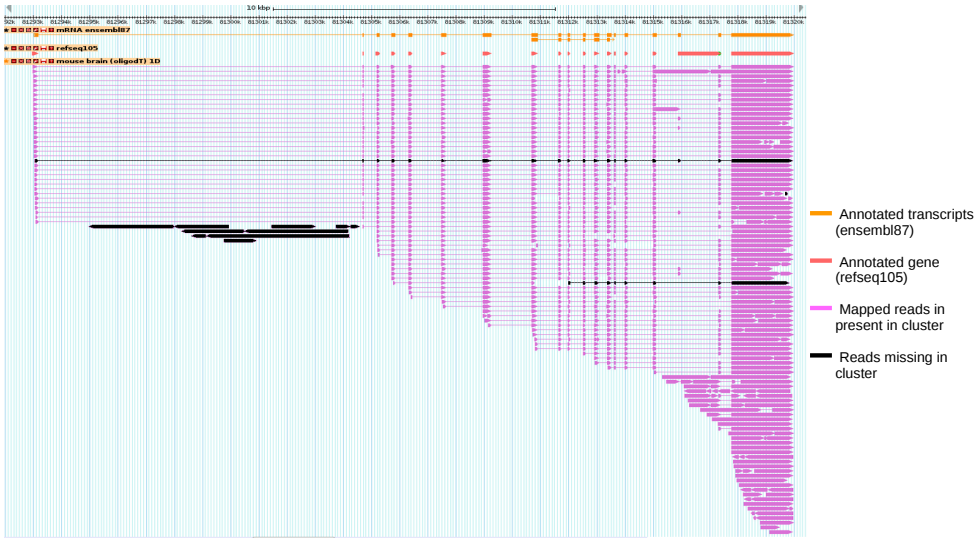

Figure 13: **Example of CARNAC-LR's output cluster in mouse.** The output of CARNAC-LR is a text file with one line per cluster, each cluster containing the read indexes. We selected a predicted cluster made of 112 reads (purple). For validation purpose these reads were mapped with BLAST on an in-house igv [195] version for mouse genome. Reads are spliced-mapped, bold parts are the mapped sequences from the reads and thin parts represents the gaps between the different mapped parts of the reads. Despite the staircase effect observed in the data, this allows to notice that several types of variants were gathered. They could all be assigned to gene Pip5k1c (chr 10:81,293,181-81,319,812), which shows no false positive was present in this cluster. 8 reads (black) present in the data are missed in this cluster. The group of 6 black reads on the left represent intronic sequences and share no sequence similarity with the others and thus could not appear in the same cluster.

set on the number of sampled values (100 by default).
We also chose to disconnect the *articulation points* of the graph to remove nodes that can be targeted as potential bridges between two correct clusters. These are nodes whose removal increases the number of connected components in the graph. Such nodes can be spotted as problematic as we do not expect a single read to be the only link between many others. Their detection can be done with a DFS in linear time for the whole graph.
Our algorithm has been also carefully designed with respect to the features of long read clustering. To obtain a $O(n.log(n))$ complexity with respect to the number n of reads, we have made the following assumption: The degree of each node is bounded by a constant, i.e. there is a limited number of transcripts that share similar exons. This ensures that the clustering coefficient of all nodes is calculated in linear time. The most complex operation is the initial sorting of nodes, first by decreasing degree value, then by decreasing $ClCo$ value, which can be achieved in $O(n.log(n))$. Moreover, since each cluster is initially built on a seed read (see paragraph 2.1.3), it intersects with a bounded number of clusters. Since the loop for making a partition from overlapping clusters is based on a scan of intersections, it is achieved in linear time with respect to the number of reads.

Approximated minimal cut The most costly phase relies on the treatment of the largest connected components. In large connected components, many clustering coefficients values are very close. Introduc-

ing a rounding factor in when computing the *ClCo* results in a neat decrease of the number of different values observed, and thus restrains drastically the number of iterations necessary for the main loop. As a consequence, the optimization only computes an upper bound of the minimal cut.

Graph pre-processing We chose to disconnect the *articulation points* of the graph to remove nodes that can be targeted as potential bridges between two correct clusters. These are nodes whose removal increases the number of connected components in the graph. Such nodes can be spotted as problematic as we do not expect a single read to be the only link between many others. Their detection can be done with a DFS in time complexity of $\mathcal{O}(\mathcal{N} + \mathcal{E})$ for the whole graph. This pre-processing is made in one pass on the initial graph.

3.3 Performances

In Figure 15 we plot wallclock runtimes of CARNAC-LR on different sizes of datasets samples from real data. For 10K, the graph characteristics were (N=4,340, E=125,172, number of connected components=403, biggest connected component=194 nodes), for 100K (N=43,588, E=435,530, number of connected components=3,970 , biggest connected components=6,010 nodes) and 1M (N=439,510, E=15,043,753, number of connected components=38,506, biggest connected component=94,099 nodes) We show a roughly linear time consumption in the size of the input in the experiment. In Figure 14 we show the gain in threading CARNAC-LR. The longest computation time is dedicated to the biggest connected component (94,099 nodes) that takes one thread. It can be seen that the memory footprint is not impacted a lot by the use of several threads.

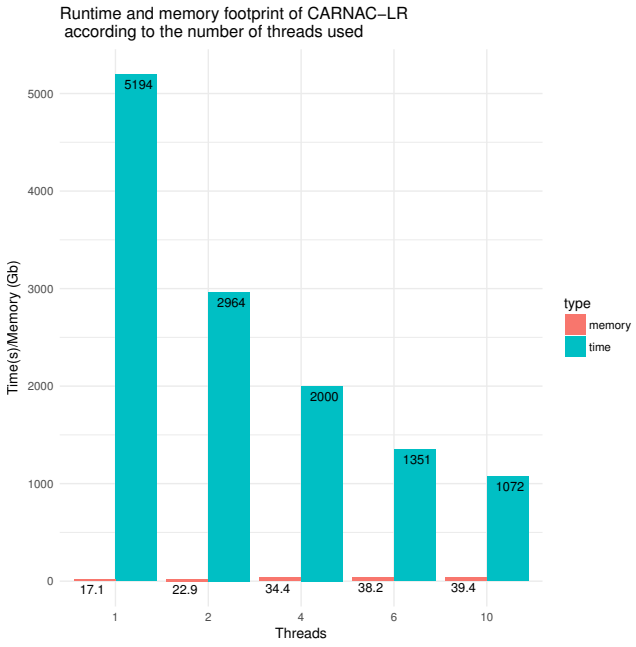

Figure 14: **Comparison of time and memory consumption and gain in throughput of CARNAC-LR on mouse dataset (1M reads) when single or multi-threaded.**

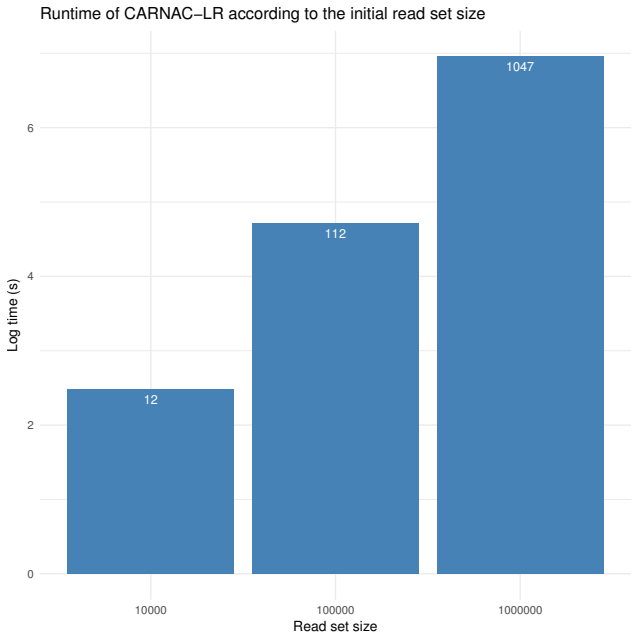

Figure 15: **Wallclock time for different sizes of datasets (10K, 100K and 1M reads) of mouse transcriptome reads (40 threads).** Time scale is log, real wallclock time values in seconds are annotated in bars.

3.4 Expected clusters on particular cases

Our clustering algorithm goal is to group RNA reads by gene. We show expected clusters on several instances in Figure 6. Constitutive transcripts lead to read that share a similar exon content and must be clustered together. Alternative splicing events or alternative transcription events lead to changes in the exon content in molecules and then in reads. If reads still share common exons, they must be put in a same cluster. However, if reads from a same gene come two very different isoforms that do not share exons, then we do not have any *a priori* information that would lead us to cluster them together. Thus they will fall in different clusters. The clustering of sequences from transcriptome reads is made difficult by the existence of multiple repeats. This first attempt to cluster RNA reads by gene is not designed to precisely assign reads from paralog genes to their original locus, we only provide first-approximation results in these cases. Our method will gather all reads from a gene family together, provided the different copies have not diverged too much and can thus be seen as a useful pre-processing step for the analysis of paralogs. At this time, it remains undisclosed which level of divergence is needed to form two separates clusters in our application.

4 Results of CARNAC-LR on several datasets

In this section we present the results of our clustering method CARNAC-LR on several instances. First, using synthetic datasets that define classic cluster resolution problems, we show it retrieve sound communities and demonstrate its interesting aspect over *modularity* based methods. Then, we include CARNAC-LR in the benchmarks already used to assess state of the art algorithms and methods to illustrate its interest on long reads in comparison to these approaches. Finally we apply CARNAC-LR on a full length transcriptome sequencing and compare its result to a mapping approach. All experiments were run on Linux distribution with 24 Intel Xeon 2.5 GHz processors, 40 threads and 200 GB RAM available.

4.1 Behavior on classic community problems

Fortunato et al. [66] proposed to test the resolution limit of community detection on a ring of 30 cliques of 5 nodes interconnected through single links. Authors. shown that *modularity*-based methods would result in a maximal resolution of clusters made of pairs of cliques. A same result is reported in Louvain's paper [21]. The Louvain algorithm finds the partition in cliques at the first level of the hierarchy and build groups of

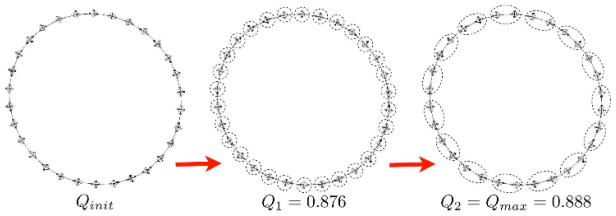

Figure 16: **The ring of cliques problem proposed by Fortunato et al.** and Louvain algorithm results, this figure is from Blondel et al([21]). The last pass of the algorithm leads to link pairs of cliques into final clusters. By contrast, our algorithm would yield results at the same definition than Louvain's first pass, where each clique is considered as a cluster in itself.

2 cliques at the second and last level. Our algorithm finds the correct partition in cliques. As our method easily found the solution to this first problem, we slightly complicated the initial example. We built a ring of 30 identical cliques like in the original paper, but we made the cliques bigger (size 7) and we put a pair of edges (instead of one edge) linking each pair of cliques (see Figure 17). The edges of the pairs involve distinct nodes from the cliques. One may expect that each original clique is found as a final cluster by methods. As a matter of comparison, we provide the resolution achieved by Louvain's algorithm on this new problem to illustrate its difficulty (see Figure 18). It cannot find the partition in cliques and moreover, the cliques are not always split at the same place. This is not a surprising result as Louvain already missed to output one clique per cluster in the original example. Contrary to *modularity*-based approaches [66, 21] CARNAC-LR successfully reported the 30 expected clusters of cliques (Figure 19).

Those instances only model simple cases, that are not representative of real data clusters that include more noise. However, they demonstrate the advantage of *ClCo* on this aspect over *modularity*.

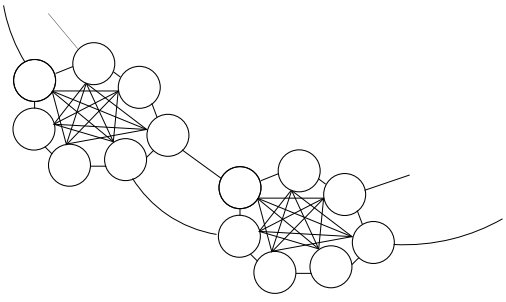

Figure 17: **We provide an example to show how 7-cliques are connected in our example design.** In total, a ring of 30 7-cliques is used.

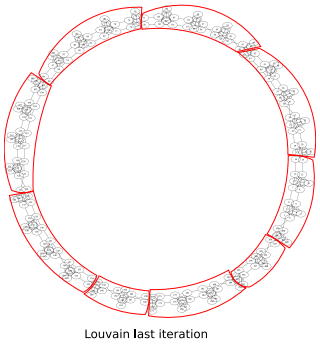

Figure 18: **We illustrate Louvain's result on this instance.** The clusters formed by Louvain are in red, spanning several cliques.

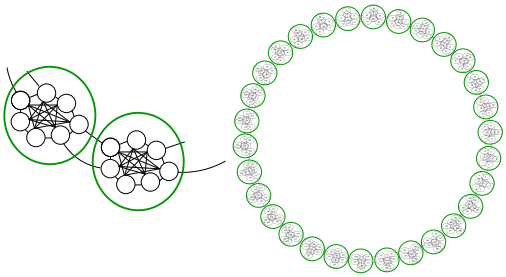

Figure 19: **CARNAC-LR result clusters on the ring of 7-cliques.** A close-up on two cliques is shown on the left. Each cluster output by CARNAC-LR is represented by a green circle. It can be seen that cliques are separated from each other as our method identifies them as independent clusters.

4.2 Comparison to state of the art

4.2.1 Comparison to algorithms for community detection

The previous benchmark is set up again, this time including CARNAC-LR. Results are presented in Tables 2.5 and 2.6. Our method has the best precision and the best overall trade-off between precision and recall as shown by the F-measure. It also has the highest Jaccard index among all tested approaches. As CARNAC-LR is conceived for general pipelines making the complete analysis of gene variants, it is important that is does not mix two unrelated genes in a same cluster. Thus our approach is more conservative than CPM, and it shows comparatively good results in any case, and furthermore needs no input parameter.

4.3 Comparison to tools for sequence clustering

In the same spirit, in included CARNAC-LR to the benchmarks comparing tools for sequence clustering. Results are presented in Tables 2.7 and 2.8. CARNAC-LR presents clear advantages in comparison to the only other tool that could compete, CD-HIT. Our pipeline is also several orders of magnitude faster than CD-HIT. Again, we highlight that such results are expected in the sense that the compared tools are not fit to ONT long reads. On the contrary, these results do not demonstrate that our pipeline would prevail in most generic cases, such as short read clustering. But they show that the challenge introduced in this chapter is better solved by our pipeline than using other tools.

	Recall (%)	Precision (%)	F-measure (%)	Jaccard index
Transitive closure	75.74	5.614	13.62	$7.3E^{-4}$
Modularity	60.70	71.16	65.51	$9.7E^{-2}$
CPM5	79.00	69.35	73.86	$3.5E^{-1}$
CPM50	49.21	89.92	63.60	$7.6E^{-2}$
Louvain	**88.58**	14.91	25.53	$1.1E^{-3}$
CARNAC-LR	65.0	**98.41**	**86.62**	**7.9 E^{-1}**

Table 2.5: Comparison with state of the art methods. The benchmark was realized on a 10K reads dataset from the mouse brain transcriptome. Recall precision and Jaccard Index are presented (see Equation 2.3, 2.1 and 2.2) to assess for the quality of communities detection. CPM5 (respectively CPM50) designates the CPM algorithm using $k = 5$ (respectively $k = 50$).

	Recall (%)	Precision (%)	F-measure (%)	Jaccard index
Connected comp.	69.04	21.76	33.09	$2.3E^{-1}$
Modularity	34.30	12.0	17.77	$2.1E^{-1}$
CPM5	55.29	29.27	38.27	$3.4E^{-1}$
CPM50	12.52	89.63	21.96	$5.2E^{-1}$
Louvain	**87.50**	1.191	2.350	$3.1E^{-2}$
CARNAC-LR	60.35	**98.86**	**74.72**	**7.1E^{-1}**

Table 2.6: Comparison with state of the art methods on a chromosome 1 10K reads sample. Reads were selected randomly from the brain transcriptome dataset. Recall precision and Jaccard Index are presented (see Equations (3,4,5). CPM5 (respectively CPM50) designates the CPM algorithm using $k = 5$ (respectively $k = 50$).

	Recall (%)	Precision (%)	F-measure (%)	Status	time (hh:mm:ss)	Memory (Mb)
CD-HIT	26.60	**99.27**	41.96	run	03:06:5	2.47
SEED	0	0	0	run	ends immediately	-
Starcode	-	-	-	error	-	-
Rainbow	-	-	-	could not be run	-	-
Tofu	-	-	-	could no be run	-	-
CARNAC-LR + Minimap	**65.0**	98.41	**86.62**	run	00:00:13	4.00

Table 2.7: **CARNAC-LR added to the benchmark of nucleic acid sequence clustering tools.**

	Recall (%)	Precision (%)	F-measure (%)	Status	time (hh:mm:ss)	Memory (Mb)
CD-HIT	11.70	**99.87**	20.95	run	03:33:2	3.11
SEED	0	0	0	run	ends immediately	-
Starcode	-	-	-	error	-	-
Rainbow	-	-	-	could not be run	-	-
Tofu	-	-	-	could no be run	-	-
CARNAC-LR + Minimap	**60.35**	98.86	**74.62**	run	00:00:16	3.99

Table 2.8: **CARNAC-LR added to the benchmark of nucleic acid sequence clustering tools, on chromosome 1 sample.**

4.4 Validation on a real size dataset

In the following we use the whole transcriptome data from mouse as well as the full "ground truth" clusters set retrieved by the Genoscope to extend our conclusions to a real size instance. We demonstrate that CARNAC-LR scales to this dataset, that is still outputs sound clusters, and then we show that it can be complementary to the mapping approach for this application that posess a reference genome.

4.4.1 Clusters goodness

In this experiment we demonstrate the quality of *de novo* clusters obtained by CARNAC-LR. We used the subset of reads that could be mapped to the mouse genome reference (501,787 reads) as a way of comparison to assess the biological relevance of our clusters. CARNAC-LR's results were computed using 43 GB RAM and 18 minutes.
The mean recall for CARNAC-LR was of 75.38% and the mean precision was 79.62%. In other words, retrieved clusters are 75.38% complete on average, and an average 79.62% portion of the clusters is composed of unmixed reads from the same gene. In order to evaluate if our method's recall and precision is consistent independently of the genes' expression levels, we computed expression bins. For a given gene, we use the number of reads of the "ground truth" cluster to approximate an expression. Any "ground truth" cluster with 5 or less reads is placed in the first bin, and so on for 5-10, 10-50 and $\geq$ 50 reads categories. Each of the four bin represents quartiles of expression, which means there is an equal number of clusters in each bin. Figure 20 presents the recalls obtained for binned expression levels and shows our approach's recall and precision remain consistent despite the heterogeneous coverage in reads.
Furthermore, we can deduce from this plot that small clusters do not bias the presented mean recall and precision, as even for big clusters (i.e. $\geq$ 50 expression bin) that are more prone to lose some reads, these metrics remain high.

4.5 Complementary of *de novo* and reference-based approaches

4.5.1 Intersection and difference with the set of mapping clusters

The mouse dataset is useful as it provides a "ground truth" for validation. However, it does not totally illustrates the interest of CARNAC-LR since this is not a real *de novo* application. Still, it is a good example to show that our *de novo* method can also be seen as an interesting complement to mapping approaches in the case a reference is available. As it does not rely on any reference information, our approach putatively yields different results than classical mapping approaches. Thus we investigate the differences between CARNAC-LR and the Genoscope mapping approach. This time we ran CARNAC-LR pipeline on the full

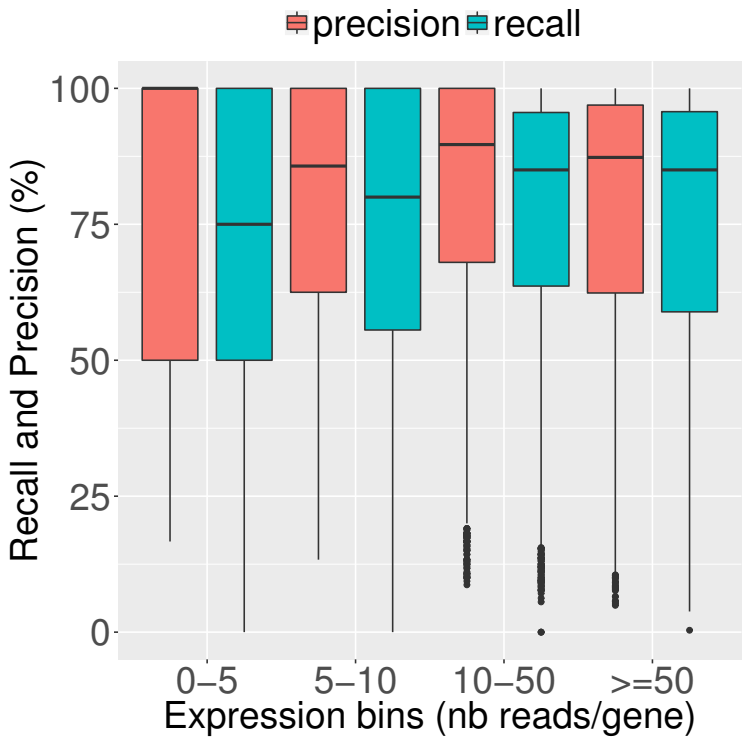

Figure 20: **Assessed mean recall and precision of CARNAC-LR+Minimap.** They were computed on mouse reads using clusters found by mapping on a reference as a "ground truth" (see Equations 2.1 and 2.2). Expression bins are computed from quartiles of expression predicted by mapping and represent the number of mapped reads by gene. Mean precision and recall over all clusters falling in theses bins were then calculated.

mouse brain transcriptome dataset (1,256,967 reads), including reads that were processed but unmapped or filtered out using the mapping approach. We compared the intersection and difference of results of our approach and mapping. CARNAC-LR+Minimap pipeline took less than three hours (using 40 threads). In comparison, the "ground truth" clusters took 15 days to be computed (using up to 40 threads). Our approach was able to place 67,422 additional reads that were absent in the mapping procedure. It resulted into 39,662 clusters. These clusters fall in two categories (i) *common clusters* with a mix of reads treated by our approach and/or processed by mapping, or (ii) *novel clusters* that contain reads treated by our approach or mapping exclusively. Each approach performed differently on these categories.

Common clusters For category (i), mapping complemented many common clusters with small amounts of reads left aside by our approach. As some reads are processed by mapping, a recall and precision can still be calculated using mapping as ground truth. We computed recall and precision based on the read fraction of clusters that could be compared with mapping. They are quite similar compared to the values obtained in the previous section (75.26% and 79.30%). This demonstrates that CARNAC-LR efficiently used the supplementary connectivity information despite the addition of potentially noisy reads.

Novel clusters Conversely CARNAC-LR shows a better ability to group reads unprocessed by mapping into novel clusters (Figure 21). CARNAC-LR output 824 novel clusters (17,189 reads) of category (ii) containing at least 5 reads. In order to evaluate the relevance of these novel clusters, we remapped reads

a posteriori, separately for each cluster, on the reference genome using a sensible approach (GMAP [251] version 2015-09-29). This operation took approximately 10 hours (using 4 threads). 19.68% of mapped reads were assigned to the MT chromosome, then chromosome 11 represented 10.85% of the reads, and other chromosomes each less than 10% of mapped reads. A third of the reads were multi-mapped (36.7%). However, on average, for each cluster 98.89% of the reads shared a common genomic locus. This is consistent with the expected results of the clustering for reads containing repeats or paralog regions (Figure 6). Finally, 5.7% of the clusters contained exclusively reads mapped at a single locus. All of them could be assigned to an annotated gene. Thus even if a reference was available, our approach was able to retrieve *de novo* expressed variants of genes that were completely missed by the mapping.

Correlation of expression levels Another way to look at these results is two consider the number of reads predicted by each method as the gene's expression, and to compare expression levels predicted by our approach and by mapping. ONT sequences start to be shown to qualify for transcript quantification in [168]. We shown that, despite the previously described differences, they are highly and linearly correlated, with a Pearson correlation coefficient of 0.80 (Figure 22).

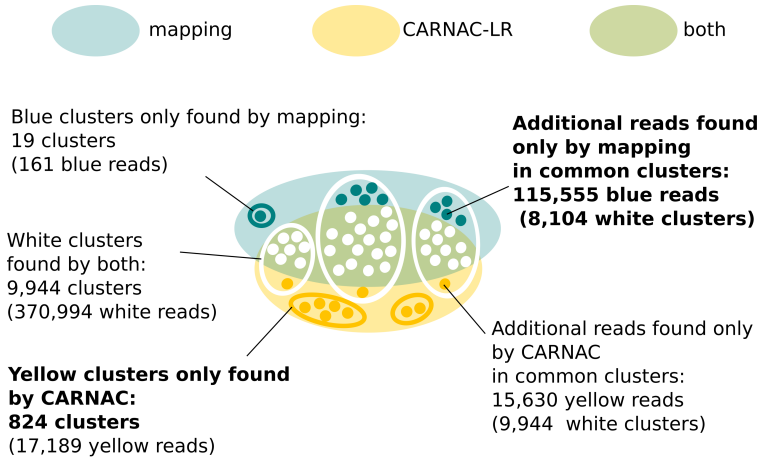

Figure 21: **Complementarity of CARNAC-LR and mapping approaches.** Only clusters of size ≥ 5 are represented. Mapping complemented common clusters a with a mean 13 reads per cluster in 90% of clusters. CARNAC-LR's supply was tenfold lower with a mean 1,3 read added to 100% of common clusters. On the other hand, CARNAC-LR retrieved 15 fold more novel cluster than mapping.

These results demonstrate that CARNAC-LR pipeline enables to retrieve clusters per expressed gene in a whole transcriptome sequencing using ONT reads. Clusters are sound in comparison to more conservative mapping approach, and the method can be helpful even when mapping is possible since it has the ability to fish back reads lost by mapping. It also presents drawbacks, such as its recall that could benefit from improvement. To that extent, we will present ideas in the discussion chapter.

5 Discussion

In this chapter we mostly presented CARNAC-LR at work on a mouse transcriptome. In order to further validate our approach we plan to apply it on more diverse datasets. Larger datasets could bring practical limits in memory and time for our approach. CARNAC-LR has been conceived with very noisy reads in mind. The noise in reads is propagated to the graph, hence we payed attention to design our clustering

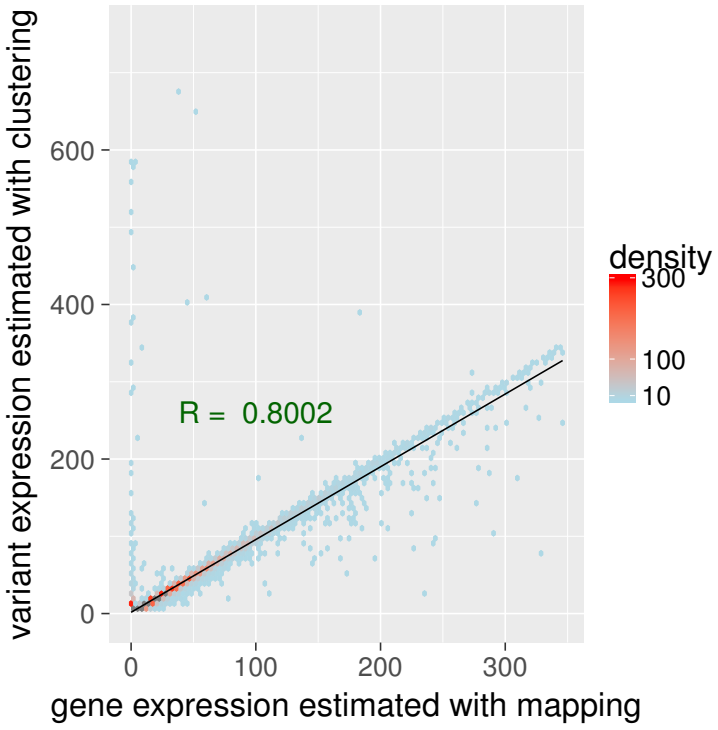

Figure 22: **Comparison of clustering and mapping approaches.** Comparison and correlation of expressions levels. Gene's expression can be inferred by counting the number of reads by gene. For each gene we counted the number of reads retrieved by mapping and we compared it to the number of reads reported by our pipeline and validated by mapping. We computed the Pearson correlation coefficient between the two (in green). Density is the number of points counted in a colored region. Despite a few outliers, we can see a strong linear correlation between the two expression levels estimations (plotted in black). Seven outliers above 750 on Y axis (up to 3327) are not shown.

method to deal with it. In the future, we might be able to determine more precisely and with more accuracy the reads that belong together. Such possibility could come from the reads' error rate that will decrease, or from more RNA-dedicated tools such as our Long Read Connector project. In this case, it is likely that the similarity graph will be cleaner, and seeking connected components might be sufficient to retrieve each gene's expressed transcripts. If the error rate decreases, it will also be easier to separate gene families. Notwithstanding, advantages and limits of CARNAC-LR algorithm as a general method for the discovery of communities, aside from the transcriptomics context, still deserve to be better understood, as suggested by the results on the ring of cliques. CARNAC-LR remains to our knowledge the only method to *de novo* cluster reads by gene using ONT. An actual limitation is that *de novo* exploitation of reads remains difficult, even after clustering. In fact CARNAC-LR is meant to be followed by other computation steps to decipher isoforms in each cluster. In Chapter 4, CARNAC-LR will be the prerequisite to partition the datasets into clusters on which a method to reduce noise and detect alternative events in reads is applied.

Chapter 3

Correction of long, spurious reads

Previous chapters led to 1-find similar reads by comparing them pairs 2-gather reads by gene using clustering. Tackling the last announced goal: read correction to obtain isoform sequences will be covered by this chapter and the next one. This is an important matter since at the moment, long reads cannot be used as is, in particular accurate protein prediction is impossible given the errors. Even in cases where references are available, many reads fail to be associated to the annotated gene they originate because mapping tools cannot deal with the error rates that are sometimes superior to the variability found in the reference. In this chapter, we essentially demonstrate that current correction methods cannot be applied for our problem. We start by presenting short-reads correction methods and explain why they were replaced by new paradigms for long reads that we present in the first place. We then expose the main contribution of this chapter, a generic tool for long read correction methods assessment. This tool aims at helping to find out limits of current correction methods. It embeds original algorithmic choices that will be found again in Chapter 4. We then demonstrate that published correction methods do not perform well on RNA and we identify their pitfalls. Thus this chapter prepares the task of designing a specific approach dedicated to obtain high-quality sequences from transcriptomic long reads.

1 Background

1.1 Short read correction

1.1.1 Methods

The most prominent strategy used in NGS reads correctors is based on k-mers spectra. A k-mer spectrum is a discrete distribution of the number of occurrences of k-mers in a given dataset, or a pool of datasets. k-mers spectrum techniques have the advantage to be quicker than previous methods based on alignment [203] and more memory efficient [256]. Thus they are more scalable and were largely adopted in the context of NGS. They are well adapted when an experiment coverage is homogeneous (i.e. most of the genomic sequencings). The intuition is that erroneous k-mer will be significantly less frequent than genomic k-mers in a dataset if the coverage is high. Thus a threshold can be determined using the occurrences distribution, that is typically multi-modal (for instance in haploid species, there are a first peak of k-mers that appear once or a few times, likely to be errors, then a second peak which amplitude depends on the sequencing depth). On large genomes, storing genome wide distribution of k-mers for k-mer counting can be memory consuming, as pointed in Chapter 1. Recent tools such as Lighter [223] rely on efficient data structures (Bloom filters) and represent real advantages over previous methods such as Musket [136]. For correction, methods try to correct as many erroneous k-mers of the reads as possible by replacing them by solid ones using heuristic strategies to make the least possible operations ("solid" k-mers are introduced in Chapter 1). A documented limitation of these methods is the possible loss of rare genomic k-mers. This can happen in scenarios of local coverage biases due to variability or sequencing biases. For same reasons, metagenomic or transcriptomic datasets that harbor heterogeneous k-mers distribution cannot be accurately corrected directly with these techniques and require more sensitive approaches.

Some methods were dedicated to RNA and uneven coverage. Rcorrector is a method that was developed for RNA-seq data correction [222], based on the De Bruijn graph and uses the graph properties to store efficiently the k-mers of the reads and to localize errors thanks to graph topology. Erroneous k-mers provoke branching regions in the graph, the different branches support either successive erroneous k-mers in vertices or real sequences. However, alternative variants also lead to these patterns. The difference between errors and variants in the graph is that erroneous k-mers are expected to be found with relatively lower coverage than real sequences. Thus Rcorrector applies local relative filters to distinguish error from expressed sequences. These filters apply on first k-mers of a branch (branches paths with a single predecessor vertex), that lead to branch pruning their if their coverage falls below the current threshold. Reads are then corrected using the path that implies the less modifications on their sequences.

Rcorrector introduces to some extent that assembly and correction are somehow related. Recently, graph methods for correction have been shown to provide better results than k-mer spectrum methods [133]. How-

ever, except from Rcorrector and a recent proposal [133], graph strategies for correction are only employed in a few methods on long reads [202, 204]. In these methods, sequences from unitigs in a De Bruijn Graph are non ambiguous assemblies of several contiguous solid k-mers that can be used to correct reads.

1.1.2 Consensus and correction

Read correction is necessary only because of actual technology pitfalls and is not responding to any biological question. However, this problem is interestingly related to many others in sequence bioinformatics. Before the advent of k-mer spectrum techniques, methods proposed to correct reads using multiple alignments [203]. The alignment of several reads on a same genomic interval allowed to define a consensus (majority) version of the base for each position of the alignment. For instance, a voting scheme that counts the base that occurred most frequently at each position of the alignment can be used. Sequences extracted from graphs can also be used for consensus. Using graphs, error handling is seen as the replacement of erroneous bases by consensus bases. Correction is not realized strictly speaking since the read content is not modified but the errors are removed.

Thus, we can say that a correction is the action to modify bases in reads to enhance the read quality by trying to get closer to the genomic sequence. A consensus is built from a set of sequences and is a single sequence of bases from the set that maximizes a given score at each position (for instance, the most frequent bases at each position). Consensus is used for correction. Well corrected reads should yield better consensus (if no bias is introduced).

Contrary to correction that is seen as a primary step where all reads can potentially be modified by the correction tool, polishing is defined as an improvement of sequences (often contigs) quality as a final touch after other steps (for instance, assembly of reads into contigs). Polishing resorts to consensus, less-error prone sequences such as contigs as "backbones" to correct reads, or aims at correcting contigs themselves. Even if it shares methodological similarities with correction, polishing is harder to evaluate since it highly depends on the consensus sequence input. However polishers can be used for read correction, and some tools present both aspects in their implementation [34]. Polishing algorithms relate more to the goal presented in Chapter 4. In the following we will deals with correctors only.

1.2 The challenge of variety of error rates and profiles in long reads

1.2.1 Why previous methods do not work

k-mers spectrum rely on the property that a sole mismatch in a read will provoke a controlled number of erroneous k-mers. Indels, as well as close (separated by less than k bases) errors are more difficult to correct using k-mer spectrum, since it is difficult to correctly assess how many k-mers are impacted in the sequence. Moreover, they use k-mer sizes large enough to ensure that k-mers are rarely repeated on average in the genome while having higher frequencies than erroneous k-mers. In the case of TGS with more than 10% errors, the previous methods no longer apply. It is way more difficult to separate erroneous from genomic k-mers using the frequencies when using long k-mers, since even genomic long k-mers are expected to be rare. Moreover, the sequencing depth obtained with NGS is at the moment hard to reach with TGS, giving less leverage to differentiate genomic from error-prone k-mers. On the other hand, small genomic k-mers are more repeated across the genome, and small k-mer sizes allow more sequencing errors to pass the solidity threshold. Finally, non systematic errors such as those found in homopolymers in ONT suggest that a third-party technology is needed to achieve a satisfying correction level with these reads.

1.3 Long read correction methods

1.3.1 Using short reads

Hybrid methods are named after their employment of NGS reads to help long read correction. Hybrid correction was the first correction paradigm to emerge for long reads, since the first experiment only produced low coverage for long reads, with particularly high error rates. In tools such as Colormap [85], Nanocorr [76], Proovread [84] or PbCR [108], short reads are directly aligned to long reads, there sequences being inserted in place of the long read alignment locus. More recent methods integrate strategies to elongate the corrected zones and to facilitate alignment. Several methods rely on contigs made of short reads. HALC [11] aligns the long reads on the short reads contigs. LoRDEC [202] and Jabba [155] align long reads on unitigs in De Bruijn graphs built using short reads. NAS [139] additionally uses the long range information of long reads to guide short reads assembly before to correct with the contigs.

1.3.2 Self-correction

Self-correction methods no longer use NGS reads for correction. One of the first method that proposed long read correction using only long reads sequences was LoRMA. LoRMA [204] follows the LoRDEC paradigm by constructing a De Bruijn Graph on the whole set of reads, then uses the graph's unitigs as consensus that can correct long reads sequences. PBDAGCon [34] also relies on graphs but is not based on a DBG. Instead, it mimics multiple sequence alignment partial order strategy, that will be presented in the next chapter. In short, performing pairwise alignments, it builds a directed acyclic graph representing the reads aligned to each others and extract a heaviest path (i.e. globally supported by most reads) in this graph as a consensus to correct reads. Falcon [35] and Canu [109] use a similar strategy to extract consensus from a DAG's path. MECAT [252] works in a different way, it starts with pairwise alignment of long reads and has an hybrid use of voting scheme (in regions containing not too many errors) and of DAGs such as in PBDAGCon. As pointed in Daccord's manuscript [231], main drawbacks of long reads correction methods are the corrected sequences fragmentation and low rate of corrected sequence compared to the input. Certain correctors make the choice to exclude reads that could not be corrected from the output. They can also report only the corrected parts and discard ambiguous subsequences, or fail to reconstitute the whole long read structure when it is mapped on complex regions of a DBG. Both hybrid and self correction methods thus often propose a trim and untrimmed version of the corrected sequences.

Self-correction seems to be proven to give better results than hybrid correctors as soon as the long read coverage is sufficient (over 30X) through the benchmarks supporting their respective publications. This can be explained by difficulties to align short reads on repeated sequences covered by long reads, or difficulties in the alignment due to the GC bias in short reads. It also seems that self-correction leads to less fragmented corrected sequences. However, such statements lacks a global study that goes over the sum of individual publications. The evaluation tool presented in this chapter aims at filling this hole and providing support for benchmarks.

1.3.3 Correction as a bottleneck in pipelines: works toward efficiency

Because it usually includes computationally expensive steps of alignment, long read correction is frequently reported as a bottleneck in long reads pipeline. More efficient implementation using SIMD (parallel computing) were proposed [239]. Very recent methods started to tackle this issue using new paradigms. For instance MECAT proposes very fast and memory efficient strategies for read mapping and correction. Daccord (unpublished) uses an assembly strategy directly on the long read sequences to perform their correction. Since it performs only assembly on short sets of k-mers from different intervals of the read, it reduces its time and memory consumption. However, Daccord still relies on a third-party prior step of precise read alignment that remains time consuming.

2 Evaluation of long reads correction methods

A large panel of methods is already available to correct long reads, with a variety of algorithmic strategies. We are lacking tools to correctly assess the pitfalls of the different methods on various sequencing scenarios, including transcriptomics. In the following we introduce a new tool, ELECTOR, dedicated to assess long read error correction methods. We also demonstrate that current correction method only partially achieve the task of correcting RNA long reads.

2.1 State of the art of correction evaluation

We start with a disclaimer: as no corrector is at the moment adapted to RNA, this part is based on genomic long reads. Thus, evaluation tools for correction methods will be presented using genomic experiments. In the end of the section we show that ELECTOR could be easily used on transcriptomics experiments as well.

We pointed the need for more comprehensive assessment of long read correction tools, both because all methods do not optimize the same feature and some can be more adapted to some genomes/sequencing experiments than others, and because such tools are useful to develop new correction methods. In the following we present an implemented tool dubbed ELECTOR (**E**vaLuation of **E**rror Correction **TO**ols for long **R**eads), developed for general long reads correction methods assessment. It is based on an original use of multiple sequence alignment, and more generally on algorithmic concepts that are used besides in this thesis work. This work was realized in the context of MASTODONS C3G project, and is also a part of Pierre Morisse's Phd thesis. Pierre contributed to ideas and participated in the implementation, in particular he is responsible for adding remapping and assembly assessment modules of our tool, and performed many of the experiments. We provided the algorithmic ideas on which the main metrics computations are based. This work is yet non published, but ELECTOR is available[1].

2.1.1 LRCstats

Correction methods quality assessment is usually based on the portion of corrected reads that could be remapped on the reference [256, 155]. Despite being interesting, this information remains incomplete, in particular it likely not to take into account poor quality reads or regions difficult to map to. In LRCstats [114] was proposed the first pipeline to allow benchmarking of long read correction methods. LRCstats aims at assessing the success of these methods to disminish the error rate in reads. It relies on read simulators (SimLoRD [225] or PBSim [170]) to produce erroneous reads with controled properties. It uses the the SAM alignment of these reads to the reference to retrieve the original sequences. LRCstats takes the corrected version of each read (after a pass of a correction method), the original version deduced from the alignment and the simulated reads to produce a three-way alignment. In this scheme, two pairs of sequences are aligned. The alignment is done with regular dynamic programming. The read is aligned to the original version and the corrected version is aligned to the original version as well. Then statistics are computed and collected from the comparison of the three versions of the sequence.

2.1.2 Current lacks to identify correction methods pros and cons

La et al. presented results on Pacific Biosciences data for hybrid correction tools using LRCstats. At the moment, no work presented a benchmark focused on self-correction methods. We also lack hindsight about the results obtained on nanopore sequences, while PacBio reads have more studies dedicated since the technology is older and more stable.

LRCstats provides reads error rate before and after correction as well as the detailed counts of every error type. However, reads error rate after correction is not a good indication of the corrector's behavior. For instance, there is no clue to be found about the putative insertion of new errors by the corrector by looking only at the error rate . This type of information is extremely useful to developers, and can be crucial

[1] `https://github.com/kamimrcht/benchmark-long-read-correction`

according to the type of biological question the reads are sequenced for. Thus, additional metrics such as recall and precision should be given as complements in order to understand correction methods pros and cons.
Aside from information that could be missing, it can be interesting to understand how a correction method performs on a genome that has particular features (size, repeats, ...). It can also be relevant to test reads with different properties. However, in experiments involving large size datasets and/or very long reads, the evaluation can be an order of magnitude longer than the correction step itself, which is not a desired property. Thus, it can be difficult for someone to correctly evaluate a correction method for a particularly large genome, or in the context of very long reads (over the Mb to the Gb recently). Both deep coverage and very long reads are though extremely useful. Long reads sequencing experiment is shown to have a log-normal read length distribution, thus high coverage helps the correction of the longest reads. Very long reads themselves are the most prone to untangle difficult assembly patterns provoked by repeats or heterozygocity. Incidentally they have a positive impact on the contigs mean length.

2.2 New methodological approach to evaluate correction

Given the absence of methods able to propose precise and accurate metrics to better render the efficiency of correction methods for long reads, we proposed an approach that could both provide these metrics and be used even on large genomes-scale experiments. Contrary to LRCstats that was based on two read pairs alignment, our approach is based on a multiple alignment of triplet of sequences. Our method compares at once the corrected, uncorrected and genomic versions of each read. We consider these three versions exist for each read, and are stored in three FASTA or FASTQ files (perfect, uncorrected and corrected files). All three files have the same number of lines and reads are sorted so that the versions of the sequences correspond to each other and appear in the same order. The three files are read to compares triplet of reads of same index in each file. We then obtain one alignment per triplet that is used to compute different metrics. The multiple alignment is realized using an efficient heuristic that combines anchoring and partial ordered alignment, that we call *seed-MSA*. First we describe the simple multiple alignment procedure, then we add the anchoring strategy.

2.2.1 Multiple alignment of triplets of sequences

Multiple sequence alignment (MSA) is not strictly defined, but can be seen as the task of rewriting more than two sequences contained in lines of a matrix so that conserved bases are in a same column. This means adding gap characters when necessary (i.e create columns with homogeneous content). An example is shown in Figure 1. Base-wise recall and precision of a method can be computed only if corrected, uncorrected

```
>0                MSA:
ACTGCTTCT           0  ...   9
>1                0 ACTGCTT-CT
ACTACTTTCT        1 ACTACTTTCT
>2                2 -CTG-TT-CT
CTGTTCT
```

Figure 1: **Toy example multiple sequence alignment of three sequences.** The MSA can be represented using a matrix where rows are sequences and columns are bases/gaps. Gaps are added in each sequence when necessary to group conserved bases.

and perfect version of the reads can be precisely compared at each position. MSA has the advantage to well group bases conserved across sequences in the alignment and to have more power than pairwise alignment to detect weakly conserved bases. Many algorithms for MSA exist, mostly heuristics since optimal solutions often suffer from intractable computational time when sequences are long, very different or numerous. Here we will rely on an algorithm where MSA in not represented in a matrix but in a DAG. This strategy is called partial order-MSA (PO-MSA) [121]. This algorithm comes with an implementation (poaV2). For our purpose we use a modified version of poaV2 that works with triplet alignment. The graph representation has the advantage to solve certain inconsistencies of algorithms based on matrix by representing differently the information of the alignment. The detailed algorithm is went through in Chapter 4, as it is the core method used to perform sequences consensus. Here we work with a particular case where only three sequences are aligned. Important messages to keep in mind is that from a given graph, a unique MSA profile can be extracted. Each nucleotide is a vertex and arcs represent consecutive bases in a sequence. The method guarantees to conserve the order of the bases in a given sequence and each sequence can be retrieved in the graph. Gaps in the alignment are directly represented in the graph topology. In short, the alignment process is the following:

1. The reference version of the read is used to initiate a linear graph (Figure 2 1))
2. The corrected read is then aligned to the reference using classic global dynamic programming (Figure 2 2))
3. The initial graph is completed with new nodes from the corrected reads. When possible (i.e. when matches occur) the graph is simplified by merging matching identical nodes (Figure 2 3))
4. Then the uncorrected version of the read is aligned against the graph using a generalization of alignment procedure detailed in Chapter 4 (Figure 2 4))
5. Again the news nodes are added to the graph and simplifications are made
6. We apply a topological sort on the DAG, so that positions are defined in matrix columns for each vertex of the graph (Figure 2 5))
7. A matrix MSA profile is output (Figure 2 5))

The rationale besides the choice of aligning first the corrected version with the reference version is that this first alignment is expected to be the most accurate. Since the read is corrected, this pair must the most similar of the three possible pairs. The alignment of two most related sequences is then more reliable and provides a better overall result [60]. Even if in our case the three versions in a triplet are expected to be alike most of the time (thus the order has little impact), this chosen order should provide at least slightly better quality alignments than another order. This alignment scheme can be generalized to any triplet of sequence alignment in any order. However, the approach we present in the following is meant to be used for triplets of sequences that come from the same genomic region.

2.2.2 Recall, precision, correct base rate

We collect the output multiple sequence alignment for each triplet of sequences. Once the MSA is computed, we have a base-wise information of the differences and similarities in nucleotide content for each three versions of a sequence. Insertions or deletions are represented by a dot "." and a corresponding nucleotide `a,c,t` or `g` in the sequence containing an insertion relatively to the two others. For each position in the MSA, if the reference, uncorrected and corrected bases are equal, the result is correct. Figure 4 presents how false negatives, false positives and true positives can be inferred from the MSA. If only the reference and uncorrected bases are the same but the corrected base is different, a new error was introduced by the corrector (false positive base). If the reference and uncorrected bases are different, there is an error to correct. In this case, if the corrected and reference bases are the same, the error was corrected (true positive base), on the contrary if the corrected and uncorrected bases are different, a new error was introduced by the corrector (false positive base). If there is an error to correct but the corrected and uncorrected bases remain the same,

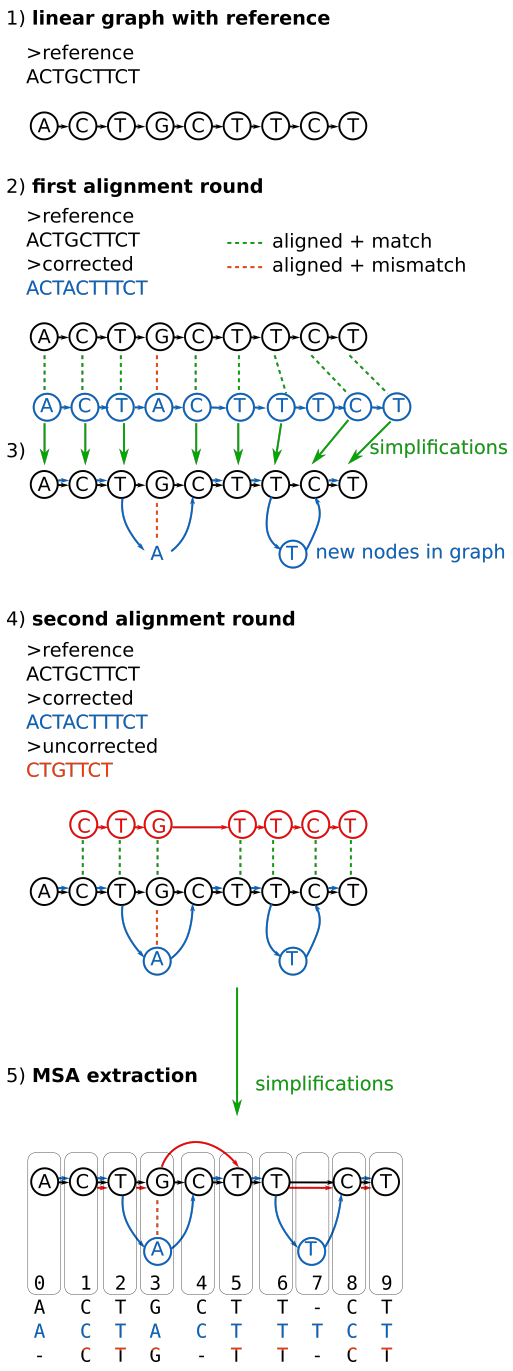

Figure 2: **Toy example of multiple sequence alignment of triplet of read versions used in ELECTOR.**

the error was not corrected (false negative base). From these observations we can compute false negative (FN), true positive (TP) and false positive (FP) counts for each read, as detailed in the Algorithm 3. We consider all bases in the MSA and count the number of bases that are equal between reference and corrected version, and divide this count by the number of bases in the reference sequence to obtain the correct base

```
1  Algorithm: Getting metrics from triplet MSA
   Data: Matrix of multiple alignment for a triplet
2  FP = 0;
3  FN = 0;
4  TP = 0;
5  foreach Column C of the MSA matrix do
6      trueNt = Column[0];
7      uncorrNt = Column[1];
8      corrNt = Column[2];
9      if not (trueNt == uncorrNt == corrNt) then
10         if trueNt == uncorrNt then
11             if uncorrNt != corrNt then
12                 ++FP;
13         else
14             if trueNt == corrNt then
15                 ++TP;
16             else
17                 if uncorrNT == corrNt then
18                     ++FN;
19                 else
20                     ++FP;
```

Figure 3: **Counting false positives (FP), false negatives (FN) and true positives (TP) in the MSA matrix.** These values are computed for each triplet of sequence that has its own MSA.

rate. Recall and precision are also computed:

$$\text{recall} = \frac{TP}{TP + FN} \quad \text{precision} = \frac{TP}{TP + FP} \tag{3.1}$$

2.2.3 Seed-MSA approach

General idea The time complexity of the PO-MSA increases linearly with the average number of branches in the DAG. Its global complexity is in $\mathcal{O}(nMN)$ with n the average number of predecessors per vertex in the graph, M the length of the sequence to be aligned and N the current number of vertices in the graph. Thus, very long reads can induce longer computation time, because in addition to their length they imply more errors and branches in the graph. We propose a seed chaining strategy inspired from Mummer's [50] or Minimap's [126] approaches, in order to divide the multiple alignment problem into smaller problems and thus, reduce the time footprint of our approach. See Figure 9 for an example. For each triplet, we compute seed k-mers that have the following properties: 1-they appear in each of the three versions of the sequence, 2- they are not repeated across any of the versions of the sequence. Once these seeds are detected, a memoization procedure is applied to find the longest subchain of valid seeds. The procedure is similar to what is used for instance in Minimap2 [127], to which the reader can refer for details.

```
positions              0123456789
genomic sequence       TCGT.ATC..
read with errors       ..GCTA.CC.
corrected read         TCGT.ATCCC
```

Figure 4: **Toy example MSA of the three versions of a sequence.** Let's first compare the genomic sequence to erroneous read. Errors in the read are reported in bold. Substitution such as in position 3 (T/C) are aligned in the MSA. An insertion with respect to the genomic sequence (position 4, 8) opens a gap in the first line. A deletion with respect to the genomic sequence (positions 0,1,6) opens a gap in the read line. The corrected read can contain: positions that must not be corrected and that are not corrected (black characters), positions that must be corrected and that have been corrected (green characters), positions that must be corrected thought that have not been corrected (orange) and positions that are inaccurately modified by the corrector (red). In the corrected read line, red positions are false positives, orange positions are false negatives and green positions are true positives. It is then possible to compute a recall and a precision.

We give an outline of the procedure in Figure 6. It ensures that seeds appear always in the same order in all three sequences. The seed k-mers have a fixed size k that is adapted according to the current observed long error rates, i.e. 9 to 15 nucleotides. These seed k-mers are ordered according to their positions in the three sequences. Thus, they delineate positions where each version of the read have an exact match of k nucleotides. We then consider that subsequences that are extracted from the same duo of seeds in each version of the read come from the same genomic location. This way we divide the global multiple alignment problem is smaller problems separated by regions of exact matches.

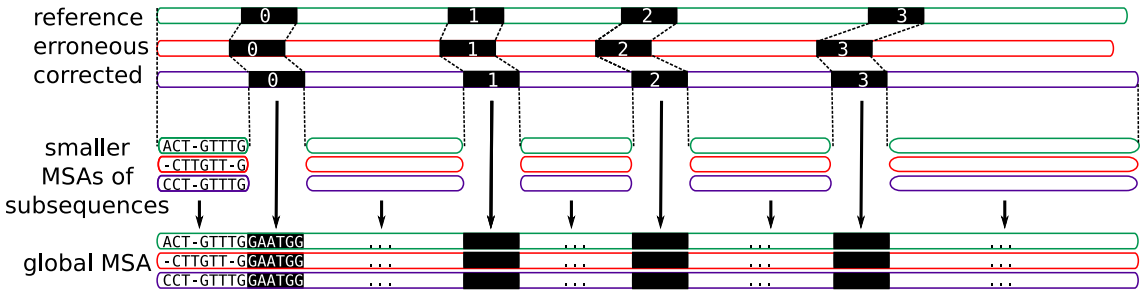

Figure 5: **Seed strategy to compute a multiple sequence alignment for a triplet of reference, uncorrected and corrected versions of a read.** Instead of computing a multiple alignment on the whole lengths of the sequences, we rather divide this problem in smaller multiple alignments. As each version is different, in order to decide were to start and end alignments we find seed k-mers (in black) that are local exact matches between the three sequences.

k value choice using multi-k strategy The optimal k size to obtain small subsequences can be difficult to predict as it depends of the initial read size and its error rate. Very long read increase the chance

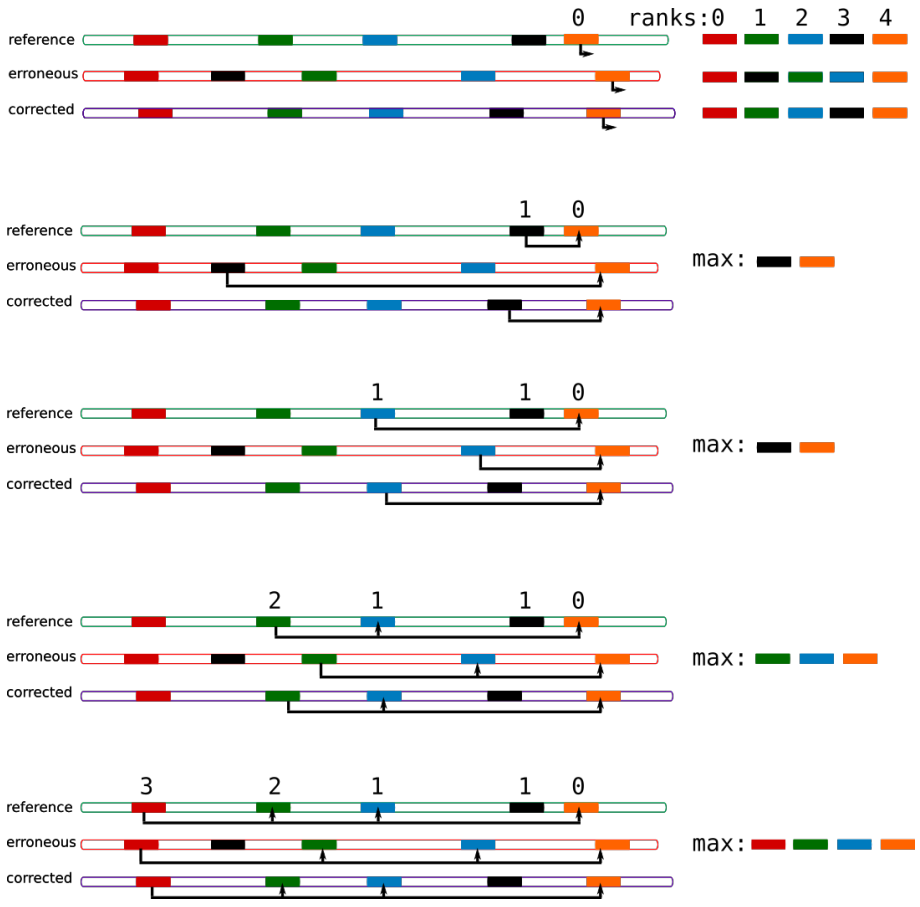

Figure 6: **Valid k-mers are shared in all three sequences and not repeated in any of them.** In this example, all colored k-mers are valid k-mers according to our seed definition. Valid k-mers are ranked in each sequence. We start with the last k-mer (orange) on the reference. The chain starting with this k-mer cannot be elongated (it is then marked with 0). We select the previous valid k-mer (black). The chain starting with this k-mer can be elongated only with the orange k-mer in all three sequences (it can be elongated with blue and green only in the erroneous sequence). We mark the black k-mer with 1. Then the maximal sub-chain length is 2. We iterate this procedure for all valid k-mers in decreasing order on the reference sequence. The maximal subchain length is not increased starting by the blue k-mer. It is increased with the green k-mer and increased again with the red one, it is then maximal. Note that a non iterative algorithm that would not start by the last k-mer would lead to the same result but is less optimized. If we obtain several maximal longest sub-chains, the first encountered is kept.

to see a k-mer repeated, that hence cannot be a seed (condition 2). Longer k-mers (around 15 nucleotides) are less prone to be seen several times in long sequences, but are harder to find in the three sequences at a time (condition 1), especially the error rate is high. Shorter k-mers (up to 9) are easier to be found in common, but are more likely to be repeated. As it is difficult to *a priori* set a k-mer size, we designed a quick iterative strategy that tests for several k value to choose the most adequate for a given triplet. We chose

to minimize the maximal interval between two seeds as the MSA algorithm bottleneck is expected to be the longest sequence. We start at the maximum k value (15), compute seeds and store the size of the maximum interval between two seeds (the maximal interval being the maximal interval observed for all intervals in the three sequences of the triplet). We decrease k and keep the new maximum interval size if it is smaller than the previous stored. If at some point the maximum interval size gets larger than the stored value, we stop the iterative process. Thus we keep a local minimum, that is mechanically not likely to be reached again by decreasing k. Indeed, if k-mers of a certain size are repeated and prevent to find seeds, smaller k-mers will be included in these larger k-mers and repeated too. We can test k values until 9 and keep the size that minimizes the maximal interval between two seeds. Figure 7 presents an example of this strategy.

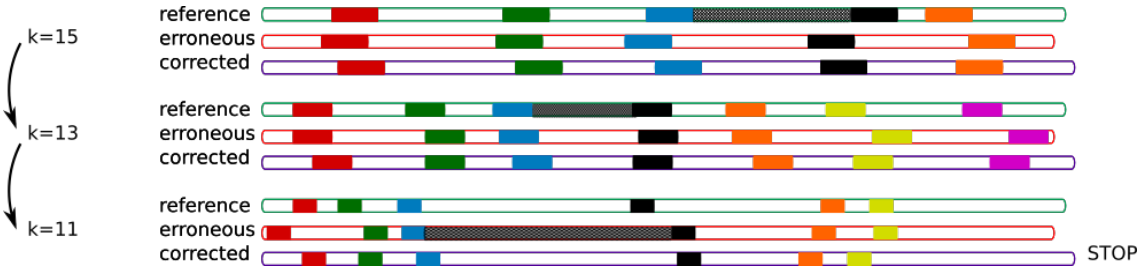

Figure 7: **Multi-k strategy to determine the best k size for intervals.** We compute seed k-mers of size 15 for a start. The maximal interval are represented in gray. The maximal interval size is recorded. Then at k=13, the maximal interval size is smaller than the previous recorded, thus it replaces the previous. This comparison is made for k values until 11. 11-mers are more prone to be repeated across the sequences, it is then harder to find close seeds. The maximal interval is then longer than the previous stored, we stop the procedure. In this example we would thus keep k=13.

Getting sequences chunks Once a size is fixed for the seeds, we start at the leftmost seed and wait for the next seed to be spaced by at least a distance that is a parameter p (in practice 20 nucleotides) on the reference sequence. Once the pair is found, we search for the next seed at at least a p distance and so on. These final seeds are reported in the two other sequences. We extract all triplet subsequences between two consecutive final seeds, as well as the left and right flanking sequences if they exist.

Reconstitute the MSA The sequences chunks are aligned with the PO-MSA algorithm, the alignment matrices being substantially smaller. Then, the seeds are replaced between each small MSA result to obtain a MSA at the scale of the whole sequence (see Figure 9).

2.2.4 Split/trimmed reads

Most of the correctors output split or trimmed corrected reads. Trimmed reads are reads whose one or both ends are missing. Split reads are corrected by parts, each part that could be corrected is output separately. In this latter case, each split part has its own MSA triplet. We duplicate the perfect and uncorrected versions of the read as many as there are split parts in order to do all multiple alignments necessary. In both split and trimmed reads, recall and precision must be computed only on the parts of the MSA that have been corrected. For trimmed reads, we detect continue windows of size over a parameter w that are gaps only appearing in the corrected read line of the MSA. Then, recall and precision are computed only out of these windows. We show an example is Figure 8. The condition of long gaps appearing only in the corrected line of the MSA must prevent to mix up long homopolymers insertions and uncorrected parts such as illustrated

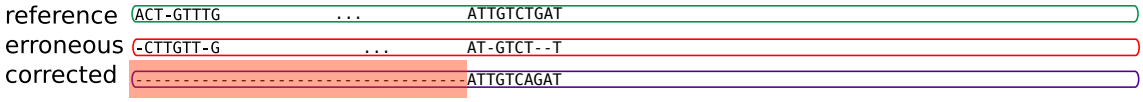

Figure 8: **Using the MSA, large windows containing only gaps appearing only in the corrected sequence (red parts) can be detected.** Such long gaps signal a trim of the read. We compute recall and precision only on the untrimmed (white) part.

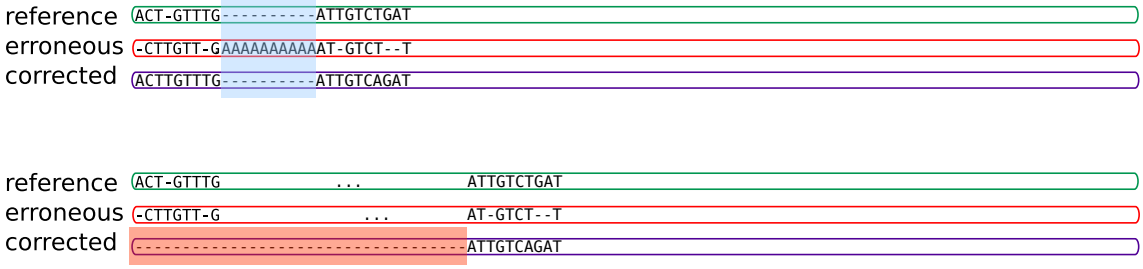

Figure 9: **Differences in MSA patterns between gaps provoked by an homopolymer insertion in the erroneous read and uncorrected parts in the final read.** In the top MSA of the example, a gap is opened in the corrected line, because of an homopolymer in the uncorrected line. However, this gap also exists in the reference (blue region). On the contrary, an uncorrected parts in a trimmed reads is materialized in a gap that appears in the line of the corrected read only, because of the missing nucleotides in this sequence.

in Figure 9. For split reads, the same window detection approach is applied on each subpart. Then recall and precision are computed only on regions that are not at the intersection of the different windows. An exemple is provided in Figure 10.

Finally, LRCstats discards reads which anchors show that they will map elsewhere than the expected region they come from, thus realizes an important gain in speed by avoiding costly divergent alignments. In our case, if a triplet yields no seeds, we do not perform the alignment and report this read separately.

2.3 ELECTOR: evaluation of long reads correction tools

2.3.1 Overview

Based on the previously described alignment strategy, we propose ELECTOR, a new evaluation tool for long read correction. It provides a range of metrics that asses correction quality, and integrates modules of read remapping and assembly metrics. It also adds results on assembly and remapping of the corrected reads (Figure 11). In the following, we present only the first module since it is the core work. The second module performs remapping and assembly, through a pipeline using BWA-MEM [125] and Miniasm [126]. ELECTOR is meant to be a user friendly tool, that delivers its results through different output formats, graphical results that can be directly integrated to users' projects. We made this tool compatible with a wide range of correctors and present novel results on self correctors. ELECTOR is based on simulated data from realistic simulation operated by state of the art tools (NanoSim [225] and SimLord [254]).

2.3.2 Input sequences

Simulated reads ELECTOR is implemented as a pipeline that is divided in two modules, an overview is shown in Figure 11. Input sequences are passed to the first and second modules. The full evaluation

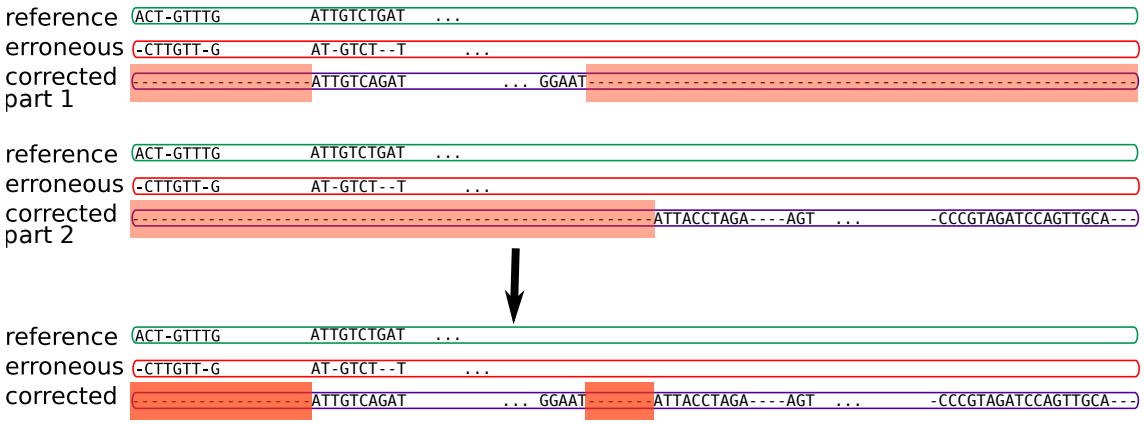

Figure 10: **A split corrected read is represented in several FASTA sequences.** In this example the read is corrected in two parts. One multiple alignment is made for each part. We detect long gaps similarly in each MSA result (yellow windows for the first corrected sequence and blue windows for the second). Then those results are combined by intersection. Windows that result from intersecting yellow and blue parts are in green. The represent remaining long uncorrected gaps after combining the different corrected subsequences of the read. Recall and precision are computed out of the green windows.

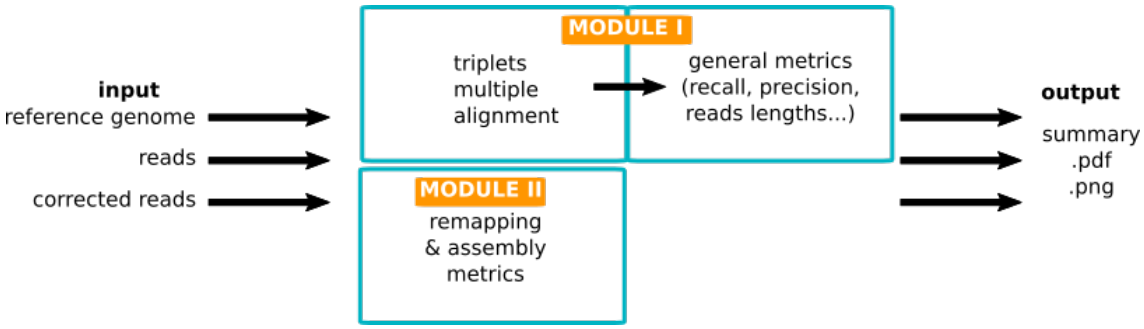

Figure 11: **Overview of ELECTOR pipeline.** Input provides the versions of the sequences at the different stages: without errors (from the reference genome), with errors (simulated or real reads) and corrected (after a correction method pass). For each sequence, a multiple alignment of the three versions is realized, and results are analyzed to provide correction quality measures. In a second part, reads are assembled using Minimap2 + Miniasm, then contigs and reads are mapped on the reference genome to provide remapping and assembly statistics. A text summary, plots and a PDF summary are output.

pipeline works with simulated reads using a reference genome provided by the user. This choice is motivated by the need to known the "true" sequences in order to precisely control the results brought by the assessed correction method. With simulation tools, we ensure we take as input sequences that will closely simulate the characteristics of long reads. Configuration of simulation, i.e. error rate, coverage, and the choice of the reference genome are the user's call.

Users are asked to simulate a dataset of reads, and to correct them with the desired correction method. The genome used for the simulation, the simulated erroneous reads and the corrected reads are then input

Experiment	Recall	Precision	Correct bases	Time
"1k" MSA	93.964%	93.479%	97.639%	11h
"1k" Splitted MSA	93.809%	93.507%	97.631%	38min
"10k" MSA	84.505 %	88.347%	95.290%	107h
"10k" Splitted MSA	84.587 %	88.278%	95.250%	42min

Table 3.1: **Comparison of the two multiple alignment strategies on a simulated datasets from *E.coli* genome.** The reads from the "1k" experiment were simulated with a 1k mean length, a 10% error rate and a coverage of 100X. The reads from the "10k" experiment were simulated with a 10k mean length, a 15% error rate and a coverage of 100X. The reads were corrected with MECAT with default parameters.

to our pipeline. ELECTOR uses headers of corrected reads to make them correspond to their uncorrected and perfect version and makes triplets for the multiple alignment. Correctly formatted and sorted files of respectively perfect, uncorrected and corrected reads are created. If necessary (split reads), uncorrected and perfect versions are duplicated in order to obtain one MSA per split part of a read.

2.3.3 General correction quality metrics

Metrics for correction quality (recall, precision...) are obtained using the seed-MSA strategy previously described using the three input files to make triplets of sequences. We filter out corrected reads which length is below a percentage threshold (in practice 1%) of the reference length. Metrics described above are not computed on these reads as we expect the alignments' quality to be poor. We also take into account that correctors distinguish corrected bases from uncorrected ones using a different case and only consider upper case bases in this situation to compute FP. For hybrid correctors, the result we present were generated using short reads simulated with ART [93].

2.3.4 Validation of the seed-MSA strategy

We first validate our seed-MSA strategy both in terms of performances and results. We show to which extent its results differ from the classic MSA approach, and the gain in speed. We expect that recall, precision and correct base rate hardly differ so that the MSA behavior is correctly reproducible. Conversely, we expect an important gain in time with seed-MSA compared to MSA. We compared multiple alignment results obtained with this seed-MSA to results obtained by the regular implementation of MSA on two datasets of different read lengths.

Results are presented in Table 3.1, they show that classic MSA and seed-MSA approaches differ only by a few digit in the presented metrics for both experiments.

However, using seed-MSA, a substantial gain of several orders of magnitude in time is achieved: while the classic MSA strategy has a quadratic runtime with respect to the read length, seed-MSA drastically reduces this drawback. As another illustration, we tried to evaluate the correction of one 100Kb read with LRCstats, which did not pass on our cluster with 200 GB memory.

2.3.5 Presentation of ELECTOR's results on several datasets

We present results of ELECTOR on several simulated datasets and several species genomes (*A. baylyi*, *E. Coli*, *S. Cerevisiae*). The datasets are presented in Table 12. We display the ELECTOR pipeline results using reads corrected on lists of hybrid and self correctors, and present main metrics output of ELECTOR's

Dataset	*A. baylyi*	*E. coli*	*S. cerevisiae*
Reference organism			
Strain	ADP1	K-12 substr. MG1655	W303
Reference sequence	CR543861	NC_000913	scf7180000000{084-13}
Genome size	3.6 Mbp	4.6 Mbp	12.2 Mbp
Simulated Pacific Biosciences data			
Number of reads	8,765	11,306	30,132
Average length	8,202	8,226	8,204
Number of bases	72 Mbp	93 Mbp	247 Mbp
Coverage	20x	20x	20x
Illumina data			
Accession number	ERR788913[1]	Genoscope[2] Sequences from Loman Lab	Genoscope[3] Sequences from Schatz Lab
Number of reads	900,000	775,500	2,500,000
Read length	250	300	250
Number of bases	224 Mbp	232 Mbp	625 Mbp
Coverage	50x	50x	50x
error rate	0.178534	0.179267	0.666190

Figure 12: **Description of the data used in the experiments.**
[1]Only a subset of the data was used.
[2]`http://www.genoscope.cns.fr/externe/nas/datasets/Illumina/ecoli/`
[3]`http://www.genoscope.cns.fr/externe/nas/datasets/Illumina/yeast/`

first module in Tables 3.2,3.3,3.4. The first module of ELECTOR computes general metrics: mean recall, precision, correct base rate, number of trimmed or split reads and mean missing size, GC content. We compare LRCstats and ELECTOR on several datasets, with LRCstats as the reference, in order to assess if ELECTOR reports similar results. Since LCRstats does not provide recall and precision, we show the comparison on correct base rates only (Table 3.5).

In terms of comparison between LRCstats and ELECTOR, the two methods relatively agree on most of the results. This shows that ELECTOR's approach enables to retrieve reliable results. Both LCRstats and ELECTOR report HALC and LoRDEC as the tools providing the best correct base rates, Canu's lower performance. However, ELECTOR reports more pieces of information. For instance, on the long reads of the *A. baylyi* dataset corrected with Nanocorr, LRCstats reports a correct base rate of 0.99422 and ELECTOR reports a correct bases rate of 0.99534 (Table 3.5), which are in accordance, but ELECTOR only reports a recall of 0.97992 (Table 3.2), meaning that Nanocorr failed to correct 2% of the erroneous bases. The tools strongly disagree on LoRMA's correction, LRCstats reporting rather bad results while ELECTOR announces more than 99% correct bases. ELECTOR's results are closer to what is presented in LoRMA's publication. This might come from issues with LoRMA's corrected reads alignment with LRCstats. Taking the example of Canu on *E. coli*, LRCstats and ELECTOR report close correct base rates (respectively 0.91439%, 0.92792%) in Table 3.5. In comparison to other correctors, Canu downperforms on this dataset. ELECTOR additionally reveals that its recall is to blame in particular: 0.61962% (most of the other tools have a recall above 99%), while it has precision of 0.95255% in Table 3.3.

Computation of these results is also more time-saving than when using LRCstats. In particular, on the *E. coli* dataset, LRCstats took an average of 3 hours and 50 minutes to evaluate the quality of the correction of the different tools, while ELECTOR only took an average of 25 minutes.

Metric	Colormap	HALC	LoRDEC	Nanocorr	Canu	LoRMA
Recall	0.99632	0.99725	0.99681	0.97992	0.63545	0.98089
Precision	0.79276	0.99426	0.9951	0.9963	0.9556	0.9582
Correct bases rate	0.96073	0.99831	0.99794	0.99534	0.92792	0.99298
Trimmed / split reads	3,137	2,603	4,385	1,584	1202	340
Mean missing size	5,699	1,314	5,211	385	79	6,656
%GC	40.5	40.40	40.5	40.5	41.0	47.4

Table 3.2: **Statistics of the long reads after correction with the different methods, as reported by ELECTOR on *A. baylyi* genome.**

Metric	Colormap	HALC	LoRDEC	Nanocorr	Canu	LoRMA
Recall	0.99632	0.9966	0.99599	0.98503	0.61962	0.98396
Precision	0.76725	0.99211	0.99275	0.99424	0.95255	0.95372
Correct bases rate	0.95362	0.99823	0.99761	0.99661	0.92499	0.99274
Trimmed / split reads	3,680	4,787	7,990	1,612	1,470	577
Mean missing size	5,982	2,374	5,577	341	76	6,153
%GC	50.8	50.8	50.9	50.8	50.7	53.0

Table 3.3: **Statistics of the long reads after correction with the different methods, as reported by ELECTOR on *E. coli* genome.**

Metric	Colormap	Nanocorr	Canu
Recall	0.99558	0.97952	0.63439
Precision	0.79184	0.98878	0.94484
Correct bases rate	0.96009	0.99472	0.9262
Trimmed / split reads	12,021	5,193	6,338
Mean missing size	5,376	416	196
%GC	38.2	38.2	38.9

Table 3.4: **Statistics of the long reads after correction with the different methods, as reported by ELECTOR on *S. cerevisiae* genome.**

dataset	evaluation tool	Colormap	HALC	LoRDEC	Nanocorr	Canu	LoRMA
A. baylyi	LRCstats	0.99897	0.99961	0.999626	0.99422	0.91439	0.78230
	ELECTOR	0.96073	0.99831	0.99794	0.99534	0.92792	0.99298
E. coli	LRCstats	0.998964	0.99940	0.99933	0.99602	0.911475	0.67024
	ELECTOR	0.95362	0.99823	0.99761	0.99661	0.92499	0.99274

Table 3.5: **Correct base rates of the long reads after correction with the different methods**, as reported by ELECTOR and LRCstats on *A. baylyi* genome and *E. coli* genomes.

2.4 How correctors perform on RNA

In our context, it would be interesting to use ELECTOR on transcriptomics read in order to analyze correction tools on RNA reads. Since RNA long reads spliced mapping has just been started to be supported by mapping methods (and no dedicated publication exist), we lack of perspective to use ELECTOR on real transcriptomic data. On the other hand, ELECTOR can be used on simulated data. Although there exist many simulation solution for long reads in the genomic context, no method currently exists for RNA. In particular the gene expression has to be modeled. Thus in the following, we provide a study of long reads correction methods on RNA that was performed separately from ELECTOR's context. This work is at the initiative of members of the ASTER ANR. In particular, Leandro Ishi, Vincent Lacroix and Rayan Chikhi realized analysis of correction quality, isoforms numbers before and after correction for each gene, and other results, using mouse transcriptome sequenced at the Genoscope such as the one used in Chapter 2. We participated in this study, specifically we identified that not all correctors achieve to conserve the isoform structures during correction. We show the experiments and results that led us to such conclusions. We additionally mention general conclusion from ASTER's study that are of general interest.

2.4.1 Specificity and caveats in the case of RNA long reads

Several hurdles can be identified about transcriptomic long read correction. First, similarly to short reads, gene expression leads to heterogeneous coverage of each isoform. Low coverage give less information for read correction. In terms of long reads specifically, the task of overlap detection is critical. As pointed out in Chapter 1, transcriptomic reads are usually less long than genomic ones, leading to a lower correction recall. Even before the correction has started, a fraction of the sequences is already lost, thus not benefiting either to correction process nor being corrected. Hybrid methods can escape this problem since they rely on the short reads coverage. However, many hybrid corrector work with assembled short reads contigs, prone to errors in the transcriptomic context when several isoform exist. Self-correction method rely on graphs where a linear consensus is searched. It is difficult to anticipate to which extent isoforms can be collapsed during this process. Thus, we present experiments in the following to better understand how correctors are able to respect each isoform structure when correcting reads. The results presented come from a preliminary study not published yet. They help understand the main pitfalls in actual RNA correction and motivate the study we present below. The principal indicators we need to observe in order to understand the quality of correction are the error rate, the loss of coverage and the isoform bias induced by correctors. We anticipate that isoform bias can occur if some isoforms are overcorrected (reads corrected to major isoform) or if rare isoforms are dropped by the correction method. The dataset used is extracted from the mouse dataset presented in Chapter 2 (750,000 reads extracted from this dataset, without rRNA). We selected hybrid correction (LorDEC, Proovread, PbCR, Colormap) and self correction tools (MECAT, Daccord, LoRMA, PBDAGCon). All tools were launched with 32 threads. Results were computed using GMAP [250] for mapping on the genome and BWA-MEM for mapping on the transcriptome and AlignQC [248] (that helps assessing alignment of corrected reads results).

2.4.2 General conclusions on RNA long reads correction

Proovread and PbCR give the longest runtimes, with more than 100 hours to correct the dataset. Concerning performances, fastest tools include LoRDEC, Daccord, LoRMA and in particular MECAT which achieves read correction in less than an hour. Trends remain the same with memory usage, MECAT and LoRDEC distinguish themselves by using less than 10 Gb while other tools use at least twice as much, and up to ten times more. All tools provide a fraction of split reads, around 30% for the tools that deliver the less of them. LoRMA is the corrector that splits the most reads (and returned 1.5 times the original number of sequences), thus achieving a lower mean length for output reads (and losing to some extent the long range information provided by the reads). Remapping rate varies goes from 85.5% with LoRDEC or Proovread to more than 99% with LoRMA (however, shorter reads are easier to map) and MECAT (initially 83.5% for raw reads). As expected, hybrid correctors take advantage of short reads to output better corrected base rates than self correctors. The same trend applies to errors in homopolymers. While most of these errors are corrected using hybrid correctors, at least 60% of them remain when using self correctors. Finally, by mapping on transcriptome it is observed that there exist cases where less isoforms exist in the output than in the input. In the following, we set up an experiment to identify scenarios that are favorable to such loss.

2.4.3 Isoform loss scenarios

Simulation For this work we need to assess the correctness in terms of base and of structure in reads after correction. We designed a simulation in order to test these two features, by passing a mixture of different isoforms to the correctors. We simulated reads for one gene only, with two different transcripts, and see the impact of the relative abundance between major and minor isoforms and of the the size of the skipped exon as parameters on the capacity of the correctors not to collapse isoforms.

We simulated a simple case of a gene with two isoforms: the major is the most expressed, the minor is the less. Exons length and number are chosen according to resemble what is reported in eukaryotes (8 exons, 200 nucleotides). A skipped exon, whose size can vary, is introduced in the middle of the inclusion isoform. We allow the ratio of minor/major isoforms (M/m) to vary in order to model a local differential in splicing isoform expression. For a coverage of C and a ratio M/m, the number of reads coming from the major isoform is MC and the number of minor isoform reads is mC. All reads are sequenced along the full-length isoform. We produce two versions of each read. The reference read is the read that represents exactly its isoform, without errors. The uncorrected read is the one in which we introduced errors. We use an error rate and profile that mimic observed R9.4 errors in ONT reads (13% error rate, with 37% of substitutions, 9% of insertions and 54% of deletions). After a corrector is applied to the read set, we obtain a triplet (reference, uncorrected, corrected) read that we use for assess the quality of the correction under several criteria.

Isoform structure In order to retrieve isoforms in presence after a pass of a corrector, we re-map the corrected reads on both reference isoforms. Since the amount of reads is reasonable we can afford dynamic programming to obtain precise alignments, so we used a Smith Waterman efficient implementation [259]. Reads are assigned to one or the other isoform according to their mapping profile result. Inclusion reads (respectively exclusion reads) are expected to map on the reference inclusion isoform with no large insertion (respectively with a large deletion, sized as the skipped exon) in the middle. Inclusion reads (respectively exclusion reads) are expected to map on the reference exclusion isoform with a large insertion of the size of the skipped exon (respectively with no large deletion) in the middle. In the output, we check if both isoforms appear, and if they do, whether it is at the right ratio. Some tools could not be run on these small simulation datasets: LoRMA, PBDAGCon (worked with 100X long reads) and MECAT. Either the tools stopped early in their execution or did not correct any read. We think that the relative short size of sequences in comparison to genomic reads was the main issue. Hybrid correctors and self correctors results are presented separately in Figures 13 and 14. The overall comparison between the two paradigms shows that hybrid correctors are less impacted by isoform collapsing phenomenon since their algorithms act less directly on the long reads. LoRDEC shows the best capacity to respect the isoforms in presence. However, in regions less covered (such as lowly expressed genes, rare transcripts), all tool report less precisely the right

amount of each isoform. Self correctors require enough material to build consensus, this is why only Daccord could be launched on 10X long reads, with rather erratic results. Even on higher coverage, not all correctors achieve to correct our simple instance and none reports exactly the expected number of sequences for each isoform. We could not derive any clear trend concerning the relative isoform ratios, even if the 90/10 ratio seems to be in favor of overcorrection to the major isoform. Skipped exon length seems to impact both hybrid and self correctors, small exons being a real challenge.

3 Discussion

In this chapter we presented ELECTOR, a tool to assess long reads correctors. In the context of this thesis, ELECTOR is particularly meant to help the development of a correction tool. Not presented ELECTOR's works in progress in this dissertation include the indel rates (and comparison with LCRstats) and percent of errors in homopolymers. We also are currently working to handle real data with ELECTOR. In this case we have to map the reads on the reference to extract a reference sequence. This task is realized with Minimap. However, this supplementary step implies that reference reads depend on mapping quality, which will have impact on presented results. We enable polyploid genome simulation, in this case each uncorrected and corrected version are linked to their corresponding reference sequence from the original haplotype, and recall, precision and correct base rate are computed adequately.

ELECTOR has the ability to adapt to RNA long reads as well, and will be useful to assess RNA read correction. Currently, the limitation is rather on the side of the data simulation. In order to build a benchmark for a RNA correction tool, we would like to start with simulated reads in order to control exactly the ground truth. But there exist no simulation tool for PacBio or ONT transcriptomics reads, most likely because it requires more work than in genomics to correctly simulate gene expression. A work in progress simulation tool for this application is presented in the next chapter.

We could see that a methodological gap exists between correcting genomic sequences and correcting transcriptomic sequences. No real generic or RNA correction method exists, all are rather tailored for DNA. According to the aspect that is looked for (base-wise accuracy, not loose any isoform,...) all correctors perform differently and none can reach good results for all requirements at a time. Thus there is no method that has synthesized all needs for RNA at the moment. The next chapter will be dedicated to present a consensus method for RNA reads. Both ELECTOR's underlying algorithms and presented correctors allowed us to introduce key concepts for the following chapter.

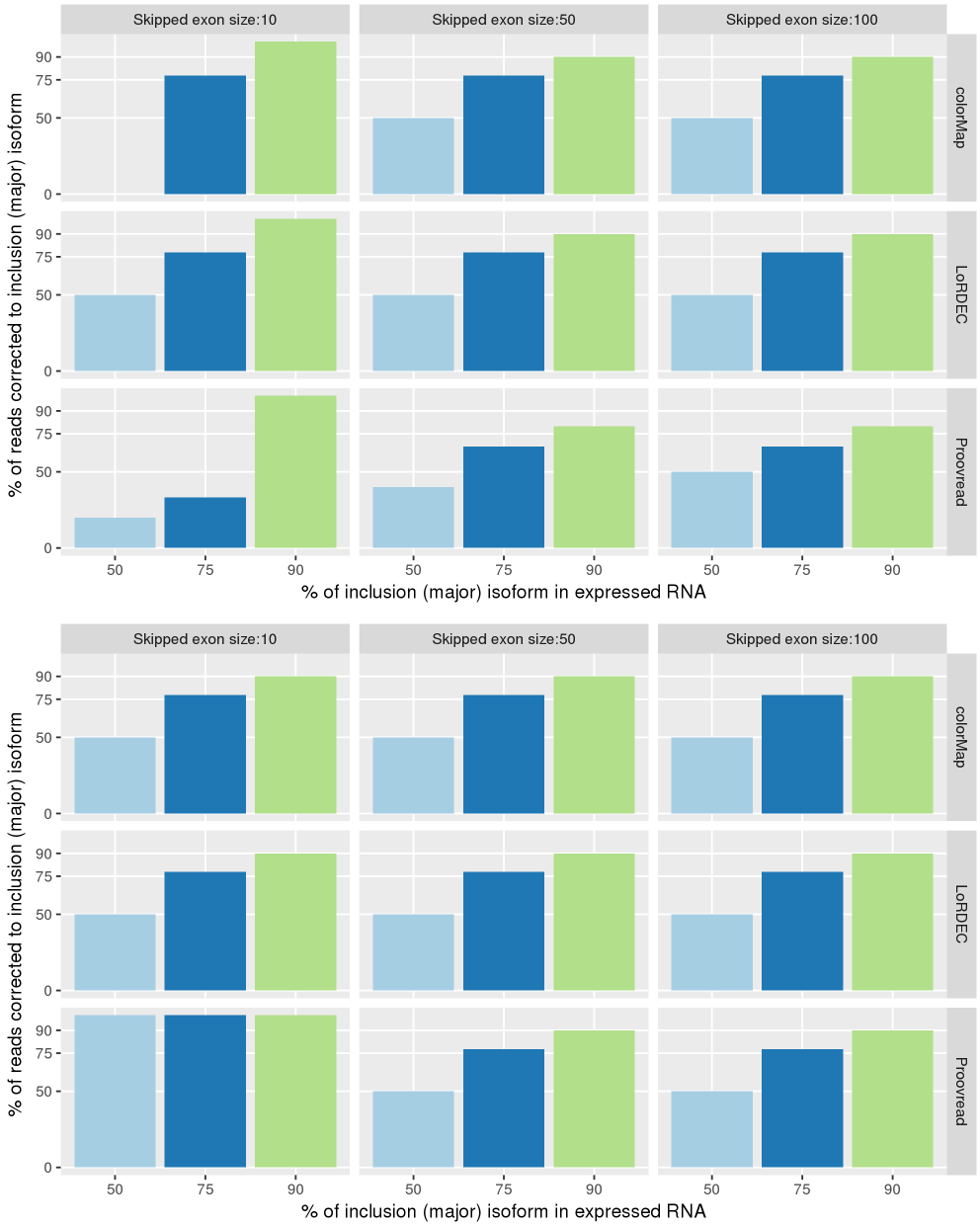

Figure 13: **Hybrid correctors and isoform structure conservation using 10X short reads coverage (up) and 100X (down).** The correctors are presented in the different lines, the alternative exon size is in column. Bars are plot for each isoform ratio (75/25; 50/50 and 90/10) on the horizontal scale. On the vertical scale, the closer a bar is to its corresponding ratio value on the horizontal, the better. For instance, the top left blue bar is a results from Proovread, for 50/50 isoform ratio in input and an exon of size 10, and we do not retrieve a 50/50 ratio after correction since the bar does not go up to 50 on the vertical axis.

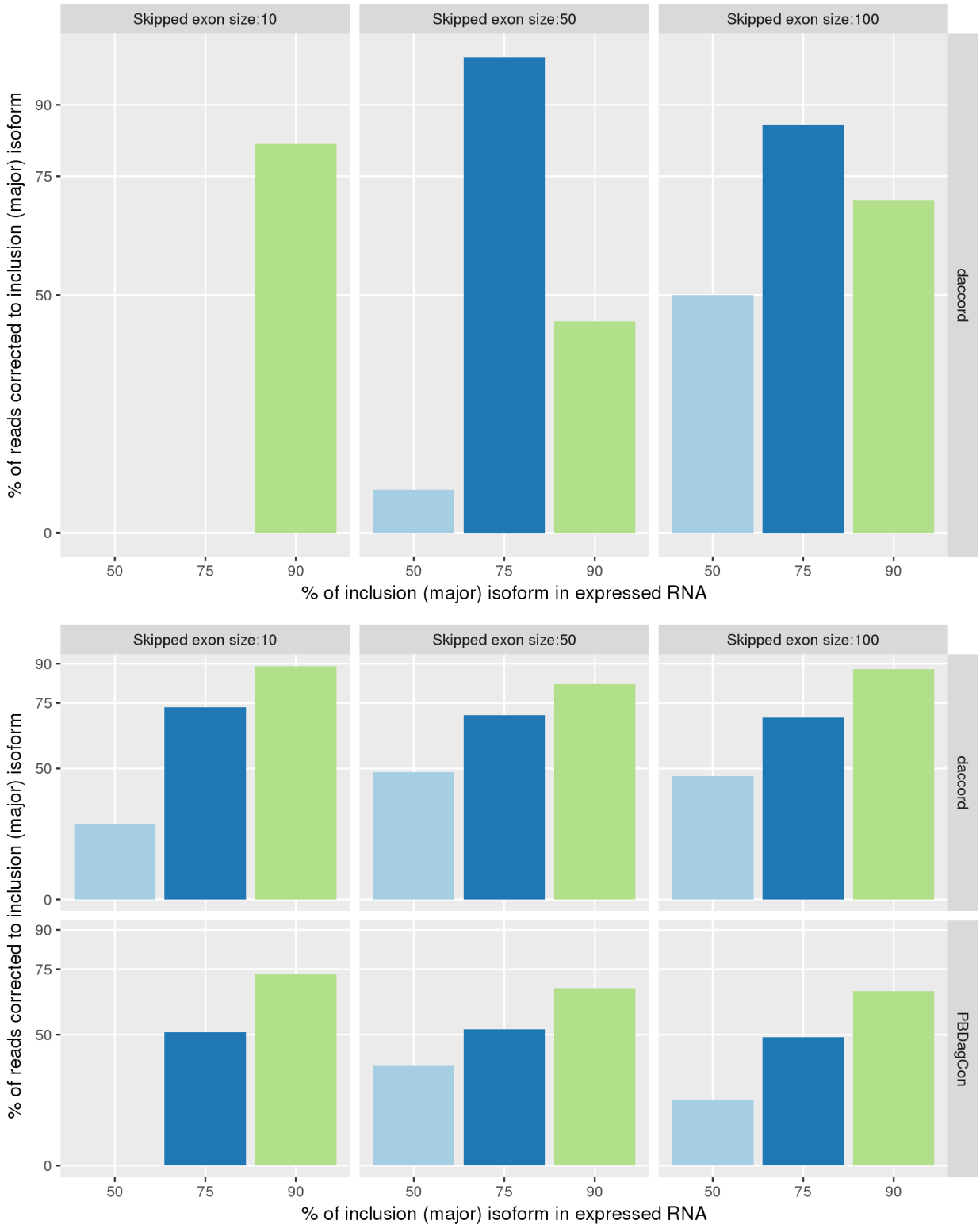

Figure 14: **Self correctors and isoform structure conservation using 10X long reads coverage (up) and 100X (down).**

Chapter 4

Towards access to corrected isoforms using long reads

In Chapter 3, we raised the point that correction methods could not be trivially adapted from genomics to transcriptomics. We also draw a parallel between correction and consensus. We aim at proposing a modular approach for going from raw long reads to isoform identification by gene. The two first modules being discussed in Chapters 1 and 2, in this last Chapter, we propose a method to compute isoforms consensus from clusters of reads coming from the same gene. We chose consensus over correction because we think that providing one consensus per isoform plus the count associated to each isoform is sufficient and less redundant than providing a corrected version for each read. But the two results are fundamentally quite similar. Against intuition, we only start reads sequence correction in the end of the pipeline. Recently in genomic assembly domain, a paradigm shift occurred, brought by long reads usage. Miniasm [126] was the first long read assembly method to propose an assembly-first method in which overlaps are found in erroneous reads to perform assembly, and assembled sequences are only corrected afterwards. Here we propose a "cluster-first" approach in which reads are first associated by genes before being corrected. This enables to divide consensus computation in several smaller problems, and to focus on each cluster content to identify isoforms from one another and provide more accurate corrected sequences.

1 Describe full-length isoforms in RNA data

1.1 Background

1.1.1 A look back to EST Consensus strategies

For EST, consensus for isoforms were looked for during the assembly phase. We recall that two main EST assemblers were PHRAP [13] or CAP3 [94]. Here we give an idea of how these methods found consensus for the ESTs. The PHRAP assembler is performing three steps: comparison, alignment and assembly of DNA sequences. The PHRAP assembler is based on an banded (local) version of SW algorithm to perform comparisons of the sequences: when a word of a certain length is perfectly matching between two sequences, PHRAP tries to extend the alignment while computing a score. Of all alignments, the highest-scored is kept and the reads are assembled. PHRAP is using the PHRED quality value to increase its accuracy. The data was not error-free, which led to discrepancies during the alignment phase. When a region has discrepancies, it is more tolerant to compute an alignment with regions with a low PHRED score than with regions with a high PHRED score that are more likely to be due to repeats and to lead to mis-assembly. The contig final sequence reflects a vote system: if there is an ambiguity at a base during the alignment, the best-quality base call is chosen.

CAP3 works on a slightly different way. The comparison and alignment steps look like PHRAP's, using a BLAST-like technique with a banded Smith Waterman alignment. Overlaps between reads are ranked according different alignment measures, several reads can be kept in a layout. A multiple sequence alignment is constructed for each contig. A consensus is created using the quality score of each base, each read is added one after another to the current consensus (the sequence is read from 5' to 3' and reads are added in order). If the average quality value of the current consensus for a base is lower than the quality value of a read added, the consensus will change according to the new read information. Others assembly strategies were develop, for instance TIGR [179] or PAVE [221] that was trying to use the mate-pair information to enhance the contig construction.

From these works, we keep the ideas of base-voting for consensus calling, and of multiple alignment to retrieve the common bases in reads.

1.1.2 Long reads

Only a few pipelines are published to date for the processing of long reads in a transcriptomic context. Mandalorion [28] and TAPIS [1] are two very recent pipelines that enable to detect alternative splicing using long reads. These methods do not work *de novo*. Mandalorion takes ONT reads and Illumina, filters and keep quality long reads that are then mapped on the reference genome using BLAT. Alternative splicing is

detected and they also provide gene expression using the short reads counts. TAPIS works in the same spirit but integrates Iso-seq (PacBio) reads, and uses the mapper GMAP [251]. Tofu [77] is the first pipeline able to work reference-free. It starts by a correction step, then clusters similar reads and extracts a consensus by isoform from the clusters. Then, these consensus can be mapped on a reference if needed. Tofu is specifically designed for PacBio reads and cannot be applied on ONT.

1.2 Objectives of our method

Our goal is to detect alternative events and call consensus from CARNAC-LR's clusters described in Chapter 2. In our case, like Tofu, we propose to rely on clustering to avoid using a reference (however we cluster reads by gene, not by isoform) and to extract several consensus from each cluster. Tofu uses PBDAGCon to call one consensus per isoform cluster. Alternative splicing is retrieved afterwards when mapping on a reference. On the contrary, we want our consensus step to 1-directly detect alternative events and 2-propose several adequate consensus. We will also use partial order graphs such as in PBDAGCon, but we will use on the original partial order alignment described in Lee's paper [121]. The rationale is that we want to extract several meaningful consensus from the alignment, instead of the unique consensus proposed by PBDAGCon. Moreover, PBDAGCon's heuristic implies that all sequences are aligned to a first backbone sequence. In our case, since several skipped exons can appear in the dataset, this is not a desired property since some exons could be missing from the first sequence (even if we choose the longest as the backbone).

1.3 Multiple sequence alignment strategy

In this section we state the work of Lee in [121] which will be our ground base for building multiple sequence alignments. The presentation of the algorithms allows us to clarify the choice of POA over other MSA strategies and to show its interest on our problem.

1.3.1 MSA

First bioinformatics applications of MSA were proteins alignment [135, 83]. Applications to RNA were mostly dedicated to retrieve RNA secondary structure [207, 262]. There is no unique formal definition for multiple sequence alignment, but we gave the main principle in the previous chapter. Classical representation of MSA are matrices. In this chapter, we will work with graph representations. In partial order-MSA (PO-MSA), the MSA is represented by a directed acyclic graph. Each base of sequence is a vertex, and arcs are drawn between consecutive bases in a sequence. Vertices record the identity of the sequences they come from as well as their positions in these sequences. Then sequences can be trivially reconstructed by following paths of vertices that share same identifiers. Thus, a very trivial simple path PO-MSA can be constructed from a single sequence, with a single source that is the first base of the sequence and a single target the is the last base of the sequence. All bases follow each other and are linked by directed arcs from left to right, resulting in a linear graph.

1.3.2 Graph construction

New sequences are gradually aligned on the graph $\mathcal{G}$. The alignment procedure is described afterwards. First we explain how, given an alignment between a path in the graph $\mathcal{S} \subseteq G$ and a new sequence $\mathcal{S}'$, the graph is enriched. A new vertex in $\mathcal{G}$ is created for each nucleotide of the sequence $\mathcal{S}'$. In the graph construction phase, a second type of undirected arcs is used, that we denote *alignment edges*. If two vertices $v \in \mathcal{S}\text{and}v' \in \mathcal{S}'$ are aligned, then an *alignment edge* $e = (v, v')$ links the pair. Once the sequence is aligned, each new vertex is looked up. If it is aligned with a vertex of the graph hence we have a couple (v, v'), then:

- ◇ if v and v' represent the same nucleotide, the vertex is *fused* (see Figure 1 a))
- ◇ else,

– if v is itself aligned to $w \in G$, with w representing the same nucleotide than v', then v' and w are *fused* (see Figure 1 d))
– else, we conserve $e = (v, v')$ (see Figure 1 b))

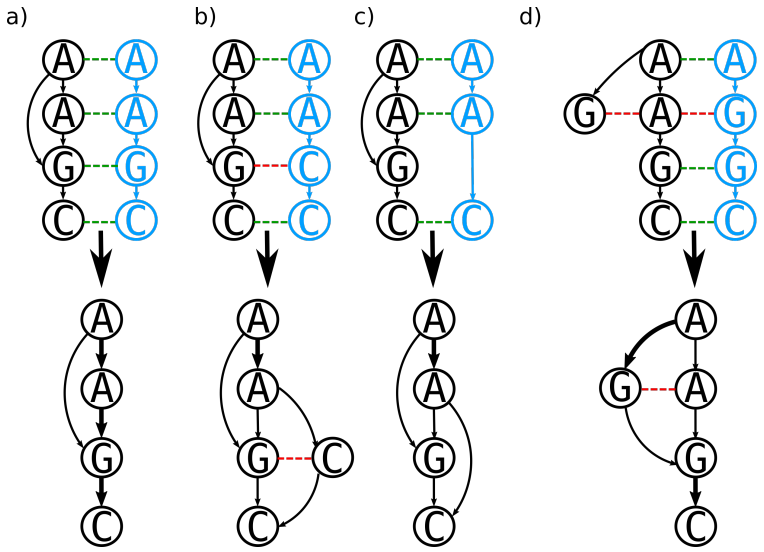

Figure 1: **DAG construction with new aligned sequences. The current graph is in black, aligned sequence to be added in blue. Green dotted edges represent a alignment with a match, red dotted represent alignment with a mismatch.** When possible, vertices are fused and arcs between two fused vertices increase in weight. Otherwise new vertices are created in the graph. When mismatches occur, we keep track of the alignment with an edge.

The *fusion* process works as such: only one of the two vertices is kept and stores the information for the two vertices, any in and out arc is also conserved. Edges $e = (v, v')$ is removed. Finally, redundant arcs are removed, that is, there is at most one arc between two vertices.New sequences are added to the graph iteratively through this scheme, in the order of a FASTA input. The resultant graph is a DAG. Within each path of this DAG, vertices are ordered with respect to each other. This order imposes that all vertices have a rank r_0, r_n, which ensures that for any vertex v of rank r_k, if v has source vertices then the ranks of these vertices are inferior to r_k. Finally, each time an arc is placed between two fused vertices, it can be weighted in the perspective of building a consensus. Arcs are weighted with the number of sequences that supported the presence of the two consecutive vertices. One can notice that the gaps are implicitly represented by the graph topology (see for instance Figure 1 c)).

1.3.3 Sequence to graph alignment

The alignment is made through an extended version of global or local classic alignment procedures. In the Needleman and Wunsch algorithm, two cursors i on *sequence* 1 and j on *sequence* 2 are considered. These cursors move from sequences left to right on each base. The pair (i, j) represents a current position in the alignment matrix. The algorithms aims at finding out which of the three possible moves is the best to do:

◇ i and j move forward, which corresponds to aligned bases (match or mismatch)
◇ i (respectively j) moves forward while j (respectively i) is fixed, which corresponds to an insertion of one base from *sequence* 1 (respectively *sequence* 2)

The score of the previous position is added to the score of the move to obtain the score of the new (i, j) position. This is repeated through all bases. When aligning a sequence on a graph, the same framework is used. A given base has only one predecessor in the sequence. In the DAG, a base can have several predecessors, i.e. vertices that are the sources of arcs entering the base's vertex. Thus, there are more moves and more previous scores to consider, starting at each predecessor. Through a loop over all predecessors of a base, we can apply the previous procedure to any case that has to be treated, and the rest of the algorithm remains the same. In Figure 2, we show the beginning of the alignment of a graph (vertical) and a sequence (horizontal). The graph is linear in its two first vertices thus the algorithms is run exaclty as the classic Needleman-Wunsch. In Figure 3, we present all the possible predecessors, moves and scores for a base in the graph that has several incoming arcs. In Figure 4 the sequence finally aligned is ready to be added in the graph.

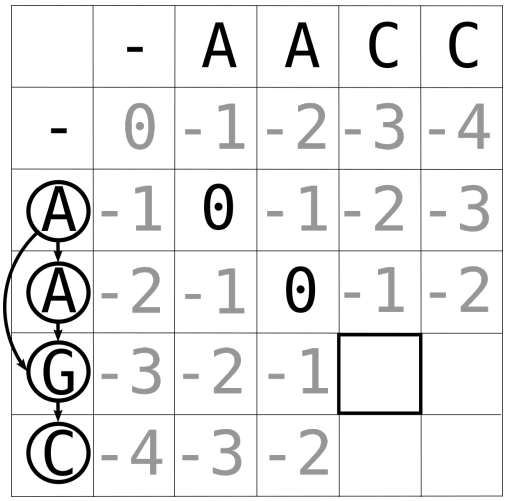

Figure 2: **The alignment can be viewed in a similar way than classic alignment schemes such as Needleman and Wunsch algorithm.** In this example, we consider a graph that was obtained by aligning two first sequences, AAGC and AGC. A new sequence AACC is to be aligned. Since the graph is ordered, each nucleotide of the graph gets a single row of the matrix. The sequence to be aligned is placed on the columns in this example.

Predecessors of a base have to be well defined. A topological sort (using the Kahn algorithm [99] or several DFS) is realized on the graph before each alignment round. This is possible since the alignment graph is a DAG. Then each vertex is guaranteed to be placed after all its predecessors.

1.3.4 Output a MSA

Since the graph has been ordered, each vertex can be assigned a column in the final MSA matrix. Vertices aligned to each others are put in the same column. Vertices store the sequences they belong to, so each row of the matrix can be filled by following paths of vertices from the same sequence (Figure 4). There is a bijection between a POA graph and a given MSA result. However the representation arbitrarily place gaps in certain cases.

1.3.5 Differences with other strategies

Since pairwise comparison of sequences complexity is in $\mathcal{O}(N^2)$ with N the length of sequences, the comparison of P sequences is in $\mathcal{O}(N^P)$ hence intractable for a large number of sequences. That is why the

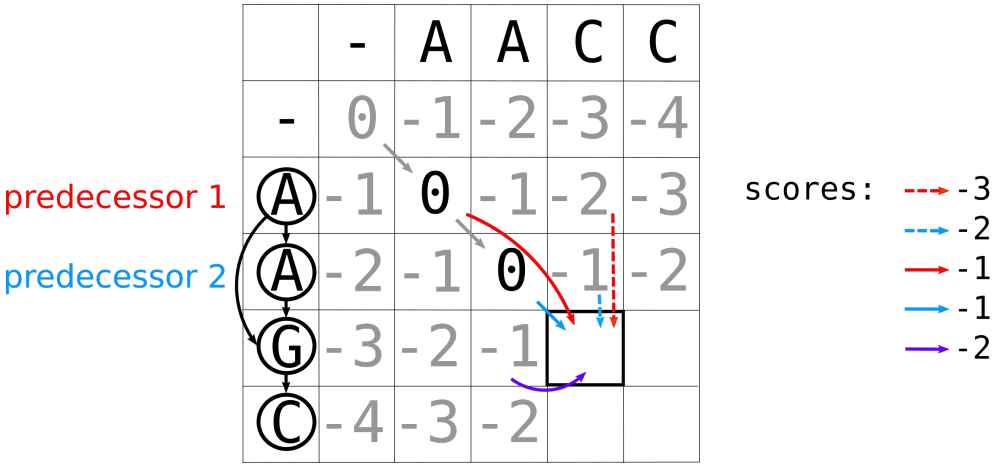

Figure 3: **The algorithm performs the same way than Needleman and Wunsch but with more possible moves and predecessors for vertices with several predecessors.** In this example, the third vertex G has several predecessors in the graph, we represented the different possible moves by colored arrows with associated scores.

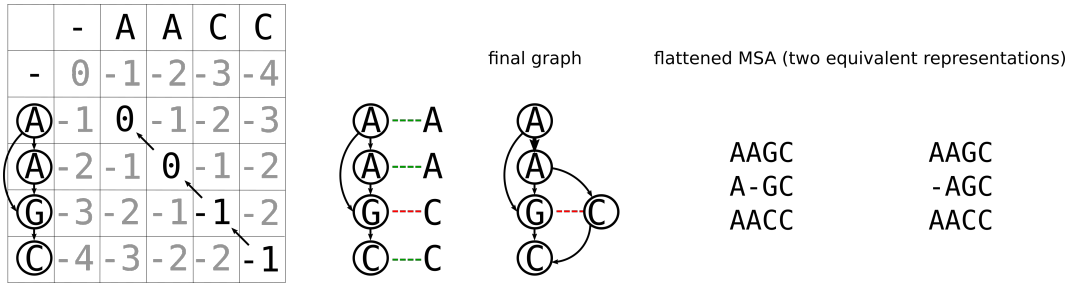

Figure 4: Integration of the sequence to the graph. We then obtain an alignment of the sequence on the DAG, in this example with a mismatch. It is inserted in the graph. A flattened alignment can be obtained.

community relies mostly on heuristics. Most works anterior to PO-MSA relied on a *progressive* strategy. Progressive strategies consist in aligning pairs of MSA to build larger MSAs. This implies to flatten each MSA of a pair to a linear profile called a consensus. This step implies loss of information, in particular when gaps and insertion occurs. The consensus inherits from choices made at gaps and insertions and fixes them for all sequences of the MSA. Hence, consensus can be computed for MSA, but a MSA cannot be retrieved only given the consensus. Moreover several MSAs can have the same consensus. Algorithms such as CLUSTAL [88] work that way, with a polynomial time complexity in $\mathcal{O}(N^2 logP)$. However, and despite enhancements proposed for instance in CLUSTALW [229], these approaches are greedy and can deliver sub-optimal alignments.On the contrary, PO-MSA helps resolve certain inconsistencies of previous multiple alignment procedures. It does not deliver an optimal MSA solution, however it keeps the whole information of the MSA in the DAG, and can guarantee the optimal alignment of each new sequence on the graph. The result will depend on the order the sequences are added in the graph. Moreover, its algorithm's speed is increased, since it is linear with the number of branches per vertices in the graph.

PO-MSA approach was brought back recently by long reads correction and polishing tools [238, 258, 34]. Its implementation was also validated for EST assembly purposes.

In order to retrieve isoforms structure, the task of grouping common elements in the MSA is more important than the overall alignment similarity. In other words, we accept to add a lot of gaps to the MSA in exchange for accurate grouping of bases from same exon sequences. Our primary goal was not to develop a new algorithm for MSA. We focused on using available methods to first present the coherence of our approach.

1.4 How exons are detected in POA results

1.4.1 General idea

In Figure 5 we outline the expected patterns of alternative variants in a cluster. The presence/absence of an exon creates a gap in the MSA. Alternative events can be classified with a small degree of resolution. Alternative start and end of transcription create gaps at the very beginning or end of the MSA. First alternative donnor or acceptor site, if they concern a long enough region to be detected, can also create such pattern. Skipped exon, other alternative donnor/acceptor and retained intron create long gaps (tens to hundreds of positions in the MSA) flanked by constitutive parts. In reads that include the alternative regions, we expected that these subsequences are at least of the size of an exon. Such a size may vary, thus we take it as a parameter, in practice set to 30 nucleotides that corresponds to what is mostly observed in eukaryotes.

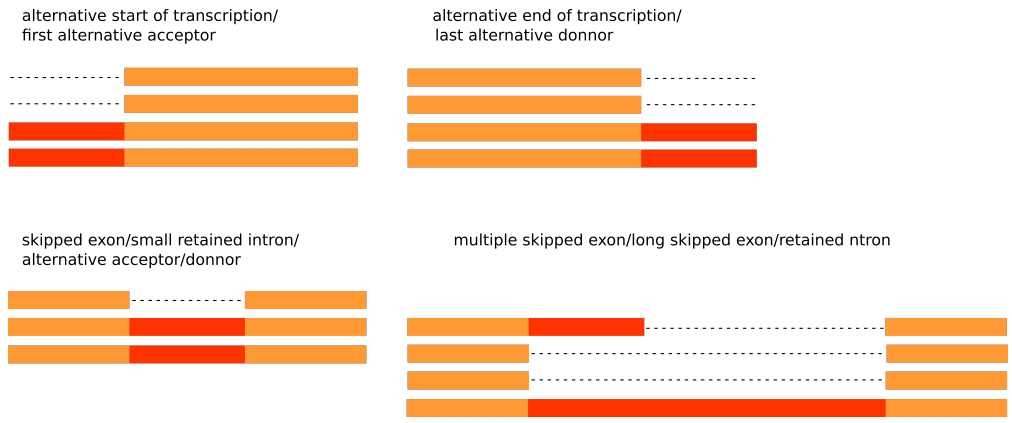

Figure 5: **Expected patterns in a MSA aligning different isoforms of a gene.** Alternative events are linked to the presence/absence of subsequences (exon, parts of exons or introns) that create long gaps in the MSA. On the contrary, constitutive exons align.

1.4.2 Gap detection

In the case of perfect sequences, the gap detection in the alignment matrix would be straightforward. Using long reads, the noise introduced in the alignment can fragment the gaps.

The first task is thus to retrieve and reconstitute gaps in the MSA. Gaps are chains of successive dashes in a MSA line. First, all gaps longer than a *gap threshold* are detected in each read R, leading to a list of pairs of coordinates for gaps that are column numbers in the MSA: $gaps(R) = (g_x = (p_0, p_1), g_y = (p_i, p_j), ...))$, with p denoting a column number (step a in Figure 6). The first coordinate of a gap is the position where the first dash was found in the MSA, the last is the position where the last dash is found. Very short gaps are more likely to be due to sequencing errors, thus not reported. In practice we set *gap threshold* to 8. Then, successive gaps $g_x = (p_i, p_j), g_{x+1} = (p_k, p_l)$ are considered. If the right coordinate of g_x and the left

coordinate of g_{x+1} are separated by less than *gap threshold*, which means a few nucleotides are included whithin a gap, a new gap $g = (p_i, p_l)$ replaces g_x and g_y (step b) in Figure 6).

Errors in sequences lead to detect gaps corresponding to a same alternative exon at slightly different positions in the different reads (see Figure 6 a) and b)). Thus we look for consensus positions for each gap in order to define alternative regions. In the set of all reported left positions P_{left} (respectively right positions), we cluster positions that do not differ from more than w nucleotides ($w = 10$ in practice). Within clusters, we chose the position that is the most reported (Figure 6 c)). Final regions are then drawn. Coordinates for gaps in all reads are ordered. Regions are defined within and between each gap coordinates. In the case several gaps overlap, the larger gap is divided in several regions according to the smaller gap(s) it spans (for instance last gap of first and second reads versus third and forth in Figure 6 d)). Reads are clustered according to whether they contain sequence or gap in each region (clusters are reported with the letter "c" on the right in Figure 6 d)), i.e. they represent the same isoform.

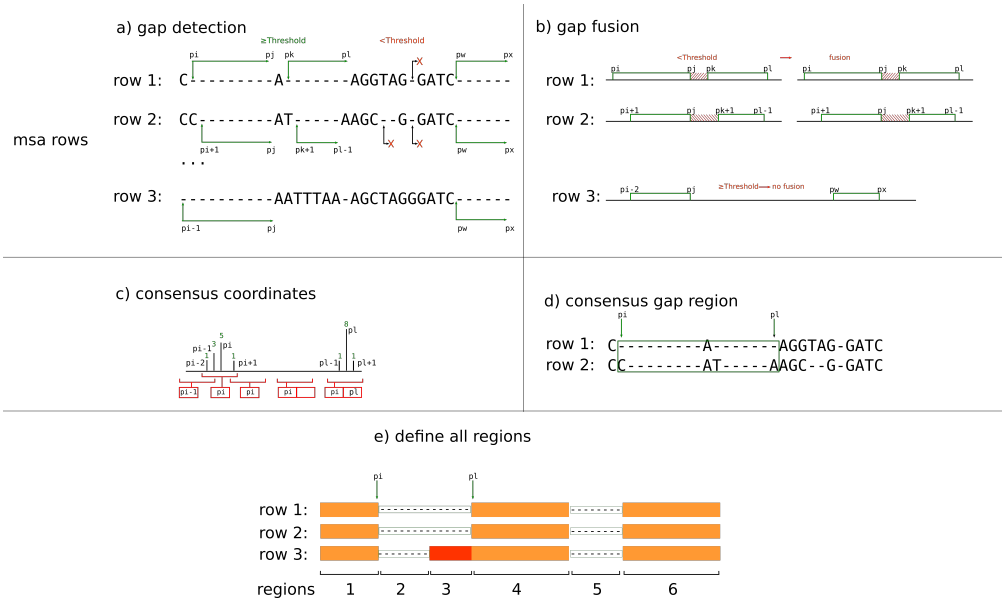

Figure 6: **Steps of alternative regions discovery in the MSA.** We present three sequences aligned that are three rows in the MSA. a) Gap detection. Gaps in green are reported, for instance gap at positions (p_i, p_j) in the first row is reported. b) Gap fusion when small island of nucleotide fragment larger gaps. Gaps in green are fused if they are separated by a small number of nucleotides (two first rows are fused to larger gaps) c) Seek a consensus position for each gap. A sliding window in red goes from the left to the right of the MSA positions. If a first gap position is present in the window, it record its value. If several gap position compete, the most reported is recorded (number of times each position is reported is written in green). While the window slides and encountered new reported positions, the current value can still be updated. For instance p_{i-2} is updated by p_i. If the window overlaps a region where no gap is reported, a new slot for a new position is created. d) Extract regions using consensus gap positions.

1.5 Consensus calling

Once regions are detected, we achieve to find common subsequences in groups of reads within clusters. We have enough information to start calling consensus with less risk of mixing different isoforms. We review the different options at our disposal to call consensus. The first strategy is to correct by isoform (Figure 7 B). The expected advantage of such approach is that the consensus is called on the whole sequence and does not suffer from approximate region definition. However, a rare isoform might not be supported by enough reads to build a consensus.

The other option is to correct by region, i.e. (groups of) exons (Figure 7 A). The main advantages are that rare isoform can still be (at least partially) corrected since they can share exons with other isoforms. This strategy also allows to compute again smaller and more accurate multiple alignments on each region. These smaller alignments should yield better quality consensus since they integrate less noise and are supposed to be realized between identical subsequences. When a region contains both gaps and exon sequences, only the reads that do not contain gaps are selected for the second multiple alignment round (for instance in Figure 7, in the first region constituted of yellow sequence, only the four first reads corresponding to the four first rows are taken into account for the consensus of this region).

Finally, a hybrid approach consists in first correcting by isoform when possible in order to obtain better definition at the exon junctions, then to perform a second round of correction using each isoform consensus and rare isoform to correct by regions (Figure 8). These three strategies are compared in the result section.

Once the isoform correction or exon correction is selected, an algorithm for consensus calling must be chosen. The most straightforward way is to vote for the majority base in each column of the MSA. A second way takes advantage of the sequence information in the MSA, and was introduced in a second paper after POA by Lee [120]. At the moment, only the voting scheme is implemented in our proof of concept. We plan to soon integrate heaviest bundling that is described in the discussion thereafter.

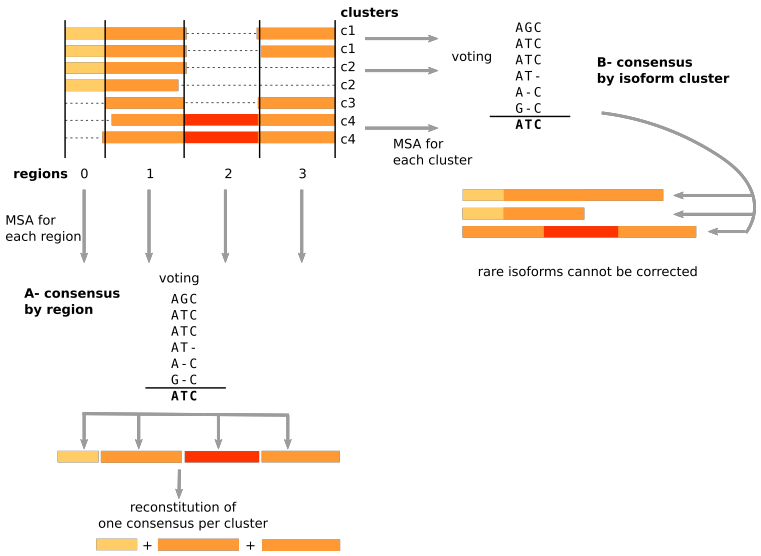

Figure 7: **Two main strategies to extract consensus from the MSA of a gene cluster.** A- each region of the MSA is corrected separately and consensus are concatenated to obtain consensus of the different isoforms. B- reads from a given isoform are used to compute a consensus.

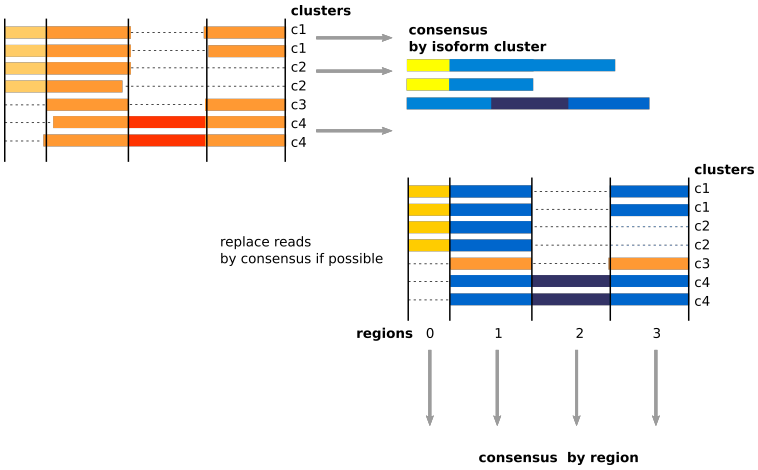

Figure 8: **Hybrid strategy for consensus calling.** First a correction per isoform is realized, then consensus are duplicated for each corrected read and uncorrected reads are added to realize a second MSA, then each region is corrected.

2 Results on simulated data

2.1 Validation protocol and simulations

The validations we present here are preliminary. We rely on the same protocol than described in Chapter 3 for benchmarking correctors against alternative splicing scenarios. Again we simulate a gene with two alternative isoforms and an error rate that looks like what is observed in ONT R9.4 chemistry. We will focus especially on the ability of the method to conserve alternative isoforms and not to collapse rare isoforms. Recall, precision and correct base rate are obtained using ELECTOR framework described in the previous chapter. In order to assess the isoform structure conservation, we use the same protocol than described in the last section of Chapter 3, in which reads are compared to both isoforms and assigned to the one it maps without long gaps.

Since our method is using only long reads information, the closest related method are self-correctors/tools for consensus, already presented in Chapter 3. Thus, the following set of methods was tested: LoRMA, PBDAGCon, daccord, MECAT.

2.2 Method validation

In the following, we present preliminary results obtained on simulated reads that demonstrate the performance of our consensus approach, and benchmark it againts other state of the art methods. Complementary results are presented in the Appendix. Our approach is denoted by "MSA" in the figures.

2.2.1 Choice of a consensus strategy

We first compare the performances of the three presented consensus strategies. Results are presented in Figure 9. All three methods aim to recover correctly the isoforms in corrected reads (this will be shown in following sections), thus we focus on other metrics that allow to choose one. The presented plots show distributions of the results on all the reads of a simulation. On the four metrics presented, the correction by cluster of isoform gives the lowest results. We believe this is due to the longer alignment to be realized. Hybrid strategy and exon region correction strategy are quite comparable on all metrics. Exon correction has

the advantage to be more conservative (more precise) and faster since it does not relies on a supplementary step. In the following, we thus chose to present only results using the region correction strategy.

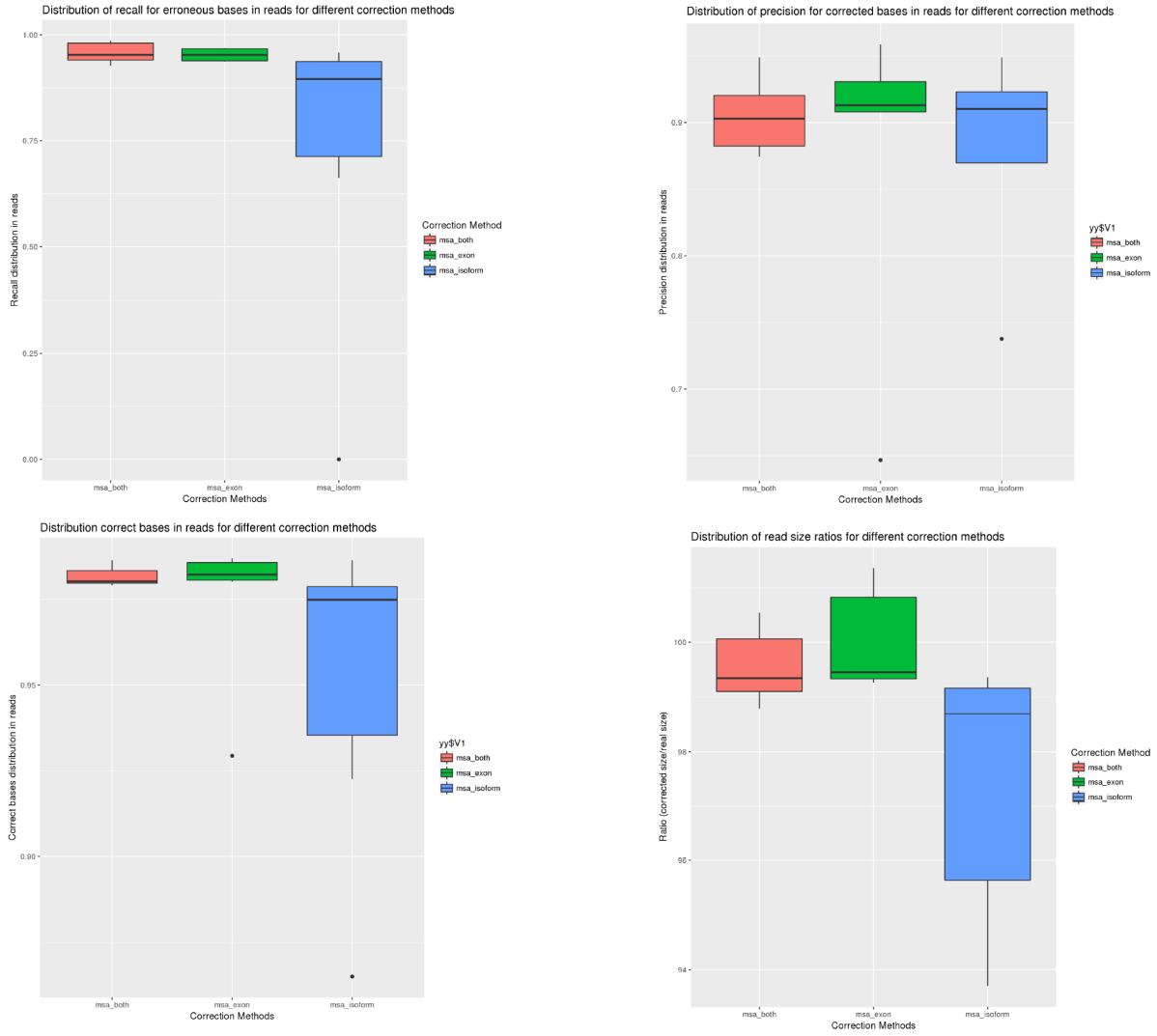

Figure 9: **From left to right, top to bottom: distribution of recalls, precisions, correct base rates and ratio of corrected over original sequence sizes for the three strategies:** consensus by cluster of isoform (MSA_isoform), consensus by region (MSA_exon) and hybrid strategy (MSA_both).

We use exon-consensus to show in Figure 10 that our method uses coverage to enhance its correction levels. In order to check for biases in remaining errors after read correction by exon-correction strategy, we plot how the error count increases along the read. We compute the cumulative number of errors over all positions of reads and recall the positions of junctions in order to verify that exon-exon junctions do not accumulate errors. We want to see whether the errors increase drastically and form clusters at junctions or if the errors remaining after the correction are rather distributed all along the read. We report two types of errors: false negatives (uncorrected bases) and false positives (wrongly modified bases). We give an

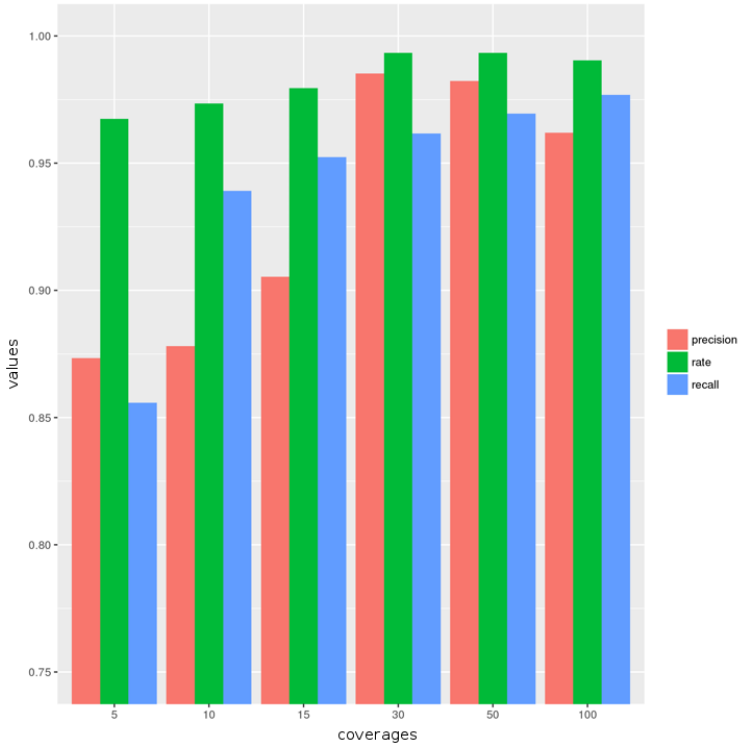

Figure 10: **Correction quality over coverage.** Precision and recall are red and blue bars, correct base ratio is the green bar, and are presented between 0.75 and 1 on the ordinates. Coverages increase in the abscissa.

example of these results on one read in Figure 11. It remains unclear if exon-junctions are particularly badly corrected using these metrics since no effect is visible in any of the simulations. A pattern seems to be the more frequent errors in the end of the read, this is because the poly A tail, being a homopolymer, is hard to correct.

2.2.2 Comparison to other correction methods

Eukaryotic RNAs show a combinatorial aspect that most likely was not in mind for the conception of these tools, which first aim is to help solving assembly. We demonstrated in Chapter III that published correctors can wrongly correct a read of an isoform to the other, even when both isoforms are fairly represented in the dataset. This is explained since in most genomic assembly paradigms, variable regions tend to be collapsed to elongate the overall contigs. These correctors also expect even and relatively high coverages. Often, both isoforms still coexist in the output dataset, but in wrong proportion. Discovering the exact reasons for this result (chimeras, bias to major isoform, missing reads) is out of the scope of this work. We focused on a pragmatic question: are the original isoforms respected and corrected in the output dataset or not?

Isoform structure conservation We present detailed behavior of the correctors when dealing with alternative isoforms. For a gene and two isoforms, we simulated 20X reads that were dispatched to one or the other isoform according to the given ratio. For instance for a ratio 90/10, the major isoform got eighteen reads while the minor isoform was present in two reads. As show in Chapter 3, correctors are prone to collapse isoforms. We investigate whether the correctors output reads according to the initial ratio, or if

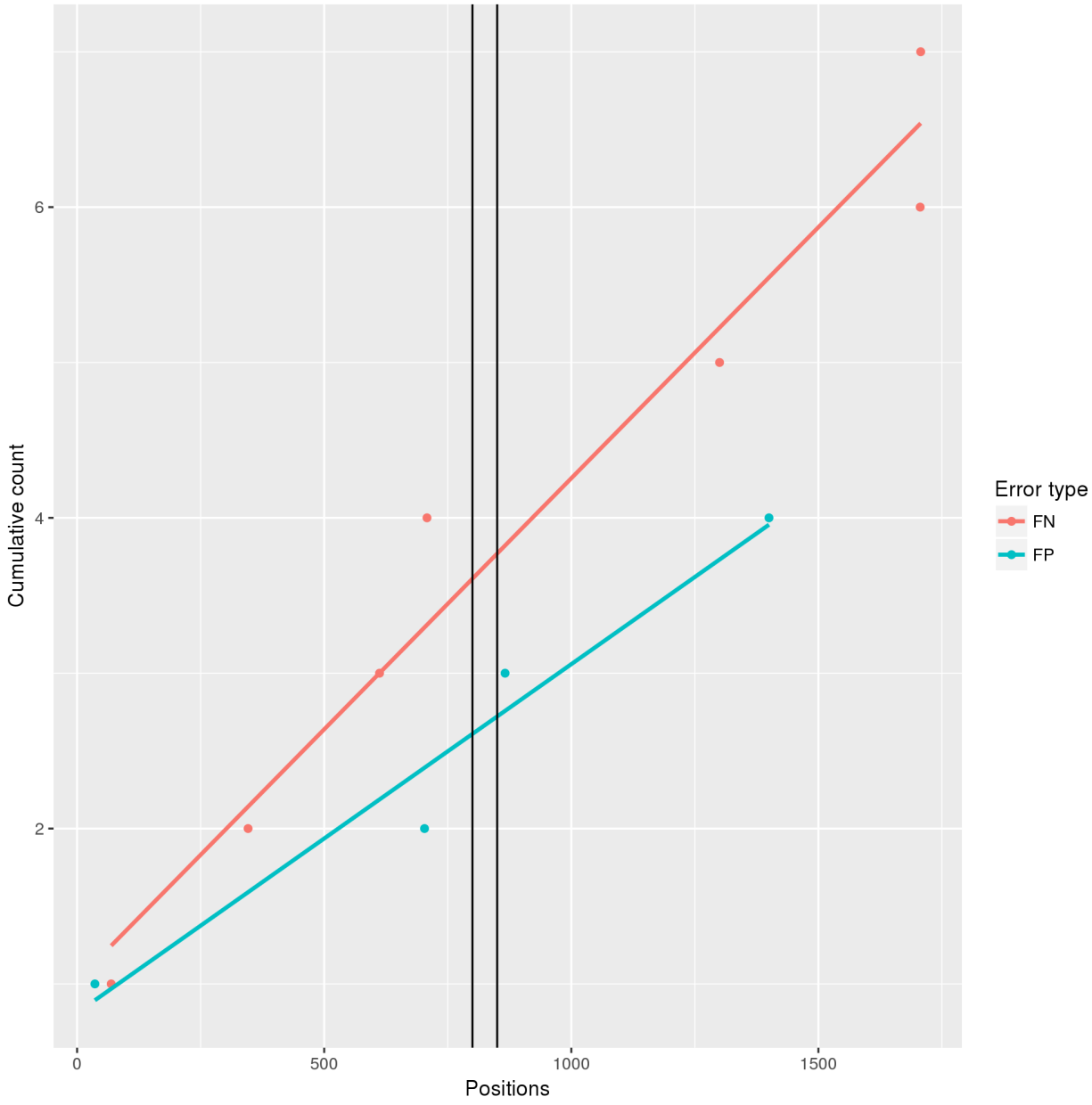

Figure 11: **Cumulative sum of remaining errors after MSA correction along with read length.** Each dot is the cumulative number of errors over all positions of a read. Vertical lines show the positions of junctions (one for exclusion forms and two for inclusion forms). FN errors (red) are erroneous bases missed by the corrector. FP errors (blue) are bases miscorrected. In this example we show a major isoform read, with included exon of length 50, and an isoform ratio of 90/10.

they tend to change this ratio. Similarly to the previous chapter, MECAT could not run on this instance, most likely because of the small size of the sequences in comparison to genomics ones. Results are shown in

Figure 12. Apart from our method, no method was clearly able to get close to the real ratio, and even at a high coverage. Longer skipped exon length does not seem to help either. They either output too less reads of the major isoform. In particular reads are missing in Lorma's output. On the contratry, MSA managed to perfectly respect the ratio.

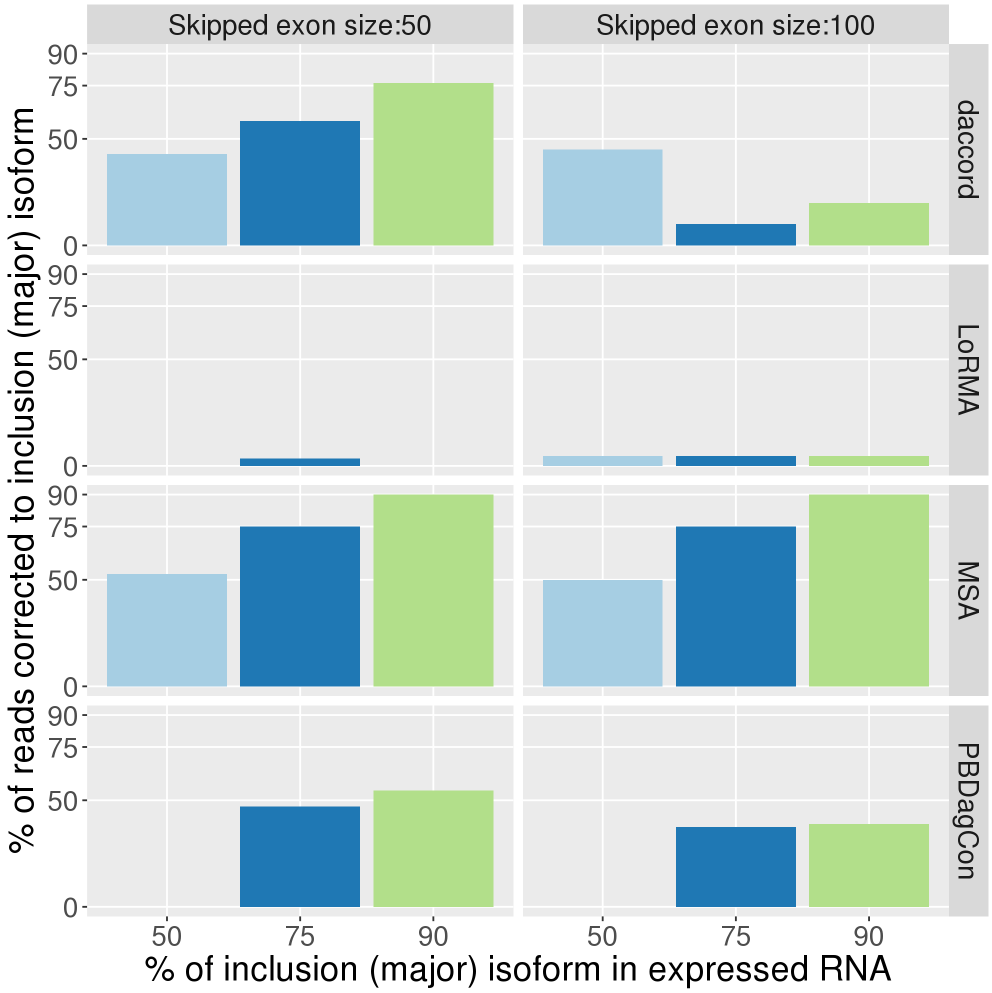

Figure 12: **Preservation of isoform ratio** Input (abscissa) and output (ordinate) major isoform ratio for varying rate of major isoform (abscissa), varying skipped exon size (vertical blocks), for each corrector that could be launched (horizontal blocks). For a 75% rate of major isoform, the ratio is 75/25, and it is expected that the colored bar goes up to 75% in ordinate so that the corrector preserved the ratio. If the colored bar is upper (respectively lower) the 75%, the corrector output more (respectively less) major isoform reads than there were in the input.

In addition, we output the ratio between corrected read size and the length of the isoform they come from. We present these sizes for each read, for the different correctors, in Figure 13. The closer to 100% this ratio is, the closer the corrected reads are from the original length of the RNA molecule. Results show MSA has the ratio closest to 100%. It also has the property not to trim reads and to produce reads of accurate

length for each isoform. The difference of size between the resultant consensus of each isoform fit the skipped exon length.

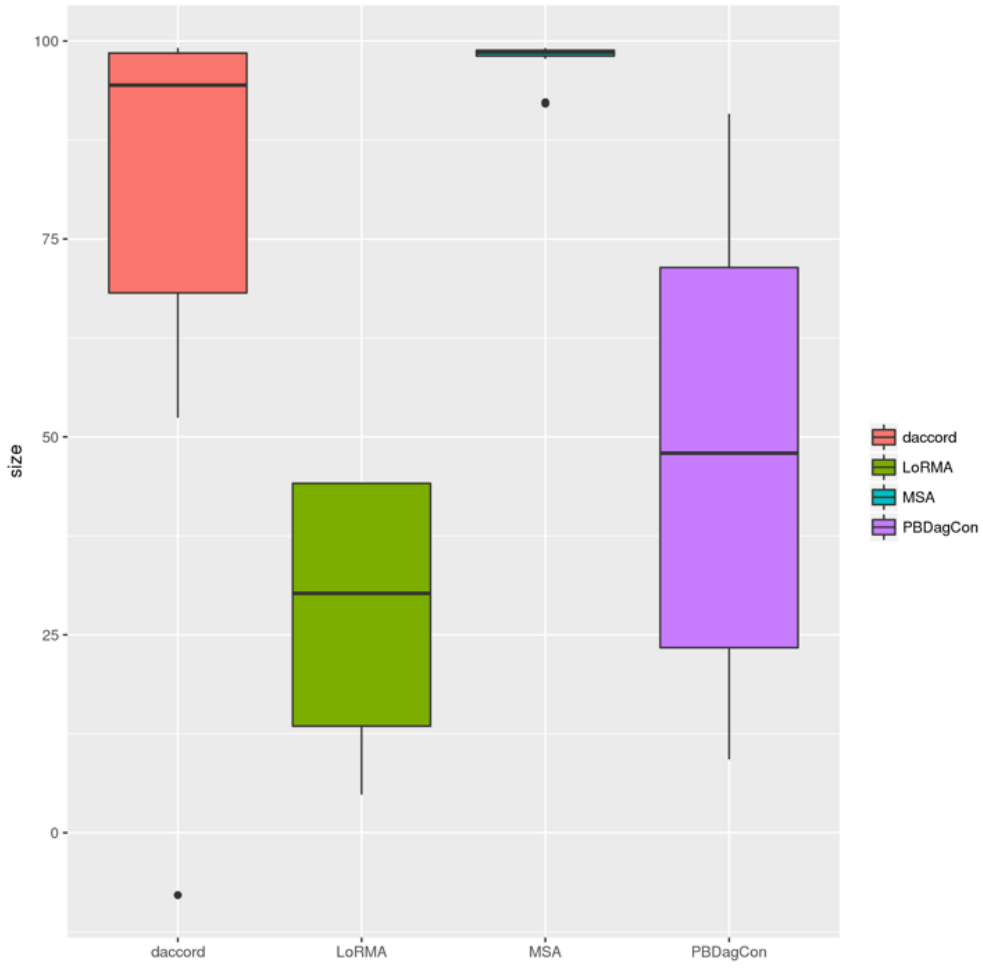

Figure 13: **Ratio of corrected over real isoform length in corrected reads.**

Using the same simulation scheme, we generate scenarios of shallow and high gene expressions (shown in Appendix). For shallow (10X) coverage, the trend is the same than previously. Our method is still able to retrieve correctly the ratio in the dataset. With a higher coverage, the main result does not change even if other corrector than MSA seem to follow a bit more the trends of each ratio.

With higher coverage, correlated with the better correction, more adequate reads sizes are observed, at the exception of LoRMA. Our method keeps showing the reads with the closest size to the real sequences.

Base-wise correction quality Recalls and precisions for each method are presented in Figure 14. Recall values are high for MSA and Daccord, and a high precision score is only obtained for MSA. Zero outliers for Daccord are reads that were miscorrected to the wrong isoform or were lost during the correction. LoRMA and PBDAGCon perform poorly for both metrics. In Figure 15, we present the rates of correct base for each read. MSA shows the best correct base rate, reaching almost 100%, followed by Daccord.

With shallow expression (10X, shown in Appendix), correction remains a hard task, as not much information is available for the correction. However, Daccord and our method can increase a the correct base rate.

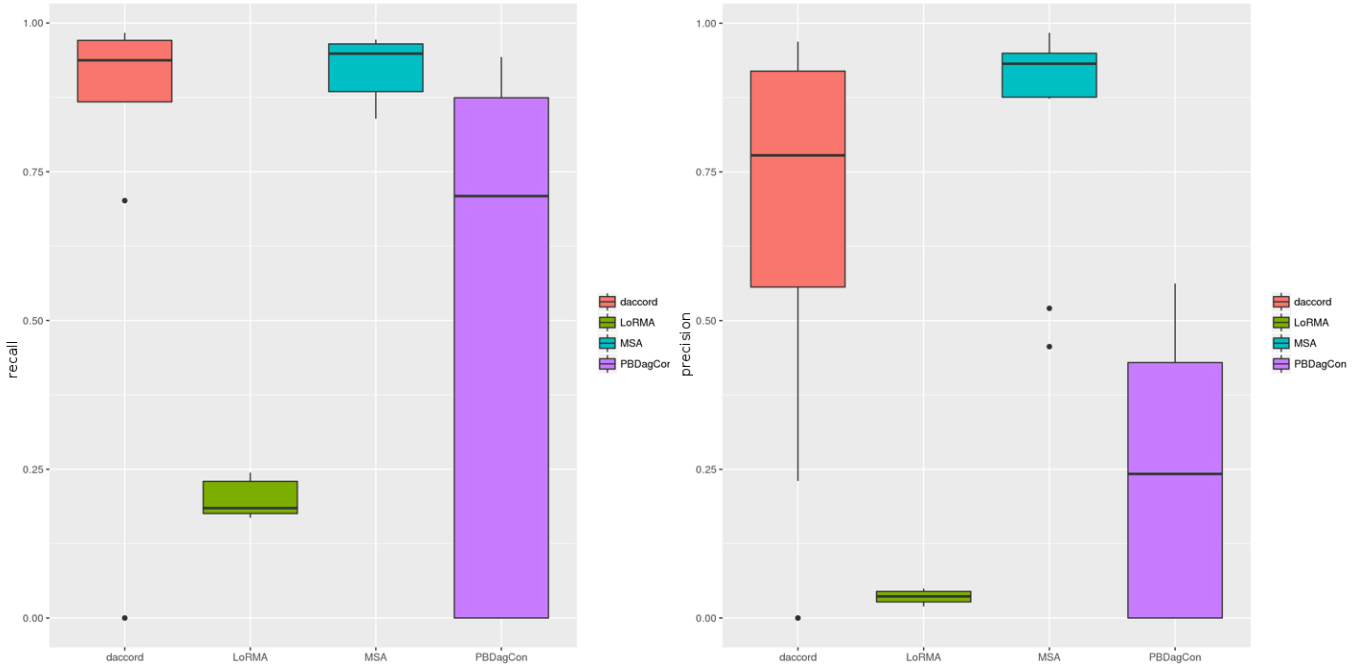

Figure 14: **Recall and precision of correctors on 20X reads.** Left plot: recalls in ordinate. Right Plot: precision in ordinate. These metrics are computed after correction for each read of a 20X experiment, using 4 different correctors in abscissa.

For all correctors, the precision of the correction is impacted. Daccord's precision is more impacted than MSA's, resulting in a difference of correct base rate in Figure 15 in favor of MSA. Other tools downperform. Note that a correct base rate that is lower or equal to the correct base rate in raw reads (0.87 on average in our simulations) means that the methods corrected nothing or that supplementary errors were added in the sequences after correction. At a high coverage (100X), our results show that correctors, in particular Daccord and at the exception of LoRMA, manage to correct many erroneous bases in the reads (shown in Appendix). They thus achieve to rise the number of correct bases in the reads that had initially a mean 87% of correct bases. Our method achieves the best result with over 99% of correct bases. On all coverages, our method competes best other tools' recalls, and has the best precision in all tests, which means it introduces the less new errors while correcting. We showed our method could take advantage of increasing coverages to enhance its correction, and get extremely close to 100% correct bases.

3 Discussion

3.0.3 Current work in progress

We presented a work in progress method to ouptut consensus for isoforms per gene cluster. We insisted on the conservation of isoforms structure after sequence correction, which is a feature not commonly represented in self correctors as show in Chapter 3. Other methods are shaped for genomic applications, and either simply cannot be run on this particular type of data, or output wrong rations. This has direct consequences on isoform discovery, isoform counting and isoform expression levels reported after such correction. On the contrary, our method is shown to strictly respect the isoform ratio, even in the case of a rare minor isoform. Our results also suggest that our method corrects reads while not requiring high coverage, which is a clear

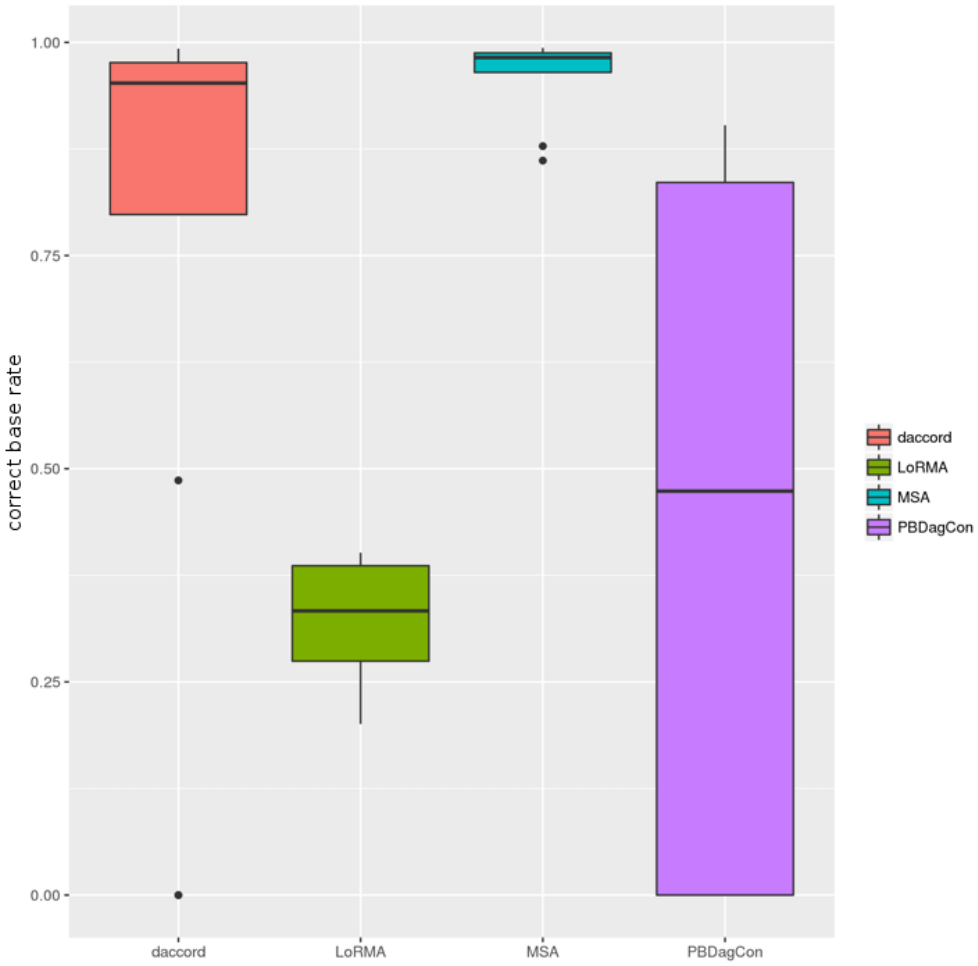

Figure 15: **Correct base rate after correction on 20X reads** Correct base rate in ordinate, computed after correction for each read of a 20X experiment, using 4 different correctors in abscissa.

advantage in RNA context. By both preserving structure and increasing sequence correctness we aim at outputting reliable reads that can be used as references and allow to infer the protein structure. The proof of concept we presented has plenty of room for improvements. Ongoing works include the detection of long gaps induced by errors in homopolymers in the MSA (that can be spotted with the redundant content in sequences), and dealing with stranger reads wrongly included within a cluster. Sequences that are too distant from a cluster will penalize the global alignment and thus the consensus quality. Our results also have to be furtherly validated, using more complex simulations and including errors in homopolymers to the reads. To precisely detect the exon junctions in case of splicing variants in a cluster remains a main challenge given the error rates. Moreover, our method does not allow to build a splicing graph in its primary definition. This is because our method can only detect differences in-between a given cluster. This is a drawback of *de novo* methods that can only extract results from information present in the initial reads, without adding annotations or reference sequences. Finally, we must investigate the maximum resolution of our method, that is, the smallest size of exon we can detect.

The presented consensus module strongly depends on the constitution of correct clusters to work with.

For the moment we use CARNAC-LR as the only tool that can provide such clusters. But each module (similarity detection/clustering/consensus) is currently independent, which allows to exchange with other possible methods. A work in progress is to present the three methods we retained (Minimap/CARNAC-LR/consensus) to process transcriptomics long reads directly embedded into a pipeline. The existence of such pipeline supposes its utilization on real size instances, thus it has to be able to scale these datasets. The consensus part has been tested only on one simulated gene at a time for the time being. However, this procedure can be launched in parallel on each gene, taking advantage of the prior clustering. Then we plan confirmations on real datasets, that always reveal more difficulties than simulated scenarios.

3.1 Details on future implementation of heaviest bundling for consensus calling

With some figures, we explain heaviest bundling [120] advantages and why it can be more interesting than a voting scheme for our application. Heaviest bundling works directly on the POA graph. The idea is to find an optimal path through a POA graph. We give the intuition of the algorithm. It starts by giving a zero score to each vertex of the POA graph (Figure 16 1)). Then, starting by the vertex first ranked in the graph order, it allocates a score equal to the weight of the arc that enters the current vertex plus the score of the source vertex (Figure 16 1)). The first vertex has a score of zero. Weights of arcs correspond to the number of sequences that supported the occurrence of the two consecutive vertices. If a vertex has several predecessors, the most weighted arc is selected (the choice is done on arcs rather than vertices in order to avoid giving to much importance to long insertions that accumulate scores in vertices) (red arcs in Figure 16 2)). If there is a tie between arcs, the source vertex with the higher score is selected (Figure 16 3). In the end a backtrack procedure starts by the vertex with the highest score and traces back a path of vertices with the higher possible score. This procedure can in certain cases produce different results from a voting scheme, mainly because it takes into consideration the local neighborhood of a base using the information that arcs carry. This information is lost during vote. An example is given in Figure 17.

3.1.1 Assess the pipeline results

We pointed out several times in this document the lack of a simulation tool for RNA long reads. We currently work on the conception on this kind of simulator. It consists in a pipeline of several published tools articulated with a novel module for read generation. This pipeline is dedicated to reproduce characteristics of ONT reads.

Several features are required for an adequate simulation. First, gene and transcripts levels should be carefully computed in order to reproduce a biologically sound scenario. Since ONT is still fast evolving, the simulator should be able to adapt to the successive chemistries. Finally it should correctly render the reads common feature so that synthetic versions and real raw reads have close characteristics. Our method is a pipeline that can be divided into four steps. For several of these steps, we relied on well established tools that we articulated with each other. It takes as input .BAM and .BAI from genome alignment as a training read set, FASTA and GTF of a reference genome and the desired final quantity of molecules. The first module builds the error model for reads. As previously mentioned, it is difficult to fit distribution for read errors, in particular since they change according to chemistries. Simulator such as Nanosim made the choice to learn error rates and profiles by training using real read datasets, and so do we. Alignments of reads from a real experiment the user wishes to mimic are passed as input, then the first module automatically deduces error rates and percentages of deletion, insertion and substitution, as well as homopolymer errors, using AlignQC [248]. Pre-computed error profiles can also be used as input. The second module extracts transcripts that will be templates for the long reads from the GTF file of the desired reference using gffreads[1]. The third module is the expression levels definition. In order to simulate expression for transcripts of the input reference, we selected the Flux Simulator that enables detailed gene expression simulation. We use

[1] `http://ccb.jhu.edu/software/stringtie/gff.shtml`

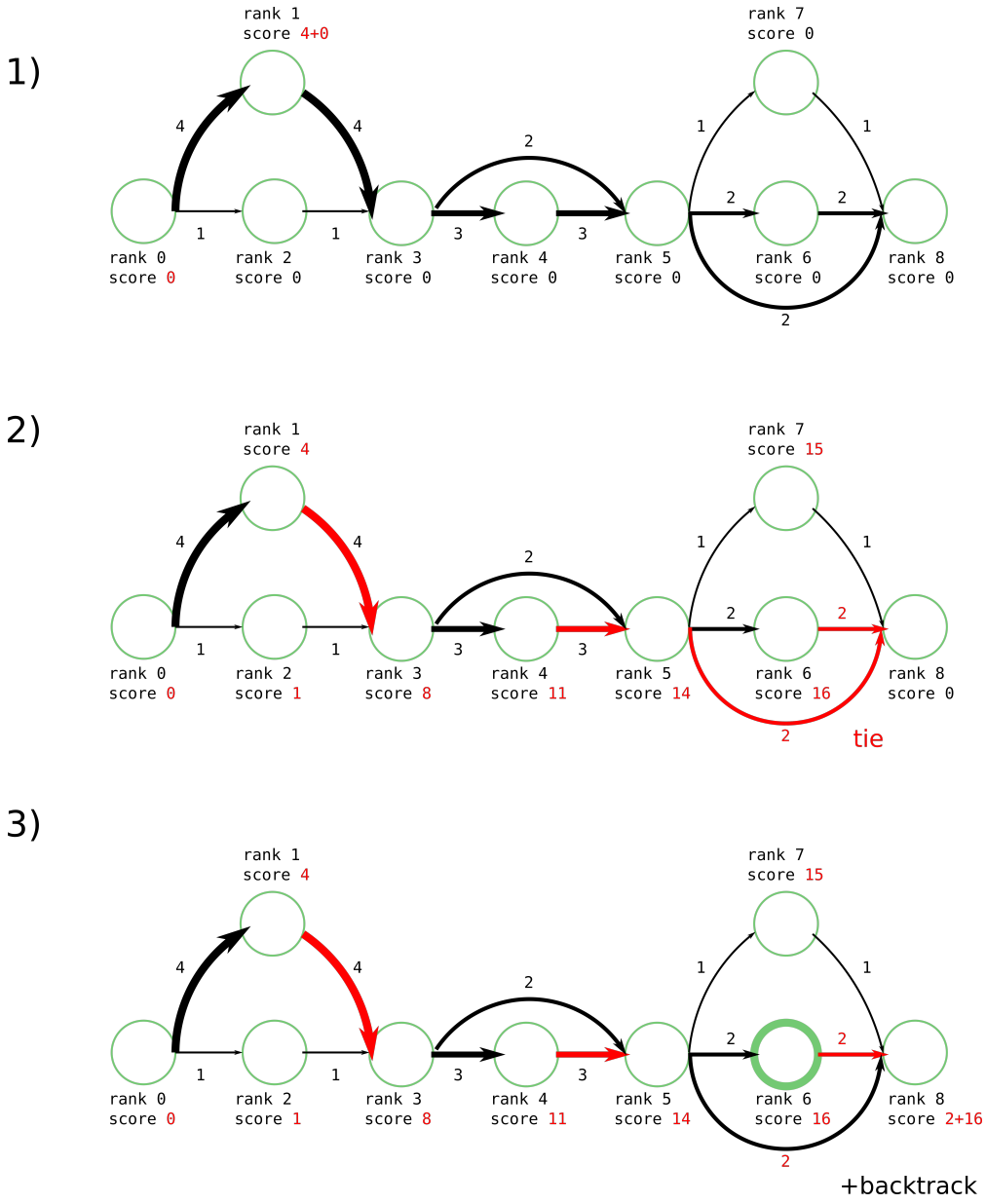

Figure 16: **Heaviest bundling intuition.** 1) initial vertex score is zero and we show how the second vertex gets its score summing the weight of the arc and of the source. 2) most weighted, red arcs are followed when there are several source vertex. 3)in case of tie the predecessor with the higher score is chosen.

expression levels to decide which quantity of reads are generated per read. The last module is a novel and efficient implementation that generates the final reads as well as their errorless versions for comparison matters. It adds the errors at positions in the sequences extracted from the GTF, deals with regular versus homopolymer errors and adds supplementary characteristics such as the staircase effect that affects length distribution, commonly encountered in this type of data. A Figure summaries the pipeline 18. This pipeline

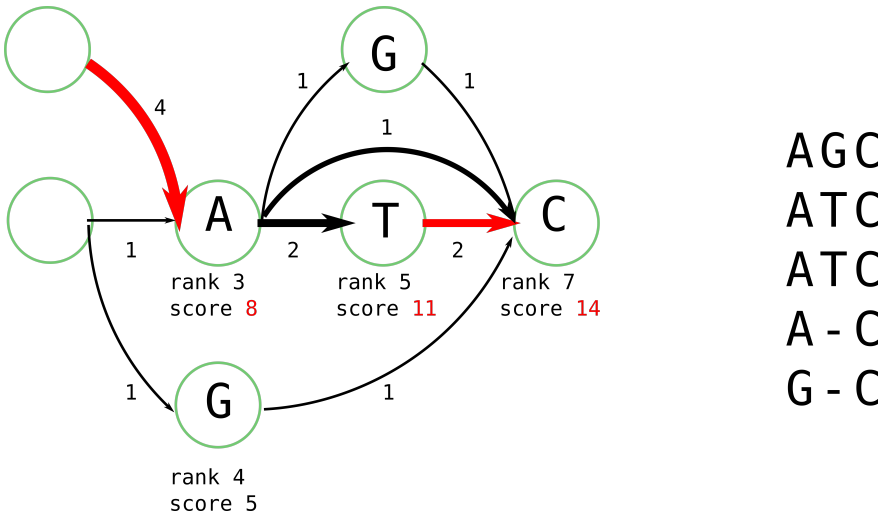

Figure 17: **Toy example comparison of voting and heaviest bundling.** On the left, the consensus is obtained using the graph. On the right, the same alignment is used for voting. Voting leads to make an arbitrary choice between either "T" or a gap in the second column. Heaviest bundling relies on more information, we can see that there are more sequences supporting going through vertex T (2) than avoiding it (1). So when at the last vertex C, the arc coming from T will be chosen. The final path and consensus will be "...ATC".

is a work in progress, already implemented and under testing. In the future it should help to model more complex scenario to test our consensus step.

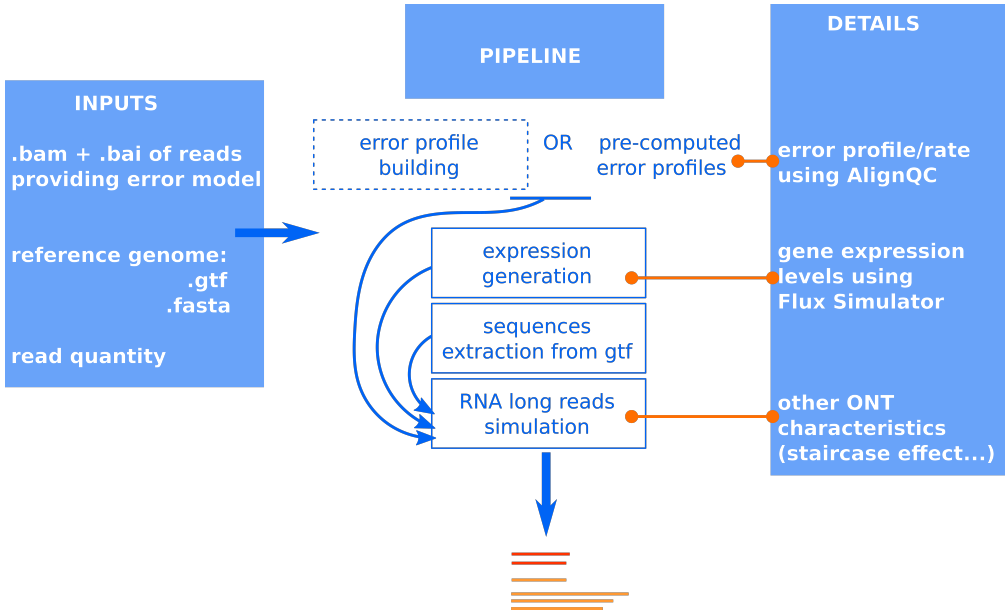

Figure 18: **Pipeline for RNA long read simulation.**

Chapter 5

Other contributions on NGS data

In this chapter we summarize several published works to which we participated, though that do not represent our main contribution during this thesis. All these works deal with NGS RNA-seq data.

1 Context

The different presented works all derive from the local transcriptome assembler KisSplice [200]. As well as classic assembly approaches, it relies on De Bruijn graphs. Contrary to transcript assemblers such as Trinity [79], KiSplice does not assemble full transcripts and rather reports topological patterns (bubbles) created by genomic and splicing variants in the graph. KisSplice original publication details how these bubbles can be enumerated in a DBG. Examples of these patterns in a DBG are shown in Figure 1. One difficulty lies in the fact that genomic repeat can induce several patterns and have to be identified. They can also create extremely complex subgraphs. KisSplice outputs bubbles in the form of a pair of FASTA sequences. Reads are then mapped on those sequences to enable event quantification. KiSplice can work with several datasets and thus deal with experimental plans with sequencing data for different biological conditions.

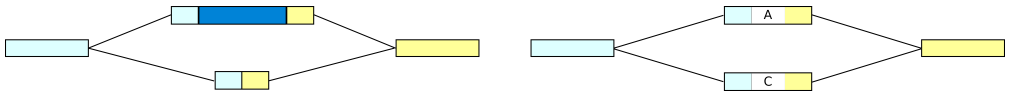

Figure 1: **Bubbles created by an exon skipping (left) and a SNP (right) in a compacted DBG.** Depending on the length of the paths of the bubble, the variation is flagged as an exon skipping (one path longer due to the retained exon, in dark blue on the left example), indel or SNP (both paths of the same lenght, as in the right example). Repeats can generate bubbles but paths usually have different lengths and a small Hamming distance. Each path of the bubble is reported in a pairwise fashion in KisSplice.

2 Dealing with complex regions in graphs

Some experiments we realized for [199] were used in a paper in 2017 [132]. This work deals with the main problem in genome assembly presented in the introduction: repeats. We mentioned that such repeats were also a hurdle in transcriptomics assembly using NGS. These repeats create problematic patterns in the graph that lower the capacity of tools such as KisSplice or other assemblers to retrieve variants of interest. In the complex subgraphs induced by repeats, the number of repeats can grow exponentially, the enumeration of bubbles in these subgraphs thus taking too much time. In Lima et al., a method is presented to avoid such regions, thus improving the results of assembly tools.

The paper shows that repeat-induced subgraphs cannot be directly identified (this is an NP-complete problem) but can be implicitly avoided. Thus, bubbles not present in such subgraphs can be enumerated. By avoiding repeat-induced subgraphs, more variants can be extracted than with previous KiSplice's algorithm. This also allows to flag putative chimeric transcripts that are partly integrated in these regions.

3 Bioinformatics for *de novo* variant discovery

KisSplice and other *de novo* tools based on DBG were presented in [119]. Here we described two applications and methodological developments realized with KisSplice.

3.1 Expressed SNPs

RNA-seq can also be used to investigate expressed SNPs. We took part to the study of Lopez-Maestre et al. 2016 [138] that used pooled RNA-seq samples to discover SNPs without a reference and was applied to non-model species. The study showed that pooled samples from model (human) and non model species enabled the identification, discovery and quantification of SNPs in expressed regions. It highlights that pooled RNA-seq can be an interesting alternative to investigate differential phenotypes when material is scarce for a single individual. This approach was compared to reference-based approaches (MPileUp and GATK [151]) on samples from the GEUVADIS project, most of the SNPs retrieved with these methods were also retrieved with KisSplice. We contributed to the conception of an R Bioconductor package, kissDE[1], for differential variant analysis using RNA-seq. The statistical model and its application to SNPs are presented in this publication.

kissDE With the current depth achieved with NGS, in the case of RNA-seq, the question of functional impact versus noise for variants is often raised for splicing variants. A significant association between a variant and a condition gives clues to point out interesting candidates for further investigation. kissDE proposes a statistical method to test for the enrichment of a variant in an experimental condition. It is meant to work with any pairwise variations: two alleles of a gene, or two splice variants of a gene. It enables to deal with pooled samples.

Tools already exist for the statistical analysis of splice variants across conditions. Either they rely on references (DEXSeq [190], CuffLinks [235]), which makes them not well-suited for non-model species and limited for novel variants, or on a simple model (MATS [215]) which takes only into account single exon skipping and overlooks the variety of possible variant types across species. MISO [101] proposes a nice output format that helps visualising the usage of alternative exons, but does not handle replicates. For SNPs, methods rely on genotypes and do not deal with pooled data. In our package kissDE, we take coverage information for pairwise alternative variants in two conditions or more (they can be SNPs, exon skipping, alternative donnor/acceptor, intron retention, indels...) and test whether a variant is enriched in one condition. kissDE takes replicates into account and requires at least two replicates per condition, with the advantage of not needing any annotation. It uses the counts reported for each variants, obtained for instance by remapping the read of the sequence of events output by KisSplice. kissDE was designed in particular in the scope of KisSplice, and can be easily integrated in a pipeline with tools from KisSplice's suite. Counts are assumed to be distributed as a negative binomial as in standard RNA-seq analysis, and we use the generalised linear models framework to build and compare two nested models with isoforms, experimental condition and interaction in the second model as effects. We do not explicit in detail the statistical model, the reader may refer to [138] and [16]. We select the pairs of variants for which the interaction term has a significant effect with a likelihood ratio test. Then we output p-values and magnitudes of the effect for each pair of isoform. Magnitude of the effect is measured using percent spliced-in (PSI denoted Ψ), that are computed for pairwise variants with $\Psi = \frac{\text{countsvariant1}}{\text{countsvariant1}+\text{countsvariant2}}$. Then the difference between the Ψ in the different experimental condition ($\Delta\Psi$) is output. An example is shown for exon retention in Figure 2. For SNPs, an equivalent metric, Δf_e for difference of allele frequency, is output.

3.2 Alternative splicing studies

We participated in the comparison of an assembly-first and a mapping-first approach to analyze RNA-seq data and find alternative splicing events by Benoit Pilven et al 2018 [16]. We worked on the assembly-first approach with the tool KisSplice.

This work provides several results. First, it proposes a pipeline of both approaches and discussion on the parameters that should be used. Secondly it discusses the complementarity of the two approaches and explains that each performs differently on several instances. As an example, mapping approach deals

[1] `https://github.com/aursiber/kissDE` and `https://www.bioconductor.org/packages/3.7/bioc/html/kissDE.html`

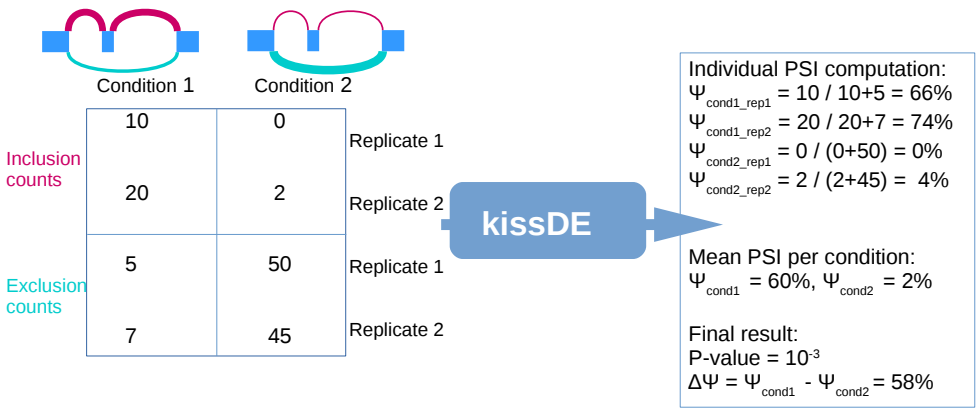

Figure 2: **Input and output of the differential analysis.** Counts for each replicate of each condition are computed for instance using KisSplice. These counts together with the experimental plan are the input of *kissDE*. In this example, we show counts for one single event, in practice *kissDE* tests all events discovered by one method to spot the differential splicing events. Provided that at least two replicates are available per condition, results are ranked using p-values and $\Delta\Psi$.

better with low expressed transcripts because the low coverage does not always allow to assemble completely the variants (some k-mers being missing in the DBG). On the contrary, variants issued from paralog or pseudogenes are easier to retrieve using assembly since they lead to multiple mapping issues. It concludes that being complementary, both approach should be led for more comprehensive transcriptome analysis. Finally, it documents a post-treatment for the study of alternative variants, that can be used after KisSplice when a reference is available, detailed in the following paragraph. Results were validated and supported by experimental results using a human SK-N-SH cell line[2].

kissplice2refgenome kissplice2refgenome enables to classify events reported by KisSplice using a reference genome. Pairwise sequences of bubbles found by KisSplice are mapped to the reference genome using a splice-mapping tool, for instance STAR [52]. The mapping results are then analysed by kissplice2refgenome. Events corresponding map in blocks on the reference genome, the number and size of blocks depending on the type of variants. They are classified in variant types according to the block patterns. For instance, exon skipping maps in three blocks (two flanking, one middle) for the sequence that includes the exon, and only the two flanking blocks are retrieved for the shorter sequence that excludes the exon. The software also reports genomic variants such as indels. In order to discriminative ambiguous patterns, such as intron retention vs deletion that yield same blocks in the alignment, we use a threshold: if the central sequence is longer than this threshold we consider that it is rather a retained intron, in the other case a deletion. We show a summary of block patterns in Figure 3.

[2] `http://genome.crg.es/encode_RNA_dashboard/hg19/35`

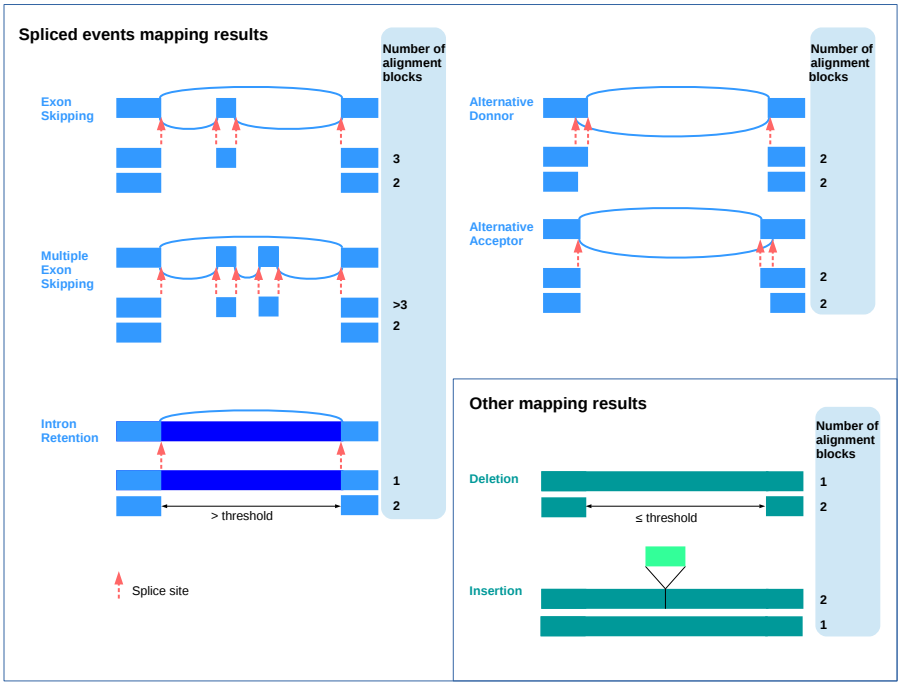

Figure 3: **Classification of KisSplice events according to the number of blocks in which they map to the reference genome.** Paths representing variants of an event are mapped on the reference. Spliced mapping results in blocks, events are then classified by kissplice2refgenome according to the block mapping patterns. (Putative) splice sites are denoted by with a red arrow.

Conclusion and Perspectives

1 Conclusion

1.1 Contributions

As a consequence of the quick evolution of TGS, the sequencing field is frequently upgraded with new types of sequences. For instance, recent long read technology ONT RNA-direct could unlock amplification biases issues in RNA sequencing and thus is promising for gene expression studies. But it shows higher error rates, at least comparatively to current nanopore reads. By proposing generic tools in Chapters 2 (CARNAC-LR), 3 (ELECTOR) and a proof of concept in Chapter 4 that are tailored to these technologies, we wish to promote and encourage a broader use of long reads and in particular ONT reads for transcriptome analysis. We proposed a modular approach composed of independent pieces of software for each task (Figure 1). Any advance in Chapter 1, such as our proposal for long reads, is meant to be channeled to CARNAC-LR. CARNAC-LR itself outputs clusters necessary for computing consensus.

We worked to allow reference-free studies with this kind of data. Such contributions remain rare at the moment and we wish to pursue. In particular, in Chapter 1 we presented a data structure, the Quasi Dictionary, and its applications, Short Read Connector tools, for which there is plenty of room for improvement towards long reads. In the meantime, Short Reads Connector starts to be used in NGS context to analyze metatranscriptomes.

Finally in Chapter 5 we gave a short summary of methodological works on NGS we participated to. These contributions aim at proposing techniques that deal with De Bruijn graphs, extract, identify and analyze variants (splicing variants, SNPs, indels) from RNA-seq data.

1.1.1 A resource-frugal data structure for indexing and querying sequences

In Chapter 1, we proposed a new probabilistic indexation scheme based on a Minimal Perfect Hash Function (MPHF) together with a fingerprint value associated to each indexed element. We have shown experiments on sets containing more than eight billion elements indexed in less than an hour and using less than 25GB of RAM. We proposed two implemented applications: SRC-counter and SRC-linker. The first estimates the abundance of a sequence in a read set. The second detects similarities between pair of reads inter or intra-read sets. Benchmarks against other tools for short reads pairwise comparison show a higher scalability of our method in comparison to classic methods such as BLAST. These works were published in [146]. Both applications were tested on several datasets including metatranscriptomics data from TARA Oceans and holobiont sequencing data of more than five billions sequences.

SRC-counter is currently used to help identify holobiont actors in very large scale experiments regarding marine holobionts. It is integrated in an assembly pipeline as a pre-processing step to assembly. We demonstrated that our approach was biologically sound by comparing our results to well-broached models of marine holobionts. Using this pipeline, we could partly retrieve actors of a not well-known protist-protist holobiont, provide and annotate transcripts with low rate of chimeras. This work has been accepted for publication in the journal Microbiome [153].

Preliminary inputs for SRC's adaptation to RNA long reads shown that SRC could be tuned to gain in speed and precision. We highlighted that a tool dedicated to retrieve pairwise similarities between RNA long reads still misses in the literature, however new advances are proposed in Minimap2's recent publication [127]. We published these results in [145]. We proposed new algorithmic solutions to fit both RNA (read local comparison to find common exons) and reads' error rates and profiles (using a generalized k-mer concept), that still have to be further tested.

1.1.2 Clustering RNA long reads from a transcriptome sequencing per genes

In Chapter 2 we presented a community detection method that took elements from clustering paradigm to retrieve reads corresponding to expressed genes. We intend to demonstrate practical advantages of this method on general community detection problems, however we have already demonstrated it could achieve a better definition than *modularity*-based methods on a classic theoretical problem instance. The main interest

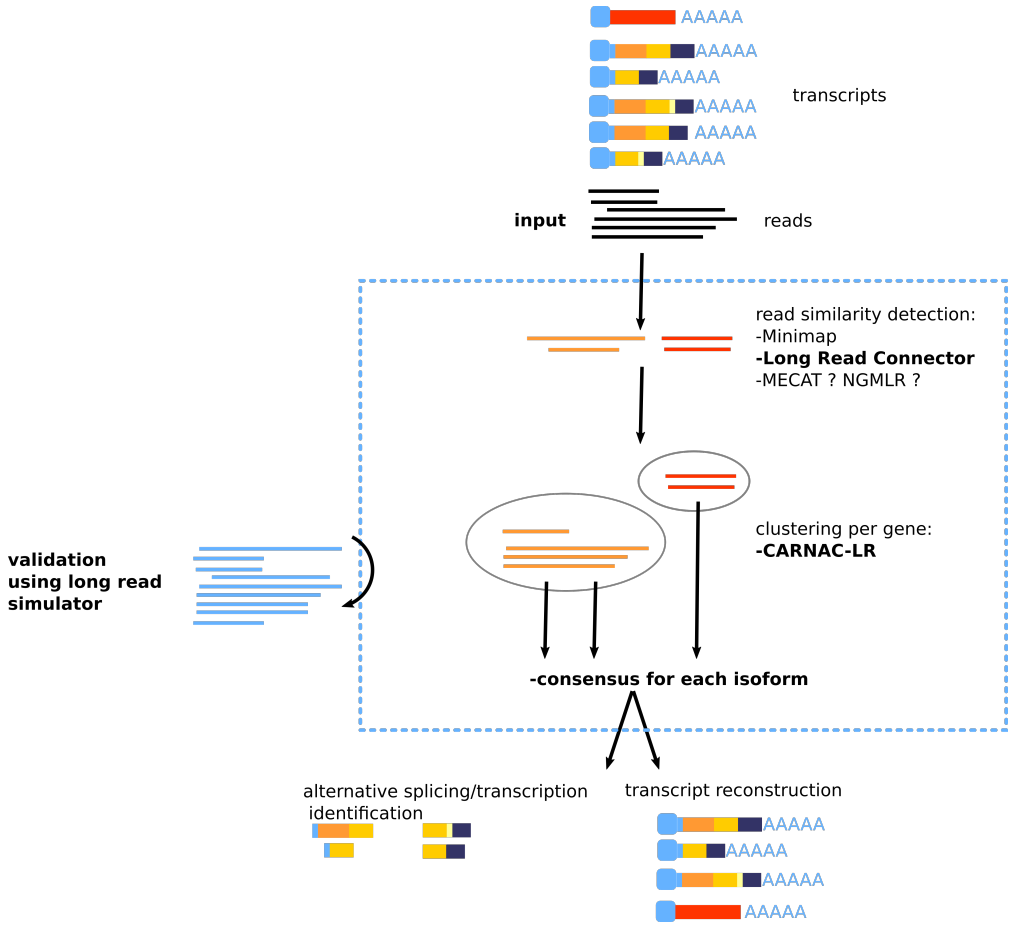

Figure 1: **Advances on transcriptome long reads sequencing pipeline.** Several solutions are proposed in Chapter 1 for the first step (pairwise comparison), including work in progress Long Read Connector. CARNAC-LR (Chapter 2) stands as the only tool to perform *de novo* clustering on ONT. The consensus step enables to obtain better quality sequences and detect alternative events at once (Chapter 4). This pipeline is scheduled to be further validated using real data and simulated data as introduced in Chapter 4.

of this method is that it combines compatibility to nanopore reads and *de novo approach*. Other methods rather focused on PacBio or work with a reference. We proposed an implementation called CARNAC-LR based on our community-detection algorithm combined to Minimap, designed for clustering long reads obtained from transcriptome sequencing in groups of expressed genes. It imposes no parameter choice to the user nor to have to select between different results such as multi-level clusterings. It takes raw long reads without filtering or correction needed. This is an important point since we demonstrated in Chapter 3 than RNA reads could be altered by correction methods. From the clusters output by CARNAC-LR, the expressed variants of each gene are obtained and related transcripts are identified, even when no reference is available. We demonstrated our method's relevance on a mouse transcriptome sequenced with ONT MinION and compared it to other community detection approaches and other sequence clustering of the literature. This was also an occasion to illustrate that *de novo* and reference-based method can be complementary in an analysis, since we have shown mapping approach and CARNAC-LR had different pros and cons. This work is in revision in NAR Methods [144].

1.1.3 Read correction/consensus and conservation of isoforms

Chapter 3's results are twofold: we introduced a tool for efficient assessment of long read correction methods and we illustrated major caveats in RNA long reads corrections. Our evaluation tool, ELECTOR, is meant to be adaptable to many different sequencing scenarios (diploid genomes, large genomes, transcriptomes) in order to help users to best define the correction they need. ELECTOR summaries the quality of the correction through various metrics. An interesting contribution in addition to ELECTOR's global pipeline is the *seed-MSA* approach used to compute main correction metrics. This approach is a multiple sequence alignment heuristic that allowed a critical gain in speed (we have shown several orders of magnitude of decrease for our runtime). This work is not published yet.

As for RNA correction, we used simulated data to better understand how different transcripts from a same gene were treated by long reads correctors. We pointed out that a main issue of self-correctors was that the conservation of alternative isoforms structures was not ensured by current correction methods. Hybrid correctors tend to not correcting a fraction of the read, or to fragment them thus loosing the long range information that is a main interest of long reads. We conclude that no corrector is completely fitted to correction of transcriptomic long reads. This lack in the state of the art motivated the work presented in Chapter 4.

In Chapter 4, we propose to compute one consensus per isoform (with in addition its count) from a transcriptome long reads sequencing instead of trying to correct reads with state of the art methods. We presented a proof of concept dedicated to produce consensus of the different isoforms in presence for each gene. This method is meant to work together with a clustering such as proposed by CARNAC-LR. The preservation of the structure was the main aspect dealt with. As in Chapter 3, we used simulated long reads for validation. Our method works *de novo* and using only long reads information. We thus compared it to self-correctors of the literature. We demonstrated in preliminary results that we could achieve better correction that state of the art methods. In particular we carefully conserve each alternative isoform present for a gene while other methods tend to collapse some isoforms, leading to information loss. This contribution is still a work in progress. In order to generate more complex scenarios that are closer real data, we are working on a RNA long reads simulator. This tool reproduces reads characteristics, as well as gene expression and alternative transcripts that are distinctive features of transcriptome sequencing. We aim at generating full transcriptome simulations on which we will train our consensus method.

1.2 Dissemination of this work

Reproducibility All softwares developed during this work are open source, versioned and available on GitHub under GPL license. Short Reads Connector relies on the GATB library which allows it to be more easily maintained. In order to help reproducibility of our experiments, when necessary we made available the scripts we used for validation, such as recall/precision/Jaccard index scripts for CARNAC-LR. Similarly, data used for the experiments is publicly available (TARA Oceans and mouse transcriptome mainly). We also have seen the holobiont paper manuscript as an opportunity to have a discussion about the right set of parameters to use with our tools, which we think is fruitful to the users. ELECTOR and the RNA simulation pipeline are also efforts towards reproducibility and production of documented benchmarks.

Projects that use our work SRC is the most stable and first tool we proposed. Therefore it is the first to be used in projects outside GenScale team. SRC is still employed in "Analyse des Données à Haut Débit en Génomique" team after our collaboration on non model holobionts. New unpublished radiolaria/dinoflagellates holobionts and non symbiotic radiolaria metatranscriptomes are currently investigated using SRC with novel strategies described in the following discussion section. The first objective is to give access to high quality transcripts on the bases of the pipeline described in Chapter 1. Then, these transcripts should help to outline genes implied in the symbiosis using sequence similarity networks.

SRC is also a part of a *de novo* pipeline for detection of small genomic variants. These variants are used in ecology to retrieve species and population structures [70].

CARNAC-LR starts to be used in a project that aims at identifying parts of chromosomes that lead to similar functions in different species of frogs (*Xenopus*) using Nanopore cDNA sequencing.

Future of this work We have shown that sequencing data undergoes quick evolution. Though we anticipated the needs concerning long reads in this work, we also think that many short reads await to reveal interesting results, however they escape from what methodology proposes. This is the case in huge data instances or for organisms without a reference as shown with holobionts. There are always less methods that offer to work reference-free than methods based on a reference, and we think this will remain an interesting front. *De novo* works helps to be more distant from the established models, and can be used even for model organisms to complement mapping-based approaches. However, it is a work in itself to well understand pros and cons of each approach. Short reads will remain a cheap access to wide depth that are crucial to investigate metagenomics, metatranscriptomics and transcriptomics. This is why works committing to lightweight data structures will remain a key for years. Thus novel algorithms or data structures such as the Quasi Dictionary or Short Reads Connector remain of interest even if technology changes.

2 Perspectives

2.1 Short Reads Connector at its best

2.1.1 Improvements on holobionts

Our first suggestion is to assign assembled transcripts to holobiont actors instead of assign reads. This would give an equivalent or greater assignation power since assembled transcripts would be longer: we could use larger k-mers to obtain more precise matches and compute a result over more k-mers in query sequences. More detailed similarity information would then be needed order to make the difference between chimeras and transcripts pertaining to the "shared" category. Another improvement that can be realized independently or in addition to this first one is to set up an iterative pipeline for SRC on holobionts (Figure 2). We saw that when actors were not well identified, the recall of the method was low. In order to increase it, we propose to iteratively enrich the banks. After a first assignment round with SRC, holobiont reads linked to an identified group (host/symbiont) can be added to the reference libraries. Then, based on these new enriched libraries, a second run of SRC can be performed on the holobiont reads. This can be implemented as an iterative pipeline: at each round, more reads will be assigned to the host or symbiont categories and will then be used as reference libraries.

2.1.2 Ideas for a "Long Reads Connector"

First, new benchmarks should be realized with the most up to date tools to distinguish which are the best to retrieve overlaps between pairs of RNA long reads. This will be the occasion to test our quasi k-mer approach. Krizanovic et al. proposed an evaluation pipeline for long read splice-mapping [111] using simulated reads, we could on the other side propose an evaluation pipeline for the overlap detection in this context. NGMLR is a mapper that was designed to retrieve genomic structural variants using long reads. These problematics can intersect splice-mapping so this tool could be interesting to evaluate. However, it is mostly developed to compare long reads to a reference, which can be a limitation. Recent, unpublished results show that Minimap2 performs well for the support of RNA reads mapping on genome[3]. Finally, new strategies such as the one employed in the correction tool MECAT [252] should be assessed. Authors of MECAT also pointed out that read comparison before the correction task itself was an important bottleneck. They also use a window strategy, as presented in the last sections of Chapter 1, and integrate a quick step to ensure the colinearity of matched k-mers that has the same goals that the seeding strategy proposed in Chapter 3.

[3] `http://bioinfo.zesoi.fer.hr/index.php/en/blog-en/56-gmap-vs-minimap2`

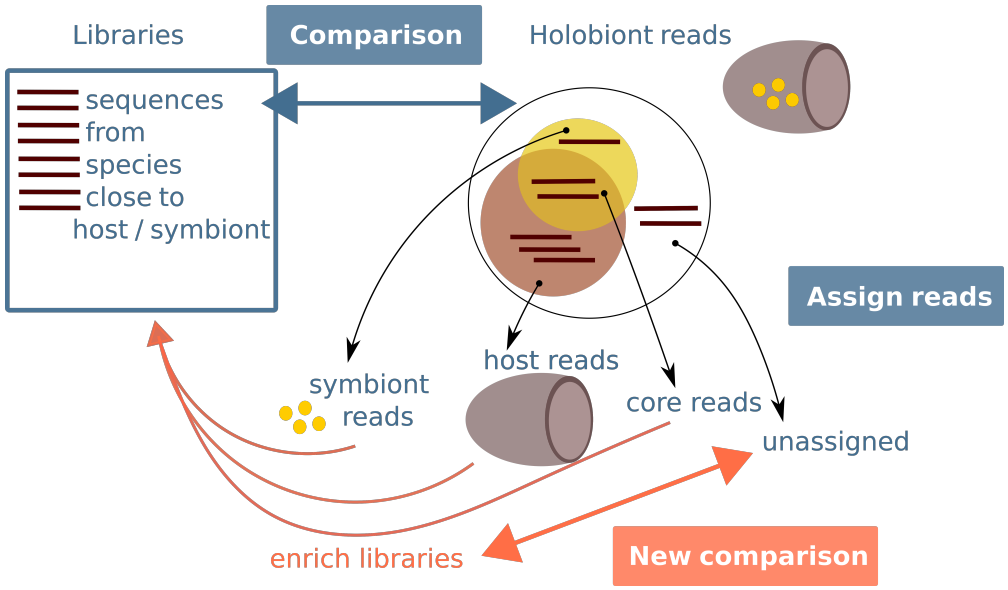

Figure 2: **Iterative pipeline for SRC in holobiont application.** Sequences assigned by SRC after a first pass are used to complete the banks and a new round is performed on the remaining reads.

2.2 Enhance CARNAC-LR

2.2.1 Food for algorithmic thoughts

We plan to test CARNAC-LR's algorithm (without pre-processing) on other classic problems of community detection using benchmarks such as proposed in [115]. in order to better situate our method in comparison to other state of the art approaches.

With mouse we shown that both recall and precision of CARNAC-LR could be enhanced. In order to see how to increase recall, we could extend the recall and precision defined in Chapter 2 to the second most covered cluster. This way, we could assess whether most expected clusters are retrieved in a single main CARNAC-LR cluster, or if they are divided in several CARNAC-LR clusters of the same range of size (Figure 3). According to the results, we will adapt strategies to fish back reads. We would also like to look in which cases isoforms are separated. The presented recall does not take into account that some isoforms do not share any exon with the rest of the isoforms expressed by a gene, thus mechanically cannot be put in a same cluster.

Cutoff rounding is for the moment very simple, we could set up strategies to find a result closer to a local minimum for the cut. Pre-processing can also be enhanced with for instance the removal of bridges (edges that connect bi-connected components) instead of articulation points. We also plan to test if several passes of this algorithm on the biggest connected components is linked with improvement in computation speed and clusters quality. Finally, the mapping method used has impacts on the input graph. We currently work at better assessing which mapper gives the best results, and at showing whether clusters remain globally the same according to mapping methods and parameters or not. In the meantime we also study how robust the algorithm is to the order in which nodes of the graph are passed in the input.

2.2.2 To go further

As mentioned in Chapter 2, for the moment the distinction between families of genes seems out of reach. Similarly, the presence of heterozygous genomic variants does not impact and is not reported by the clustering.

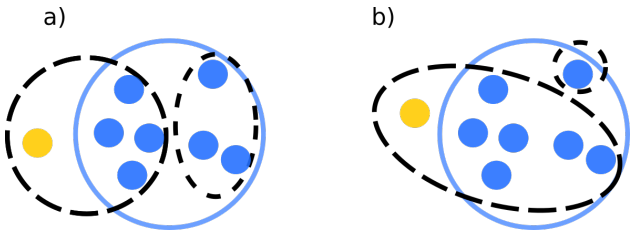

Figure 3: **Analyze of ground truth versus result clusters.** Ground truth cluster is shown in colored, plain stroke. Result clusters are dashed. In scenario a), the blue ground truth cluster is over-clustered in more than one main result cluster. In scenario b), the majority of the information contained in the blue ground truth cluster is captured by a single result cluster. Other results clusters covering it are small or singletons.

In order to make CARNAC-LR fit more with the content of current experiments, we think of integrating short reads in some ways. The first idea would be to correct long reads using short reads prior to clustering, which should help cleaning the similarity graph. Uncorrected reads could be kept. However, this has the disadvantages of current correction methods presented in Chapter 3: possible isoform collapsing and loss of long range information. Another way is to assemble short reads into contigs and to integrate these contigs to the similarity graph. Given their lower error rate, they should increase the connectivity of the genes they belong to, thus making easier to delineate communities.

A last, very natural application for CARNAC-LR would be metatranscriptomics. The main properties of datasets remain the same, however this would be substantial work to make CARNAC-LR scale to larger datasets.

2.3 Read correction

2.3.1 RNA short Read correction using graphs

As an extension of the work proposed in BCOOL [133] corrector, we would like to correct RNA short reads using two solidity thresholds, an absolute threshold to remove extremely low covered unitigs and local relative thresholds such as used in KisSplice [200] or the read corrector Rcorrector [222] to make the difference between variants and errors and conserve the information of variable regions. The difference between this proposal and the literature is that we would integrate information at the read level to decide whether a path represents an error, while Rconnector uses extremely local information (successor k-mers of a branching k-mer), and can therefore take wrong decisions due to local coverage depletion. Similarly, only very local information is taken into account in KisSplice and reads are not corrected. An intuition of the increased accuracy that would be proposed by such a method in comparison to state of the art is shown in Figure 4.

Not only a more conservative correction would be useful to the numerous RNA-seq projects, but we would also like to better apprehend the influence of a good correction of short reads on hybrid correction. Our intuition is that well-corrected reads could better map (for methods that directly map short reads on long reads) or provide better assemblies (for methods that use unitigs of graphs as correction templates).

2.3.2 Long read self correction

Our work on long reads correction led us detect some of the main pitfalls of current methods. We started very preliminary work for a self corrector that would avoid long pairwise or multiple alignment steps generally required at the beginning of the correction process. We think that a seed-chaining strategy such as the one discussed for ELECTOR in Chapter 4 could replace the alignment step while being more efficient, as advocated in Minimap's paper. With the set of seeds, we can delineate regions in reads that can be corrected

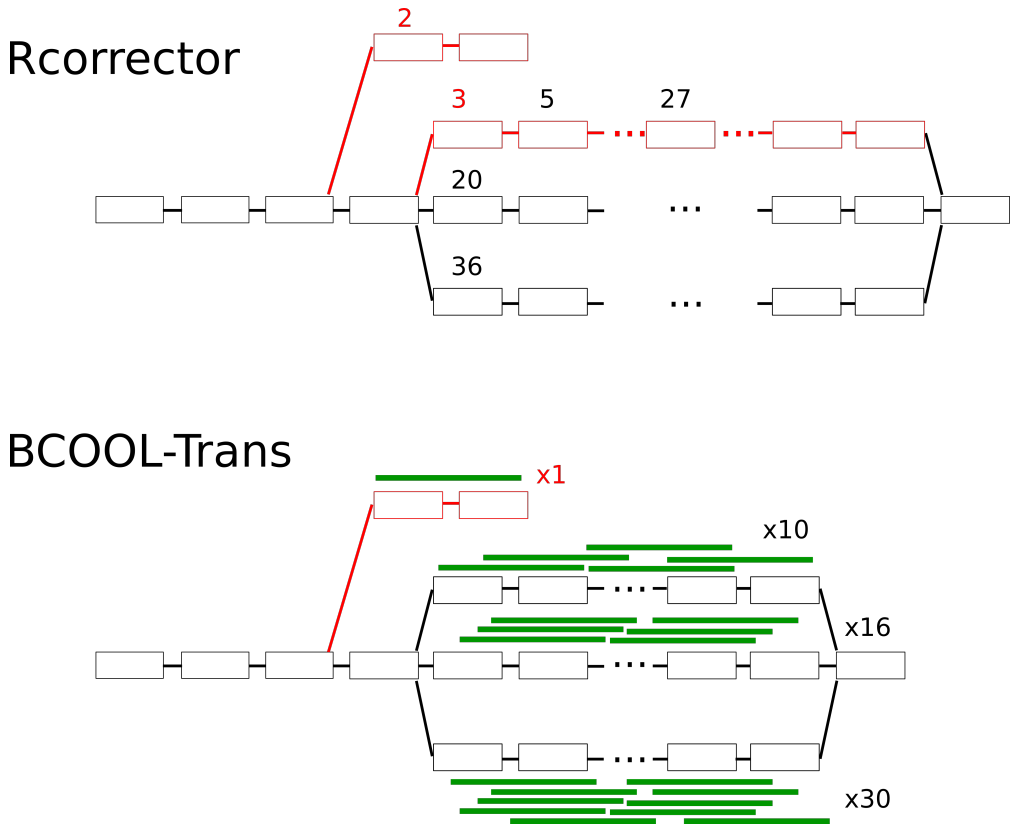

Figure 4: **Comparison between Rcorrector's approach to detect errors in the graph and an approach based on BCOOL (BCOOL-Trans).** In Rcorrector, both red unitigs will be flagged as erroneous paths because their local k-mer coverage is low relatively to other possible paths. k-mers occurrences are reported above their corresponding vertices. Our approach would only discard one of these two branches, because it would take into account information carried by reads mapping to each paths instead of k-mer counts. In this example the branch discarded by Rcorrector because of a local low k-mer count is in fact supported by a relatively abundant set of reads and will be kept as a correct path. Reads mapped on those paths can be then corrected. In the case of Rcorrector such reads would have been wrongly converted to sequences of other paths.

together. Then, according to the complexity (i.e. level of errors) of the region to be corrected, we design an adaptive strategy for consensus calling. Many regions can be corrected using a MSA and heaviest bundling such as described in Chapter 4. We mentioned in Chapter 3 that several methods applied De Bruijn graph to obtain consensus from a set of data. For the most complex regions, we plan to use "micro-assembly" (i.e. assembly of very small k-mers, of size 5 to 10) of k-mers found in the dataset between seed pairs with a De Bruijn graph to extract consensus. Consensus obtained by one or the other strategy that can be injected in reads to correct errors. We show an outline of the procedure in Figure 5, and we give further details for this method in the Appendix.

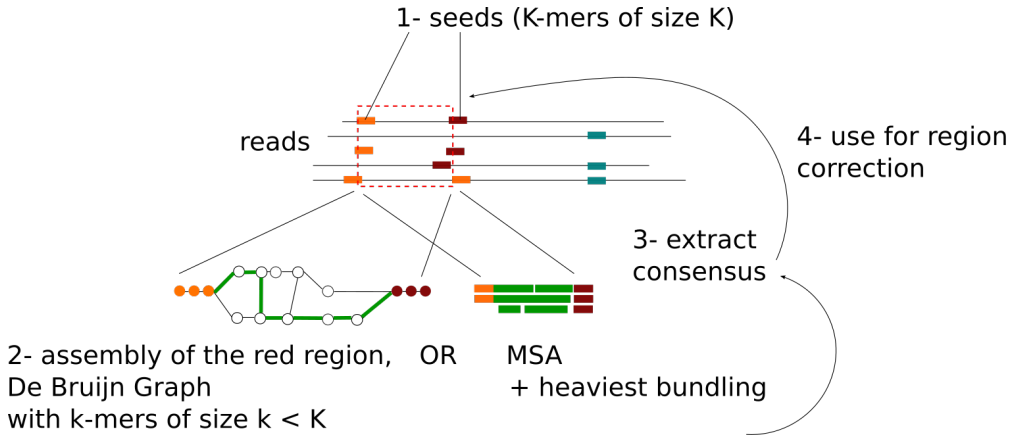

Figure 5: **Outline of the correction using seed chaining and De Bruijn graph based micro-assembly.** The red dashed region is corrected by micro-assembly of small k-mers of this region, or MSA from sequences of this region. Other, not shown, regions are defined between pairs of seeds.

2.4 Towards a comprehensive pipeline for *de novo* study of transcriptomes with long reads

2.4.1 Enhance the consensus step

Currently our consensus step is based on multiple sequence alignment, that can be rather long if reads are long or if clusters are large. We can think of several strategies to speed-up this step. First, the MSA could be replaced by heuristics such as the one used in PBDAGCon. A partial order graph is also built but sequences are mapped onto a single sequence instead of mapping to the graph. However, the graph is updated after each alignment in a quite similar way than Lee's algorithm. Other possibilities include the adaptation of the seed-POA strategy presented in Chapter 3 to RNA.

For highly expressed genes, it is possible that we can afford to sample before alignment. However, this has to be shown. Experiments tend to show than from 30X coverage the maximal quality for consensus is almost reached. This means that, using exon-correction strategy, only a fraction of sequences of the regions could be used to compute consensus (in case of highly expressed regions), thus making the alignment quicker. Finally we plan to implement and test heaviest bundling presented in the end of Chapter 4 for consensus calling.

2.4.2 CARNAC-LR and consensus pipeline

The idea is to channel clusters found by CARNAC-LR to the consensus step. Such pipeline was the underlying condition to have input clusters for the method presented in Chapter 4. This is a work in progress, as well as the integration of a mapping tool to compute the similarity graph. An advantage is that consensus computation can be computed in parallel for several clusters. The simulation tool we currently work on will help us precisely assess clustering and consensus results, coupled with ELECTOR.

We plan to validate the simulation pipeline using mouse transcriptomic data, we will show the properties of our simulation and compare them to available mouse transcriptome ONT reads sequenced on the MinION platform at the Genoscope. Such pipeline could also be compared to Tofu on PacBio reads.

2.5 Final note

Principal contributions of this work relate to the treatment of long reads of transcriptomics sequencing, aiming at retrieving the content in alternative transcription and splicing variants in these datasets. We also presented works regarding data structures scaling meta genomics or transcriptomics projects, and several applications to short reads. We think that hybrid methods that integrate short and long reads should be considered, not by using short reads to correct long reads but to really exploit the potential and different characteristics of each technology. Future contributions include two main axis: efficient methods for metatranscriptomics (through space and time saving data structures) and *de novo* methods for handling transcriptomics long reads (notably good consensus methods still lack).

Appendix

1 Appendix to Chapter 2: details on methods

1.1 Algorithm

We provide an outline of CARNAC-LR's main algorithm. The split procedure we refer to is described in the main document.

Algorithm: Main
Data: Graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$
Result: A cutoff CC_{min}, and a partition P of $\mathcal{N}$
$P = \{\mathcal{C}(n_1), \mathcal{C}(n_2), ...\mathcal{C}(n_k)\}$ such that
all $\mathcal{C}(n_i)$ have a clustering coefficient $\geq CC_{min}$,
and result in a minimal cut of $\mathcal{G}$ with value Cut_{min}
foreach *node* n *of* $\mathcal{N}$ **do**
    Compute its degree $deg(n)$;
    Compute its clustering coefficient $CC(n)$ (equ. (1) in main document);
$\mathcal{N} \leftarrow \mathcal{N} \setminus \{n \mid CC(n) = 0$ or is an articulation node in $\mathcal{G}\}$
$S\mathcal{N} \leftarrow sort\_deg(\mathcal{N})$; *% Sorted list of nodes*
$CC_{min} = 0$; $Cut_{min} = \infty$
% Loop over possible cutoff values
foreach *cutoff sampled in CC* **do**
    % Compute possibly overlapping clusters
    foreach n *in* $S\mathcal{N}$ *such that* $CC(n) \geq$ *cutoff* **do**
        $\mathcal{C}(n) \leftarrow \{n\} \cup neighbours(n)$; *% Initial clusters*
        $S_{cl}(n) \leftarrow$ Sorted list of clusters containing n in decreasing value of CC
        $\mathcal{N}_{Inter} \leftarrow$ Nodes in the intersection of 2 clusters
        $\mathcal{N}_{cl} \leftarrow sort\_degc(\mathcal{N}_{Inter})$

    % Make a partition from overlapping clusters
    foreach *node* n *in* $\mathcal{N}_{cl}$ **do**
        $x = S_{cl}(n)[1]$ *% Representative element of the cluster*
        foreach y *in* $S_{cl}(n)[2 : length(S_{cl}(n))]$ **do**
            $CC_{xy} \leftarrow$clustering coefficient of $x \cup y$ (equ. (2) in main document);
            if $CC_{xy} \geq$ *cutoff* **then**
                $x \leftarrow$ **Merge**(x, y); *%* $\mathcal{C}(x) = \mathcal{C}(x) \cup \mathcal{C}(y)$
            else
                $x \leftarrow$ **Split**$(x, y, \mathcal{E})$
                % n is discarded from one of the clusters
                % x refers now to the cluster containing n
    Cut$\leftarrow$ number of inter-cluster edges in $\mathcal{E}$;
    % Update partition with the minimal cut value
    if $Cut < Cut_{min}$ **then**
        $Cut_{min} \leftarrow Cut$; $CC_{min} \leftarrow$cutoff; $P \leftarrow \{\mathcal{C}(n)\}$

Figure 1: **Main algorithm for the clustering based on minimal cut to find pseudo-cliques.** The Split step is detailed in Chapter 2. The Merge step is not detailed as it is more trivial. Procedure sort_deg sort nodes in decreasing value of deg, then of CC. Procedure sort_degc sort nodes in decreasing value of deg, then of CC for the representative of the cluster.

1.2 Example of problematic nodes

We recall that the auxiliary condition for a node n_i to be a community seed is

$$\forall n_i, ClCo_i \in]cutoff, \theta_2[\Rightarrow deg(n_i) \leq \theta_1 \tag{5.1}$$

θ_1 and θ_2 are the values such that 1% of the observed degrees are greater than θ_1 and 1% of the observed $ClCo$ are lower than θ_2 (1st and 99th percentiles). It was motivated by empiric observations such as presented in Figure 2. We show, using values computed on the whole mouse transcriptome datasets, that some nodes fall in the worst case scenario that we control using the second condition to be a seed node. These nodes are then prevented to form clusters. We show examples of such nodes in the mouse dataset in Appendix.

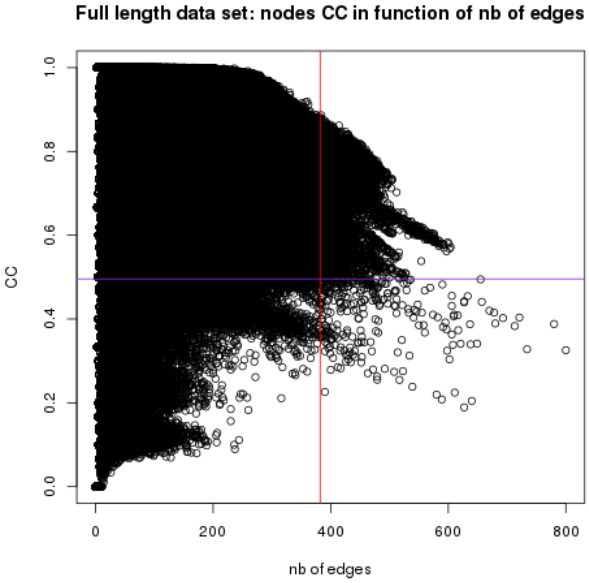

Figure 2: **In this figure we present the $ClCo$ and degrees computed on nodes from the whole mouse dataset.** We plot the degrees 99th percentile and $ClCo$ 1st percentile in color. We can see that a small population of nodes are both very well connected and have a very low $ClCo$ value. They are likely to resemble stars.

2 Appendix to Chapter 4: supplementary results

We provide supplementary results to show our consensus approach results on more scenarios, using different coverages.

2.0.1 Isoform structure conservation

First we plot results for isoform conservation at low and high coverages that were not shown in main document (Figures 3). Results for both shallow and high coverage do not differ from 20X coverage presented in Chapter 4. With a higher coverage, the main result does not change even if other correctors than MSA seem to follow a bit more the trends of each ratio.

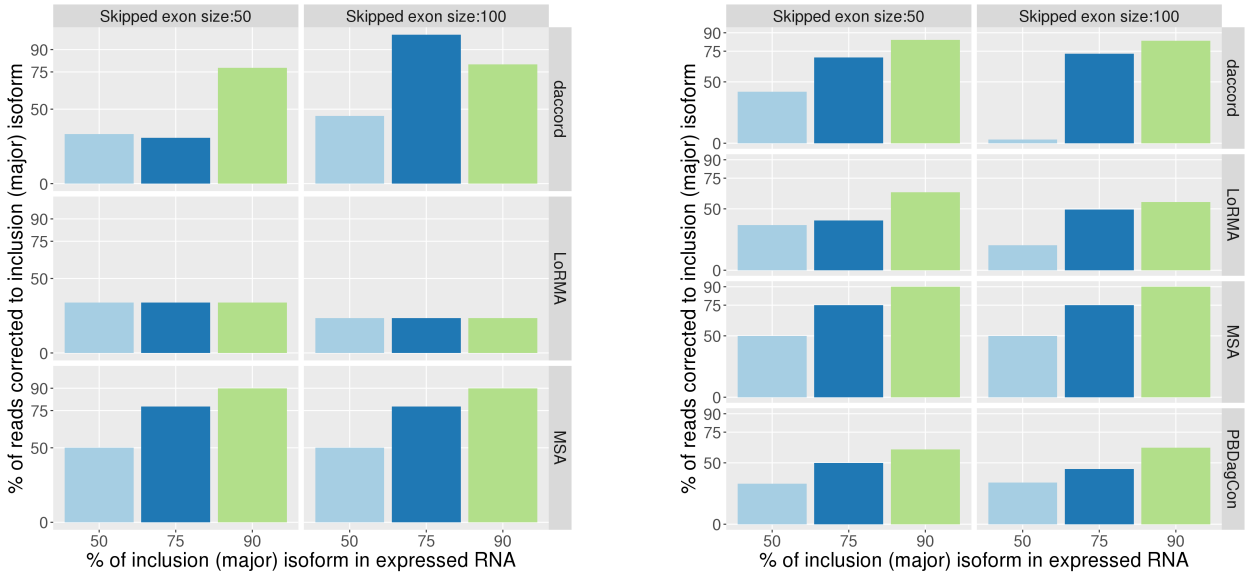

Figure 3: **Preservation of isoform ratio for low and high coverages** Input (abscissa) and output (ordinate) major isoform ratio for varying rate of major isoform (abscissa), varying skipped exon size (vertical blocks), for each corrector that could be launched (horizontal blocks). For a 75% rate of major isoform, the ratio is 75/25, and it is expected that the colored bar goes up to 75% in ordinate so that the corrector preserved the ratio. If the colored bar is upper (respectively lower) the 75%, the corrector output more (respectively less) major isoform reads than there were in the input. Low coverage is the left plot (10X), high coverage is the right plot (100X).

We also show corrected reads size over real molecule size ratios for shallow and high coverages (Figure 4). MSA's ratio remains close to 1 for shallow coverage. Other methods output reads shorted than expected. Higher coverages allow them to get closer lengths.

2.0.2 Base-wise correction quality

In this paragraph we detail recall, precision and correct base rates output by methods at shallow and high coverages.

Daccord and MSA manage to correct most of the erroneous bases with low coverage (PBDAGCon cannot be run). However, we show in Figure 5 that for all correctors, the precision of the correction is impacted. Correct base rates thus follow the same trends (Figure 6).

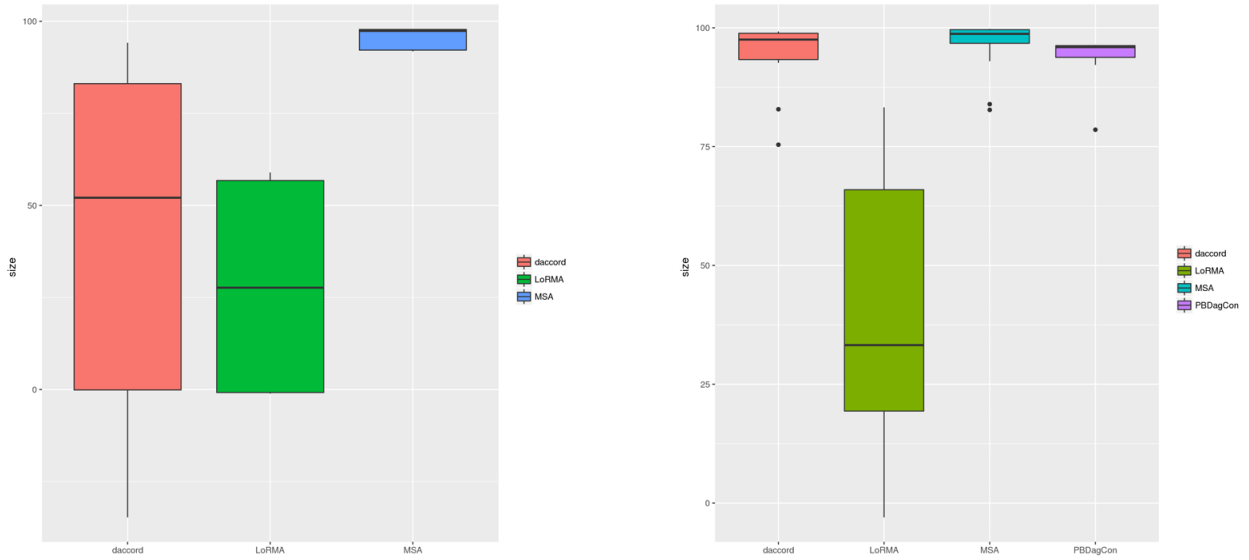

Figure 4: **Ratio of corrected over real isoform length in corrected reads** Low coverage is the left plot (10X), high coverage is the right plot (100X).

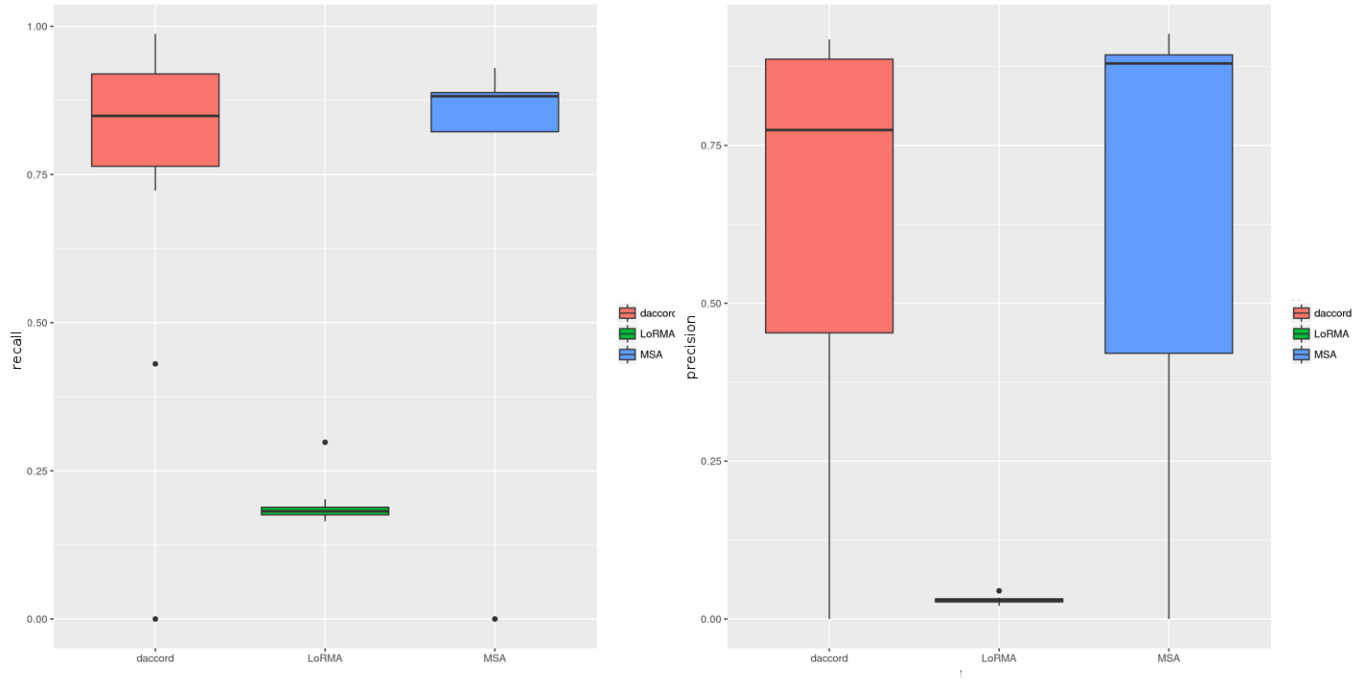

Figure 5: **Recall and precision of correctors on 10X reads** Left plot: recalls in ordinate. Right Plot: precision in ordinate. These metrics are computed after correction for each read of a 10X experiment, using 3 different correctors in abscissa.

Figures 7 and 8 show that Daccord, PBDAGon and our approach took advantage of the higher coverage to perform correction.

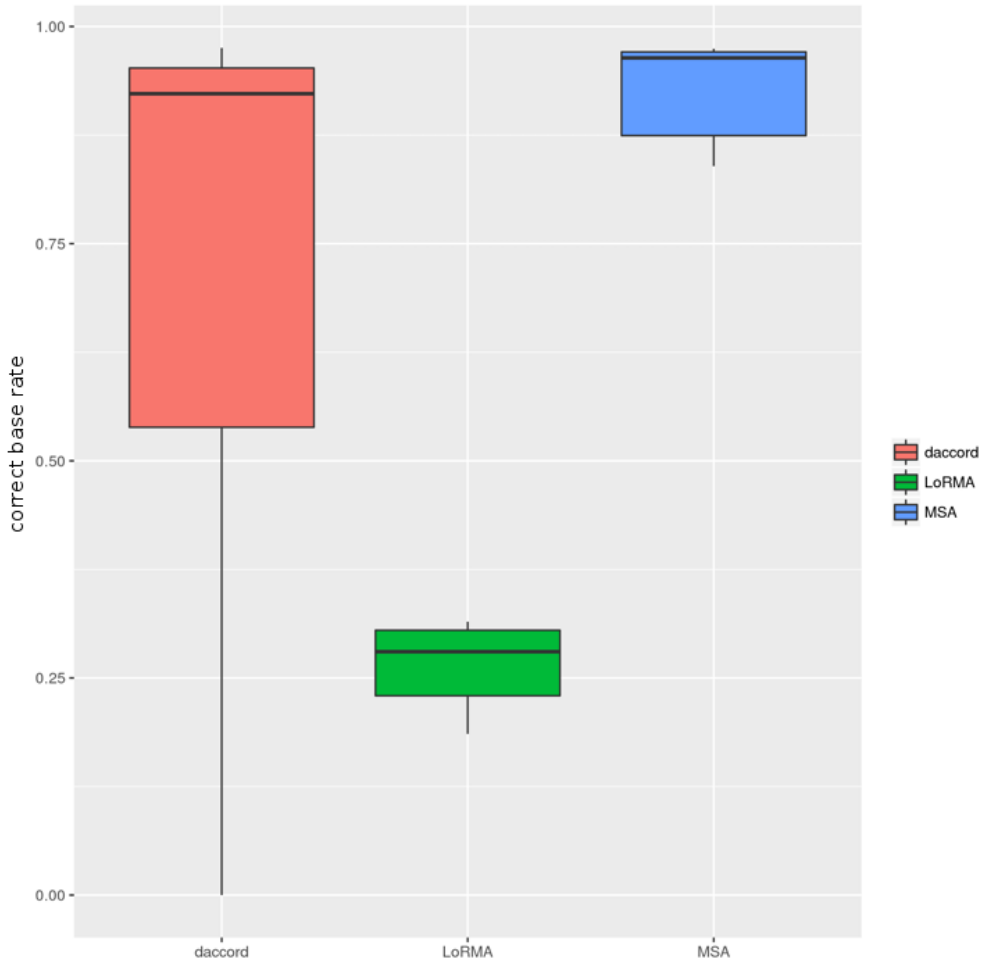

Figure 6: **Correct base rate after correction on 10X reads** Correct base rate in ordinate, computed after correction for each read of a 10X experiment, using 3 different correctors in abscissa.

3 Appendix to conclusion: long reads correction

3.0.3 Extract subsequences from a read set for correction

We deliver more details on our ideas for long read self correction. The very first step consists in finding similar regions in reads in an alignment-free way. Contrary other long read correction methods, we propose an alignment-free scheme that should enable to speed up the initial step that all correctors share: finding similar regions to be corrected together. This step allows to separate reads in several subsequences that will be independently corrected. These subsequences are shared by several reads in the set, and will be compared with each other in some way to build a local consensus. The same seed chaining paradigm than in ELECTOR (Chapter 3) is used, though generalized to n reads instead of a triplet. In ELECTOR we required that all three reads harbor all seeds. In this case, since we compare a larger set of reads, it is very unlikely to find a sufficiently large set of seeds shared by all reads that respects the conditions given in Chapter 3 and have the advised size for seeds given the error rate, i.e. around 15 nucleotides. Thus, we relax the condition that

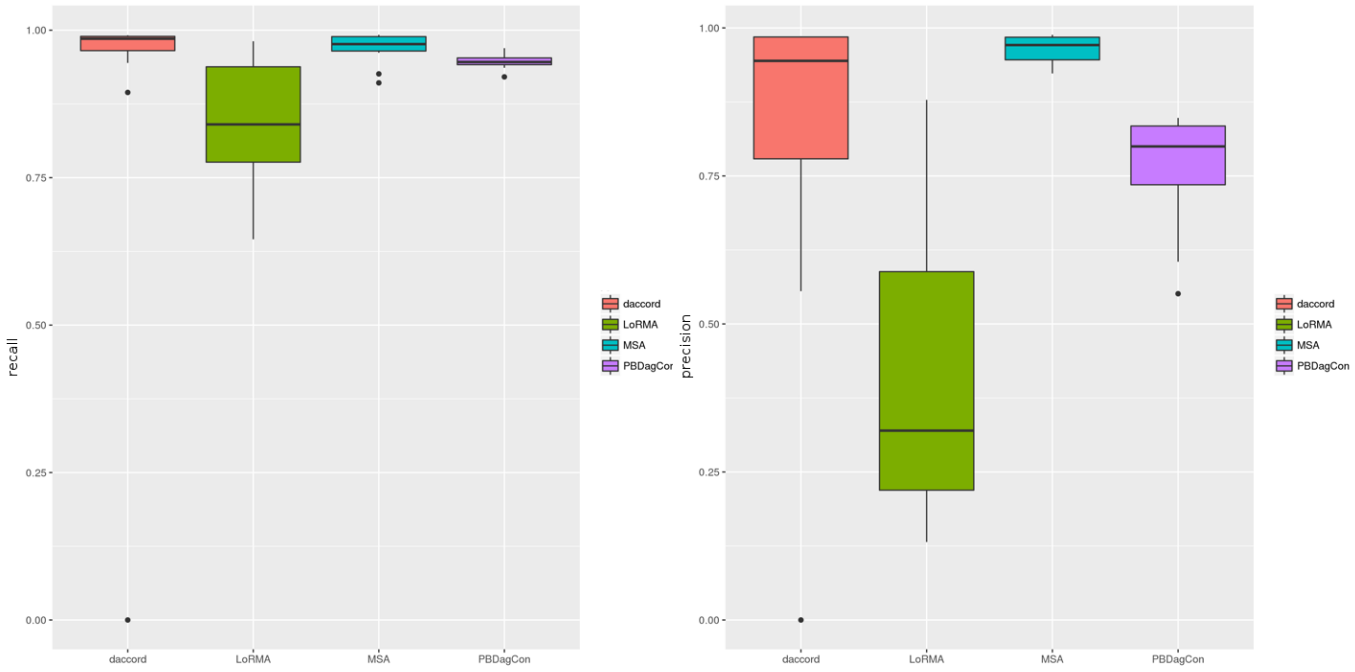

Figure 7: **Recall and precision of correctors on 100X reads** Left plot: recalls in ordinate. Right Plot: precision in ordinate. These metrics are computed after correction for each read of a 100X experiment, using 3 different correctors in abscissa.

seeds must be present in all reads, and introduce a parameter that is the minimum number of reads that must contain the k-mer so that is can be a seed. Other condition (non repeated k-mer, order) remain the same. We can also compute an approximate distance between consecutive seeds using the occurrences of seed pairs seen in reads. We have the following information at our disposal (Figure 9):

- ◇ a set of seeds,
- ◇ their order (how seeds are placed relatively to one another from left to right in the read set),
- ◇ which read contains which seeds,
- ◇ distances between seeds.

Two consecutive seeds flank subsequences in reads, this set of subsequences is called a region. Since not all reads contain all pairs of seeds, we introduce a supplementary strategy to extract the subsequences of a region when seeds are missing in a read. If one seed of the pair is present, using the distance between this pair of seed, it is possible to estimate where the other seed must be placed in the sequence, then to extract the corresponding region (Figure 10). We still have to define to which extend distances must be used. While distance between consecutive seeds seem rather safe, distances between distant seeds may suffer from bad estimation due to the accumulation of errors in the sequences. In the example of Figure 10, the green region is not extracted from the second read because the next seed in this read was too distant from the current region.

This procedure is repeated for each region. We end up with several subsets of subsequences from reads to be corrected.

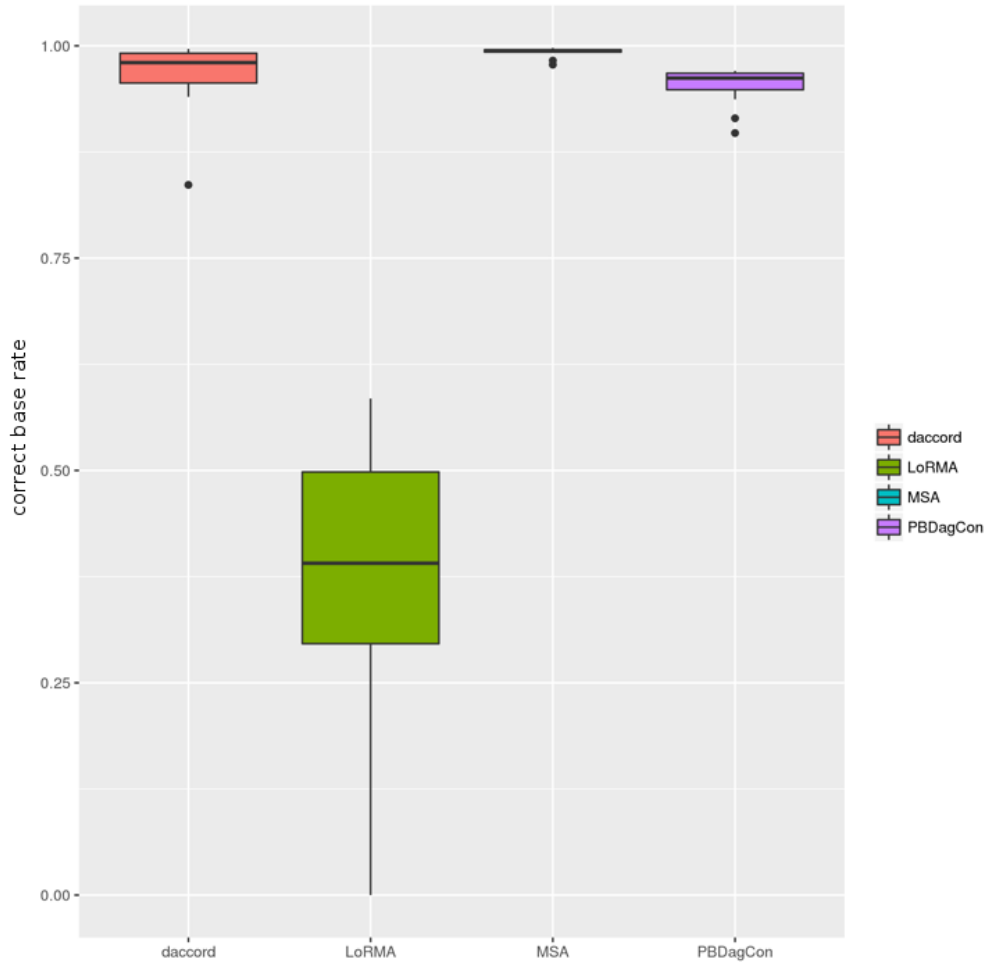

Figure 8: **Correct base rate after correction on 100X reads** Correct base rate in ordinate, computed after correction for each read of a 10X experiment, using 3 different correctors in abscissa.

3.1 Compute a consensus per region

Our work with ELECTOR, as well as other correction methods, suggested that multiple sequence alignment is a strong tool to extract accurate consensus from a set of sequences. However, it suffers from tractability limits when large, very divergent and numerous sequences are compared. The division in regions described earlier aims at reducing the size of the multiple alignment problems to smaller instances, just as in ELECTOR. In practice, we think that regions must be at most a hundred bases long. However, sometimes it is impossible to find a set of seeds so that all regions are rather small, moreover even in small regions, some subsequences can be particularly divergent. Thus, we introduce a prior step to MSA that aims at assessing the complexity of the region, and an alternative strategy for consensus computing in complex regions.

First, for a given region we extract small k-mers from its sequences, together with their occurrences. These k-mers are smaller than the k-mers used to find seeds (we think of a size 5 to 9). Using these k-mers we build a De Bruijn graph. Contrary to De Bruijn graphs usually encountered in practice for assembly, this De Bruijn graph is small (build on short sequences, with less k-mers due to the sequences and k-mers size),

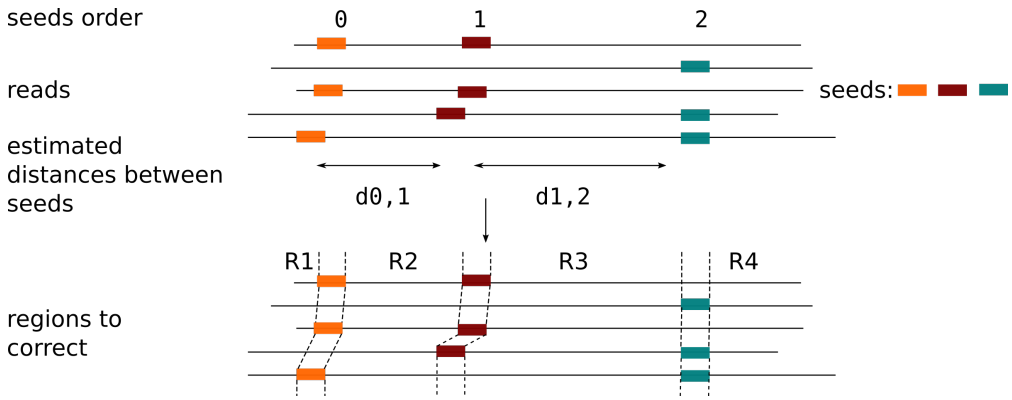

Figure 9: **Seed strategy to divide reads in regions to be corrected.** Seeds (colored) are found in a set of reads, they are ordered and distances can be approximated between pairs of seeds. Subsequences between two consecutive seeds are said to be included in a "region" that will be corrected. Flanking subsequences before the leftmost and after the rightmost seeds are also regions. In this example we define four regions using three seeds. In read 4, the subsequence is extracted using the distance between seed 0 and 1.

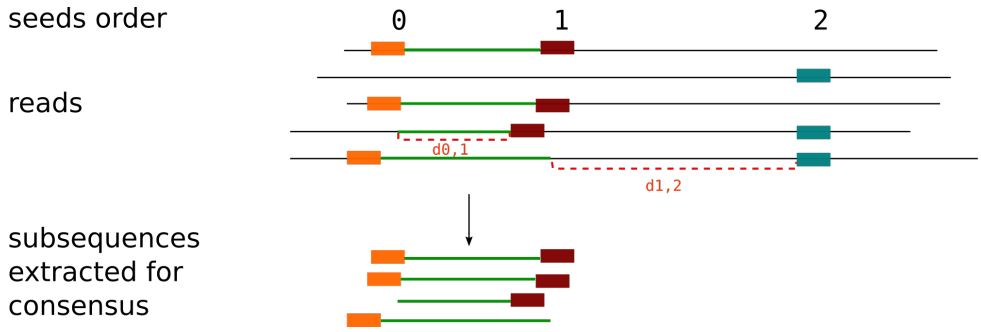

Figure 10: **How subsequences are extracted from a given region.** When the pair of seeds that delineate a region is present, the subsequence is trivially extracted. Note that we include both seeds in the sequence. When one seed is missing, it can be retrieved using pairs of seeds distances recorded, if the next seed is not too far. In read two, the next seed is too far to safely extract the subsequence of the green region.

which means storing associated pieces of information to k-mers (such as occurrence, sequences of origin...) remain cost efficient. By measuring the complexity of the graph (i.e. whether it contains a lot of intricated bubbles), we reach a proxy of the complexity of the region. A region with not too many errors will produce a rather flat graph (right example in Figure 11) while error-prone regions will produce dense graphs (left example in Figure 11). Flat graphs indicate that the region can be processed by MSA to find out a consensus. The MSA can be realized using POA such as in Chapter 4, which has the advantage to allow to directly apply a heaviest bundling algorithm on the partial order graph (described in Chapter 4) to find a consensus.

A complex region will be longer to align using MSA. In this case, we propose to use De Bruijn graph traversal instead. Using the build De Bruijn graph and information its nodes carry, we can seek a heaviest path based on the occurrences of k-mers. Several algorithms are possible, we plan to first test a greedy way that elongates the path by selecting the next most weighted k-mer. The rationale of using such small k-mer

size is twofold:

- ◇ the smaller the k-mers, the most likely they appear in many reads, thus the correct version of a region is solid in the graph
- ◇ the graph has less chance to be fragmented: we can find a single path that traverses the region to correct it

A key is that the space between two seeds must be rather small (at most a hundred nucleotides). Assembly at such size is possible if the interval between two seeds is not too long, thus k-mers are not likely to be repeated. Moreover, consensus will be more easily computed from small subsequences. This is also true for the MSA.

Both MSA and De Bruijn graph consensus must start and end by seeds sequences. In the De Bruijn graph, an initial path must be found between the k-mer that once concatenated, yield the initial left seed sequence. The same requirement exists for the right seed. We can imagine starting from each extremity of the graph and looking for pairs of path that meet at some point in the graph (a strategy also employer in Daccord). In the MSA, seed sequences are conserved and should be aligned altogether across sequences, thus can be precisely spotted in the partial order graph. The heaviest bundling must be performed starting at the first nucleotide of the left seed and end at the last right seed. This is an important condition since extra (or missing) nucleotides may be introduced when extracting subsequences that are not flanked by both seeds of the pair.

The De Bruijn graph strategy has the advantage to be more time-saving but can have limitations in repeated regions. The MSA strategy is more conservative and handles repeats, outputs consensus that are expected to be more accurate, but only on rather close sequences.

In both case, we obtain a consensus that is a single sequence. Using distances between seed pairs to compare with the length of the consensus can help us to quick assess if our result is incorrect. Thus, too short or too long consensus regarding the estimated distance can be discarded to make the correction more conservative. There must exist case where no correct consensus can be found.

3.2 Read correction using the consensus

Once a consensus is found for a region, it can replaced the corresponding subsequence in the reads. The trivial case is to replace a read's subsequence when the two seeds are present (Figure 11 and Figure 12 1)). In other cases, it is sometimes still possible to correct even when successive seeds are not present in a read. Once all consensus are found, if all consensus between two given seeds could be computed, they can be concatenated and replace the sequence between the two distant seeds (Figure 5 2)). If a consensus could not be found for a given region, it is not corrected in any reads. If the concatenation of several consensus includes a region where no consensus was found, it is not performed (Figure 5 3)).

3.3 Remarks

This preliminary work is based on several parameters: size and number of seeds, size of regions, size of small k-mers for the DBG. Several experiment in Daccord's preprint should help us to rasterize our set of parameters. Daccord's approach also advocates for sliding windows. In our case, we only presented no-overlapping regions. Overlapping regions could be investigated as well. In order to enhance the path finding in De Bruijn graph, that is sensible to repeated k-mers, we can imagine to build a colored De Bruijn graph in which k-mers recall to which sequence they belong. This could allow to follow path that are not chimeric. Repeated k-mers could also recall how many times there were seen repeated on average in the sequences, to allow finding walks in a controlled way (i.e. going though a node a limited number of times). Finally, this work is at the moment designed in the genomic case. At the moment the limiting step is that we do not have a method to find chained seeds in the case of alternative sequences.

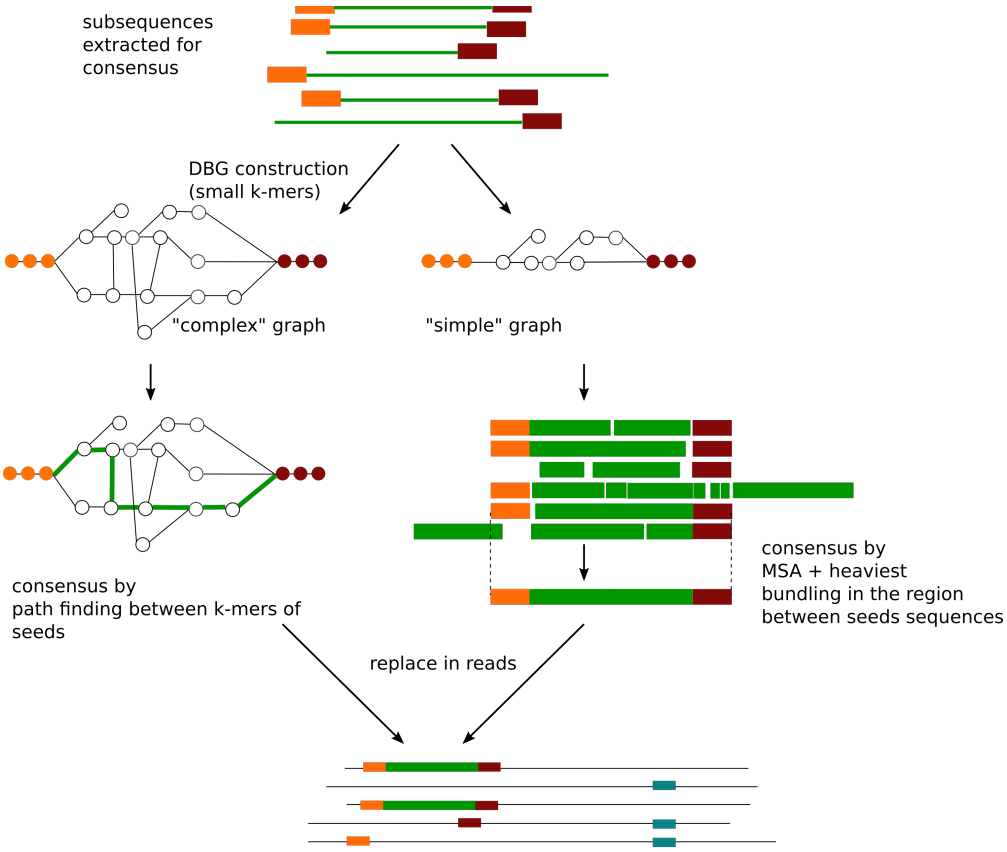

Figure 11: **Consensus strategies.** A De Bruijn graph is first built from small k-mers extracted from the region. Using this graph we assess if the region contains a high rate of errors. We adapt our strategy according to the complexity of the region. Complex regions are treated by path finding in the De Bruijn graph, the consensus being a path supported by k-mers of high occurrence in the set. Less complex regions are processed by MSA + heaviest bundling to output accurate consensus.

Bibliography

[1] Abdel-Ghany, S. E., Hamilton, M., Jacobi, J. L., Ngam, P., Devitt, N., Schilkey, F., Ben-Hur, A., and Reddy, A. S. (2016). A survey of the sorghum transcriptome using single-molecule long reads. *Nature communications*, 7:11706.

[2] Adamcsek, B., Palla, G., Farkas, I. J., Derényi, I., and Vicsek, T. (2006). Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023.

[3] Alberts, B. (2017). *Molecular biology of the cell*. Garland science.

[4] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

[5] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J.

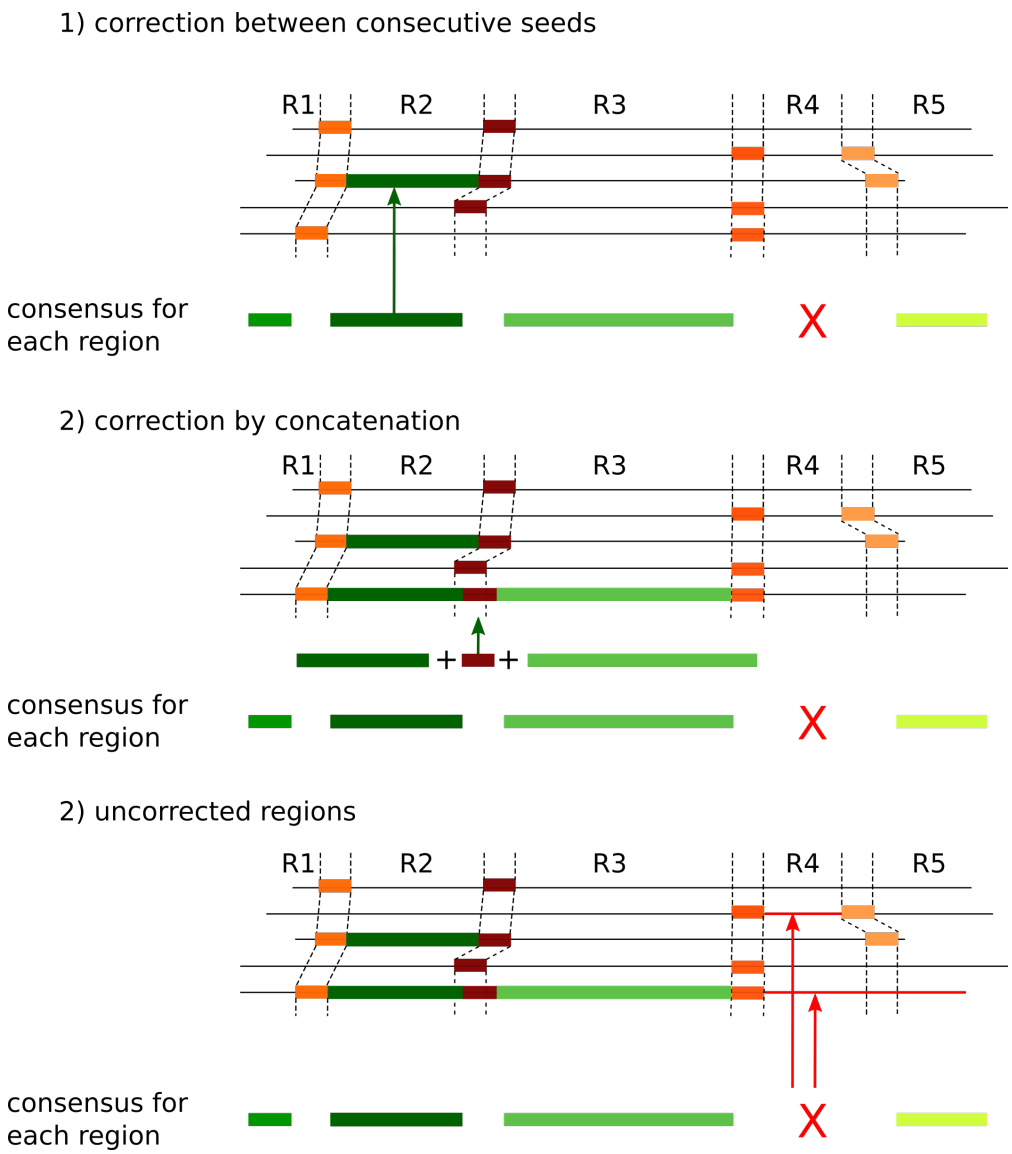

Figure 12: **Strategies to correct read with consensus.** 1) Replace a subsequence flanked with consecutive seeds with the consensus. 2) Concatenate consensus to correct a larger subsequence between distant seeds. 3) Missing consensus lead to uncorrected regions.

(1997). Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402.

[6] Anamika, K., Verma, S., Jere, A., and Desai, A. (2016). Transcriptomic profiling using next generation sequencing-advances, advantages, and challenges. In *Next Generation Sequencing-Advances, Applications and Challenges*. InTech.

[7] Andersson, L., Archibald, A. L., Bottema, C. D., Brauning, R., Burgess, S. C., Burt, D. W., Casas, E., Cheng, H. H., Clarke, L., Couldrey, C., et al. (2015). Coordinated international action to accelerate

genome-to-phenome with faang, the functional annotation of animal genomes project. *Genome biology*, 16(1):57.

[8] Au, K. F., Sebastiano, V., Afshar, P. T., Durruthy, J. D., Lee, L., Williams, B. A., van Bakel, H., Schadt, E. E., Reijo-Pera, R. A., Underwood, J. G., et al. (2013). Characterization of the human esc transcriptome by hybrid sequencing. *Proceedings of the National Academy of Sciences*, 110(50):E4821–E4830.

[9] Balzano, S., Corre, E., Decelle, J., Sierra, R., Wincker, P., Da Silva, C., Poulain, J., Pawlowski, J., and Not, F. (2015). Transcriptome analyses to investigate symbiotic relationships between marine protists. *Frontiers in microbiology*, 6:98.

[10] Bao, E., Jiang, T., Kaloshian, I., and Girke, T. (2011). Seed: efficient clustering of next-generation sequences. *Bioinformatics*, 27(18):2502–2509.

[11] Bao, E. and Lan, L. (2017). Halc: High throughput algorithm for long read error correction. *BMC bioinformatics*, 18(1):204.

[12] Barash, Y., Calarco, J. A., Gao, W., Pan, Q., Wang, X., Shai, O., Blencowe, B. J., and Frey, B. J. (2010). Deciphering the splicing code. *Nature*, 465(7294):53.

[13] Bastide, M. and McCombie, W. R. (2007). Assembling genomic dna sequences with phrap. *Current Protocols in Bioinformatics*, pages 11–4.

[14] Benjamini, Y. and Speed, T. P. (2012). Summarizing and correcting the gc content bias in high-throughput sequencing. *Nucleic acids research*, 40(10):e72–e72.

[15] Benoit, G., Peterlongo, P., Mariadassou, M., Drezen, E., Schbath, S., Lavenier, D., and Lemaitre, C. (2016). Multiple Comparative Metagenomics using Multiset k-mer Counting. pages 1–17.

[16] Benoit-Pilven, C., Marchet, C., Chautard, E., Lima, L., Lambert, M.-P., Sacomoto, G., Rey, A., Cologne, A., Terrone, S., Dulaurier, L., et al. (2018). Complementarity of assembly-first and mapping-first approaches for alternative splicing annotation and differential analysis from rnaseq data. *Scientific reports*, 8(1):4307.

[17] Berget, S. M., Moore, C., and Sharp, P. A. (1977). Spliced segments at the 5́terminus of adenovirus 2 late mrna. *Proceedings of the National Academy of Sciences*, 74(8):3171–3175.

[18] Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015a). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630.

[19] Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015b). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630.

[20] Biard, T., Pillet, L., Decelle, J., Poirier, C., Suzuki, N., and Not, F. (2015). Towards an integrative morpho-molecular classification of the collodaria (polycystinea, radiolaria). *Protist*, 166(3):374–388.

[21] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

[22] Bolisetty, M. T., Rajadinakaran, G., and Graveley, B. R. (2015). Determining exon connectivity in complex mrnas by nanopore sequencing. *Genome biology*, 16(1):204.

[23] Bork, P., Bowler, C., De Vargas, C., Gorsky, G., Karsenti, E., and Wincker, P. (2015). Tara oceans studies plankton at planetary scale.

[24] Bouck, J., Yu, W., Gibbs, R., and Worley, K. (1999). Comparison of gene indexing databases. *Trends in Genetics*, 15(4):159–162.

[25] Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525.

[26] Burke, J., Davison, D., and Hide, W. (1999a). d2_cluster: a validated method for clustering est and full-length cdna sequences. *Genome research*, 9(11):1135–1142.

[27] Burke, J., Davison, D., and Hide, W. (1999b). d2_cluster: a validated method for clustering est and full-length cdna sequences. *Genome research*, 9(11):1135–1142.

[28] Byrne, A., Beaudin, A. E., Olsen, H. E., Jain, M., Cole, C., Palmer, T., DuBois, R. M., Forsberg, E. C., Akeson, M., and Vollmers, C. (2017). Nanopore long-read rnaseq reveals widespread transcriptional variation among the surface receptors of individual b cells. *Nature Communications*, 8:16027.

[29] Carneiro, M. O., Russ, C., Ross, M. G., Gabriel, S. B., Nusbaum, C., and DePristo, M. A. (2012). Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC genomics*, 13(1):375.

[30] Carvalho, A. B., Dupim, E. G., and Goldstein, G. (2016). Improved assembly of noisy long reads by k-mer validation. *Genome Research*, 26(12):1710–1720.

[31] Chacko, E. and Ranganathan, S. (2009). Comprehensive splicing graph analysis of alternative splicing patterns in chicken, compared to human and mouse. *BMC genomics*, 10(1):S5.

[32] Chaisson, M. J. and Tesler, G. (2012). Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory. *BMC bioinformatics*, 13(1):238.

[33] Chikhi, R., Rizk, G., et al. (2013). Space-efficient and exact de bruijn graph representation based on a bloom filter. *Algorithms for Molecular Biology*, 8(22):1.

[34] Chin, C.-S., Alexander, D. H., Marks, P., Klammer, A. A., Drake, J., Heiner, C., Clum, A., Copeland, A., Huddleston, J., Eichler, E. E., et al. (2013). Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nature methods*, 10(6):563–569.

[35] Chin, C.-S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., O'Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., et al. (2016). Phased diploid genome assembly with single-molecule real-time sequencing. *Nature methods*, 13(12):1050.

[36] Chong, Z., Ruan, J., and Wu, C.-I. (2012). Rainbow: an integrated tool for efficient clustering and assembling rad-seq reads. *Bioinformatics*, 28(21):2732–2737.

[37] Christoffels, A., van Gelder, A., Greyling, G., Miller, R., Hide, T., and Hide, W. (2001). Stack: sequence tag alignment and consensus knowledgebase. *Nucleic Acids Research*, 29(1):234–238.

[38] Chu, J., Mohamadi, H., Warren, R. L., Yang, C., and Birol, I. (2016). Innovations and challenges in detecting long read overlaps: an evaluation of the state-of-the-art. *Bioinformatics*, 33(8):1261–1270.

[39] Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

[40] Cock, P. J., Fields, C. J., Goto, N., Heuer, M. L., and Rice, P. M. (2009). The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771.

[41] Commons, W. (2018). File:difference dna rna-en.svg — wikimedia commons, the free media repository. [Online; accessed 24-May-2018].

[42] Consortium, E. P. et al. (2004). The encode (encyclopedia of dna elements) project. *Science*, 306(5696):636–640.

[43] Consortium, E. P. et al. (2007). Identification and analysis of functional elements in 1% of the human genome by the encode pilot project. *nature*, 447(7146):799.

[44] Cormen, T. H. (2009). *Introduction to algorithms*. MIT press.

[45] Cusack, B. P., Arndt, P. F., Duret, L., and Crollius, H. R. (2011). Preventing dangerous nonsense: selection for robustness to transcriptional error in human genes. *PLoS genetics*, 7(10):e1002276.

[46] Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., and Yannakakis, M. (1994). The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894.

[47] David, M., Dzamba, M., Lister, D., Ilie, L., and Brudno, M. (2011). Shrimp2: sensitive yet practical short read mapping. *Bioinformatics*, 27(7):1011–1012.

[48] Davy, S. K., Allemand, D., and Weis, V. M. (2012). Cell biology of cnidarian-dinoflagellate symbiosis. *Microbiology and Molecular Biology Reviews*, 76(2):229–261.

[49] de Lannoy, C., de Ridder, D., and Risse, J. (2017). The long reads ahead: de novo genome assembly using the minion [version 2; referees: 2 approved]. *F1000Research*, 6(1083).

[50] Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O., and Salzberg, S. L. (1999). Alignment of whole genomes. *Nucleic acids research*, 27(11):2369–2376.

[51] Djebali, S., Davis, C. A., Merkel, A., Dobin, A., Lassmann, T., Mortazavi, A., Tanzer, A., Lagarde, J., Lin, W., Schlesinger, F., et al. (2012). Landscape of transcription in human cells. *Nature*, 489(7414):101.

[52] Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T. R. (2013). Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21.

[53] Dost, B., Wu, C., Su, A., and Bafna, V. (2011). Tclust: A fast method for clustering genome-scale expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(3):808–818.

[54] Drezen, E., Rizk, G., Chikhi, R., Deltel, C., Lemaitre, C., Peterlongo, P., and Lavenier, D. (2014). Gatb: Genome assembly & analysis tool box. *Bioinformatics*, 30(20):2959–2961.

[55] Eckmann, J.-P. and Moses, E. (2002). Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the national academy of sciences*, 99(9):5825–5829.

[56] Edery, P., Marcaillou, C., Sahbatou, M., Labalme, A., Chastang, J., Touraine, R., Tubacher, E., Senni, F., Bober, M. B., Nampoothiri, S., et al. (2011). Association of tals developmental disorder with defect in minor splicing component u4atac snrna. *Science*, 332(6026):240–243.

[57] Edgar, R. C. (2010). Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461.

[58] Escalona, M., Rocha, S., and Posada, D. (2016). A comparison of tools for the simulation of genomic next-generation sequencing data. *Nature Reviews Genetics*, 17(8):459.

[59] et al, P. (2008). Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing.

[60] Feng, D.-F. and Doolittle, R. F. (1997). Converting amino acid alignment scores into measures of evolutionary time: a simulation study of various relationships. *Journal of molecular evolution*, 44(4):361–370.

[61] Fichot, E. B. and Norman, R. S. (2013). Microbial phylogenetic profiling with the pacific biosciences sequencing platform. *Microbiome*, 1(1):10.

[62] Fiore, C. L., Labrie, M., Jarett, J. K., and Lesser, M. P. (2015). Transcriptional activity of the giant barrel sponge, xestospongia muta holobiont: molecular evidence for metabolic interchange. *Frontiers in microbiology*, 6:364.

[63] Foissac, S. and Sammeth, M. (2007). Astalavista: dynamic and flexible analysis of alternative splicing events in custom gene datasets. *Nucleic acids research*, 35(suppl 2):W297–W299.

[64] Forster, D., Bittner, L., Karkar, S., Dunthorn, M., Romac, S., Audic, S., Lopez, P., Stoeck, T., and Bapteste, E. (2015). Testing ecological theories with sequence similarity networks: marine ciliates exhibit similar geographic dispersal patterns as multicellular organisms. *BMC biology*, 13(1):16.

[65] Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3-5):75–174.

[66] Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41.

[67] Fortunato, S., Latora, V., and Marchiori, M. (2004). Method to find community structures based on information centrality. *Physical review E*, 70(5):056104.

[68] Friedman, K. J., Kole, J., Cohn, J. A., Knowles, M. R., Silverman, L. M., and Kole, R. (1999). Correction of aberrant splicing of the cystic fibrosis transmembrane conductance regulator (cftr) gene by antisense oligonucleotides. *Journal of Biological Chemistry*, 274(51):36193–36199.

[69] Fuda, N. J., Ardehali, M. B., and Lis, J. T. (2009). Defining mechanisms that regulate rna polymerase ii transcription in vivo. *Nature*, 461(7261):186.

[70] Gauthier, J., Mouden, C., Suchan, T., Alvarez, N., Arrigo, N., Riou, C., Lemaitre, C., and Peterlongo, P. (2017). Discosnp-rad: de novo detection of small variants for population genomics. *bioRxiv*, page 216747.

[71] Ghodsi, M., Liu, B., and Pop, M. (2011). Dnaclust: accurate and efficient clustering of phylogenetic marker genes. *BMC bioinformatics*, 12(1):271.

[72] Girvan, M. and Newman, M. E. (2002a). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

[73] Girvan, M. and Newman, M. E. (2002b). Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826.

[74] Gonzalez-Garay, M. L. (2016). Introduction to isoform sequencing using pacific biosciences technology (iso-seq). In *Transcriptomics and Gene Regulation*, pages 141–160. Springer.

[75] Good, B. H., de Montjoye, Y.-A., and Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106.

[76] Goodwin, S., Gurtowski, J., Ethe-Sayers, S., Deshpande, P., Schatz, M. C., and McCombie, W. R. (2015). Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, 25(11):1750–1756.

[77] Gordon, S. P., Tseng, E., Salamov, A., Zhang, J., Meng, X., Zhao, Z., Kang, D., Underwood, J., Grigoriev, I. V., Figueroa, M., et al. (2015). Widespread polycistronic transcripts in fungi revealed by single-molecule mrna sequencing. *PloS one*, 10(7):e0132628.

[78] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., et al. (2011a). Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature biotechnology*, 29(7):644–652.

[79] Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., et al. (2011b). Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data. *Nature biotechnology*, 29(7):644.

[80] Grant, G. R., Farkas, M. H., Pizarro, A. D., Lahens, N. F., Schug, J., Brunk, B. P., Stoeckert, C. J., Hogenesch, J. B., and Pierce, E. A. (2011). Comparative analysis of rna-seq alignment algorithms and the rna-seq unified mapper (rum). *Bioinformatics*, 27(18):2518–2528.

[81] Griebel, T., Zacher, B., Ribeca, P., Raineri, E., Lacroix, V., Guigó, R., and Sammeth, M. (2012). Modelling and simulating generic rna-seq experiments with the flux simulator. *Nucleic acids research*, 40(20):10073–10083.

[82] Guidi, L., Chaffron, S., Bittner, L., Eveillard, D., Larhlimi, A., Roux, S., Darzi, Y., Audic, S., Berline, L., Brum, J. R., et al. (2016). Plankton networks driving carbon export in the oligotrophic ocean. *Nature*, 532(7600):465.

[83] Gupta, S. K., Kececioglu, J. D., and Schäffer, A. A. (1995). Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2(3):459–472.

[84] Hackl, T., Hedrich, R., Schultz, J., and Förster, F. (2014). proovread: large-scale high-accuracy pacbio correction through iterative short read consensus. *Bioinformatics*, 30(21):3004–3011.

[85] Haghshenas, E., Hach, F., Sahinalp, S. C., and Chauve, C. (2016). Colormap: Correcting long reads by mapping short reads. *Bioinformatics*, 32(17):i545–i551.

[86] Heber, S., Alekseyev, M., Sze, S.-H., Tang, H., and Pevzner, P. A. (2002). Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl 1):S181–S188.

[87] Hentschel, U., Usher, K. M., and Taylor, M. W. (2006). Marine sponges as microbial fermenters. *FEMS microbiology ecology*, 55(2):167–177.

[88] Higgins, D. G. and Sharp, P. M. (1988). Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244.

[89] Hoang, N. V., Furtado, A., Mason, P. J., Marquardt, A., Kasirajan, L., Thirugnanasambandam, P. P., Botha, F. C., and Henry, R. J. (2017). A survey of the complex transcriptome from the highly polyploid sugarcane genome using full-length isoform sequencing and de novo assembly from short read sequencing. *BMC genomics*, 18(1):395.

[90] Hoegh-Guldberg, O. (1999). Climate change, coral bleaching and the future of the world's coral reefs. *Marine and freshwater research*, 50(8):839–866.

[91] Holter, N. S., Mitra, M., Maritan, A., Cieplak, M., Banavar, J. R., and Fedoroff, N. V. (2000). Fundamental patterns underlying gene expression profiles: simplicity from complexity. *Proceedings of the National Academy of Sciences*, 97(15):8409–8414.

[92] Hopcroft, J. and Tarjan, R. (1973). Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378.

[93] Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2011). Art: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

[94] Huang, X. and Madan, A. (1999). Cap3: A dna sequence assembly program. *Genome research*, 9(9):868–877.

[95] Huson, D. H., Auch, A. F., Qi, J., and Schuster, S. C. (2007). Megan analysis of metagenomic data. *Genome research*, 17(3):377–386.

[96] Jain, M., Fiddes, I. T., Miga, K. H., Olsen, H. E., Paten, B., and Akeson, M. (2015). Improved data analysis for the minion nanopore sequencer. *Nature methods*, 12(4):351.

[97] Jiang, H., Ling, Z., Zhang, Y., Mao, H., Ma, Z., Yin, Y., Wang, W., Tang, W., Tan, Z., Shi, J., et al. (2015). Altered fecal microbiota composition in patients with major depressive disorder. *Brain, behavior, and immunity*, 48:186–194.

[98] Jonsson, P. F., Cavanna, T., Zicha, D., and Bates, P. A. (2006). Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. *BMC bioinformatics*, 7(1):2.

[99] Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562.

[100] Karsenti, E., Acinas, S. G., Bork, P., Bowler, C., de Vargas, C., Raes, J., Sullivan, M., Arendt, D., Benzoni, F., Claverie, J. M., Follows, M., Gorsky, G., Hingamp, P., Iudicone, D., Jaillon, O., Kandels-Lewis, S., Krzic, U., Not, F., Ogata, H., Pesant, S., Reynaud, E. G., Sardet, C., Sieracki, M. E., Speich, S., Velayoudon, D., Weissenbach, J., and Wincker, P. (2011). A holistic approach to marine Eco-systems biology. *PLoS Biology*, 9.

[101] Katz, Y., Wang, E. T., Airoldi, E. M., and Burge, C. B. (2010). Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature methods*, 7(12):1009–1015.

[102] Keeling, P. J., Burki, F., Wilcox, H. M., Allam, B., Allen, E. E., Amaral-Zettler, L. A., Armbrust, E. V., Archibald, J. M., Bharti, A. K., Bell, C. J., et al. (2014). The marine microbial eukaryote transcriptome sequencing project (mmetsp): illuminating the functional diversity of eukaryotic life in the oceans through transcriptome sequencing. *PLoS Biology*, 12(6):e1001889.

[103] Kent, W. J. (2002). Blat – the blast-like alignment tool. *Genome research*, 12(4):656–664.

[104] Khiste, N. and Ilie, L. (2017). Hisea: Hierarchical seed aligner for pacbio data. *BMC Bioinformatics*, 18(1):564.

[105] Kiełbasa, S. M., Wan, R., Sato, K., Horton, P., and Frith, M. C. (2011). Adaptive seeds tame genomic sequence comparison. *Genome research*, 21(3):487–493.

[106] Kim, D., Langmead, B., and Salzberg, S. L. (2015). Hisat: a fast spliced aligner with low memory requirements. *Nature methods*, 12(4):357.

[107] Kirsch, A. and Mitzenmacher, M. (2006). Less hashing, same performance: building a better bloom filter. In *European Symposium on Algorithms*, pages 456–467. Springer.

[108] Koren, S., Schatz, M. C., Walenz, B. P., Martin, J., Howard, J. T., Ganapathy, G., Wang, Z., Rasko, D. A., McCombie, W. R., Jarvis, E. D., et al. (2012). Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, 30(7):693.

[109] Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736.

[110] Kornblihtt, A. R., Schor, I. E., Alló, M., Dujardin, G., Petrillo, E., and Muñoz, M. J. (2013). Alternative splicing: a pivotal step between eukaryotic transcription and translation. *Nature reviews Molecular cell biology*, 14(3):153.

[111] Krizanovic, K., Echchiki, A., Roux, J., Sikic, M., Morales-Rozo, A., Tenorio, E. A., Carling, M. D., Cadena, C. D., Kordonowy, L., MacManes, M., et al. (2017). Evaluation of tools for long read rna-seq splice-aware alignment. *Bioinformatics*, 1:7.

[112] Kuehner, J. N., Pearson, E. L., and Moore, C. (2011). Unravelling the means to an end: Rna polymerase ii transcription termination. *Nature reviews Molecular cell biology*, 12(5):283.

[113] Kurtz, S. (1999). Reducing the space requirement of suffix trees. *Software-Practice and Experience*, 29(13):1149–71.

[114] La, S., Haghshenas, E., and Chauve, C. (2017). Lrcstats, a tool for evaluating long reads correction methods. *Bioinformatics*, 33(22):3652–3654.

[115] Lancichinetti, A., Fortunato, S., and Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110.

[116] Langmead, B. (2010). Aligning short sequencing reads with bowtie. *Current protocols in bioinformatics*, pages 11–7.

[117] Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357.

[118] Lappalainen, T., Sammeth, M., Friedländer, M. R., AC't Hoen, P., Monlong, J., Rivas, M. A., Gonzalez-Porta, M., Kurbatova, N., Griebel, T., Ferreira, P. G., et al. (2013). Transcriptome and genome sequencing uncovers functional variation in humans. *Nature*, 501(7468):506.

[119] Le Bras, Y., Collin, O., Monjeaud, C., Lacroix, V., Rivals, É., Lemaitre, C., Miele, V., Sacomoto, G., Marchet, C., Cazaux, B., et al. (2016). Colib'read on galaxy: a tools suite dedicated to biological information extraction from raw ngs reads. *GigaScience*, 5(1):9.

[120] Lee, C. (2003). Generating consensus sequences from partial order multiple sequence alignment graphs. *Bioinformatics*, 19(8):999–1008.

[121] Lee, C., Grasso, C., and Sharlow, M. F. (2002). Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464.

[122] Li, B. and Dewey, C. N. (2011). Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC bioinformatics*, 12(1):323.

[123] Li, B., Fillmore, N., Bai, Y., Collins, M., Thomson, J. A., Stewart, R., and Dewey, C. N. (2014). Evaluation of de novo transcriptome assemblies from rna-seq data. *Genome biology*, 15(12):553.

[124] Li, F., Jiang, C., Krausz, K. W., Li, Y., Albert, I., Hao, H., Fabre, K. M., Mitchell, J. B., Patterson, A. D., and Gonzalez, F. J. (2013). Microbiome remodelling leads to inhibition of intestinal farnesoid x receptor signalling and decreased obesity. *Nature communications*, 4:2384.

[125] Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*.

[126] Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110.

[127] Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 1:7.

[128] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760.

[129] Li, H. and Durbin, R. (2010). Fast and accurate long-read alignment with burrows–wheeler transform. *Bioinformatics*, 26(5):589–595.

[130] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079.

[131] Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

[132] Lima, L., Sinaimeri, B., Sacomoto, G., Lopez-Maestre, H., Marchet, C., Miele, V., Sagot, M.-F., and Lacroix, V. (2017). Playing hide and seek with repeats in local and global de novo transcriptome assembly of short rna-seq reads. *Algorithms for molecular biology*, 12(1):2.

[133] Limasset, A., Flot, J.-F., and Peterlongo, P. (2018). Toward perfect reads: self-correction of short reads via mapping on de bruijn graphs. In *RECOMB-seq*.

[134] Limasset, A., Rizk, G., Chikhi, R., and Peterlongo, P. (2017). Fast and scalable minimal perfect hashing for massive key sets. *arXiv preprint arXiv:1702.03154*.

[135] Lipman, D. J., Altschul, S. F., and Kececioglu, J. D. (1989). A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415.

[136] Liu, Y., Schröder, J., and Schmidt, B. (2012). Musket: a multistage k-mer spectrum-based error corrector for illumina sequence data. *Bioinformatics*, 29(3):308–315.

[137] Loman, N. J., Quick, J., and Simpson, J. T. (2015). A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature methods*, 12(8):733.

[138] Lopez-Maestre, H., Brinza, L., Marchet, C., Kielbassa, J., Bastien, S., Boutigny, M., Monnin, D., Filali, A. E., Carareto, C. M., Vieira, C., et al. (2016). Snp calling from rna-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. *Nucleic acids research*, 44(19):e148–e148.

[139] Madoui, M.-A., Engelen, S., Cruaud, C., Belser, C., Bertrand, L., Alberti, A., Lemainque, A., Wincker, P., and Aury, J.-M. (2015). Genome assembly using nanopore-guided long and error-free dna reads. *BMC genomics*, 16(1):327.

[140] Mahé, F., Rognes, T., Quince, C., de Vargas, C., and Dunthorn, M. (2014). Swarm: robust and fast clustering method for amplicon-based studies. *PeerJ*, 2:e593.

[141] Maillet, N., Collet, G., Vannier, T., Lavenier, D., and Peterlongo, P. (2014). COMMET: comparing and combining multiple metagenomic datasets. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 94–98. IEEE.

[142] Maillet, N., Lemaitre, C., Chikhi, R., Lavenier, D., and Peterlongo, P. (2012). Compareads: comparing huge metagenomic experiments. In *BMC bioinformatics*, volume 13, page S10. BioMed Central.

[143] Manber, U. and Myers, G. (1993). Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948.

[144] Marchet, C., Lecompte, L., Da Silva, C., Cruaud, C., Aury, J. M., Nicolas, J., and Peterlongo, P. (2018a). Clustering de novo by gene of long reads from transcriptomics data. *bioRxiv*, page 170035.

[145] Marchet, C., Lecompte, L., Limasset, A., Bittner, L., and Peterlongo, P. (2018b). A resource-frugal probabilistic dictionary and applications in bioinformatics. *Discrete Applied Mathematics*.

[146] Marchet, C., Limasset, A., Bittner, L., and Peterlongo, P. (2016). A resource-frugal probabilistic dictionary and applications in (meta)genomics. In Holub, J. and Žďárek, J., editors, *Proceedings of the Prague Stringology Conference 2016*, pages 85–97", Czech Technical University in Prague, Czech Republic.

[147] Marsaglia, G. (2003). Xorshift rngs. *Journal of Statistical Software*, 8(14):1–6.

[148] Martin, J. A. and Wang, Z. (2011). Next-generation transcriptome assembly. *Nature Reviews Genetics*, 12(10):671.

[149] Matera, A. G. and Wang, Z. (2014). A day in the life of the spliceosome. *Nature reviews Molecular cell biology*, 15(2):108.

[150] McCreight, E. M. (1976). A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM)*, 23(2):262–272.

[151] McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., et al. (2010). The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303.

[152] Mei, J., Zhao, J., Yang, X., and Zhou, W. (2011). Remote protein homology detection using a modularity-based approach. In *Information Science and Technology (ICIST), 2011 International Conference on*, pages 1287–1291. IEEE.

[153] Meng, A., Marchet, C., Corre, E., Peterlongo, P., Alberti, A., Da Silva, C., Wincker, P., Pelletier, E., Probert, I., Decelle, J., et al. (2017a). A de novo approach to disentangle partner identity and function in holobiont systems.

[154] Meng, A., Marchet, C., Corre, E., Peterlongo, P., Alberti, A., Da Silva, C., Wincker, P., Pelletier, E., Probert, I., Decelle, J., Le Crom, S., Not, F., and Bittner, L. (2017b). A de novo approach to disentangle partner identity and function in holobiont systems. working paper or preprint.

[155] Miclotte, G., Heydari, M., Demeester, P., Rombauts, S., Van de Peer, Y., Audenaert, P., and Fostier, J. (2016). Jabba: hybrid error correction for long sequencing reads. *Algorithms for Molecular Biology*, 11(1):10.

[156] Miele, V., Penel, S., Daubin, V., Picard, F., Kahn, D., and Duret, L. (2012). High-quality sequence clustering guided by network topology and multiple alignment likelihood. *Bioinformatics*, 28(8):1078–1085.

[157] Modrek, B. and Lee, C. (2002). A genomic view of alternative splicing. *Nature genetics*, 30(1):13–19.

[158] Morin, R. D., Bainbridge, M., Fejes, A., Hirst, M., Krzywinski, M., Pugh, T. J., McDonald, H., Varhol, R., Jones, S. J., and Marra, M. A. (2008). Profiling the hela s3 transcriptome using randomly primed cdna and massively parallel short-read sequencing. *Biotechniques*, 45(1):81.

[159] Mott, R. (1997). Est_genome: a program to align spliced dna sequences to unspliced genomic dna. *Bioinformatics*, 13(4):477–478.

[160] Muller-Parker, G., D'elia, C. F., and Cook, C. B. (2015). Interactions between corals and their symbiotic algae. In *Coral Reefs in the Anthropocene*, pages 99–116. Springer.

[161] Myers, G. (2014a). Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer.

[162] Myers, G. (2014b). Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer.

[163] Needleman, S. B. and Wunsch, C. D. (1970). 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.

[164] Newman, M. E. (2003). The structure and function of complex networks. *SIAM review*, 45(2):167–256.

[165] Newman, M. E. (2004a). Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330.

[166] Newman, M. E. (2004b). Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133.

[167] Novák, P., Neumann, P., and Macas, J. (2010). Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. *BMC bioinformatics*, 11(1):378.

[168] Oikonomopoulos, S., Wang, Y. C., Djambazian, H., Badescu, D., and Ragoussis, J. (2016). Benchmarking of the oxford nanopore minion sequencing for quantitative and qualitative assessment of cdna populations. *Scientific reports*, 6:31602.

[169] Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., and Phillippy, A. M. (2016). Mash: fast genome and metagenome distance estimation using minhash. *Genome biology*, 17(1):132.

[170] Ono, Y., Asai, K., and Hamada, M. (2013). Pbsim: Pacbio reads simulator—toward accurate genome assembly. *Bioinformatics*, 29(1):119–121.

[171] Oshlack, A., Robinson, M. D., and Young, M. D. (2010). From rna-seq reads to differential expression results. *Genome biology*, 11(12):220.

[172] Palla, G., Barabási, A., and Vicsek, T. (2007). Quantifying social group evolution. *Nature*, 446(7136):664–667.

[173] Pan, Q., Shai, O., Lee, L. J., Frey, B. J., and Blencowe, B. J. (2008). Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature genetics*, 40(12):1413.

[174] Pardalos, J. A. and Resende, M. (1999). On maximum cli, ue problems in very large graphs.

[175] Patro, R., Duggal, G., and Kingsford, C. (2015). Salmon: accurate, versatile and ultrafast quantification from rna-seq data using lightweight-alignment. *bioRxiv*, page 021592.

[176] Patro, R., Mount, S. M., and Kingsford, C. (2014). Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462.

[177] Pattillo, J., Veremyev, A., Butenko, S., and Boginski, V. (2013). On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1-2):244–257.

[178] Pérez-Ortín, J. E., Alepuz, P., Chávez, S., and Choder, M. (2013). Eukaryotic mrna decay: methodologies, pathways, and links to other stages of gene expression. *Journal of molecular biology*, 425(20):3750–3775.

[179] Pertea, G., Huang, X., Liang, F., Antonescu, V., Sultana, R., Karamycheva, S., Lee, Y., White, J., Cheung, F., Parvizi, B., et al. (2003). Tigr gene indices clustering tools (tgicl): a software system for fast clustering of large est datasets. *Bioinformatics*, 19(5):651–652.

[180] Pesant, S., Not, F., Picheral, M., Kandels-Lewis, S., Le Bescot, N., Gorsky, G., Iudicone, D., Karsenti, E., Speich, S., Troublé, R., et al. (2015). Open science resources for the discovery and analysis of tara oceans data. *Scientific data*, 2:150023.

[181] Philippe, N., Salson, M., Commes, T., and Rivals, E. (2013). Crac: an integrated approach to the analysis of rna-seq reads. *Genome Biology*, 14(3):R30.

[182] Pinzón, J. H., Kamel, B., Burge, C. A., Harvell, C. D., Medina, M., Weil, E., and Mydlarz, L. D. (2015). Whole transcriptome analysis reveals changes in expression of immune-related genes during and after bleaching in a reef-building coral. *Royal Society open science*, 2(4):140214.

[183] Porrua, O. and Libri, D. (2013). Rna quality control in the nucleus: the angels' share of rna. *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms*, 1829(6):604–611.

[184] Poursani, E. M., Soltani, B. M., and Mowla, S. J. (2016). Differential expression of oct4 pseudogenes in pluripotent and tumor cell lines. *Cell Journal (Yakhteh)*, 18(1):28.

[185] Probert, I., Siano, R., Poirier, C., Decelle, J., Biard, T., Tuji, A., Suzuki, N., and Not, F. (2014). Brandtodinium gen. nov. and b. nutricula comb. nov.(dinophyceae), a dinoflagellate commonly found in symbiosis with polycystine radiolarians. *Journal of phycology*, 50(2):388–399.

[186] Quackenbush, J., Liang, F., Holt, I., Pertea, G., and Upton, J. (2000). The tigr gene indices: reconstruction and representation of expressed gene sequences. *Nucleic Acids Research*, 28(1):141–145.

[187] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663.

[188] Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106.

[189] Reuter, J., Spacek, D. V., and Snyder, M. (2015). High-throughput sequencing technologies. *Molecular Cell*, 58(4):586 – 597.

[190] Reyes, A., Anders, S., and Huber, W. (2012). Analyzing RNA-seq data for differential exon usage with the DEXSeq package.

[191] Rizk, G., Lavenier, D., and Chikhi, R. (2013a). Dsk: k-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–653.

[192] Rizk, G., Lavenier, D., and Chikhi, R. (2013b). DSK: K-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–653.

[193] Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M., and Yorke, J. A. (2004). Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369.

[194] Robertson, G., Schein, J., Chiu, R., Corbett, R., Field, M., Jackman, S. D., Mungall, K., Lee, S., Okada, H. M., Qian, J. Q., et al. (2010). De novo assembly and analysis of RNA-seq data. *Nature methods*, 7(11):909–912.

[195] Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., Lander, E. S., Getz, G., and Mesirov, J. P. (2011). Integrative genomics viewer. *Nature biotechnology*, 29(1):24.

[196] Rodwell, G. E., Sonu, R., Zahn, J. M., Lund, J., Wilhelmy, J., Wang, L., Xiao, W., Mindrinos, M., Crane, E., Segal, E., et al. (2004). A transcriptional profile of aging in the human kidney. *PLoS biology*, 2(12):e427.

[197] Rohwer, F., Seguritan, V., Azam, F., and Knowlton, N. (2002). Diversity and distribution of coral-associated bacteria. *Marine Ecology Progress Series*, 243:1–10.

[198] Romero, I. G., Pai, A. A., Tung, J., and Gilad, Y. (2014). Rna-seq: impact of rna degradation on transcript quantification. *BMC biology*, 12(1):42.

[199] Sacomoto, G., Sinaimeri, B., Marchet, C., Miele, V., Sagot, M.-F., and Lacroix, V. (2014). Navigating in a sea of repeats in rna-seq without drowning. In *International Workshop on Algorithms in Bioinformatics*, pages 82–96. Springer.

[200] Sacomoto, G. A., Kielbassa, J., Chikhi, R., Uricaru, R., Antoniou, P., Sagot, M.-F., Peterlongo, P., and Lacroix, V. (2012). Kissplice: de-novo calling alternative splicing events from rna-seq data. *BMC bioinformatics*, 13(Suppl 6):S5.

[201] Sakharkar, M. K., Chow, V. T., and Kangueane, P. (2004). Distributions of exons and introns in the human genome. *In silico biology*, 4(4):387–393.

[202] Salmela, L. and Rivals, E. (2014). Lordec: accurate and efficient long read error correction. *Bioinformatics*, 30(24):3506–3514.

[203] Salmela, L. and Schröder, J. (2011). Correcting errors in short reads by multiple alignments. *Bioinformatics*, 27(11):1455–1461.

[204] Salmela, L., Walve, R., Rivals, E., and Ukkonen, E. (2016). Accurate self-correction of errors in long reads using de bruijn graphs. *Bioinformatics*, 33(6):799–806.

[205] Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467.

[206] Sangwan, N., Xia, F., and Gilbert, J. A. (2016). Recovering complete and draft population genomes from metagenome datasets. *Microbiome*, 4(1):8.

[207] Sankoff, D. (1985). Simultaneous solution of the rna folding, alignment and protosequence problems. *SIAM journal on applied mathematics*, 45(5):810–825.

[208] Schadt, E. E., Monks, S. A., Drake, T. A., Lusis, A. J., et al. (2003). Genetics of gene expression surveyed in maize, mouse and man. *Nature*, 422(6929):297.

[209] Schuler, G. D. (1997). Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *Journal of Molecular Medicine*, 75(10):694–698.

[210] Schulz, M. H., Zerbino, D. R., Vingron, M., and Birney, E. (2012). Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092.

[211] Schwarz, J. A., Brokstein, P. B., Voolstra, C., Terry, A. Y., Miller, D. J., Szmant, A. M., Coffroth, M. A., and Medina, M. (2008). Coral life history and symbiosis: functional genomic resources for two reef building caribbean corals, acropora palmata and montastraea faveolata. *BMC genomics*, 9(1):97.

[212] Selosse, M.-A. and Strullu-Derrien, C. (2015). Origins of the terrestrial flora: a symbiosis with fungi? In *BIO Web of Conferences*, volume 4, page 00009. EDP Sciences.

[213] Senior, J. K. (1951). Partitions and their representative graphs. *American Journal of Mathematics*, 73(3):663–689.

[214] Sharon, D., Tilgner, H., Grubert, F., and Snyder, M. (2013). A single-molecule long-read survey of the human transcriptome. *Nature biotechnology*, 31(11):1009–1014.

[215] Shen, S., Park, J. W., Huang, J., Dittmar, K. A., Lu, Z.-x., Zhou, Q., Carstens, R. P., and Xing, Y. (2012). MATS: a bayesian framework for flexible detection of differential alternative splicing from RNA-seq data. *Nucleic acids research*, page gkr1291.

[216] Siegl, A., Kamke, J., Hochmuth, T., Piel, J., Richter, M., Liang, C., Dandekar, T., and Hentschel, U. (2011). Single-cell genomics reveals the lifestyle of poribacteria, a candidate phylum symbiotically associated with marine sponges. *The ISME journal*, 5(1):61.

[217] Simister, R. L., Deines, P., Botté, E. S., Webster, N. S., and Taylor, M. W. (2012). Sponge-specific clusters revisited: a comprehensive phylogeny of sponge-associated microorganisms. *Environmental Microbiology*, 14(2):517–524.

[218] Simpson, J. T., Workman, R. E., Zuzarte, P., David, M., Dursi, L., and Timp, W. (2017). Detecting dna cytosine methylation using nanopore sequencing. *nature methods*, 14(4):407.

[219] Smith, T. F. and Waterman, M. S. (1981). Comparison of biosequences. *Advances in applied mathematics*, 2(4):482–489.

[220] Smith-Unna, R., Boursnell, C., Patro, R., Hibberd, J. M., and Kelly, S. (2016). Transrate: reference-free quality assessment of de novo transcriptome assemblies. *Genome research*, 26(8):1134–1144.

[221] Soderlund, C., Johnson, E., Bomhoff, M., and Descour, A. (2009). Pave: program for assembling and viewing ests. *Bmc Genomics*, 10(1):1.

[222] Song, L. and Florea, L. (2015). Rcorrector: efficient and accurate error correction for illumina rna-seq reads. *Gigascience*, 4(1):48.

[223] Song, L., Florea, L., and Langmead, B. (2014). Lighter: fast and memory-efficient sequencing error correction without counting. *Genome biology*, 15(11):509.

[224] Sović, I., Šikić, M., Wilm, A., Fenlon, S. N., Chen, S., and Nagarajan, N. (2016). Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature communications*, 7.

[225] Stöcker, B. K., Köster, J., and Rahmann, S. (2016). Simlord: Simulation of long read data. *Bioinformatics*, 32(17):2704–2706.

[226] Sunagawa, S., Coelho, L. P., Chaffron, S., Kultima, J. R., Labadie, K., Salazar, G., Djahanschiri, B., Zeller, G., Mende, D. R., Alberti, A., et al. (2015). Structure and function of the global ocean microbiome. *Science*, 348(6237):1261359.

[227] Tazi, J., Bakkour, N., and Stamm, S. (2009). Alternative splicing and disease. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1792(1):14–26.

[228] Thanh, N. M., Jung, H., Lyons, R. E., Njaci, I., Yoon, B.-H., Chand, V., Tuan, N. V., Thu, V. T. M., and Mather, P. (2015). Optimizing de novo transcriptome assembly and extending genomic resources for striped catfish (pangasianodon hypophthalmus). *Marine genomics*, 23:87–97.

[229] Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*, 22(22):4673–4680.

[230] Tilgner, H., Jahanbani, F., Gupta, I., Collier, P., Wei, E., Rasmussen, M., and Snyder, M. (2018). Microfluidic isoform sequencing shows widespread splicing coordination in the human transcriptome. *Genome research*, 28(2):231–242.

[231] Tischler, G. and Myers, E. W. (2017). Non hybrid long read consensus using local de bruijn graph assembly. *bioRxiv*, page 106252.

[232] Torney, D. C., Burks, C., Davison, D., and Sirotkin, K. M. (1990). Computation of d2: a measure of sequence dissimilarity. In *Computers and DNA: the proceedings of the Interface between Computation Science and Nucleic Acid Sequencing Workshop, held December 12 to 16, 1988 in Santa Fe, New Mexico/edited by George I. Bell, Thomas G. Marr*. Redwood City, Calif.: Addison-Wesley Pub. Co., 1990.

[233] Toseland, A., Moxon, S., Mock, T., and Moulton, V. (2014). Metatranscriptomes from diverse microbial communities: assessment of data reduction techniques for rigorous annotation. *BMC genomics*, 15(1):901.

[234] Trapnell, C., Pachter, L., and Salzberg, S. L. (2009). Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111.

[235] Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D. R., Pimentel, H., Salzberg, S. L., Rinn, J. L., and Pachter, L. (2012). Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature protocols*, 7(3):562–578.

[236] Tress, M. L., Bodenmiller, B., Aebersold, R., and Valencia, A. (2008). Proteomics studies confirm the presence of alternative protein isoforms on a large scale. *Genome biology*, 9(11):R162.

[237] Turunen, J. J., Niemelä, E. H., Verma, B., and Frilander, M. J. (2013). The significant other: splicing by the minor spliceosome. *Wiley Interdisciplinary Reviews: RNA*, 4(1):61–76.

[238] Vaser, R., Sovic, I., Nagarajan, N., and Sikic, M. (2016). Fast and accurate de novo genome assembly from long uncorrected reads. *bioRxiv*, page 068122.

[239] Vaser, R., Sović, I., Nagarajan, N., and Šikić, M. (2017). Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, 27(5):737–746.

[240] Wang, B., Tseng, E., Regulski, M., Clark, T. A., Hon, T., Jiao, Y., Lu, Z., Olson, A., Stein, J. C., and Ware, D. (2016). Unveiling the complexity of the maize transcriptome by single-molecule long-read sequencing. *Nature communications*, 7:11708.

[241] Wang, G.-S. and Cooper, T. A. (2007). Splicing in disease: disruption of the splicing code and the decoding machinery. *Nature Reviews Genetics*, 8(10):nrg2164.

[242] Wang, K., Singh, D., Zeng, Z., Coleman, S. J., Huang, Y., Savich, G. L., He, X., Mieczkowski, P., Grimm, S. A., Perou, C. M., et al. (2010). Mapsplice: accurate mapping of rna-seq reads for splice junction discovery. *Nucleic acids research*, 38(18):e178–e178.

[243] Wang, Y., Leung, H. C., Yiu, S., and Chin, F. Y. (2012). Metacluster 5.0: a two-round binning approach for metagenomic data for low-abundance species in a noisy sample. *Bioinformatics*, 28(18):i356–i362.

[244] Wang, Z., Gerstein, M., and Snyder, M. (2009). Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57.

[245] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684):440.

[246] Webster, N. S. and Taylor, M. W. (2012). Marine sponges and their microbial symbionts: love and other relationships. *Environmental Microbiology*, 14(2):335–346.

[247] Wei, Z.-G., Zhang, S.-W., and Zhang, Y.-Z. (2017). Dmclust, a density-based modularity method for accurate otu picking of 16s rrna sequences. *Molecular informatics*, 36(12).

[248] Weirather, J. L., de Cesare, M., Wang, Y., Piazza, P., Sebastiano, V., Wang, X.-J., Buck, D., and Au, K. F. (2017). Comprehensive comparison of pacific biosciences and oxford nanopore technologies and their applications to transcriptome analysis. *F1000Research*, 6.

[249] Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):R46.

[250] Wu, T. D., Reeder, J., Lawrence, M., Becker, G., and Brauer, M. J. (2016). Gmap and gsnap for genomic sequence alignment: enhancements to speed, accuracy, and functionality. *Statistical Genomics: Methods and Protocols*, pages 283–334.

[251] Wu, T. D. and Watanabe, C. K. (2005). Gmap: a genomic mapping and alignment program for mrna and est sequences. *Bioinformatics*, 21(9):1859–1875.

[252] Xiao, C.-L., Chen, Y., Xie, S.-Q., Chen, K.-N., Wang, Y., Han, Y., Luo, F., and Xie, Z. (2017). Mecat: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads. *nature methods*, 14(11):1072.

[253] Xie, Y., Wu, G., Tang, J., Luo, R., Patterson, J., Liu, S., Huang, W., He, G., Gu, S., Li, S., et al. (2014). Soapdenovo-trans: de novo transcriptome assembly with short rna-seq reads. *Bioinformatics*, 30(12):1660–1666.

[254] Yang, C., Chu, J., Warren, R. L., and Birol, I. (2017). Nanosim: nanopore sequence read simulator based on statistical characterization. *GigaScience*, 6(4):1–6.

[255] Yang, J. and Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213.

[256] Yang, X., Chockalingam, S. P., and Aluru, S. (2012). A survey of error-correction methods for next-generation sequencing. *Briefings in bioinformatics*, 14(1):56–66.

[257] Yang, Y. and Smith, S. A. (2013). Optimizing de novo assembly of short-read rna-seq data for phylogenomics. *BMC genomics*, 14(1):328.

[258] Ye, C. and Ma, Z. S. (2016). Sparc: a sparsity-based consensus algorithm for long erroneous sequencing reads. *PeerJ*, 4:e2016.

[259] Zhao, M., Lee, W.-P., Garrison, E. P., and Marth, G. T. (2013). Ssw library: an simd smith-waterman c/c++ library for use in genomic applications. *PloS one*, 8(12):e82138.

[260] Zorita, E., Cuscó, P., and Filion, G. J. (2015a). Starcode: sequence clustering based on all-pairs search. *Bioinformatics*, 31(12):1913–1919.

[261] Zorita, E., Cusco, P., and Filion, G. J. (2015b). Starcode: sequence clustering based on all-pairs search. *Bioinformatics*, 31(12):1913–1919.

[262] Zuker, M. and Sankoff, D. (1984). Rna secondary structures and their prediction. *Bulletin of mathematical biology*, 46(4):591–621.